A Brief Guide to SPC File Format
and
Using GSPCIO

Thermo Galactic

NOTE: This document describes the "new" SPC file format. The "old" format that was used in SpectraCalc and LabCalc does not support multifiles or the Audit Log and has a 256 byte Main header. However, GRAMS and the file I/O utilities must be capable of reading both formats.

A more detailed description of the SPC file format is available in the document "**Galactic Universal Data Format Specification**." This document provides a guide for programmers to implement data reading or data writing routines for files in Thermo Galactic SPC data file format. The "official" description of the SPC file format is contained in the file **SPC.H**. These documents are included in the Thermo Galactic SPC File Developer's Kit which can be downloaded from the Thermo Galactic website (www.thermogalactic.com/instruments/spcdata/spc_sdk.zip).

Table of Contents

SPC File Format

The SPC file is the standard Thermo Galactic file format. This is a binary file containing all of the information necessary for displaying and processing spectral data in GRAMS and related applications.

Single Files and Multifiles

The SPC file format allows the storage of either one or multiple spectra, plus associated data. An SPC file can be either a single file or a multifile. A **Single file** is a file containing just one spectrum. A **Multifile** is a file containing more than one spectrum. Each individual spectrum in a Multifile is referred to as a **subfile**.

A spectrum file can be viewed as an array where each Y point has a corresponding X value. It can either be a Y-only or an XY file.

- The **Y-only** file is the standard format. Y-only files have X values that are equally spaced. A Y-only file stores only the Y data, with the starting and ending X-axis limits stored in the header. (With this information, GRAMS and other Thermo Galactic products can calculate the X value of any data point in the file "on-the-fly.") The X-axis limits are called the Frequency First Point (**ffp**) and Frequency Last Point (**flp**).
- The **XY** file has X values that are not equally spaced. In this case, the X and Y values are both stored (two arrays of data points, one array for the Y values and a second array for the corresponding X values). The ffp/flp limits are ignored.

A multifile is simply a file with more than one spectrum (subfile) stored in it. There are three basic types of multifiles:

Y-Only	All of the subfiles have the same evenly spaced X values. Only the Y values are stored.
XYY	All of the subfiles have the same unevenly spaced X values. The X values are stored once and the Y values are stored for each subfile.
XYXY	Each subfile has its own separate X and Y arrays.

SPC File Sections

The SPC data format itself is broken down into three main categories (blocks) of information and it is important for developers to have a basic understanding of them in order to make the best use of the GSPCIO library.

In the order that they appear in the file format, these are the **Main Header Block**, the **Data Block**, and the **Log Block**.

Main Header Block

A 512 byte **Main Header** contains information defining the global file parameters and has the following fields:

Byte #	Data Type	Description
0	byte	File type flag. Each bit contains a different parameter: 0 = Y data is stored in 16-bit precision (instead of 32-bit) 2 = Use Experiment extension, not SPC 4 = Multifile 8 = If a Multifile, Z values are randomly ordered 16 = If a Multifile, Z values are ordered, but not even 32 = Use custom axis labels (obsolete) 64 = If an XY file and a Multifile, each subfile has its own X array 128 = XY file
1	byte	SPC file version: 4B = New format 4D = Old LabCalc format
2	byte	Experiment type code (see SPC.H for values)

3	char	Exponent for Y values (80h = as floating point):
		FloatY = (2^Exponent)*IntegerY/(2^32)32-bit
		FloatY = (2^Exponent)*IntegerY/(2^16)16-bit

4	long	Number of points in the file (if not XYXY)
8	double	First X coordinate
15	double	Last X coordinate
23	long	Number of subfiles
27	byte	X units type code (see SPC.H for values)
28	byte	Y units type code
29	byte	Z units type code
30	byte	Posting disposition (see GRAMSDDE.H)

31	long	Compressed Date:
		minutes= 6 bits
		hour= 5 bits
		day= 5 bits
		month= 4 bits
		year=12 bits

35	char * 9	Resolution description text
43	char * 9	Source instrument description text
52	word	Peak point number for interferograms
54	float * 8	Spare
86	char * 130	Memo
216	char * 30	X, Y, and Z custom axis strings (combined)

246	long	Byte offset to Log Block
250	long	File modification flag (see SPC.H for values)
254	byte	Processing code (see GRAMSDDE.H)
255	byte	Calibration level + 1
256	word	Sub-method sample injection number
258	float	Floating data multiplier concentration factor
262	char * 48	Method file
310	float	Z subfile increment for even Z Multifiles
314	long	Number of W planes
318	float	W plane increment
322	byte	W axis units code
324	char * 187	Reserved

Data Block

The **Data Block** is the body of the file containing the actual trace data.

1. For each subfile, a 32 byte subheader
2. All of the X coordinates of the subfiles, if appropriate
3. All of the Y values

XY Data Block

For XY type single files and XYY Multifiles, the X axis data are stored in a block following the Main Header block and before the Subheader Block as single precision floats.

Subheader Block

Each subfile has a 32 byte subheader block. This block exists even for single files.

Byte #	Data Type	Description
1	byte	Subfile flags. Each bit contains a different parameter: 1 = Subfiles changed 8 = Do not use peak table file 128 = Subfile modified by arithmetic
2	char	Exponent for subfile's Y values (80h = as floating point)
3	word	Subfile index number
5	float	Subfile's starting Z value
9	float	Subfile's ending Z value
13	float	Subfile's noise value to use peak picking
17	long	Number of points if XYXY Multifile
21	long	Number of co-added scans
25	float	W axis value
29	char * 4	Reserved

For **XY single files and XYY Multifiles**, the Y data are stored after each subfile header. They can be stored in one of two formats:

- ❑ as fixed point integers that are multiplied by the Exponent value
- ❑ as single precision floating point values (Note: GSPCIO always saves in this format)

For **XYXY Multifiles**, the data for each subfile is stored with all of the X coordinates (as singles) followed by all of the Y values in one of the above formats.

The SPC file format allows for the handling of 4-dimensional data. This is done using the **W** axis, which is tracked as follows:

1. In the Main header block are parameters specifying the number of W planes, the W increment, and the W axis units.
2. The subfile headers then have a parameter setting the W value for that subfile.
3. The total number of subfiles are then divided by the number of W planes and consecutive subfiles are "bundled" in each W plane. This means that the total number of subfiles must be an even multiple of the number of W planes.

For example, if there are 5 W planes and 20 subfiles, each block of 4 subfiles will belong to a W plane and will each have the same W value.

Directory Block

For XYXY Multifiles, this block contains the offset pointers for each subfile.

Log Block

The Log Block consists of three parts:

- ❑ the Log Header block
- ❑ the binary Log Data block
- ❑ the ASCII Log Text block

Log Header Block

The Log Header is a 64 byte block containing the following fields:

Byte #	Data Type	Description
0	Long	Size of the Log block (in bytes)
3	Long	Size of the memory block occupied by the Log block = the Log block size rounded-up to the nearest even multiple of 4096
7	Long	Offset to the Text section of the Log block
11	Long	Size of the Binary Log block
15	Long	Size of the disk area
19	Char * 44	Reserved

Log Data Block

The Binary area of the Log block is used to store data that is required for processing the file, but not displayed. For example, the imaginary data in an NMR file.

ASCII Log Text Block

The ASCII Log Text block serves a dual purpose:

1. To store acquisition parameters and other information about the file that is necessary for processing.

2. To record any changes made to the file by processing programs. This is referred to as the Audit Log.

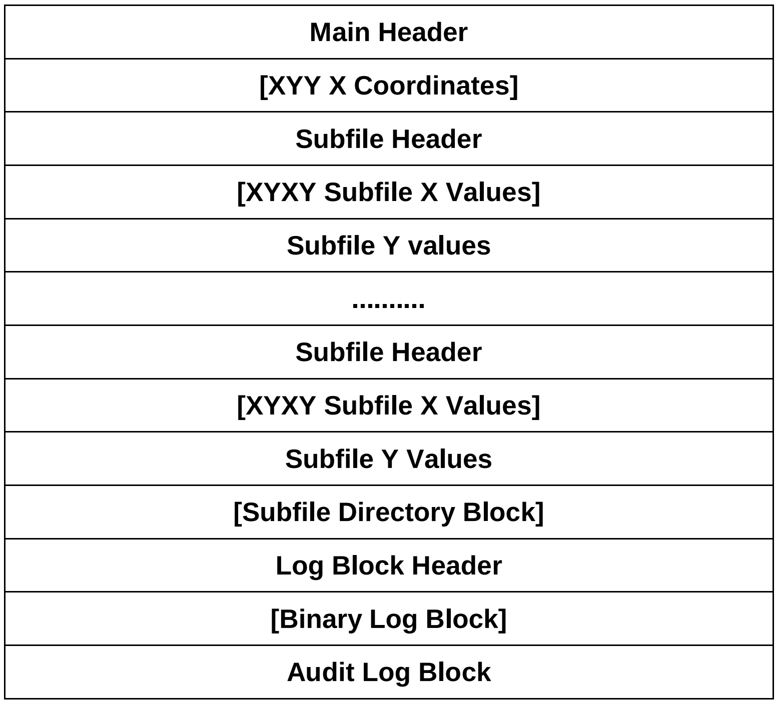

Figure 1. Diagram of SPC File Structure.

Using GSPCIO

This tutorial gives basic information and tips on using the utility in applications. All of the code examples are in Visual Basic. They assume that suitable variables have been declared and set to valid values. The methods, properties, and events of GSPCIO are described in detail in the Help file **GSPCIO.CHM**.

Note that index variables for the numbers of points and subfiles should be declared as **Long** to handle the largest allowed values.

Defining GSPCIO for Use in Programs

Before using GSPCIO, a variable of its type must be defined in the General Declarations section of the program – either in the Main module if it is global to the project or at the appropriate module level. This is done as follows:

```
Public SPCIO As New GSpcIOLib.GspcIO
```

In addition, the object must be added to the References list for the program by selecting:

Project / References / Galactic SpcIO Library

The variable SPCIO will be used in all of the following examples. Remember to free the variable when you are done with it or exit the program using:

```
Set SPCIO = Nothing
```

Reading File Properties

File Type

The basic types of SPC files are: Y-only, XY (for single files) / XYY (for Multifiles) and XYXY Multifiles. The type is determined as follows:

```
FileType = SPCIO.FileType
```

GSPCIO will return an integer value for one of the following defined types:

0 = spcFileTypeEven

1 = spcFileTypeXYY

2 = spcFileTypeXYXY

X and Y Values

The Y values are acquired using the call **SPCIO.YPoints**. Getting the X values is more complicated since the call used depends on whether the file is Y-only or XY. Therefore, the following code must be used:

```
pts = SPCIO.NumPoints
If SPCIO.FileType = spcFileTypeEven then
  FFP = SPCIO.FirstX
  FLP = SPCIO.LastX
  For i = 0 to pts - 1
     X(i) = FFP + i * (FLP – FFP) / (pts-1)
  next I
  else
     X = SPCIO.XPoints
     FFP = X(0)
     FLP = X(pts-1)
end if
```

Note that the X and Y arrays <u>must</u> be dimensioned from 0 to the number of points–1.

X Axis Limits

The starting and ending X-axis limits of a spectrum are read using the **SPCIO.FirstX** (= ffp) and the **SPCIO.LastX** (= flp) methods.

Note that for XY files, these values do not necessarily correspond to the actual X-axis limits. In this case, you can get the actual limits by reading the file's X array (using **SPCIO.XPoints**) and reading the values of the 0th and last elements of the X-axis array. For XYXY Multifiles, it is necessary to scan through all of the subfiles and get the outermost X limits (when doing so, be sure to handle the data running from both low to high and high to low).

Axis Labels

There are two basic types of axis labels in an SPC file: a "standard" label defined by a numerical code in the Main file header, and a "custom" label defined by a string in the Main header block. Because the two values can be set independently, you cannot count on the one not being used to be null. The following code gets the X-axis type code and label:

```
If SPCIO.XLabel <> SPCIO.XCustomLabel then
   XType = SPCIO.XType
   XCustomLabel =""
Else
   XType = 0
   XCustomLabel = SPCIO.XCustomLabel
End If
```

Subfiles

GSPCIO numbers the subfiles in a Multifile from 0 to the number of subfiles –1. Each subfile actually has two Z values: a starting "time" and an ending "time". This was to allow for a scan taking a finite period. The subfiles are accessed as follows:

```
NumSubs = SPCIO.NumSubfiles
For i = 0 to NumSubs-1
  SPCIO.SubfileIndex = i
  ZStart = SPCIO.ZStart
  ZEnd = SPCIO.ZEnd
Next i
```

Audit Log

The Audit Log section of the SPC file serves two functions:

1. To record instrument and data acquisition information for use in processing the data

2. To provide a record of changes made to the file by processing programs

Because GSPCIO does not have a call to return the number of lines in the Audit Log, you need to acquire the information in a loop until an error condition occurs. When reading the Log text, you need to specify the line number being read.

```
i = 0
Do While SPCIO.GetLogLine(i, Text(i)) <> -1
  LineLen = SPCIO.GetLogLine(i, Text(i))
  i = i + 1
Loop
```

The Log text can also be read-in all at once and then parsed by looking for the <CR><LF> line delimiters. Note that you have to check on both characters because it is possible for them to be in reversed order in the file.

```
NumChars = SPCIO.LogTextSize
LogString = SPCIO.LogText
i = 0
j = 0

Do While i <= NumChars
  i = i+1
  Char = Mid$(LogString,i,1)
  If (Char<>vbCr and Char<>vbLf) then
     Text(j) = Text(j) & Char
  Else
     j = j+1
  End If
Loop
```

To write to the Audit Log, you just keep adding lines. GSPCIO automatically updates the pointer information in the Log block header.

```
For i = 0 to NumLines
  SPCIO.AppendLogLine LogText(i)
Next i
```

Creating SPC Files

The code used depends on the type of the file being created. Here, the different file types are presented separately for clarity. In practice, a single code block using SELECT and IF statements to handle all of the types would usually be used. The code samples for Multifiles assume the use of a Recordset (or GdbGrid object) to store the source (single file) spectra. They could just as easily be read from disk files or an array of bytes (also known as a **blob**).

It should be noted that GSPCIO always writes the Y data in floating point format.

Y-only Single Files

For Single files, you can just create the file by adding points to the created spectrum. GSPCIO requires that the X array not be passed for Y-only files:

```
SPCIO.CreateSPC spcFileTypeEven
SPCIO.FirstX = FFP
SPCIO.LastX = FLP
SPCIO.AddPoints = Y
```

XY Single Files

For XY files, the X axis values must be explicitly passed. To facilitate future use of the file, it is recommended that the First and Last X values be set to the corresponding values in the X axis array.

```
SPCIO.CreateSPC spcFileTypeXYY
SPCIO.FirstX = X(0)
SPCIO.LastX = X(pts-1)
SPCIO.AddPoints Y,X
```

Y-Only Multifiles

To create Multifiles, you successively add each subfile. You must also set the Z order type and Z values. The Z axis type can be one of the following:

- **ZTypeEven**: The subfile Z values are evenly spaced. Only the first Z value and Z Increment are used.
- **ZTypeOrdered**: The subfile Z values are either all ascending or descending, but do not have to be even.
- **ZType Random**: Any Z values are allowed for any subfile.

Note that the subfile numbers are indexed starting at 0. Also, a null placeholder is passed in the X axis field of the GSPCIO call.

```
SPCIO.CreateSPC spcFileTypeEven
SPCIO.FirstX = FFP
SPCIO.LastX = FLP
SPCIO.ZIncrementTypeDefault = ZType
ReDim Y(0 to NumPoints-1)

For i = 0 to NumSubfiles-1
  RS.AbsolutePosition = i+1
  SPCIO.OpenBlob RS.Fields("Trace").Value
  Y = SPCIO.YPoints
  SPCIO.AddSubfile Y, vbNull, ZValues(i), ZValues(i)
Next i
```

XYY Multifiles

To generate an XYY Multifile, you must explicitly pass the (global) X axis array. In the sample below, the X axis array must be the same for each source file. It is re-read in each pass of the loop for convenience.

```
SPCIO.CreateSPC spcFileTypeXYY
SPCIO.ZIncrementTypeDefault = ZType
ReDim Y(0 to NumPoints-1)
ReDim X(0 to NumPoints-1)

For i = 0 to NumSubfiles-1
  RS.AbsolutePosition = i+1
  SPCIO.OpenBlob RS.Fields("Trace").Value
  Y = SPCIO.YPoints
  X = SPCIO.XPoints
  SPCIO.AddSubfile Y, X, ZValues(i), ZValues(i)
Next I

SPCIO.FirstX = X(0)
SPCIO.LastX = X(NumPoints-1)
```

XYXY Multifiles

For XYXY Multifiles, the X data is different for each subfile, but must always be in the same orientation. This means that the data arrays must be re-dimensioned for each source file. The outermost points must be determined when scanning through the loop. This is incorporated into the file generation loop here, but it could also be made a separate scan through the data.

```
SPCIO.CreateSPC spcFileTypeXYXY
SPCIO.ZIncrementTypeDefault = Ztype

For i = 0 to NumSubfiles-1
  RS.AbsolutePosition = i+1
  SPCIO.OpenBlob RS.Fields("Trace").Value
  NumPoints = SPCIO.NumPoints
  ReDim Y(0 to NumPoints-1)
  ReDim X(0 to NumPoints-1)
  Y = SPCIO.YPoints
  X = SPCIO.XPoints
  SPCIO.AddSubfile Y, X, ZValues(i), ZValues(i)

  If i = 0 Then                                 'Get outermost X values
     FFP = SPCIO.X(0)
     FLP = SPCIO.X(NumPoints-1)
  End If

  If X(0) < X(NumPoints-1) Then
     if X(0)<FFP then FFP = X(0)
     if X(NumPoints-1)>FLP then FLP = X(NumPoints-1)
  Else
     if X(0)>FFP then FFP = X(0)
     if X(NumPoints-1)<FLP then FLP = X(NumPoints-1)
  End If

Next I

SPCIO.FirstX = FFP
SPCIO.LastX = FLP
```

Disk Files

Reading from Disk

GSPCIO can also be used to read an SPC file directly from disk. In this case, it is necessary to validate that the file is in SPC format. An error code is returned if it is not.

```
On Error Resume Next
SPCIO.OpenFile FileName
ecode = Err.Number
On Error GoTo 0
Select Case ecode
  case 0
     MsgBox "This is a valid SPC file.  You may proceed."
  case –1
     MsgBox "Unable to open file.  File does not exist or" _
            & " you do not have rights to it."
  case –3
     MsgBox "This file is not in SPC format, but is a recognized" _
            & " spectrum file that can be converted."
End Select
```

Writing to Disk

Either a single (sub)file or an entire Multifile can be saved to disk:

```
SPCIO.SaveFile FileName, SubfileIndex
```

If the subfile index = -1, then the entire Multifile is saved.