# Galactic Industries Corp.
# Universal Data Format Specification

**Document History:**

| | | |
|---|---|---|
| 10-21-94 | SCS | Original version (GRAMSUI.TXT). |
| 05-18-95 | JHD/BMG | Add NMR log text extensions. |
| 07-13-95 | JHD | Add AIA log text extensions. |
| 09-18-96 | JHD/PEH | Update NMR log text extensions, reformat to split DRIVER commands from converter log text entries, add sections for other techniques (NIR & UV/VIS), add new DRIVER commands for UV/VIS instruments |
| 10-09-96 | JHD/DEA/PEH | Incorporated changes from DEA & PEH to NIR section and DRIVER commands sections respectively. |
| 11-08-96 | JHD | Split DRIVER commands into separate document: “Galactic Industries Universal Driver Specification”. Current document now only covers the Galactic SPC file format Log information. |
| 09-03-97 | JHD | Added definition of `fexper` main header parameter for experimental technique. Extend unit type tables and add new descriptions of SPC header parameters and 4D data sets. Also added more examples and detailed descriptions of header parameters and file structures. Added all header files as Appendices. |

# Table of Contents

## Purpose

This document provides a guide for programmers to implement data reading or data writing routines for files in Galactic Industries' SPC data file format. This format provides for storage of a variety of different types of data taken from laboratory analytical instrumentation. As part of its products, Galactic Industries provides a series of data file conversion utilities that can translate data files taken from industry standard formats (i.e. ASCII, JCAMP, AIA/ANDI/netCDF), or from vendor specific binary formats into the Galactic SPC format. All of Galactic's software products use the SPC format to store and retrieve spectral and chromatographic data.

The Galactic SPC format consists of a main header describing the contents of the file followed by a binary storage area for the instrumentation data and an optional "Log block" for storing other relevant information about the data which cannot be stored in the main header block. In general, this block is used to store ASCII strings of important instrument or sample information that may be relevant to visualizing or processing the data when it is loaded into a Galactic software program. It is also used by Galactic software to store information about modifications made to the attached data. However, the Log block may also be used to store binary data as well. The text area of this block is known as the Log Text block, while the binary portion is called the Log Binary block. In addition to the main header and data areas (which are always present), data files in Galactic SPC format can contain none, either or both Log Text and Log Binary data in the Log block at the end of the file.

The Log Text block is a "free form" area where ASCII text information is stored. Although there is no defined format for the information in Log Text block (it can be used to store any type ASCII text information), Galactic has defined a system of text "keys" that are used to store important information when files are translated into SPC format from another source file. In addition, data acquisition software written by Galactic Industries that interfaces directly to laboratory instrumentation use this system of "keys" to store relevant sample and instrumental parameter information along with the raw data.

The Log Text block is primarily for use with Galactic-written data file converters, but information may be placed in this block by software from other companies who write/convert data files to the Galactic SPC file format or by Galactic-written data processing programs. Note that most of the described "key" parameters are optional, and Galactic file converters need not fill in all possible parameters in the Log Text block unless otherwise noted by the technique (i.e. FT-IR, NIR, UV/VIS, NMR, etc.) specification in this document. This set of key names is designed to take care of the following situations in an attempt to maintain data file "completeness" when it is brought into the Galactic software environment:

1. Data acquisition parameters used on the instrument to collect the data using Galactic software or similar parameters from a foreign file imported with a Galactic-written file converter.
2. Data processing parameters required to properly manipulate, process and/or display the data.
3. Other relevant data that may exist in a foreign file for which there is no corresponding structure in the Galactic SPC file format which is required for describing the sample or data.
4. Other relevant data that may exist in a foreign file which may be required if the file is to be exported back to the exact same foreign format.
5. Processing histories and electronic signatures.

The main goal behind implementation of these standard "keys" is to allow data to be transferred between Galactic SPC files and foreign file formats with minimal loss of important information. While in most cases complete interchangability cannot achieved, the goal is to provide a mechanism to transfer as much relevant information as possible when data is brought into the Galactic SPC file format.

In addition to textual information, some file formats have large portions of extra information that must be maintained as a complete binary block within the SPC file, but this information is not required by Galactic software to display or process the spectrum or chromatogram. The Galactic SPC format provides a Log Binary block for storage of this information. Possible uses of this block include:

1. Storing a copy of a large block of binary information that is required for calculations, but should not be displayed when the file is loaded into Galactic software (i.e. imaginary data in an NMR spectrum).
2. Storing a copy of the file header for a complex file format so that the SPC file may be exported back to the foreign format completely .
3. Storing a complete copy of the foreign binary file such that data integrity and GLP (Good Laboratory Practices) can be maintained when the file is brought back to the native system or the data is archived for future use.

In order to simplify and standardize the process of developing conversion routines, this document was written to describe the set of standard ASCII Log Text “keys” and other storage mechanisms used to store information in a Galactic SPC file. Thus, the hope is that programmers both within Galactic Industries and at other organizations can build utilities to translate data to and from Galactic’s SPC file format and maintain the maximum data completeness possible.

## Other Relevant Documents

The following documents are also relevant for programmers implementing conversion routines to/from Galactic's SPC file format. In addition, these documents should be referred to by Galactic programmers implementing data acquisition routines.

The following documents, source code files, some sample program source code and a copy of this specification are all available from Galactic Industries Corp. in the "Galactic SPC File Software Developer's Kit". This kit is available from Galactic Industries Corp. as a Pkzip file called SPC_SDK.ZIP.

### *Galactic SPC Binary File Format (SPC.H)*

The source code header file SPC.H describes the complete SPC data file structure and all relevant header parameters. This file is absolutely required to properly understand the format and construct properly formatted SPC data files. See Appendix A.

### *Galactic File Converter Specification (GCDLL.H)*

This document describes how to write a Windows DLL-based file converter that will plug into the Galactic Convert engine (GRAMSCNV.EXE). DLL's written to this specification may also be called directly by other Windows programs for conversion services. It describes the function calls and argument lists that must be supported in the DLL and provides for both 16-bit and 32-bit interfaces. There are sample code libraries available as well. This specification can only be used for writing converter DLLs that are compatible with Galactic Convert V2.00 or higher. Prior versions of the convert engine will not work with converters written to this specification.

This source code header file is provided to Galactic programmers for implementation of conversion routines within Galactic products, but can also be provided to other programmers who wish to use the Galactic Convert engine to translate files to/from the SPC format as well. Refer to this file for more information. See Appendix B.

### *GRAMS DDE Interface (GRAMSDDE.H)*

This is a source code file that describes the complete Windows Dynamic Data Exchange (DDE) Client and Server interface in all GRAMS Level II compliant products. This includes information on how to develop a DDE server to interface with GRAMS as the client, and how to send data to GRAMS when it is the server. There are specific items in the SPC data file structure which must be filled in when sending data via DDE that are not required when simply converting data to disk-based SPC files for loading and manipulation in the GRAMS products. This file, other documentation and sample source code are included in a complete developer's kit for the GRAMS DDE interface available from Galactic Industries upon request. See Appendix C.

### *Galactic Industries Universal Driver Specification*

The Log Text entries used for the data file Log Text parameters are, in almost all cases, the same arguments that are used by the DRIVER command interface to Galactic-written ABC data acquisition drivers. This document describes the interfaces that are allowed and those that must be supported when writing a Galactic ABC data acquisition driver. This document is not required to develop data file conversion routines for SPC files, and is primarily only of interest to Galactic programmers when developing data acquisition drivers.

# Converting Data Files To and From Galactic SPC Format

When converting data files between different formats, it is desirable to maintain as much information as possible from the original data. This is certainly required by GLP, but is also a good idea to maintain the "completeness" of the data as usually every piece of information stored in a file is relevant to the data contained therein.

Internally, Galactic Industries has set two policies for all the conversion routines it creates:

- All attempts should be made to maintain data integrity when importing data from the native format into Galactic SPC. However, unless the native file format is simple enough to easily reconstruct converters will not support export from Galactic SPC files back to the native format. Galactic will continue to support and enhance the export capabilities of the converters for “industry standard” formats such as JCAMP and ASCII.
- As of December 1996 all Galactic-written converters are designated to follow the Galactic File Converter (GCDLL) Specification (see information in source code file GCDLL.H) to provide as much portability and flexibility as possible in terms of both the conversion software and the data files.

## *Conversion from a Foreign Format to Galactic SPC*

When data is converted from foreign file formats into Galactic SPC format, every attempt should be made to include all relevant information that is available in the foreign file. This includes filling in the standard SPC main file header items (i.e. X units, Y units, Date, Time, Resolution Text, Memo Text) as well as all instrument parameters possible in the Log Text block.

Galactic has defined sets of standand Log Text keys for commonly used parameters for a variety of instrumental techniques. When at all possible, these keys must be used when converting data to the SPC format to insure the completeness of the data file. While the use of the Log Text parameters is not required to create a valid Galactic SPC file, it allows a greater degree of interchangeability of data to other foreign formats. In general, the Galactic-defined Log Text keys are specific only to each different instrumental technique. In addition, at this time, complete sets of keys are not defined for some techniques. As new conversion routines are developed either by Galactic or third parties that define parameters for these techniques, they will be adopted into the Galactic recommend set and published in future updates of this document. For complete information on the currently defined Log Text keys for instrumental parameters, see the " Universal Log Text Parameters" chapter later in this document

## *Conversion from Galactic SPC to a Foreign Format*

When data is exported from a Galactic SPC file to a foreign file format, the converting program should look for any and all relevant items in the Log Text block to determine common parameters that can be exported along with the raw data. Note that some foreign formats do not support fields for all the data that may exist in the Log Text block, thus those parameters must be ignored on export. In addition, some foreign file formats will support many more parameter fields than are available in the parameters in the Log Text block. In this case, the export must fill in these fields with reasonable defaults that will allow the foreign software to load the exported data file without failure.Finally, when exporting data from a Galactic SPC file with no Log Text block, or no relevant entries in the Log Text block that match required parameters in the foreign file format, the converter should fill in these fields with reasonable defaults. These defaults should be set such that the foreign software can load the exported data file without failure.

In some cases, it will not be possible to properly export data from a Galactic SPC file to a foreign file format. There are many possible reasons for this:

1. Critical information required for constructing the foreign file is not available in the Galactic SPC file and there are no reasonable default values that can be universally used for all possible data files in the foreign format.
2. The documentation on the foreign file format is not complete enough to determine reasonable default values for required parameters.
3. The data structures of the foreign format are too complex to reconstruct, or reconstruction would result in loss of accuracy or precision in the data. Examples of this are file formats that do not use the standard Intel word size and/or ordering, or use byte or word compression methods on the raw data.
4. The information in the foreign file format is not stored in a single data file, but uses complex structures to split the information into many different files and/or directories.

## The Galactic SPC File Format

NOTE: The following information requires a basic understanding of the Galactic SPC binary format structures. Please refer to the SPC.H header file for reference to the variables and structures discussed here.

There are actually 2 different forms of the Galactic SPC file format: a simple array oriented record format used by the DOS-based products Spectra Calc® and Lab Calc®, and a redesigned version used by the newer Windows-based products such as GRAMS/386®, GRAMS/32® and other future products. The differences between these formats are documented in the SPC.H source code. The former format only allows for evenly spaced data arrays, while the latter allows for non-evenly spaced data, and dynamic data array sizes in multi record files to provide complete support for all possible different types of analytical data.

All conversion routines written by Galactic Industries after July 1996 only create SPC data files in the latter format. While future Galactic software products will continue to read all the different formats, they will only write the newer redesigned file format.

All the information in this document only refers to files stored in the “newer” GRAMS version of the SPC file format. The main reason being, that the older format does not allow for Log data storage. If back compatibility with the DOS products is required for the SPC files, then the old format must be used. However note that none of the information in this document regarding the Log Text and Log Binary information is relevant. It is highly recommended that programmers implementing data file translation routines to store data in Galactic’s SPC file format only support the newer file format. However, programmers wishing to write routines to translate SPC data files into other formats must be prepared to read all the different SPC data formats.

As noted previously, the SPC format was designed to be flexible to allow for storage of many different data types. The final construction of an SPC file depends very heavily on the type of data it will hold. The SPC format also allows for different representations of the binary “Y” data. The `ftflgs`, `fversn` and `fexp` parameters in the `SPCHDR` structure (see SPC.H) determine both mechanism of the Y data storage (single or multi-record, evenly spaced X axis or data pairs, enumerated label types or custom strings, constant or dynamic record length, etc.), the format version type of the file (i.e. old or new) and the binary type of the Y values.

The remainder of this section is devoted to describing the important structures and parameters in Galactic SPC files and differentiating the data storage mechanisms offered by the format.

## *The SPCHDR Structure (Main File Header)*

This section describes all the parameters in the SPCHDR structure and how they are used to describe the information stored in the SPC data file. One piece of terminology to note is that in SPC files, the X axis is always the horizontal axis (i.e. wavelength/frequency for spectra, time for chromatograms), the Y axis is always the vertical axis (response), and the Z axis is always the planar axis (event, time or secondary wavelength/frequency).

### `Ftflgs`*: File Data Type Flags*

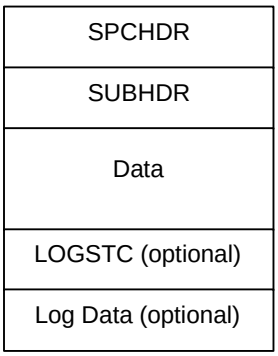
This determines structural organization of the SPC data file. As noted above, the structural layout of an SPC file will depend on the type of data it is being use to store. This value is set by OR'ing together the data file type bit flags:

| Defined Flag | `Ftflgs` Value | Description |
|---|---|---|
| TSPREC | 0x01h | Y data blocks are 16 bit integer (only if `fexp` is NOT 0x80h) |
| TCGRAM | 0x02h | Enables `fexper` in older software (not used) |
| TMULTI | 0x04h | Multifile data format (more than one subfile) |
| TRANDM | 0x08h | If TMULTI and TRANDM then Z values in SUBHDR structures are in random order (not used) |
| TORDRD | 0x10h | If TMULTI and TORDRD then Z values are in ascending or descending ordered but not evenly spaced. Z values read from individual SUBHDR structures. |
| TALABS | 0x20h | Axis label text stored in `fcatxt` separated by nulls. Ignore `fxtype`, `fytype`, `fztype` corresponding to non-null text in `fcatxt`. |
| TXYXYS | 0x40h | Each subfile has unique X array; can only be used if TXVALS is also used. Used exclusively to flag as MS data for drawing as "sticks" rather than connected lines. |
| TXVALS | 0x80h | Non-evenly spaced X data. File has X value array preceding Y data block(s). |

In all cases of SPC files the `ftflgs` value in SPCHDR can be OR'ed with TSPREC to flag the Y data arrays being stored in 16 bit (rather than 32 bit) values. However, this only applies for block scaled integer format data values. If `fexp` is set to 0x80h to flag floating point data values, DO NOT set TSPREC! See the description of the SPCHDR parameter `fexp` for more details.

TALABS is used to flag axis unit label strings being stored in `fcatxt` rather than using the `fxtype`, `fytype`, `fztype` enumerated label types. See the description of the SPCHDR parameter `fcatxt` for more details.

The following diagrams show the layout of different types of SPC files. The parameters in the square brackets [ ] indicate the other possible `ftflgs` bit values that can be OR'ed with the main values to produce the same type of file, with other attributes. Note in all cases, the LOGSTC and Log Data blocks are optional with some exceptions. See the description of the SPCHDR parameter `flogoff` for more details.

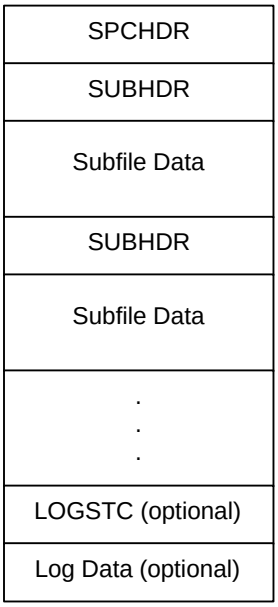



***Single File, Evenly Spaced X Values***

`Ftflgs` = Null

This is the most common form of SPC files. It is typically used to store data for a single spectrum or chromatogram where the X axis data point spacing is evenly throughout the data array. Galactic software calculates the X value of a given point in the Data array from the data point index from the beginning of the array using `fnpts`, `ffirst` and `flast` in SPCHDR:

X value = index · (`flast` – `ffirst`)/(`fnpts` – 1).



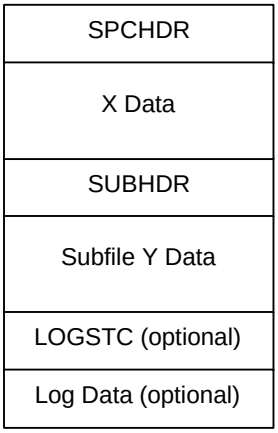

***Multifile, Evenly Spaced X Values***

`Ftflgs` = `TMULTI [ | TORDRD ]`

This form is generally used for multidimensional data sets. Examples include hyphenated chromatography-spectroscopy techniques (except GC-MS) and full spectrum kinetics experiments. It is also used for response map data from imaging experiments, as well as multidimensional spectroscopies such as 2D-IR, and 2D-NMR. As with the previous form, the X values are assumed to be evenly spaced.

The `fnsubs` parameter in SPCHDR determines the number of repeats of the SUBHDR and Subfile Data structures in the file.

The Z axis values can be stored in a variety of ways, depending on how they are spaced. This is determined by the settings of the `TORDRD` bits in the `ftflgs` parameter in SPCHDR. Files with `TORDRD` not set have evenly spaced Z values which are determined by `fzinc` in SPCHDR, or by difference from the `subtime` and `subnext` values in the first SUBHDR if `fzinc` is null. Files with non-evenly spaced Z values will have `TORDRD`, and the Z value for each subfile is stored in the leading SUBHDR for every subfile. (Although allowed, `TRANDM` should not be used.)

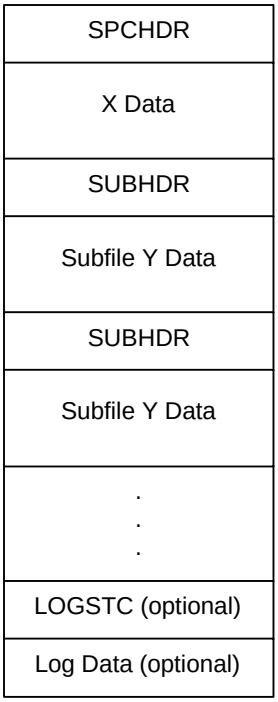



***Single File, Unevenly Spaced X values***

`Ftflgs` = TXVALS

This form is used for single record data sets where the spacing of the X values is not uniform. This is typically used for data such as Raman spectra from CCD/diode cameras, and TOF MS spectra. Some chromatography systems change the data collection rate during the run, which would also require this format.

In this format, the entire array of X values is stored in the file in IEEE 32-bit floating point format. There exactly the same number of data points (specified by `fnpts`) in both the X and Y value data blocks. Although the X values are fully specified by the X array, the `ffirst` and `flast` parameters must also be set to the first and last values in the X array, respectively. The X values must be in either increasing or decreasing order; not random order.



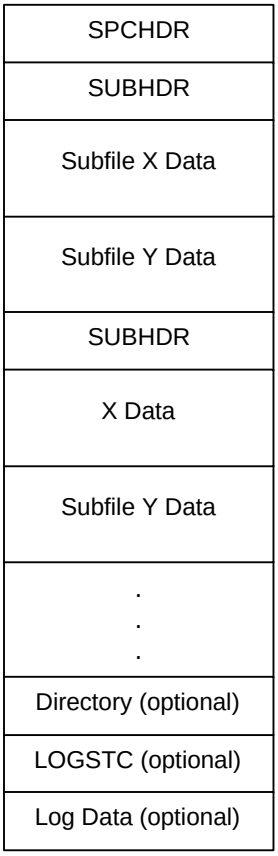

***Multifile, Unevenly Spaced X Values, Common X Array***

`Ftflgs` = `TMULTI | TXVALS [ | TORDRD ]`

This is effectively the mutifile form of the previous format. This form is generally used for multi-spectral kinetics experiments using Raman/CCD/Diode detectors. It can also be used for multi-chromatogram Diode array runs if the time point spacing is not constant (in this case the wavelength values for each chromatogram would be the Z values).

Note that in this form, the same set of X values is used for all the subfiles in the multifile. Thus every subfile must have the same number of data points as the X array which is specified by `fnpts`.

`TORDRD` can be applied to specify the spacing of the Z values as with multifiles with evenly spaced X values (above).




***Multifile, Unevenly Spaced X Values, Unique X Arrays***

`Ftflgs` = `TMULTI | TXYXYS | TXVALS [ | TORDRD ]`

This format is generally only used for one type of data: GC-MS spectra. This format allows every subfile to have a unique X array, thus a different number of data points. In this case, the `fnpts` parameter in SPCHDR should normally be set to null since the number of data points in each subfile is specified by the `subnpts` parameter in each SUBHDR structure. However, if the optional Directory of subfile pointers (SSFSTC) is included at the end of the file, then `fnpts` must be non-null and hold an offset pointer to the beginning of the directory.

When TXYXYS is included in `ftflgs`, all Galactic software will draw the data points a “sticks” originating from Y=0 (the traditional representation for MS spectra) rather than connecting the data points.

Note that a variant of this format can also be used for a single thresholded or centroided MS spectra (i.e. non-multifile). In this case, do not include `TMULTI` with the other settings for `ftflgs`. The Directory is not necessary for a single spectrum.

**`Fversn`*: SPC File Format Version***

This is the file format version; normally this should always be set to 0x4Bh for "new format" Galactic SPC files. Although SPC.H makes mention of a 0x4C setting for new format SPC files that allows for a different word order, most Galactic software products do not support it.

Files in the older Spectra/Lab Calc format use 0x4D for this parameter. However, while the offset and format of this value are the same between the old and new format SPC files, the rest of the headers themselves are substantially different. This document only describes the newer format information. For more documentation the old SPC file format refer to the `OSPCHDR` structure in SPC.H.

**`Fexper`*: Instrumental Experiment Technique***

This is the "experiment type" flag. In most Galactic SPC files in circulation today, the `fexper` parameter has been left set at null. However, Galactic plans to use this value as an internal flag to identify the instrumental technique that created the original data. This will allow future products be "smart" and tailor the interface to perform operations that are typically done on that type of data, while possibly disallowing other operations that don't make any sense. All future file conversion routines should take advantage of these flags when creating SPC files.

There are a series of predefined flags in SPC.H for setting this parameter documented in the following table. Note that these flags are mutually exclusive and cannot be OR'ed together.

| Defined Flag | Fexper Value | Instrumental Technique |
|---|---|---|
| SPCGEN | 0x00h | General (could be anything) |
| SPCGC | 0x01h | Gas Chromatogram |
| SPCCGM | 0x02h | General Chromatogram (same as SPCGEN and TCGRAM in `ftflgs`). |
| SPCHPLC | 0x03h | HPLC Chromatogram |
| SPCFTIR | 0x04h | FT-IR, FT-NIR, FT-Raman Spectrum (Can also be used for scanning IR.) |
| SPCNIR | 0x05h | NIR Spectrum (Usually multi-spectral data sets for calibration.) |
| SPCUV | 0x06h | UV-VIS Spectrum (Can be used for single scanning UV-VIS-NIR.) |
| | 0x07h | *Not Defined – Do not use.* |
| SPCXRY | 0x08h | X-ray Diffraction Spectrum |
| SPCMS | 0x09h | Mass Spectrum (Can be GC-MS, Continuum, Centroid or TOF.) |
| SPCNMR | 0x0Ah | NMR Spectrum |
| SPCRMN | 0x0Bh | Raman Spectrum (Usually Diode Array, CCD, etc. not for FT-Raman.) |
| SPCFLR | 0x0Ch | Fluorescence Spectrum |
| SPCATM | 0x0Dh | Atomic Spectrum |
| SPCDAD | 0x0Eh | Chromatography Diode Array Spectra |

**`Fexp`*: Data Block Scaling Exponent or Floating Point Data Flag***

This is the scaling exponent for the Y data values in the Data Block. In most SPC files created by Galactic software, the Y values in the Data Block(s) are represented as fixed-point signed fractions scaled by a single `fexp` exponent value. These are similar to integers except that the binary point is above the most significant bit rather than below the least significant bit. This format was designed to allow for storing data in higher precision than is offered by IEEE 32-bit floating point representation, but uses the same amount of disk space.

To convert the Galactic fixed format Y values to floating point representation:

FloatY = (2^Exponent)*IntegerY/(2^32)

or: FloatY = (2^Exponent)*IntegerY/(2^16) (only if TSPREC in ftflgs)

For example, a Y value of 0x40000000h represents 0.25 and a value of 0xC0000000h represents -0.25. If the fexp exponent is 2 then they represent 1 and -1 respectively.

Note that if the file is a multifile (see ftflgs) then there are separate exponents for each subfile stored in subexp in the preceding SUBHDR. This same conversion applies to the Y data for that particular Data Block immediately following the SUBHDR.

SPC files can also be formatted to store the Y data values as IEEE 32-bit floats. In this case the fexp value (or subexp values in the SUBHDR structures for multifiles) must be set to 0x80h. Currently, all Galactic-developed software products can read any and all SPC file formats, but new converters save SPC files using the floating-point format. However older Galactic converters always saved SPC files in the fixed integer format.

When developing new data file conversion routines, the Y data should always be stored in 32-bit IEEE floating point representation. The only exception would be if the original format of the data required higher precision than is offered by this format (i.e. if the original data was 32-bit integer). As of December 1996 all new Galactic-written data file conversion routines will output SPC files using the IEEE 32-bit floating point format unless more precision is required due to the original format of the data.

**Fnpts*: Number of Data Points or* TXYXYS *Directory Offset***

This is the number of data points in the X and Y data array(s) in the file. If the TXYXYS bit is turned on in the ftflgs parameter, then fnpts will normally be null since the number of data points is different for every subfile and is specified by the subnpts parameter in the SUBHDR structures.

However, the SPC file format allows for a separate Directory of subfile pointers to be stored immediately following the last subfile in the multifile. A non-null value for fnpts flags the existence of the Directory, and the fnpts parameter must be the offset to the beginning of the Directory. The Directory is a series of fnsubs repeats of the SSFSTC structure which gives offset pointers to the beginning of the SUBHDR for each subfile. This allows software to quickly "look up" an individual subfile without having to scan through all the subfiles. It also allows software to modify a single subfile, change its size, and place it at the end of the subfile list, as long as the corresponding pointer in the directory is also changed. This also allows for the subfiles to be stored in the multifile in a random Z order, although this is generally not recommended.

**Ffirst*: X Value of First Data Point***

X value corresponding to the first data point in the Y data array. In the case of XY type files (ftflgs includes TXVALS) this should be the same as the first value in the X data array. Ignored for files where ftflgs includes TXYXYS.

**Flast*: X Value of Last Data Point***

X value corresponding to the last data point in the Y data array. In the case of XY type files (ftflgs includes TXVALS) this should be the same as the last value in the X data array. Ignored for files where ftflgs includes TXYXYS.

**Fnsub*: Number of Subfiles***

Number of repeats of SUBHDR and data blocks in the file. In cases where `ftflgs` does not include `TMULTI`, this will normally be 1 or null.

**`Fxtype`, `Fytype`, *and* `Fztype`: *Enumerated X,Y and Z Axis Unit Label Types***

Galactic has defined a series of enumerated axis unit "types" that are allowed for SPC files. This allows for software to know the type of data stored in the file, and whether certain operations are allowed (i.e. convert to Absorbance from Transmission units). The SPC.H file defines the enumerated X, Y and Z axis unit type numbers. For convenience, they are duplicated here, however the ONLY source of the currently defined unit strings is maintained in the SPC.H source code. The `fxtype` and `fztype` unit parameters use the same enumerated list, while the `fytype` has its own list.

Note that not all the unit type values listed are supported by all Galactic products. The lists have been extended as the new products and routines for new data types were developed. The values listed here are in decimal, NOT hexidecimal.

It is not necessary to use one of the enumerated type values listed. If the unit type string required is not in the lists below, then SPC file main header block provides for "custom text" unit labels to be stored in the header instead. See the `fcatxt` parameter for more information.

| Defined Flag | `Fxtype` or `Fztype` Value | X or Z Unit Label |
|---|---|---|
| XARB | 0 | Arbitrary |
| XWAVEN | 1 | Wavenumber (cm-1) |
| XUMETR | 2 | Micrometers |
| XNMETR | 3 | Nanometers |
| XSECS | 4 | Seconds |
| XMINUTS | 5 | Minutes |
| XHERTZ | 6 | Hertz |
| XKHERTZ | 7 | Kilohertz |
| XMHERTZ | 8 | Megahertz |
| XMUNITS | 9 | Mass (M/z) |
| XPPM | 10 | Parts per million |
| XDAYS | 11 | Days |
| XYEARS | 12 | Years |
| XRAMANS | 13 | Raman Shift (cm-1) |
| XEV | 14 | Electron Volts (eV) |
| ZTEXTL | 15 | X,Y,Z text labels in `fcatxt` (old 4Dh version only) |
| XDIODE | 16 | Diode Number |
| XCHANL | 17 | Channel |
| XDEGRS | 18 | Degrees |
| XDEGRF | 19 | Temperature (F) |
| XDEGRC | 20 | Temperature (C) |
| XDEGRK | 21 | Temperature (K) |
| XPOINT | 22 | Data Points |
| XMSEC | 23 | Milliseconds (mSec) |
| XUSEC | 24 | Microseconds (uSec) |
| XNSEC | 25 | Nanoseconds (nSec) |
| XGHERTZ | 26 | Gigahertz (GHz) |
| XCM | 27 | Centimeters (cm) |
| XMETERS | 28 | Meters (m) |
| XMMETR | 29 | Millimeters (mm) |
| XHOURS | 30 | Hours |
| XDBLIGM | 255 | Double interferogram (no display labels) |

| Defined Flag | Fytype Value | Y Unit Label |
|---|---|---|
| YARB | 0 | Arbitrary Intensity |
| YIGRAM | 1 | Interferogram |
| YABSRB | 2 | Absorbance |
| YKMONK | 3 | Kubelka-Munk |
| YCOUNT | 4 | Counts |
| YVOLTS | 5 | Volts |
| YDEGRS | 6 | Degrees |
| YAMPS | 7 | Milliamps |
| YMETERS | 8 | Millimeters |
| YMVOLTS | 9 | Millivolts |
| YLOGDR | 10 | Log (1/R) |
| YPERCNT | 11 | Percent |
| YINTENS | 12 | Intensity |
| YRELINT | 13 | Relative Intensity |
| YENERGY | 14 | Energy |
| | 15 | ****** NOT USED ******* |
| YDECBL | 16 | Decibel |
| | 17 – 18 | ****** NOT USED ******* |
| YDEGRF | 19 | Temperature (F) |
| YDEGRC | 20 | Temperature (C) |
| YDEGRK | 21 | Temperature (K) |
| YINDRF | 22 | Index of Refraction [N] |
| YEXTCF | 23 | Extinction Coeff. [K] |
| YREAL | 24 | Real |
| YIMAG | 25 | Imaginary |
| YCMPLX | 26 | Complex |
| YTRANS | 128 | Transmission (ALL TYPES >=128 ARE ASSUMED TO HAVE INVERTED PEAKS!) |
| YREFLEC | 129 | Reflectance |
| YVALLEY | 130 | Arbitrary or Single Beam with Valley Peaks |
| YEMISN | 131 | Emission |

**`Fpost`*: DDE Posting Disposition Code***

Posting disposition code for SPC data sent to GRAMS/386 or GRAMS/32 via DDE. See GRAMSDDE.H for more information. Do not use when creating data file conversion routines; should normally be set to null.

**`Fdate`*: Data File Collection Date/Time***

This is the date and time the data in the file was collected/stored. This is not the same as the date and time stamp put on the SPC file by the operating system when it is stored. The information is encoded as unsigned integers into a 32-bit value for compactness as follows (***most significant bit is on the left***):

| Year (12 bits) | Month (4 bits) | Day (5 bits) | Hour (5 bits) | Minute (6bits) |
|---|---|---|---|---|

**`Fres`*: Resolution Description Text***

This is a short string (9 characters, but must be null terminated) describing the resolution of the data in the file. This is not necessarily the same as the data point spacing. For example, in and FT-IR spectrum, a commonly used resolution for data collection is 4 $cm^{-1}$, however the data point spacing for these spectra is ~1.93 $cm^{-1}$. In this case the text might say "4 cm-1". This string is not required, but if it is not used, it must be set to null.

**`Fsource`*: Source Instrument Description Text***

This is a short string (9 characters, but must be null terminated) identifying the source instrument name or model. This string is not required, but if it is not used, it must be set to null.

**`Fpeakpt`*: Interferogram Peak Point Number***

If the data in the file is an interferogram, (normally flagged by `fytype`=2), then this is the data point index in the Y array of the ZPD point, or point of maximum absolute response. Note that the data point index starts at zero (0) at the first value in the Y data array. If this is not known, or the data is not an interferogram set this value to null.

**`Fspare[8]`*: Array Basic Value Storage***

These are 8 IEEE 32-bit floating point values used by Galactic data processing routines (primarily written into the SPC file by an Array Basic application) to store private values inside an SPC file. These were widely used in products prior to GRAMS/386 V3.0, however have been supplanted by the use of textual parameters in the Log Text block. In general, it is a good idea to leave these values set to null.

**`Fcmnt`*: Memo/Comment/Description Text***

This is a 130 character text string memo or comment that describes the data in the file. This string must be null terminated.

**`Fcatxt`*: Custom Axis Unit Label Text***

Note that when custom unit labels are used, `ftflgs` must include TALABS. When using "custom unit" labels, some data processing tasks in Galactic software require data having specific enumerated unit types in order to work correctly. In addition that conversions from these SPC files out to other foreign formats may not be reliable.

**`Flogoff`*: File Offset to LOGSTC Structure***

This is an offset pointer to the beginning of the file Log Data. Normally this points to the beginning of the LOGSTC structure at the end of the file. If there is no Log Data in the SPC file, then `flogoff` must be set to null.

**`Fmods`*: File Modification Flags***

These are a series of bit-encoded flags that identify certain classes of operations that have been applied to the data since it was originally collected. The bit codes correspond to letters that are representative of the operation performed. These were widely used in GRAMS products prior to GRAMS/386 V3.0, however have been supplanted by the use of textual parameters in the Log Text block. Refer to SPC.H for more detailed information.

**`Fprocs`*: DDE Processing Code***

Post processing code for SPC data sent to GRAMS/386 or GRAMS/32 via DDE. See GRAMSDDE.H for more information. Do not use when creating data file conversion routines; should normally be set to null.

**`Flevel`*: Chromatogram Calibration Level***

For Galactic internal use only; do not use. Must be set to null.

**`Fsampin`*: Chromatogram Sample Injection Number***

For Galactic internal use only; do not use. Must be set to null.

**`Ffactor`*: Chromatogram Concentration Multiplier***

For Galactic internal use only; do not use. Must be set to null.

**`Fmethod`*: Chromatogram Method/Program/Data Filenames List***

For Galactic internal use only; do not use. Must be set to null.

**`Fzinc`*: Multifile Z Value Subfile Increment***

For multifiles with evenly spaced Z axes (`ftflgs` includes `TMULTI`, but not `TORDRD`), this is the spacing value between the individual subfiles. The initial value is taken from the `subfirst` value in the first subfile's SUBHDR, and then the Z value for subsequent files is given by:

Z value(n) = `subfirst`(1) + (`fzinc` ∗ n)

where "n" is the subfile number. However, the Z value spacing can also be calculated from the first subfile in a different way. If `fzinc` is null, then the subfile spacing is calculated by:

`fzinc` = `subnext`(1) – `subfirst`(1)

Note that this only applies to multifiles that use even Z spacing. All other multifiles must have explicit values set in `subfirst` in every single SUBHDR. In general, it is a good idea to put values in for `subfirst` in every SUBHDR, even if the file will have even Z spacing. This allows future products to randomly seek through the file and retrieve the complete data for a single subfile without having to rely on information in SPCHDR other than `fnpts`, `fnsubs` and `ftflgs`.

**Fwplanes*: Multifile 4D Data Number W Planes***

The fwplanes allows a using a multifile to store 4D data from experiments like spectral surface mapping. This is accomplished by identifying a series of subfiles to be interpreted as a volume of data with ordinate Y values along three dimensions (X,Z,W).

When fwplanes is non-null, then groups of subfiles are interpreted as separate planes along a W axis. The fwplanes value gives the total number of planes (groups of subfiles) in the SPC file and thus it must divide evenly into the total number of subfiles ( fnsub).

**Fwinc*: Multifile 4D W Plane Increment Value***

For 4D files (fwplanes is non-null) fwinc is the constant W spacing value between the planes of data. If fwinc is non-zero, then the W axis values are evenly spaced beginning with the subwlevel values in the SUBHDR structure for the first subfile and incremented by fwinc after each group of fwplanes subfiles. Thus the W value for subsequent planes is given by:

W value(n) = suwlevel(1) + (fzinc ∗ n)

where “n” is the W plane number.

If fwinc is null, then the planes are allowed to have non-evenly-spaced W axis values as given by the subwlevel in the SUBHDR structure for the first subfile in each plane's group. However, the W axis values must be ordered so that the plane values always increase or decrease. Also all subfiles in the plane should have the same subwlevel value.

In general, equally-spaced W planes are recommended since some software may not handle fwinc=0.

**Fwtype*: W Axis Unit Label Type***

The fwtype gives the W axis unit type value. This uses the same enumerated list of enumerated axis unit types as fxtype and fztype. Note that unlike the other 3 axis unit labeling methods, there is no allocated storage for a W axis custom unit string in fcatxt. Thus the W axis can only be labeled using a standard Galactic unit type.

**Freserv*: Reserved Storage for Future Expansion***

Do not write any data in this area. In addition, make sure that this area of the SPCHDR structure is initialized to nulls to avoid problems with any future definitions of parameters in this area.

## *The SUBHDR Structure (Subfile Header)*

In the new Galactic SPC file format, every individual data block is preceded by an SUBHDR structure. This gives information on the data in the following data block.

**Subflgs*: Subfile Data Flags***

This value should always be set to null for files that are created by conversion routines. However if a subfile in an SPC file has been manipulated by Galactic software and saved back to disk, it is possible that some bit flags will be set in this value. See SPC.H for more information.

**Subexp*: Subfile Data Block Scaling Exponent or Floating Point Data Flag***

Similar to fexp in SPCHDR, this is the scaling exponent for the data in the following Y data block for a single subfile in a multifile. See the previous description of fexp for information on

the scaling calculations. If the value of `subexp` is 0x80h, then the Y data block for the subfile is stored as IEEE32-bit float.

Note that the data storage format of the Y data blocks should be homogeneous throughout all subfiles in the multifile. This means that the subfiles should either all be stored as floating point format (`subexp=0x80h` for all subfiles) or they should all be stored as block scaled integers. If the latter format is used, each subfile can have a different exponent, but the file cannot also have individual subfiles where the Y data stored in floating point format mixed in.

**`Subindx`*: Subfile Index***

Integer index of the subfile number in the whole multifile. Note that these should increase monotonically starting at zero (0) for the first subfile, going up to `fnsubs – 1` for the last subfile.

**`Subtime`*: Subfile Z Axis Value***

This is the Z value for the subfile. This is typically the time the trace data in the following data block was collected during the experiment (i.e. spectra in hyphenated chromatography experiments), but can actually be any value representing a 3rd dimension for the data collected.

This value can be null in the subfile `SUBHDR` structures if `ftflgs` includes only `TMULTI` (i.e. does not include `TORDRD`) and the `fzinc` value is non-null. However, remember the Z value of the first subfile must be set to the starting Z value for the series, otherwise zero is assumed.

If `ftflgs` does include `TORDRD`, then every subfile `SUBHDR` must have a value for `subtime`.

**`Subnext`*: Subfile Ending Z Axis Value***

This is the “end” Z value for the subfile. In some experiments there is a time span over which the data for the subfile was collected, but this may not necessarily be the same as the spacing between the SUBHDR `subtime` values. For example, FT-IR kinetics runs can have multiple subfiles each of which was calculated from a number of co-added scans. If the time to collect the co-added scans for a single subfile was only 10 seconds, but the subfile spacing was set to be 30 seconds during the run, then the `subnext` parameter should be used to indicate the time actually required to collect the subfile. Thus the `subtime` value is the time point in the experiment that the collection of the subfile data began, while the `subnext` is the time that the subfile data was completed, and the next subfile would have a `subtime` different from both of these values.

Unless this capability is required, the value of the `subnext` parameter should be set to null, or set to the same value as the `subtime`. However, if the multifile is set up to have an evenly spaced Z axis and the `fzinc` parameter is null, then the `subtime` and `subnext` values in the very first subfile must be set to different values to indicate the Z value spacing.

**`Subnois`*: Subfile Peak Picking Noise Level***

This is used by Galactic software to store the calculated noise level in the Y data block for Galactic’s peak picking algorithm. This value should always be set to null for files that are created by conversion routines. However if a subfile in an SPC file has been manipulated by Galactic software and saved back to disk, it is possible that a value will be stored for `subnois`.

**`Subnpts`*: Subfile Number of Data Points***

Indicates the number of data points in the following subfile X and Y data arrays for files with `ftflgs` set to include `TXYXYS`. For all other file types, it must be set to null.

**Subscan*: Subfile Number of Scans***

The number of co-added scans collected to produce the subfile data. Generally only set by Galactic-written data acquisition routines. In general, this value should be set to null for files that are created by conversion routines.

**Subwlevel*: Floating W axis value (if fwplanes non-zero)***

This is the subfile W axis values for 4D data sets. This value will only be used when the fwplanes value in the SPCHDR structure is non-null. Refer to fwplanes and fwinc in the SPCHDR documentation for more information.

**Subresv[4]*: Subfile Reserved Storage for Future Expansion***

Do not write any data in this area. In addition, make sure that this area of the SUBHDR structure is initialized to nulls to avoid problems with any future definitions of parameters in this area.

## *The SSFTC Structure (Subfile Offset Pointers Directory)*

This structure will generally only be found in files with the TXVALS and TXYXYS flags set in the ftflags value in the SPCHDR structure. This structure is effectively a lookup pointer to an individual subfile record in the multifile. In this type of multifile the subfile records are variable length, and therefore software cannot perform calculated jumps to specific subfiles. A directory of fnsubs repeats of these subfile pointer structures allows software to quickly jump to any subfile.

The existence of the directory is flagged by a non-null value for the fnpts parameter in the SPCHDR structure. In this case, the fnpts parameter is interpreted as an offset pointer in the file to the beginning of the directory structure.

**Ssfposn*: Byte Offset to beginning of subfile SUBHDR(n)***

Byte offset from the beginning of the file to the beginning of the SUBHDR structure for the subfile.

**Ssfsize*: Byte size of complete subfile record***

Byte size in the complete subfile record (SUBHDR + X data block + Y data block).

**Ssftime*: Z value of subfile record***

This is effectively a duplicate of the subfile SUBHDR subtime value.

## *The LOGSTC Structure (Log Data Header)*

The Galactic SPC file Log Data is a free-form dynamic data block at the end of the file. The existence of log data is flagged by a non-null value of flogoff in the SPCHDR structure which is the byte offset to the beginning of the LOGSTC structure.

The LOGSTC structure normally immediately follows the main binary data block (or subfile directory in the case of TXYXYS files), and immediately precedes the Log Text block.

**Logsizd*: Byte size of log disk block***

The logsizd describes the size (in bytes) of the complete log block (LOGSTC structure and Log data block).

**`Logsizm`*: Byte size of log memory block required***

The `logsizm` should be the `logsizd` size of the block rounded to the next larger multiple of 4096.

**`Logtxto`*: Byte offset to Log Text data***

The `logtxto` is the byte offset FROM THE BEGINNING OF `LOGSTC` to the beginning of the Log Text data block.

**`Logbins`*: Byte size of log binary block***

This is the size in bytes of the private binary data block which is assumed to immediately follow the `LOGSTC` structure, and immediately precede the Log Text data block. This binary block area can be used by other programs to store their own private data inside an SPC file. In general, Galactic software programs do not use the data in this area, however it is carried along intact every time the file is modified and then saved back to disk. The one notable exception is for NMR data files, which use this area to store the imaginary array from quadrature detection NMR experiments.

Note that this binary data block is read into memory when the file is loaded and thus can cause memory usage to go up if this private data block is very large.

**`Logsizd`*: Byte size of log disk block***

This is the size in bytes of a private binary block which is assumed to immediately follow the `logbins` data block. The purpose of this block is the same as for `logbins`, however this data is not loaded into memory along with the file when using Galactic software products. Thus when the file is modified and saved back to disk, this data is not carried along.

**`Logspar`*: Reserved storage for future expansion***

Do not write any data in this area. In addition, make sure that this area of the `LOGSTC` structure is initialized to nulls to avoid problems with any future definitions of parameters in this area.

## *Data File Extensions*

Galactic'c DOS based software packages supported only a few file extensions for data files. For Spectra Calc, only files with ".SPC" extensions were supported. For Lab Calc, data files with either ".SPC" or ".CGM" extensions were recognized. Again, as mentioned above, these files used a different file format with no Log block for storing extra data.

However, Galactic's GRAMS/386 and GRAMS/32 software packages allow for a variety of file extensions on the data files according to the following table. These were designed to allow users to more easily organize and segregate their data by analytic techniques when many files were stored in the same directory. Creating files in GRAMS/386 or GRAMS/32 using these extensions (either via the use of Array Basic programs, or the GRAMS Convert Application file converters), also sets the `fexper` parameter in `SPCHDR` to match the experiment extension (although newer designed file converters force the `fexper` value to an internally set value rather than letting the user choose the type).

In general these experiment-oriented file extensions are not used by many users. Thus the standard convention for file naming is to use ".SPC" file extensions for files that contain spectral data, and ".CGM" extensions for chromatographic data.

Some experiments create both sets of spectral data and chromatographic data (i.e. GC-IR, GC-MS, HPLS-DAS, etc.). The SPC file format does not provide a way to store both pieces of information in the same file. However, all Galactic software products will automatically "link" the

data together if the spectral data is stored in a Galactic SPC file with a ".SPC" file extension, and the chromatogram data is stored in the same directory, with the same base name, but using a ".CGM" file extension (i.e. DATA.SPC and DATA.CGM). Therefore, when presented with data file from hyphenated techniques, many Galactic-written file conversion routines will store the spectral data in a multi-record ".SPC" file, and the chromatogram(s) in a separate ".CGM" file.

| **Extension** | **Data File Type** |
|---|---|
| SPC | Spectrum (General case) |
| CGM | Chromatogram (General case) |
| GC | Gas Chromatogram |
| LC | Liquid Chromatogram |
| HC | HPLC Chromatogram |
| FIR | Infrared (FT-IR) Spectrum |
| IR | Near Infrared (NIR) Spectrum |
| VIS | Visible Spectrum |
| UV | Ultraviolet (Visible) Spectrum |
| XRY | X-ray Spectrum |
| MS | Mass Spectrum |
| NMR | Nuclear Magnetic Resonance (NMR) Spectrum |
| RMN | Raman Spectrum |
| EFL | Fluorescence Spectrum |
| AAS | Atomic Spectrum |
| DAS | Diode Array Spectrum |
| OTR | Other Data Type |

# Universal Log Text Parameters

## *Log Text Information Format*

All information in the Log Text block must be in ASCII characters. The parameter keys are stored in the Log Text block of the SPC file as a string characters, followed by an equals sign ("=", ASCII 61), and another string of characters. Each entry should terminate with a carriage return (CR, ASCII 13) and a line feed (LF, ASCII 10). The lines can be in one of the following formats:

```
KeyName = ########
```

or

```
KeyName = Text
```

where "########" represents a numerical value as ASCII text, and "Text" represents a text string of characters. Each particular parameter always uses either a numeric value or a quoted text value (although numbers may optionally be followed by text after a space). Most parameters use numbers. The following sections denote what parameters are followed by ASCII values, and which are followed by text strings. Some parameters that are followed by text strings require that the text match a certain string in a list of allowed strings. All routines that read information from the Log Text block must be written such that the key names are not context sensitive (i.e. "Scans = " can be interpreted as easily as "SCANS ="), but by convention the key names are capitalized when writing information into the Log Text.

Note that it is not required to use every parameter for a given file type. This is especially true if the parameter does not exist in the file being translated or is not used by the piece of instrumentation/software writing the SPC file. However, all attempts should be made to translate as many or all parameters as possible to maintain data portability between different file formats.

For example, a Galactic SPC file containing an FT-IR spectrum may contain the following information in the Log Text block:

```
SCANS = 10
SPEED = 15
LWN = 15799.7
APOD = Triangular
PHASE = Self
```

Note also that some numeric parameter values have inherent units. However, the units are not always expressed. In the Log Text, the units are optional but must be separated from the number by at least one space character (ASCII 32) if used:

```
BEGX = 4000 cm-1
```

## *General Data File Description Parameters*

Before considering storing information in the Log Text block, all reasonable attempts must be made to fill in the information in the main header block of the SPC file. The main header block stores most of the general information required to properly display data in Galactic software. This includes the X limits, number of points, number of sub-files, the axis types, resolution text, memo text, and instrument source text. Therefore, such items such must always be stored in the main header block in the areas provided (see SPC.H). However, these same items can optionally be duplicated in the Log Text block

The following table lists the Log Text keys for the common parameters that can possibly be found within every Galactic SPC file. Note that these are duplicates of parameters that are already part of the main SPC file header.

| Log Text Key | Type | Description |
|---|---|---|
| BEGX | Value | X position of first trace point |
| ENDX | Value | X position of last trace point |
| NPTS | Value | Number of trace points (0=XY pairs) |
| BEGZ | Value | Z position of first subfile |
| ENDZ | Value | Z position of last subfile |
| NSUBS | Value | Number of subfiles |
| XTYPE | Value | X axis type number (as defined in SPC.H) |
| YTYPE | Value | Y axis type number (as defined in SPC.H) |
| ZTYPE | Value | Z axis type number (as defined in SPC.H) |
| RES | Text string | Collection X axis resolution (not necessarily the same as the data point spacing!)<br>(Set to -1 if automatic or unknown) |
| NAME | Text string | Data file name (optionally including path and extension) |
| MEMO | Text string | Primary comment/description text (up to 129 bytes) |
| COMMENT | Text string | Secondary comment/description text (use MEMO first) |
| USER | Text string | User's or Analyst's name |
| MODEL | Text string | The unique instrument name without spaces. This is used in log text and with "IGET" to determine the type of instrument or data acquisition or the file converter name. By convention "MODEL=name" is placed at the beginning of log text for a file. |

***Galactic Enumerated Unit Labels***

The SPC.H file defines the enumerated X, Y and Z axis unit type numbers. For convenience, they are also duplicated below. Be aware that this list is NOT used in compiling code and the ONLY source of the currently defined unit strings is maintained in these source code header files. In addition, it is not necessary to use one of the enumerated type values listed. The SPC file main header block provides for “custom text” unit labels to be stored in the header instead. Note however, that when using “custom unit” labels, some data processing tasks in Galactic software require that the data use a certain enumerated unit type, and that conversions from these SPC files out to other foreign formats may not be reliable. Also note that not all unit type values are supported by all Galactic products as the lists have been extended as the new products and routines for new data types were developed.

Possible settings for XTYPE and ZTYPE and the corresponding unit label:

| Value | X or Z Unit Label |
|---|---|
| 0 | Arbitrary |
| 1 | Wavenumber (cm-1) |
| 2 | Micrometers |
| 3 | Nanometers |
| 4 | Seconds |
| 5 | Minutes |
| 6 | Hertz |
| 7 | Kilohertz |
| 8 | Megahertz |
| 9 | Mass (M/z) |
| 10 | Parts per million |
| 11 | Days |
| 12 | Years |
| 13 | Raman Shift (cm-1) |
| 14 | Electron Volts (eV) |
| 15 | X,Y,Z text labels in fcatxt (old 4Dh version only) |
| 16 | Diode Number |
| 17 | Channel |
| 18 | Degrees |
| 19 | Temperature (F) |
| 20 | Temperature (C) |
| 21 | Temperature (K) |
| 22 | Data Points |
| 23 | Milliseconds (mSec) |
| 24 | Microseconds (uSec) |
| 25 | Nanoseconds (nSec) |
| 26 | Gigahertz (GHz) |
| 27 | Centimeters (cm) |
| 28 | Meters (m) |
| 29 | Millimeters (mm) |
| 30 | Hours |
| 255 | Double interferogram (no display labels) |

Possible settings for YTYPE and the corresponding unit label:

| Value | Y Unit Label |
|---|---|
| -1 | Reference Arbitrary Energy |
| 0 | Arbitrary Intensity |
| 1 | Interferogram |
| 2 | Absorbance |
| 3 | Kubelka-Munk |
| 4 | Counts |
| 5 | Volts |
| 6 | Degrees |
| 7 | Milliamps |
| 8 | Millimeters |
| 9 | Millivolts |
| 10 | Log (1/R) |
| 11 | Percent |
| 12 | Intensity |
| 13 | Relative Intensity |
| 14 | Energy |
| 15 | ****** NOT USED ******* |
| 16 | Decibel |
| 17 – 18 | ****** NOT USED ******* |
| 19 | Temperature (F) |
| 20 | Temperature (C) |
| 21 | Temperature (K) |
| 22 | Index of Refraction [N] |
| 23 | Extinction Coeff. [K] |
| 24 | Real |
| 25 | Imaginary |
| 26 | Complex |
| 128 | Transmission (ALL TYPES >=128 ARE ASSUMED TO HAVE INVERTED PEAKS!) |
| 129 | Reflectance |
| 130 | Arbitrary or Single Beam with Valley Peaks |
| 131 | Emission |

### *FT-IR Data Files*

The following Log Text keys can be used in Galactic SPC files containing FT-IR spectra.

| Log Text Key | Type | Description |
|---|---|---|
| SCANS | Value | Number of co-added scans |
| SCANSBG | Value | Number of co-added scans in background file |
| NAMEBG | Text string | Background spectrum filename |
| GAIN | Value | Detector Gain Factor |
| SPEED | Text string | Scan Speed or Velocity text (may be text like "Slow") |
| VELOCITY | Value | Scan Speed or Velocity number (-1 = automatic or unknown) |
| LWN | Value | Laser Wavenumber (usually ~15799.7 cm-1) |
| RAMANFREQ | Value | Raman Laser Frequency in Wavenumbers |
| RAMANPWR | Value | Raman Laser Power |
| JSTOP | Value | J-Stop aperture width |
| BSTOP | Value | B-Stop aperture width |
| DET | Text string | Detector type name |
| SRC | Text string | Source type name |
| BSPLITTER | Text string | Beam Splitter name |
| PURGE | Value | Purge Delay before scanning |
| ZFF | Value | Zero Filling Factor (0=none, 1=2x, 2=4x, etc.) |
| IRMODE | Value | FTIR Laser Sampling Frequency and Raman Mode<br>2 = Mid-IR mode<br>1 = Near-IR mode<br>-2 = Raman-shift (negative => Raman) |
| APOD | Text item | Apodization function type name:<br>"None" (boxcar apodization)<br>"Triangular"<br>"Weak" Norton-Beer<br>"Medium" Norton-Beer<br>"Strong" Norton-Beer<br>"Happ-Genzel"<br>"Bessel"<br>"Cosine"<br>"Filler"<br>"Kaiser" Bessel<br>"Gaussian"<br>"Tradezoidal" |
| PHASE | Text item | Phase correction type name:<br>"Self"<br>"Magnitude"<br>"Background"<br>"Stored" Phase<br>"None" |
| PHASEPTS | Value | Number of interferogram points used for phase correction |

| Log Text Key | Type | Description |
|---|---|---|
| IGRAMSIDE | Text item | Interferogram mode name:<br>"Double" sided<br>"Single" sided<br>"Left" single sided<br>"Right" single sided |
| IGRAMDIR | Text item | Interferogram collection direction name:<br>"Bi" interferometer scans in both direction<br>"Uni" interferometer scans in one direction<br>"Forward" uni-directional<br>"Reverse" uni-directional |
| POLARIZER | Value | Polarizer Angle |
| LOWPASS | Value | Low Pass Filter Cutoff |
| HIGHPASS | Value | High Pass Filter Cutoff |
| FILTER | Text string | Optical Filter Name |
| SMOOTH | Value | Number of averaged readings per data point. |
| BEAM | Text item | Single beam mode name:<br>"Sample" (Does not replace existing background).<br>“Background" for future ratioing with sample. |
| AVGTIME | Value | Time for A/D averaging of each point reading (msec). If the A/D has a fixed time and point readings are averaged, then this time should be divided by the approximate A/D time to get the number of readings. |
| BDELAY | Value | Begin delay in seconds |
| SDELAY | Value | Scan delay in seconds |
| CHANNEL | Value | Detector or beam path channel used for acquisition. |

## *NIR (Quant) Data Files*

Many NIR instrument software packages use multi-record file formats where each record contains a spectrum and additional information and/or data related to the individual sample. Since the Galactic SPC multifile format does not allow for easy storage of this information within the subfile records, the extra information should be imported according to the following specifications.

In addition, these files can have “constituent” information in them which is used for quantitative calibration purposes. This information is not imported into the Galactic SPC file, but is stored in a separate ASCII file known as a Galactic Calibration File List (CFL). The format of the CFL files is described here as well as the Log Text keys.

The following Log Text keys can be used in Galactic SPC files containing NIR spectra.

| Log Text Key | Type | Description |
|---|---|---|
| NODE | Value | Identifies the instrument connection node number |
| PPROCxx | Text item | Text description of additional processing applied to entire data set. There can be multiple entries allowed and they are NOT required to be in numerical order, and are NOT required to have a complete sequence (i.e. file could have only PPROC2 and PPROC8). Some possible values are, (although others are allowed):<br>"Constant Multiplication"<br>"File Multiplication"<br>"Constant Division"<br>"File Division"<br>"Constant Addition"<br>"File Multiplication"<br>"Constant Subtraction"<br>"File Subtraction"<br>"Back Correlation"<br>"Normalize"<br>"Mean Center"<br>“Variance Scaled”<br>“Smooth”<br>"SG Smooth"<br>"First Derivative"<br>"Second Derivative"<br>"Log Y"<br>“MSC” (Multiplicative Scatter Correction)<br>“SG 1st Derivative”<br>“SG 2nd Derivative”<br>“SNV” (Standard Normal Variate)<br>“SNV + Detrend”<br>“Divide by Wavelength (xxx)” (xxx is wavelength value)<br>“FFT” |
| CHANNEL | Value | Detector or beam path channel used for acquisition. |

| Log Text Key | Type | Description |
|---|---|---|
| SENSORCNT | Value | If the "SENSORCNT=" value is non-zero, each "SUBFILEx=" line is followed by a "SENSORVALx=" line where "x" is the subfile number beginning with “1” for the first subfile. |
| SENSORNAMEx | Text string | Sensor names. If the sensor count is non-zero, then the "SENSORCNT=" label is followed by "SENSORNAME1=" “SENSORNAME2=", etc. up to the number of sensors identified by SENSORCNT. |
| SUBFILEx | Formatted text string | Extra text information for subfile number “x”. This is a set of comma delimited strings in the following format: MM/DD/YY, HH:MM:SS, “Memo text” Example: SUBFILE43= 04/03/1996, 22:31:45, "MeOH" |
| SENSORVALx | Comma delimited values | Sensor values for subfile number “x” as comma delimited floating point values. Must be SENSORCNT number of values. |

There can be (optionally) at least one subfile label section for each subfile in the file. If used, the subfile section starts with the label "SUBFILEx=" (where "x" is the subfile number beginning with “1”) followed by the subfile date and time. The date and time are fixed width fields of the form MM/DD/YYYY HH:MM:SS separated by commas. The time should be in 24 hour format. The last item on the "SUBFILEx=" label line is the subfile description in double quotes. If no description is available, "" is used.

If the "SENSORCNT=" value is non-zero, each "SUBFILEx=" line is followed by a "SENSORVALx=" line where "x" is the again subfile number. Following the label are the sensor values as comma delimited text of floating point values.

***Constituent Concentration Data - Calibration File List (CFL File)***

Many NIR instrument file formats store concentration values and constituent name information used to create quantitative calibrations from the spectral data. The Galactic SPC file has no room for this information as the number of constituents and thus the size of the extra data is completely dynamic. While it is possible that the Log Text key name specifications could be extended in the future to include this information as part of the Galactic SPC file, the current mechanism is to write the information into a Galactic ASCII text Calibration File List (CFL) file. There are currently two formats allowed for the CFL file, however all future converters should only import to the “New” format. When importing the concentration data, the base name of the CFL file should match the base name of the output Galactic SPC data file (i.e. importing the foreign file “SAMPLE.DAT” would generate a Galactic multifile of the spectral data called “SAMPLE.SPC”, and a Galactic CFL file called “SAMPLE.CFL” of the concentration data).

When exporting data to a foreign NIR data file format, the Galactic-written converters should check for the existence of a matching CFL file. If there is a CFL file that matches the base name of the Galactic SPC file being exported, the conversion routine MUST export both the spectral data and concentration data. Also note that export converters will need to support BOTH CFL file formats as there is no guarantee that the CFL file is always in the new format and the user is not expected to manually translate it.

The old CFL format is from previous versions of Galactic’s quantitative applications and is still supported for import. However, this format is no longer used and new foreign file converters and Array Basic applications are only expected to save concentration data into the new CFL format. However, file converters must support the both the old and new CFL file formats on export.

**Old CFL Format Example**

```
3   =NUMBER OF COMPONENTS
8   =NUMBER OF SAMPLES
Sample Calibration file for PLSplus tutorial - Synthetic test mixtures
A
B
C
1         2         3         4         5         6         7         8         9         10
GOOD #1
.1        .35       .55
GOOD #2
.15       .4        .45
GOOD #3
.2        .45       .35
GOOD #4
.25       .5        .25
GOOD #5
.3        .1        .6
GOOD #6
.35       .15       .5
GOOD #7
.4        .2        .4
GOOD #8
.45       .25       .3
```

In the old CFL file format it is very important that there are NO BLANK LINES, especially at the top of the file.

| | |
|---|---|
| Line 1 | Contains the number of components (M). A comment may follow, however there must be AT LEAST ONE SPACE between the comment and the number. |
| Line 2 | Contains the number of samples (N). A comment may follow, however there must be AT LEAST ONE SPACE between the comment and the number. |
| Line 3 | Contains a comment line. |
| Line 4 | Contains the name of the first component. Continue typing the name of each component on a separate line up to the number of components (M) listed in Line 1. Component names may be up to 15 characters long. |
| Line 4+M | The next line simply labels 10 columns for visual examination. |
| Line 5+M | The rest of the lines contain the sample file name (with NO FILE EXTENSION!), and on the following line(s), the corresponding concentration data. Follow the example as shown. The values may be delimited by a space, comma or tab. If you have more than 10 components in the unknown, the concentration data may continue on the next line but the name of the next file must start on the following line. There must be N (from Line 2) sample name, concentration line pairs in the file. |

The new Galactic CFL format was designed specifically for users that create calibration lists using text editor or spreadsheet programs. It is currently only supported by the PLSplus/IQ quantitative analysis product, however all future products will use it as well. It uses a comma delimited format to store the spectral file names, and concentration values (if applicable). Therefore, it is not necessary to have the number of components and number of samples in the file header.

**New CFL Format**

```
Sample Calibration file for PLSplus/IQ tutorial - Synthetic test mixtures
File/Comp,Constit A,Constit B,Constit C
GOOD.SPC #1,0.10,    0.35,    0.55
GOOD.SPC #2,0.15,    0.40,    0.45
GOOD.SPC #3,0.20,    0.45,    0.35
GOOD.SPC #4,0.25,    0.50,    0.25
GOOD.SPC #5,0.30,    0.10,    0.60
GOOD.SPC #6,0.35,    0.15,    0.50
GOOD.SPC #7,0.40,    0.20,    0.40
GOOD.SPC #8,0.45,    0.25,    0.30
```

In the new CFL file format, it is very import that there are NO BLANK LINES!. The number of samples (N) is determined by counting the number of non-blank lines after the component list line (Line 2). The count is stopped when the first blank line, or a line that has fewer concentration values than the first sample is encountered.

| | |
|---|---|
| Line 1 | File comment (MUST NOT BE BLANK!). |
| Line 2 | Must start with "File/Comp", then list constituent names using commas as separators. All characters between the commas are considered significant, thus ", Stuff ," will result in a constituent name " Stuff ". The number of constituents (M) is determined by the number of comma separated constituent names found on this line. If there are no names following the "File/Comp" leader, then the rest of the file is assumed to have NO CONCENTRATION DATA and there must be no concentration values following the comma after the file name. |
| Line 3 | File name of 1st sample spectrum, followed by the comma separated concentration values of the constituents listed in line 2. Files can optionally have the extension, but if it is not present, then ".SPC" is assumed. When reading concentration values any non-numeric characters (i.e. spaces) between the commas are ignored. |
| Line N+2 | File name and concentrations for last sample spectrum. |

## *UV/VIS (NIR) Data Files*

The following Log Text keys can be used in Galactic SPC files containing UV/VIS/NIR spectra. Note that these are different from the parameters used by Galactic NIR (Quant) Data Files. This set of parameters is primarily used by Galactic ABC data acquisition drivers to store relevant data collection parameters. However, some foreign file formats may included this information as well, so this list is provided to help achieve data completeness for those files that do.

| Log Text Key | Type | Description |
|---|---|---|
| NAMESTD | Text string | Standard reference correction filename |
| NAMEZR | Text string | Zero reference (dark reference) correction filename |
| NAMEBG | Text string | Background or baseline spectrum filename |
| GAIN | Value | Detector Gain Factor |
| SLIT1 | Value | Slit aperture width (-1 = automatic or unknown) |
| SLIT2 | Value | Slit aperture width (-1 = automatic or unknown) |
| DET | Text string | Detector type name |
| SRC | Text string | Source type name |
| BEAM | Text item | Beam mode name:<br>“Double”<br>“Single Front”<br>“Single Rear”<br>“Double Reverse”<br>“Dual Single” |
| BDELAY | Value | Begin Delay in seconds |
| SDELAY | Value | Scan delay in seconds |
| PMT | Value | Photo multiplier Tube voltage (-1 = automatic or unknown) |
| DETCHG | Value | Detector changeover (NIR to UV/VIS) wavelength in nm |
| DETCOR | Value | Detector correction<br>0 = off<br>1 = on |
| SRCCHG | Value | Source changeover wavelength in nm |
| GRTCHG | Value | Grating changeover wavelength in nm |
| INDEPENDENT | Value | Spectral range independent collection:<br>0 = off, use same instrument parameters for whole spectrum<br>1 = allow different parameters for UV/VIS and NIR range |
| SCANMODE | Text item | Scan type:<br>“Spectrum” (0)<br>“Time scan” (1)<br>“Multi-wl time” scan (2) |
| SIGNOISE | Value | Signal to noise processing<br>0 = no S/N processing<br>1 = use S/N processing |
| SNLEVEL | Value | Acceptable S/N ratio |
| SNTIMEOUT | Value | Time out (in sec) for S/N processing in scan |
| NIR_RES | Value | NIR data interval (INDEPENDENT = 1 only) |

| Log Text Key | Type | Description |
|---|---|---|
| NIR_AVERAGING | Value | NIR averaging time (INDEPENDENT = 1 only) |
| NIR_SBW | Value | NIR slit width (INDEPENDENT = 1 only) |
| NIR_ENERGY | Value | NIR energy level (INDEPENDENT = 1 only) |
| NIR_SLITHT | Value | NIR slit height (INDEPENDENT = 1 only):<br>0 = Full<br>1 = 1/3 height |
| CORRECTION | Value | Sample baseline correction:<br>0 = off<br>1 = on |
| CORR_TYPE | Text item | Type of correction name:<br>“Normal” (0)<br>“Zero line” (1)<br>“Zero SRA” (2)<br>“Zero*Std.Ref.” (3) (uses NAMESTD file) |
| SCANTYPE | Text item | Scan type name:<br>“Sample” (0)<br>“Baseline” (1)<br>“Zero line” (2) |

## *NMR Data Files*

The following information describes how the Galactic SPC format was extended to provide support for 1D NMR data. This makes use of the both the Log Text block (for storing acquisition parameters) and the Log Binary block (for storing the imaginary data in complex number data sets). Note that some Log text parameters are REQUIRED, and must be present in the Log Text block in order for Galactic software to properly process the data. Also, some parameters may only be present in processed NMR spectra, and not FID data.

NOTE: These parameters have been defined for 1D NMR data only. This specification will be extended if there is a decision made to accommodate 2D NMR data in the future.

NOTE: All the following parameters MUST appear as the first set of information in the Galactic file text header record. In addition, the last item in the NMR part of the Log Text parameters must be "NMREND=NMREND". This is used by the Galactic NMR processing routines to recognize if additional processing has been applied to the data file. It is also used to flag that there is no more NMR-related information in the Log Text after that point.

### *Complex (Imaginary) Data Storage - Binary Log Block*

NOTE: The following information requires an understanding of the Galactic SPC binary format. Refer to the SPC.H header file for a complete description of the Log data area.

Most NMR instruments generate complex data (data that has arrays of both “real” and “imaginary” numbers). In order to meet the dual requirements that the data must display the “real” part of the spectrum properly for plotting, general manipulation and publication, but retain the “imaginary’ part for future NMR-related manipulations (such as re-phasing), Galactic has designed a split file format.

In all cases, whether the data is an FID or Spectrum, only the “real” portion is stored in the main data block of the SPC file. The imaginary data is stored in an additional binary block as part of the Log Text block at the end of the file (see SPC.H for more information). Although the main data block in an SPC file can be stored as either 32 bit-block scaled integers or 32-bit IEEE floating point values, the “imaginary” binary data block can ONLY be stored as 32-bit IEEE floats. When loading an SPC file for display of the Y values, Galactic software only reads the data from the main data block, and ignores any information in the log text block.

When creating a Galactic NMR file for SPC data, the `flogoff` value in the SPCHDR structure MUST be set to the byte offset to the beginning of the LOGSTC structure. This normally immediately follows the main binary data block:

```
typedef struct          /* log block header format */
   {
   DWORD logsizd;       /* byte size of disk block */
   DWORD logsizm;       /* byte size of memory block */
   DWORD logtxto;       /* byte offset to text */
   DWORD logbins;       /* byte size of binary area (after logstc*/
   DWORD logdsks;       /* byte size of disk area (after logbins)*/
   char logspar[44];    /* reserved (must be zero) */
   } LOGSTC;
```

The `logsizd` describes the size (in bytes) of the complete log block (LOGSTC, binary block and text data). The `logsizm` should be the `logsizd` size of the block rounded to the next larger power of 4096. The `logtxto` is the byte offset FROM THE BEGINNING OF LOGSTC to the beginning of the log text data. In other words, the binary log data should immediately follow the LOGSTC structure in the file. The `logbins` specifies the size of the binary block (in bytes) that follows LOGSTC.

Note that the size of the main data block and the binary log block MUST be exactly the same. This is required since the Galactic NMR processing routines require that the “real” and “imaginary” array have exactly the same number of data points.

***Informational Parameters***

These parameters are used to include extra information about the sample and source of the data in the file. These are all optional parameters and are not required for proper processing.

| Log Text Key | Type | Description |
|---|---|---|
| INSTRUM | Text string | Text description of the instrument/software |
| SOLVENT | Text string | Text name of the solvent used (i.e. "CDCl3", "D20", "DMSO", "Acetone", "CD30D", "MeOH", etc.) |
| NUCLEUS | Text string | Nucleus name for tuning frequency "1H" for Proton "13C" for Carbon 13 "19F" for Fluorine 19 General format is "AtomicnumberElement" for all others. |

***Acquisition Parameters***

These parameters are generally related to the modes used to collect the data from the NMR instrument. Many of these are REQUIRED since Galactic uses them during data processing.

| Log Text Key | Type | Description |
|---|---|---|
| DETMODE | Text item | Detection Quadrature mode name (REQUIRED):<br>"SIM" for simultaneous (use Complex FFT)<br>"SEQ" for sequential (use Redfield FFT)<br>"REAL" for real-only (use Real FFT) |
| NUCFREQ | Value | Nucleus tuning frequency in MHz (REQUIRED)<br>(aka Carrier Freq., Observed Freq., Center Freq.) |
| SW_HZ | Value | Spectral band width measured in Hertz (REQUIRED if SW_ppm not used)<br>(aka Sweep width, Spectral width) |
| DWELL | Value | Dwell time in Seconds (Should equal 1/SW_Hz) |
| DELAY | Value | Acquisition delay in Seconds |
| ACQTIME | Value | Acquisition time in Seconds (REQUIRED)<br>(Should be equal to NPTS*DWELL, or NPTS/SW_HZ.) |
| SCANS | Value | Actual number of scans performed<br>(aka Count of actual responses recorded (CT)) |
| REQSCAN | Value | Number of scans requested during acquisition<br>(aka Number of transients (NT), Number of Acquisitions (NA)) |
| FLTFREQ | Value | Filter Frequency in Hertz |
| PULSWD | Value | Pulse Width in Microseconds |
| SW_PPM | Value | Spectral band width measured in PPM (REQUIRED if SW_HZ not used)<br>(aka Sweep width, Spectral width) |
| AB_APPS | Text boolean | File subsequently processed by AB program<br>"TRUE" or "FALSE" |
| FIDFILE | Text string | Text filename (no path!) of Galactic FID file (*.FIG) corresponding to spectrum This tag can only exist if the spectrum or FID was either processed with Galactic's NMR routines OR was read into the NMR routine. In general, only files processed using Galactic's NMR routines will have this field set. |
| FID | Text boolean | Current data type indicator (REQUIRED)<br>"TRUE" for raw FID data<br>"FALSE" for FFT'd data (REQUIRES Processing Parameters below) |
| SPC_REAL | Text boolean | REAL data only, no IMAG record. (REQUIRED)<br>"TRUE" or "FALSE" |

***Processing Parameters***

These parameters are generally added to the log text block by the Galactic NMR processing applications. In general, file converters will not be able to get this information from the foreign file format or the values will be different than what is expected by the Galactic processing routines.

| Log Text Key | Type | Description |
|---|---|---|
| DCOFF | Value | Direct Current Offset correction. This is the data point number threshold to use when calculating the DC offset. This value indicates the number of data points, starting from the last value, and ending at the first data point, that is to be used in the DC offset calculation. |
| ZFF | Text item | Zero Filling Factor Text field that is used to indicate then $2^n$K value by which to increase the data size by.<br>“NONE”<br>“2xNPTS” (i.e. 2 times the number of data points in the file)<br>“4xNPTS”<br>“8xNPTS”<br>“16xNPTS”<br>“32xNPTS” |
| APOD | Text item | Apodization filter function type name. (Note this is the same key for Universal Parameters, but the list of allowed strings is different.)<br>"NONE"<br>"MATCH_EXP" (Matched Exponential)<br>"MATCH_GAUSS" (Matched Gaussian)<br>"BOXCAR" (Boxcar)<br>"COS_SQ" (Cosine)<br>"SINE_BELL" (Sine)<br>"TRAPEZOID" (Trapezoid)<br>"LORENTZ-GAUSS" (Lorentz-Gauss)<br>"OPT_RES" (Optimal Resolution) |
| APODP(0)<br>APODP(1) | Value | Some apodization filters require one or more fit parameter (REQUIRED if APOD present) |
| PHMOD | Text item | Phase mode used for spectrum (NOT SUPPORTED YET)<br>"NORMAL" (Normal spectrum phasing using PH0,PH1,PV0&PV1)<br>"MAGNITUDE" (Magnitude spectrum = sqrt(Real^2+Imag^2))<br>"POWER" (Power spectrum = (Real^2+Imag^2))<br>(Note if PHMOD<>"Normal" then user cannot re-phase.) |
| PH0 | Value | Zero order phase angle (degrees) |
| PH1 | Value | First order phase angle (degrees) |
| PV0 | Value | Zero order pivot (data point #, 0=first) |
| PV1 | Value | First order pivot (data point #, 0=first) |
| AUTOTYP | Text item | Text name of autophasing method used (NOT SUPPORTED) |

| Log Text Key | Type | Description |
|---|---|---|
| SPC_REV | Text boolean | Reverse spectrum flag<br>"TRUE" the file has been flipped<br>"FALSE" file read in does not change |
| DELTA_PPM | Value | Shift in PPM is the value that is subtracted from the full range beginning value (SW_PPM) and ending value (0) resulting in the new range in PPM. |

***Integration Parameters***

The following are used to display the integrals for the file when it is loaded into the Galactic NMR processing routines. These are entirely optional, but if one is present, they all must be.

| Log Text Key | Type | Description |
|---|---|---|
| INTBAS | Text boolean | All integral regions were baselined<br>"TRUE" or "FALSE" |
| INTOFF | Text boolean | Integral regions were offset<br>"TRUE" or "FALSE" |
| BASE | Text boolean | Interactive baseline indicator<br>"TRUE" or "FALSE" |
| NORM | Value | Region normalization value |
| MASK_ON | Text boolean | Use MASK regions to first excluding spectral information for calculating integrals and for display purposes<br>"TRUE" or "FALSE" |
| IREGNx | Space delimited values | Integral parameters for a single integral region. There can be any number of IREGNx log text entries (where "x" is the region number), and they must appear in sequence and in increasing order (i.e. IREGN1, IREGN2, IREGN3, etc.) The data is stored as a list of text values that are space (ASCII 32) delimited. The format is as follows:<br>Left_Edge Right_Edge Area Normalized_Area Multiplier Offset Tilt<br>The edge values must be specified in terms of PPM The Offset and Tilt values are the baseline correction parameters for the integral region. They are specified in terms of a fraction (0-100%) of 1% of the largest Y value in the masked real data array. |
| MREGNx | Space delimited values | Integral masking parameters for a single mask region. There can be any number of MREGNx log text entries (where "x" is the region number), and they must appear in sequence and in increasing order (i.e. MREGN1, MREGN2, MREGN3, etc.)The data is stored as a list of text values that are space (ASCII 32) delimited. The format is as follows:<br>Left_Edge Right_Edge<br>The edge values must be specified in terms of PPM. |

***Peak Picking Parameters***

These are used to display the peak positions for the file when it is loaded into the Galactic NMR processing routines. These are entirely optional, but if the peaktable is present, they all must be with the exception of the PEAK_MODE.

| Log Text Key | Type | Description |
|---|---|---|
| PEAK_TBL | Text string | Text filename of Galactic peak table for spectrum. Normally this will be <datafile>.PKN. |
| PEAK_MODE | Text item | Peak picking mode used:<br>"MASK_PKP" presets a masked region where peaks are not to be picked, then auto picks the rest of the spectrum using the threshold and option values.<br>If the user adds values to the table manually, both modes become obsolete. |
| THRESH | Value | Peak picking threshold value (corresponds to Galactic Threshold Y Level) |
| SENS | Value | Peak picking sensitivity value (corresponds to Galactic Sensitivity) |
| SMOOTH | Value | Peak picking smoothing value (corresponds to Galactic Smoothing)<br>Valid only if PICK_TYPE is set to EXTREMA (1) |
| PICK_TYPE | Value | Peak picking type:<br>0 = Maximum<br>1 = Extrema (also REQUIRES SMOOTH value) |
| WINDOW | Value | Size of peak window in data points when Adding a new peak. (Window used will actually be +/- WINDOW from the current position). |
| PK_MARK | Text item | Peak mark type:<br>"None"<br>"Arrow"<br>"V"<br>"Numbers" |
| PK_LABEL | Value | Peak labeling type number. Corresponds to the GRAMS internal Array Basic code number for peak labeling [i.e. 7 for Smart Labels]. |
| LBL_FORM | Value | Peak label digits. Integer number of digits after decimal point in peak labels. Allowed values are 1 to 4. |
| TR_OBJ | Value | Trace box object number. Allowed value is 1 only |
| PEAK_MASK | Text boolean | Peak marking masking used:<br>"TRUE" the mask is used to block out portions of the spectrum. This is always TRUE if PEAK_MODE = MASK_PKP.<br>"FALSE" for no masking |

| Log Text Key | Type | Description |
|---|---|---|
| PREGNx | Space delimited values | Peak picking masking parameters for a single region. There can be any number of PREGNx log text entries (where “x” is the region number), and they must appear in sequence and in increasing order (i.e. PREGN1, PREGN2, PREGN3, etc.) The data is stored as a list of text values that are space (ASCII 32) delimited. The format is as follows:<br>Left_Edge Right_Edge<br>The edge values must be specified in terms of PPM. |

***Other Considerations to Note for NMR Data***

Array Basic NMR processing programs must add the appropriate processing log text entries when manipulating data. This includes FFTing FIDs to spectra, phasing, and referencing. If the log text processing information is already present after import, but the data is FID type, then Galactic written processing retains the parameters imported from the instrument in memory and upon FFTing the FID will automatically apply those values. These values can be re-processed in the software at any point.

NMR FID data must always have the X axis in Seconds, with FFP=0 and FLP=ACQTIME [or NPTS/SW_Hz]. The maximum intensity of the data must be at the FFP end with the decay of the signal going towards the FLP end of the trace.

NMR spectra that have been FFT'd from FIDs must have X axis units in either Hertz (spanning -BANDWD/2 to +BANDWD/2) or PPM. Default is for the axis to be in PPM.

## *Mass Spectroscopy Data Files*

Currently there are no established standards for Log Text parameters for Mass Spectroscopy data files. However, any standard that is established will have to differentiate between the different types of spectrometers (i.e. quadrapole, sector, TOF, etc.) and deal with specific parameters related to each. It is likely that this category can cover all single scan as well as hyphenated (i.e. GC-MS) Mass Spectroscopy data.

## *Chromatogram Data Files*

Currently there are no established standards for Log Text parameters for Chromatography data files. However, any standard that is established will have to differentiate between the different types of chromatographs (i.e. HPLC, GC, SEC, GPC, etc.) and deal with specific parameters related to each. As chromatography data has a large variety of information required to maintain complete GLP compliance, it is likely that only a subset of all possible information available will be able to be carried along in a Galactic SPC file.

## *Diode Array Chromatography Detector Data Files*

Currently there are no established standards for Log Text parameters for Diode Array Chromatography Detector data files.

## *Diode Array / CCD Spectrometer Data Files*

Currently there are no established standards for Log Text parameters for Diode Array / CCD Spectrometer data files. Note this category is NOT the same as Diode Array Chromatography Detectors, but instead for CCD spectrometers used for applications like Raman spectroscopy, or spectroscopic imaging.

## *Xray Diffraction Spectrometer Data Files*

Currently there are no established standards for Log Text parameters for Xray Diffraction Spectrometer data files.

## *Fluorescence Spectrometer Data Files*

Currently there are no established standards for Log Text parameters for Fluorescence Spectrometer data files.

### *Data Files Created from AIA Data Interchange (ANDI/netCDF) Files*

There are many other data parameters which may be needed from time to time. When a setting is not available in the above list, the AIA name from the AIA FTIR or Chromatography Data Interchange specifications can be used. Alternatively, a new Galactic parameter name can be devised. However, to avoid duplications, it is important that all names used, either AIA or Galactic, be recorded in a single place (which is this document for now). A Galactic parameter should be used if possible. A new Galactic name should be created only if the equivalent AIA name has not already been used.

In order to assure proper transfer of key AIA sample information from file to file, any AIA names used as keys in the log text should appear between "`AIA_info_begin`" and "`AIA_info_end`" entries. The key names should be the AIA names as they appear in the template specification, complete with underbars as needed.

For example (taken from an AIA Chromatography V1.0 file):

```
AIA_info_begin
dataset_completeness = C1+C2
aia_template_revision = 1.0
netcdf_revision = 2.0
languages = English
dataset_origin = PE Nelson Turbochrom
operator_name = JDX
company_method_name = gasoline
sample_id = 1
sample_name = Gasoline
sample_injection_volume = 1.00
sample_amount = 1.00
detection_method_name = gasoline
detector_name =
detector_unit = uV
retention_unit = time in seconds
uniform_sampling_flag = Y
autosampler_position =
AIA_info_end
```

Currently, in Galactic's file converters for the AIA/ANDI/netCDF file formats, there is no attempt made to map the instrument or other parameters to the standard or terchnique-specific Log Text entries.

# Appendix A - SPC.H Header File

```
/**********************************************************************
*   FILENAME:  spc.h
*   AUTHOR:    Steven Simonoff 11-25-90 (from spc.inc)
*   MRU:       5-6-97 (New X,Y types, fexper types, float ssftime, 4D data)
*
*       Contains the Spectrum Information Header structure definitions.
*       Based on zspchdr.inc from Lab Calc (tm).
*       Must #include <windows.h> before this file.
*
*   Copyright (C) 1986-1997 by Galactic Industries Corp.
*   All Rights Reserved.
**********************************************************************

* The following defines a trace header and file format.
* All floating values in SPC files are in the IEEE standard formats.

* There are two basic formats, new and old.  FVERSN flags the format and
* also serves as a check byte.        The new format has header values in the file
* exactly as they appear below.  The old format has slightly different file
* header formatting which is translated as it is read into memory. New features
* like XY data and audit log information are not supported for the old format.

* The new format allows X,Y pairs data to be stored when the TXVALS flag is set.
* The main header is immediately followed by any array of fnpts 32-bit floating
* numbers giving the X values for all points in the file or subfiles.  Note
* that for multi files, there is normally a single X values array which applies
* to all subfiles.  The X values are followed by a subfile header and fixed
* point Y values array or, for multi files, by the subfiles which each consist
* of a subfile header followed by a fixed-point Y values array.  Note that
* the software may be somewhat slower when using X-values type files.

* Another X,Y mode allows for separate X arrays and differing numbers of
* points for each subfile.  This mode is normally used for Mass Spec Data.
* If the TXYXYS flag is set along with TXVALS, then each subfile has a
* separate X array which follows the subfile header and preceeds the Y array.
* An additional subnpts subfile header entry gives the number of X,Y values
* for the subfile (rather than the fnpts entry in the main header).  Under
* this mode, there may be a directory subfile pointers whose offset is
* stored in the fnpts main header entry.  This directory consists of an
* array of ssfstc structures, one for each of the subfiles.  Each ssfstc
* gives the byte file offset of the begining of the subfile (that is, of
* its subfile header) and also gives the Z value (subtime) for the subfile
* and is byte size.  This directory is normally saved at the end of the
* file after the last subfile.        If the fnpts entry is zero, then no directory
* is present and GRAMS/32 automatically creates one (by scanning through the
* subfiles) when the file is opened.  Otherwise, fnpts should be the byte
* offset into the file to the first ssfstc for the first subfile.  Note
* that when the directory is present, the subfiles may not be sequentially
* stored in the file.  This allows GRAMS/32 to add points to subfiles by
* moving them to the end of the file.

* Y values are represented as fixed-point signed fractions (which are similar
* to integers except that the binary point is above the most significant bit
* rather than below the least significant) scaled by a single exponent value.
* For example, 0x40000000 represents 0.25 and 0xC0000000 represents -0.25 and
* if the exponent is 2 then they represent 1 and -1 respectively.  Note that
* in the old 0x4D format, the two words in a 4-byte DP Y value are reversed.
* To convert the fixed Y values to floating point:
*       FloatY = (2^Exponent)*FractionY
* or:   FloatY = (2^Exponent)*IntegerY/(2^32)          -if 32-bit values
* or:   FloatY = (2^Exponent)*IntegerY/(2^16)          -if 16-bit values

* Optionally, the Y values on the disk may be 32-bit IEEE floating numbers.
* In this case the fexp value (or subexp value for multifile subfiles)
* must be set to 0x80 (-128 decimal).  Floating Y values are automatically
* converted to the fixed format when read into memory and are somewhat slower.
* GRAMS/32 never saves traces with floating Y values but can read them.
```

```
* Thus an SPC trace file normally has these components in the following order:
*       SPCHDR          Main header (512 bytes in new format, 224 or 256 in old)
*      [X Values]       Optional FNPTS 32-bit floating X values if TXVALS flag
*       SUBHDR          Subfile Header for 1st subfile (32 bytes)
*       Y Values        FNPTS 32 or 16 bit fixed Y fractions scaled by exponent
*      [SUBHDR ]        Optional Subfile Header for 2nd subfile if TMULTI flag
*      [Y Values]       Optional FNPTS Y values for 2nd subfile if TMULTI flag
*       ...             Additional subfiles if TMULTI flag (up to FNSUB total)
*      [Log Info]       Optional LOGSTC and log data if flogoff is non-zero

* However, files with the TXYXYS ftflgs flag set have these components:
*       SPCHDR          Main header (512 bytes in new format)
*       SUBHDR          Subfile Header for 1st subfile (32 bytes)
*       X Values        FNPTS 32-bit floating X values
*       Y Values        FNPTS 32 or 16 bit fixed Y fractions scaled by exponent
*      [SUBHDR ]        Subfile Header for 2nd subfile
*      [X Values]       FNPTS 32-bit floating X values for 2nd subfile
*      [Y Values]       FNPTS Y values for 2nd subfile
*       ...             Additional subfiles (up to FNSUB total)
*      [Directory]      Optional FNSUB SSFSTC entries pointed to by FNPTS
*      [Log Info]       Optional LOGSTC and log data if flogoff is non-zero

* Note that the fxtype, fytype, and fztype default axis labels can be
* overridden with null-terminated strings at the end of fcmnt.      If the
* TALABS bit is set in ftflgs (or Z=ZTEXTL in old format), then the labels
* come from the fcatxt offset of the header.  The X, Y, and Z labels
* must each be zero-byte terminated and must occure in the stated (X,Y,Z)
* order.  If a label is only the terminating zero byte then the fxtype,
* fytype, or fztype (or Arbitrary Z) type label is used instead.  The
* labels may not exceed 20 characters each and all three must fit in 30 bytes.

* The fpost, fprocs, flevel, fsampin, ffactor, and fmethod offsets specify
* the desired post collect processing for the data.  Zero values are used
* for unspecified values causing default settings to be used.  See GRAMSDDE.INC
* Normally fpeakpt is zero to allow the centerburst to be automatically located.

* If flogoff is non-zero, then it is the byte offset in the SPC file to a
* block of memory reserved for logging changes and comments.  The beginning
* of this block holds a logstc structure which gives the size of the
* block and the offset to the log text.  The log text must be at the block's
* end.  The log text consists of lines, each ending with a carriage return
* and line feed.  After the final line's CR and LF must come a zero character
* (which must be the first in the text).  Log text requires V1.10 or later.
* The log is normally after the last subfile (or after the TXYXYS directory).

* The fwplanes allows a series of subfiles to be interpreted as a volume of
* data with ordinate Y values along three dimensions (X,Z,W).  Volume data is
* also known as 4D data since plots can have X, Z, W, and Y axes.  When
* fwplanes is non-zero, then groups of subfiles are interpreted as planes
* along a W axis.  The fwplanes value gives the number of planes (groups of
* subfiles) and must divide evenly into the total number of subfiles (fnsub).
* If the fwinc is non-zero, then the W axis values are evenly spaced beginning
* with subwlevel for the first subfile and incremented by fwinc after each
* group of fwplanes subfiles.  If fwinc is zero, then the planes may have
* non-evenly-spaced W axis values as given by the subwlevel for the first
* subfile in the plane's group.  However, the W axis values must be ordered so
* that the plane values always increase or decrease.  Also all subfiles in the
* plane should have the same subwlevel.  Equally-spaced W planes are recommended
* and some software may not handle fwinc=0.  The fwtype gives the W axis type.
*****************************************************************************/
```

```
typedef struct
   {
   BYTE   ftflgs;      /* Flag bits defined below */
   BYTE   fversn;      /* 0x4B=> new LSB 1st, 0x4C=> new MSB 1st, 0x4D=> old format */
   BYTE   fexper;      /* Instrument technique code (see below) */
   char   fexp;        /* Fraction scaling exponent integer (80h=>float) */
   DWORD  fnpts;       /* Integer number of points (or TXYXYS directory position) */
   double ffirst;      /* Floating X coordinate of first point */
   double flast;       /* Floating X coordinate of last point */
   DWORD  fnsub;       /* Integer number of subfiles (1 if not TMULTI) */
   BYTE   fxtype;      /* Type of X axis units (see definitions below) */
   BYTE   fytype;      /* Type of Y axis units (see definitions below) */
   BYTE   fztype;      /* Type of Z axis units (see definitions below) */
   BYTE   fpost;       /* Posting disposition (see GRAMSDDE.H) */
   DWORD  fdate;       /* Date/Time LSB: min=6b,hour=5b,day=5b,month=4b,year=12b */
   char   fres[9];     /* Resolution description text (null terminated) */
   char   fsource[9];  /* Source instrument description text (null terminated) */
   WORD   fpeakpt;     /* Peak point number for interferograms (0=not known) */
   float  fspare[8];   /* Used for Array Basic storage */
   char   fcmnt[130];  /* Null terminated comment ASCII text string */
   char   fcatxt[30];  /* X,Y,Z axis label strings if ftflgs=TALABS */
   DWORD  flogoff;     /* File offset to log block or 0 (see above) */
   DWORD  fmods;       /* File Modification Flags (see below: 1=A,2=B,4=C,8=D..) */
   BYTE   fprocs;      /* Processing code (see GRAMSDDE.H) */
   BYTE   flevel;      /* Calibration level plus one (1 = not calibration data) */
   WORD   fsampin;     /* Sub-method sample injection number (1 = first or only ) */
   float  ffactor;     /* Floating data multiplier concentration factor (IEEE-32) */
   char   fmethod[48]; /* Method/program/data filename w/extensions comma list */
   float  fzinc;       /* Z subfile increment (0 = use 1st subnext-subfirst) */
   DWORD  fwplanes;    /* Number of planes for 4D with W dimension (0=normal) */
   float  fwinc;       /* W plane increment (only if fwplanes is not 0) */
   BYTE   fwtype;      /* Type of W axis units (see definitions below) */
   char   freserv[187]; /* Reserved (must be set to zero) */
   } SPCHDR;

#define SPCHSZ sizeof(SPCHDR) /* Size of spectrum header for disk file. */
```

```
/*************************************************************************
* In the old 0x4D format, fnpts is floating point rather than a DP integer,
* ffirst and flast are 32-bit floating point rather than 64-bit, and fnsub
* fmethod, and fextra do not exist.  (Note that in the new formats, the
* fcmnt text may extend into the fcatxt and fextra areas if the TALABS flag
* is not set.  However, any text beyond the first 130 bytes may be
* ignored in future versions if fextra is used for other purposes.)
* Also, in the old format, the date and time are stored differently.
* Note that the new format header has 512 bytes while old format headers
* have 256 bytes and in memory all headers use 288 bytes.  Also, the
* new header does not include the first subfile header but the old does.
* The following constants define the offsets in the old format header:

* Finally, the old format 32-bit Y values have the two words reversed from
* the Intel least-significant-word-first order.  Within each word, the
* least significant byte comes first, but the most significant word is first.
*************************************************************************/

typedef struct
   {
   BYTE  oftflgs;
   BYTE  oversn;      /* 0x4D rather than 0x4C or 0x4B */
   short oexp;        /* Word rather than byte */
   float onpts;       /* Floating number of points */
   float ofirst;      /* Floating X coordinate of first pnt (SP rather than DP) */
   float olast;       /* Floating X coordinate of last point (SP rather than DP) */
   BYTE  oxtype;      /* Type of X units */
   BYTE  oytype;      /* Type of Y units */
   WORD  oyear;       /* Year collected (0=no date/time) - MSB 4 bits are Z type */
   BYTE  omonth;      /* Month collected (1=Jan) */
   BYTE  oday;        /* Day of month (1=1st) */
   BYTE  ohour;       /* Hour of day (13=1PM) */
   BYTE  ominute;     /* Minute of hour */
   char  ores[8];     /* Resolution text (null terminated unless 8 bytes used) */
   WORD  opeakpt;
   WORD  onscans;
   float ospare[7];
   char  ocmnt[130];
   char  ocatxt[30];
   char  osubh1[32];  /* Header for first (or main) subfile included in main header */
   } OSPCHDR;

/*************************************************************************
* This structure defines the subfile headers that preceed each trace in a
* multi-type file.  Note that for evenly-spaced files, subtime and subnext are
* optional (and ignored) for all but the first subfile.  The (subnext-subtime)
* for the first subfile determines the Z spacing for all evenly-spaced subfiles.
* For ordered and random multi files, subnext is normally set to match subtime.
* However, for all types, the subindx must be correct for all subfiles.
* This header must must always be present even if there is only one subfile.
* However, if TMULTI is not set, then the subexp is ignored in favor of fexp.
* Normally, subflgs and subnois are set to zero and are used internally.
*************************************************************************/

#define SUBCHGD 1     /* Subflgs bit if subfile changed */
#define SUBNOPT 8     /* Subflgs bit if peak table file should not be used */
#define SUBMODF 128   /* Subflgs bit if subfile modified by arithmetic */

typedef struct
   {
   BYTE  subflgs;     /* Flags as defined above */
   char  subexp;      /* Exponent for sub-file's Y values (80h=>float) */
   WORD  subindx;     /* Integer index number of trace subfile (0=first) */
   float subtime;     /* Floating time for trace (Z axis coordinate) */
   float subnext;     /* Floating time for next trace (May be same as beg) */
   float subnois;     /* Floating peak pick noise level if high byte nonzero */
   DWORD subnpts;     /* Integer number of subfile points for TXYXYS type */
   DWORD subscan;     /* Integer number of co-added scans or 0 (for collect) */
   float subwlevel;   /* Floating W axis value (if fwplanes non-zero) */
   char  subresv[4];  /* Reserved area (must be set to zero) */
   } SUBHDR;

#define FSNOIS fsubh1+subnois+3 /* Byte which is non-zero if subnois valid */
```

```
/* This structure defines the entries in the XY subfile directory. */
/* Its size is guaranteed to be 12 bytes long. */

typedef struct
   {
   DWORD ssfposn;      /* disk file position of beginning of subfile (subhdr)*/
   DWORD ssfsize;      /* byte size of subfile (subhdr+X+Y) */
   float ssftime;      /* floating Z time of subfile (subtime) */
   } SSFSTC;

/* This structure defines the header at the beginning of a flogoff block. */
/* The logsizd should be large enough to hold the text and its ending zero. */
/* The logsizm is normally set to be a multiple of 4096 and must be */
/* greater than logsizd.  It is normally set to the next larger multiple. */
/* The logdsks section is a binary block which is not read into memory. */

typedef struct         /* log block header format */
   {
   DWORD logsizd;      /* byte size of disk block */
   DWORD logsizm;      /* byte size of memory block */
   DWORD logtxto;      /* byte offset to text */
   DWORD logbins;      /* byte size of binary area (immediately after logstc) */
   DWORD logdsks;      /* byte size of disk area (immediately after logbins) */
   char logspar[44];   /* reserved (must be zero) */
   } LOGSTC;

/* Possible settings for fxtype, fztype, fwtype. */
/* XEV and XDIODE - XMETERS are new and not supported by all software. */

#define XARB   0       /* Arbitrary */
#define XWAVEN 1       /* Wavenumber (cm-1) */
#define XUMETR 2       /* Micrometers (um) */
#define XNMETR 3       /* Nanometers (nm) */
#define XSECS  4       /* Seconds */
#define XMINUTS 5      /* Minutes */
#define XHERTZ 6       /* Hertz (Hz) */
#define XKHERTZ 7      /* Kilohertz (KHz) */
#define XMHERTZ 8      /* Megahertz (MHz) */
#define XMUNITS 9      /* Mass (M/z) */
#define XPPM   10      /* Parts per million (PPM) */
#define XDAYS  11      /* Days */
#define XYEARS 12      /* Years */
#define XRAMANS 13     /* Raman Shift (cm-1) */

#define XEV    14      /* eV */
#define ZTEXTL 15      /* XYZ text labels in fcatxt (old 0x4D version only) */
#define XDIODE 16      /* Diode Number */
#define XCHANL 17      /* Channel */
#define XDEGRS 18      /* Degrees */
#define XDEGRF 19      /* Temperature (F) */
#define XDEGRC 20      /* Temperature (C) */
#define XDEGRK 21      /* Temperature (K) */
#define XPOINT 22      /* Data Points */
#define XMSEC  23      /* Milliseconds (mSec) */
#define XUSEC  24      /* Microseconds (uSec) */
#define XNSEC  25      /* Nanoseconds (nSec) */
#define XGHERTZ 26     /* Gigahertz (GHz) */
#define XCM    27      /* Centimeters (cm) */
#define XMETERS 28     /* Meters (m) */
#define XMMETR 29      /* Millimeters (mm) */
#define XHOURS 30      /* Hours */

#define XDBLIGM 255    /* Double interferogram (no display labels) */
```

```
/* Possible settings for fytype.  (The first 127 have positive peaks.) */
/* YINTENS - YDEGRK and YEMISN are new and not supported by all software. */

#define YARB   0      /* Arbitrary Intensity */
#define YIGRAM 1      /* Interferogram */
#define YABSRB 2      /* Absorbance */
#define YKMONK 3      /* Kubelka-Monk */
#define YCOUNT 4      /* Counts */
#define YVOLTS 5      /* Volts */
#define YDEGRS 6      /* Degrees */
#define YAMPS  7      /* Milliamps */
#define YMETERS 8     /* Millimeters */
#define YMVOLTS 9     /* Millivolts */
#define YLOGDR 10     /* Log(1/R) */
#define YPERCNT 11    /* Percent */

#define YINTENS 12    /* Intensity */
#define YRELINT 13    /* Relative Intensity */
#define YENERGY 14    /* Energy */
#define YDECBL 16     /* Decibel */
#define YDEGRF 19     /* Temperature (F) */
#define YDEGRC 20     /* Temperature (C) */
#define YDEGRK 21     /* Temperature (K) */
#define YINDRF 22     /* Index of Refraction [N] */
#define YEXTCF 23     /* Extinction Coeff. [K] */
#define YREAL  24     /* Real */
#define YIMAG  25     /* Imaginary */
#define YCMPLX 26     /* Complex */

#define YTRANS 128    /* Transmission (ALL HIGHER MUST HAVE VALLEYS!) */
#define YREFLEC 129   /* Reflectance */
#define YVALLEY 130   /* Arbitrary or Single Beam with Valley Peaks */
#define YEMISN 131    /* Emission */

/* Possible bit FTFLGS flag byte settings. */
/* Note that TRANDM and TORDRD are mutually exclusive. */
/* Code depends on TXVALS being the sign bit.  TXYXYS must be 0 if TXVALS=0. */
/* In old software without the fexper code, TCGRAM specifies a chromatogram. */

#define TSPREC 1      /* Single precision (16 bit) Y data if set. */
#define TCGRAM 2      /* Enables fexper in older software (CGM if fexper=0) */
#define TMULTI 4      /* Multiple traces format (set if more than one subfile) */
#define TRANDM 8      /* If TMULTI and TRANDM=1 then arbitrary time (Z) values */
#define TORDRD 16     /* If TMULTI abd TORDRD=1 then ordered but uneven subtimes */
#define TALABS 32     /* Set if should use fcatxt axis labels, not fxtype etc.  */
#define TXYXYS 64     /* If TXVALS and multifile, then each subfile has own X's */
#define TXVALS 128    /* Floating X value array preceeds Y's  (New format only) */

/* FMODS spectral modifications flag setting conventions: */
/*  "A" (2^01) = Averaging (from multiple source traces) */
/*  "B" (2^02) = Baseline correction or offset functions */
/*  "C" (2^03) = Interferogram to spectrum Computation */
/*  "D" (2^04) = Derivative (or integrate) functions */
/*  "E" (2^06) = Resolution Enhancement functions (such as deconvolution) */
/*  "I" (2^09) = Interpolation functions */
/*  "N" (2^14) = Noise reduction smoothing */
/*  "O" (2^15) = Other functions (add, subtract, noise, etc.) */
/*  "S" (2^19) = Spectral Subtraction */
/*  "T" (2^20) = Truncation (only a portion of original X axis remains) */
/*  "W" (2^23) = When collected (date and time information) has been modified */
/*  "X" (2^24) = X units conversions or X shifting */
/*  "Y" (2^25) = Y units conversions (transmission->absorbance, etc.) */
/*  "Z" (2^26) = Zap functions (features removed or modified) */
```

```
/* Instrument Technique fexper settings */
/* In older software, the TCGRAM in ftflgs must be set if fexper is non-zero. */
/* A general chromatogram is specified by a zero fexper when TCGRAM is set. */

#define SPCGEN     0    /* General SPC (could be anything) */
#define SPCGC      1    /* Gas Chromatogram */
#define SPCCGM     2    /* General Chromatogram (same as SPCGEN with TCGRAM) */
#define SPCHPLC    3    /* HPLC Chromatogram */
#define SPCFTIR    4    /* FT-IR, FT-NIR, FT-Raman Spectrum or Igram (Can also be used for
                           scanning IR.) */
#define SPCNIR     5    /* NIR Spectrum (Usually multi-spectral data sets for
                           calibration.) */
#define SPCUV      7    /* UV-VIS Spectrum (Can be used for single scanning UV-VIS-NIR) */
#define SPCXRY     8    /* X-ray Diffraction Spectrum */
#define SPCMS      9    /* Mass Spectrum  (Can be single, GC-MS, Continuum, Centroid or
                           TOF.) */
#define SPCNMR     10   /* NMR Spectrum or FID */
#define SPCRMN     11   /* Raman Spectrum (Usually Diode Array, CCD, etc. use SPCFTIR for
                           FT-Raman.) */
#define SPCFLR     12   /* Fluorescence Spectrum */
#define SPCATM     13   /* Atomic Spectrum */
#define SPCDAD     14   /* Chromatography Diode Array Spectra */
```