ASD FieldSpec Data Format Definition

The first step to reading an ASD format data file is to read the file's header. This header is 484 bytes in size and has a structure that is defined in Appendix I. This data structure references several other structure definitions that are listed in Appendix III. If you are writing your program in C or C++ you can save yourself some time by getting the **SPECIO.H**, **TIME.H** and **GPS.H** header files from ASD.

Next, you'll need to determine the type and size of the data portion of the file. The header contains several variables that provide this information. The **data_format** variable, located at a byte offset of 199 from the beginning of the file, is a byte variable that contains a value defining the spectrum's data format. Use the values listed in Appendix II to determine the format. For example, if **data_format** has a value of '0' the data is in 4 byte floating point format. The **channels** variable, located at a byte offset of 204 from the beginning of the file, is an unsigned integer (2 bytes) that contains the number of data values in the spectrum. Once the format and size of the data block is known, it is a simple matter to read in an array (starting at a byte offset of 484) of size **channels** and type defined by **data_format**.

Now that the data is read, you'll still need a few more variables to make use of the data. The **ch1_wavel** and **wavel_step** variables are used to provide wavelength values for each of the data values. These are both floating point values and are located at byte offsets of 191 and 195 respectively. The **ch1_wavel** variable contains the wavelength value in nanometers corresponding to the first data value; the **wavel_step** variable contains the wavelength interval between each data value. The **data_type** variable, a byte variable located at a byte offset of 186, defines the data type. The data types corresponding to the values of **data_type** are listed in Appendix II. For example, if the **data_type** variable has a value of '1', the spectrum contained in the data block of the file is a reflectance spectrum. The **instrument** variable, a byte variable located at a byte offset of 431, defines the type of instrument used to collect the spectrum. The instruments corresponding to the values of **instrument** are listed in Appendix II. For example, if the **instrument** variable has a value of '4', the file was collected using a FieldSpec® FR. Other variables can be found in Appendix I.

Of course, if you want to keep things really simple, you can simply assume all of the above values and just read in the data. For example, if you know that you are going to read in only reflectance spectra collected by a FieldSpec® FR, you can simply skip over the first 484 bytes of the file and then read in 2151 floating point values. For other ASD instruments, the number of data points are as follows: FieldSpec® UV/VNIR — 512; FieldSpec® HH — 512; FieldSpec® UV/VNIR-CCD — 1024. For all ASD's current instruments (this doesn't include the PSII), reflectance, (ir)radiance, and raw (DN) spectra are stored in floating point format. This can be confirmed by subtracting 484 from the file size (in bytes) and then dividing by the number of data points — if the data format is floating point you'll get a value of 4; if integer a value of 2. The wavelength of the first data point and the step between data points can be found in the **ASD.INI** (DOS Version – RS^2) or **ASDCFG.ini** (Windows version - RS^3) file that came with your instrument.

Appendix I: Details of file header structure

file offset	size	variable type	variable name	description
	3	char	co[3];	// Three character company name
3	157	char	comments[157];	// comment field
160	18	struct tm	when;	// time when spectrum was saved
178	1	byte	program_version;	// ver. of the programcreatinf this file.
				// major ver in upper nibble, min in lower
179	1	byte	file_version;	// spectrum file format version
180	1	byte	itime;	// Not used after v2.00
181	1	byte	dc_corr;	// 1 if DC subtracted, 0 if not
182	4	time_t (==long)	dc_time;	// Time of last dc, seconds since 1/1/1970
186	**1**	**byte**	**data_type;**	**// see *_TYPE below**
187	4	time_t (==long)	ref_time;	// Time of last wr, seconds since 1/1/1970
191	**4**	**float**	**ch1_wavel;**	**// calibrated starting wavelength in nm**
195	**4**	**float**	**wavel_step;**	**// calibrated wavelength step in nm**
199	**1**	**byte**	**data_format;**	**// format of spectrum.**
200	1	byte	old_dc_count;	// Num of DC measurements in the avg
201	1	byte	old_ref_count;	// Num of WR in the average
202	1	byte	old_sample_count;	// Num of spec samples in the avg
203	1	byte	application;	// Which application created APP_DATA
204	**2**	**ushort**	**channels;**	**// Num of channels in the detector**
206	128	APP_DATA	app_data;	// Application-specific data
334	56	GPS_DATA	gps_data;	// GPS position, course, etc.
390	4	ulong	it;	// The actual integration time in ms
394	2	int	fo;	// The fo attachment's view in degrees
396	2	int	dcc;	// The dark current correction value
398	2	uint	calibration;	// calibration series
400	2	uint	instrument_num;	// instrument number
402	4	float	ymin;	// setting of the y axis' min value
406	4	float	ymax;	// setting of the y axis' max value
410	4	float	xmin;	// setting of the x axis' min value
414	4	float	xmax;	// setting of the x axis' max value
418	2	uint	ip_numbits;	// instrument's dynamic range
420	1	byte	xmode;	// x axis mode. See *_XMODE
421	4	byte	flags[4];	// Flags (0 = AVGFIX'ed)
425	2	unsigned	dc_count;	// Num of DC measurements in the avg
427	2	unsigned	ref_count;	// Num of WR in the average
429	2	unsigned	sample_count;	// Num of spec samples in the avg
431	**1**	**byte**	**instrument;**	**// Instrument type. See defs below**
432	4	ulong	bulb;	// The id number of the cal bulb
436	2	uint	swir1_gain;	// gain setting for swir 1
438	2	uint	swir2_gain;	// gain setting for swir 2
440	2	uint	swir1_offset;	// offset setting for swir 1
442	2	uint	swir2_offset;	// offset setting for swir 2
444	4	float	splice1_wavelength;	// wavelength of VNIR and SWIR1 splice
448	4	float	splice2_wavelength;	// wavelength of SWIR1 and SWIR2 splice
452	12	char	when_in_ms[12];	// fill to 484 bytes
464	20	byte	spare[20];	// fill to 484 bytes
	484	total size of header in bytes		

Appendix II: Values need to interpret some of the variable listed in Appendix I

Spectrum data type (variable data_type at byte offset 186):

```
#define RAW_TYPE         (byte)0
#define REF_TYPE         (byte)1
#define RAD_TYPE         (byte)2
#define NOUNITS_TYPE     (byte)3
#define IRRAD_TYPE       (byte)4
#define QI_TYPE          (byte)5
#define TRANS_TYPE       (byte)6
#define UNKNOWN_TYPE     (byte)7
#define ABS_TYPE         (byte)8
```

Spectrum data format (variable data_format at byte offset 199):

```
#define FLOAT_FORMAT     (byte)0
#define INTEGER_FORMAT   (byte)1
#define DOUBLE_FORMAT    (byte)2
#define UNKNOWN_FORMAT   (byte)3
```

Instrument type that created spectrum (variable instrument at byte offset 431):

```
#define UNKNOWN_INSTRUMENT           (byte)0
#define PSII_INSTRUMENT              (byte)1
#define LSVNIR_INSTRUMENT            (byte)2
#define FSVNIR_INSTRUMENT            (byte)3
#define FSFR_INSTRUMENT              (byte)4
#define FSNIR_INSTRUMENT             (byte)5
#define CHEM_INSTRUMENT              (byte)6
#define FSFR_UNATTENDED_INSTRUMENT   (byte)7
```

Appendix III: Details of data structures referenced in Appendix I

```
struct tm
{
  int   tm_sec;         // seconds [0,61]
  int   tm_min;         // minutes [0,59]
  int   tm_hour;        // hour [0,23]
  int   tm_mday;        // day of month [1,31]
  int   tm_mon;         // month of year [0,11]
  int   tm_year;        // years since 1900
  int   tm_wday;        // day of week [0,6] (Sunday = 0)
  int   tm_yday;        // day of year [0,365]
  int   tm_isdst;       // daylight savings flag
};
```

```
typedef long time_t;
```

APP_DATA
This is a 128 byte field that is used for storing results produced by various real-time processing routines.

```
struct GPS_DATA
{
      double      true_heading;
      double      speed;
      double      latitude, longitude;
      double      altitude;
      struct
      {
            unsigned havecomm : 1;
            unsigned terrain : 2;
            unsigned datum   : 6;
            unsigned dist_sp_units : 2;
            unsigned alt_units : 2;
            unsigned mag_var : 2;
            unsigned nav : 1;
      } flags; // these are bit fields totaling to 2 bytes

      char  hardware_mode;

      time_t  timestamp;
      struct
      {
            unsigned corrected : 1;
            unsigned filler    : 15;
      } flags2; // these are bit fields totaling to 2 bytes

      unsigned char satellites[5];

      char  filler[2];
};
```