# USGS\_SpectralDataReader Documentation

## What This Code Does

This code reads the spectral reflectance library splib06a produced by the USGS. After loading the data, the library will return either a pointer to one of the splib06a records or a fully parsed spectral record in a specpr formatted object called USGSRecord.

### Limitations

Though everything is in place to do so, this code does not attempt to provide access to the text data records within the splib06a file. It only strives to retrieve spectral record data.

## Algorithm Walkthrough

### Reading

This code opens the splib06a file. And then steps through is in 1536 (the USGS record format size) byte chunks. For each 1536 byte specpr record, the code allocates memory on the Heap and then copies the record to that memory.

### Accessing

The user passes an index value to one of 2 get functions. The code then accesses an vector element with that index and either returns a pointer to a specpr record or a parsed USGSRecord object.

## Class Style Guide

1. Readability was chosen over efficiency so that debuggers and porters could follow the algorithm easily
2. the class members and methods are grouped by overall function
3. All member variables are private.
4. All member variables have public get/set accessor functions.
5. When appropriate member variables have update accessor functions.
6. Static variables start with a lowercase "s".
7. Member variables start with a lowercase "m".
8. Local variables start with a lowercase "l".
9. Loop variables are a single lowercase letter. In our case we used "i"
10. Method return variables are all labeled "rvalue". They are used to insure that a method only has 1 exit point.
11. Return values (rvalue) are defaulted to indicate failure either through assignment to "false" or "NaN"
12. Accessor methods start with a lowercase "a".
13. Calculation methods start with a lowercase "c".
14. Void returns are discouraged.

## Class Structures

struct USGSRecord{

bool icflag[32];

unsigned char ititl[40];

unsigned char usernm[8];

unsigned int iscta;

unsigned int isctb;

unsigned int jdatea;

unsigned int jdateb;

unsigned int istb;

unsigned int isra;

unsigned int isdec;

unsigned int itchan;

unsigned int irmas;

unsigned int revs;

unsigned int iband[2];

unsigned int irwav;

unsigned int irespt;

unsigned int irecno;

unsigned int itpntr;

unsigned char ihist[60];

unsigned char mhist[296];

unsigned int nruns;

unsigned int siangl;

unsigned int seangl;

unsigned int sphase;

unsigned int iwtrns;

unsigned int itimch;

float xnrm;

float scatim;

float timint;

float tempd;

float /\*use with flag bits 00\*/ data[256] ;

float /\*use with flag bits 10\*/ cdata[383];

int /\*use with flag bits 01\*/ itxtpt;

int /\*use with flag bits 01\*/ itxtch;

unsigned char /\*use with flag bits 01\*/ itext[1476];

unsigned char /\*use with flag bits 11\*/ tdata[1532];

};

The USGSRecord structure holds data for a Specpr format record. The individual elements of the structure are described by the USGS documentation at

<http://speclab.cr.usgs.gov/specpr-format.html>

A copy of which is listed in the appendix.

## Class Static Members

public:

static const int sRecordLength = 1536;

sRecordLength:

The number of bytes in a Specpr record as defined by USGS

## Class Instance Members

private:

vector<char \* >mRecords;

mRecords:

A vector of pointers that are dynamically allocated to hold Specpr records.

## Class Accessor Methods

### Get Methods

//get methods

char\* aGetRecordBinaryData(int index);

USGSRecord aGetRecordUSGSFormat(int index);

Int aGetNumRecords();

int aGetNumCompiledRecords();

aGetRecordBinaryData:

Returns a pointer to the ith Specpr record

aGetRecordUSGSFormat:

Returns a USGSRecord structure populated by the ith Specpr record.

aGetNumRecords:

Returns the number of Specpr records in the splib06a file.

aGetNumCompiledRecords

Returns the number of spectral data records (which may contain more than one Specpr record i.e. the spectral data may take more than one Specpr record to fully quantify) in the splib06a file.

### Set Methods

//set methods

bool aCreateRecord();

aCreateRecord:

Allocates Heap memory for the elements of of mRecords.

## Class Calculation Methods

bool cIntelByteFlip(unsigned char \* data, int bytes);

bool cLoadRecords();

cIntelByteFlip:

Splib06a is recorded in the Sun Microsystems endian order. This method flips the endian order to an intel compatible order.

cLoadRecords:

Reads Splib06a and loads the binary data into mRecords.

## How To Use This Class

## Prerequisites

1. Your code must have a #include <USGS\_SpectralDataReader.h> statement, and the header file must be in you include library path.
2. The USGS library must be in the following relative path to the executing code:

./SpectralData/splib06a

## Reading the library

USGS\_SpectralDataReader lUSGSDataRecord = USGS\_SpectralDataReader();

lUSGSDataRecord.cLoadRecords();

## Accessing the Library

int lIndex = 0;

USGS\_SpectralDataReader::USGSRecord lRecord = lUSGSDataRecord.aGetRecordUSGSFormat(lIndex);

## Appendix

### USGS Specpr Format

The following format data was taken from <http://speclab.cr.usgs.gov/specpr-format.html>

In November of 2015

SPECtrum Processing Routines

Data Format

3/4/88

Length

Variable Description (bytes)

------------------------------------------------------------------------------

icflag 32 one-bit flags: 4

(low bit = 0, high bit = 31)

bit 00 continuation data flag

=0 first record of a spectrum consists of:

header then 256 data channels

=1 continuation data record consisting of:

bit flags followed by 1532 bytes of

real data (bit 1=0) (383 channels)

or 1532 bytes of text (bit 1=1).

A maximum of 12 continuation records

are allowed for a total of 4852

channels (limited by arrays of 4864)

or 19860 characters of text (bit 1=1).

bit 01 text/data flag:

=0 the data in the array "data" is data

=1 the data in the array "data" is ascii text

as is most of the header info.

bit 02 flag to indicate whether or not the data for

the error bar (1 sigma standard deviation of

the mean) is in the next record set.

=0: no errors, =1: errors in next record set.

bit 03 RA, Dec / Long., Lat flag:

=0 the array "ira" and "idec" corresponds to

the right ascension and declination of an

astronomical object.

=1 the array "ira" and "idec" correspond to

the longitude and latitude of a spot on a

planetary surface.

bit 04 variable iscta universal time/civil time flag

=0 cta is civil time

=1 cta is universal time

bit 05 variable isctb universal time/civil time flag

=0 ctb is civil time

=1 ctb is universal time

bit 06 unused

bit 07 unused

bit 08 unused

bit 09 unused

bit 10 unused

bit 11 unused

bit 12 unused

bit 13 unused

bit 14 unused

bit 15 unused

bit 16 unused

bit 17 unused

bit 18 unused

bit 19 unused

bit 20 unused

bit 21 unused

bit 22 unused

bit 23 unused

bit 24 unused

bit 25 unused

bit 26 unused

bit 27 unused

bit 28 unused

bit 29 unused

bit 30 unused

bit 31 unused

bit 32 unused

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* case 1: bit 00 = 0, bit 01 = 0 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

icflag Bit flags: 32 bits, see above. 4

ititl 40 character title which describes the 40

data (ascii, character\*40).

usernm The name of the user that created the data record 8

(ascii, character\*8).

iscta Civil or Universal time when data was 4

last processed (integer\*4 in scaled seconds).

Scaled by 24000. See flag #04.

isctb Civil or Universal time at the start of 4

the spectral run (integer\*4 in scaled seconds).

Scaled by 24000. See flag #05.

jdatea Date when data was last processed 4

Stored as integer\*4 Julian Day number \*10

jdateb Date when the spectral run began 4

Stored as integer\*4 Julian Day number \*10

istb Siderial time when the spectral run started. 4

(integer\*4 in scaled seconds).

Scaled by 24000. See flag #05.

isra Right ascension coordinates of an astronomical 4

object, or longitude on a planetary surface

(integer\*4 numbers in seconds \*1000)

(RA in RA seconds, Longitude in arc-seconds)

See flag #06.

isdec Declination coordinates of an astronomical 4

object, or latitude on a planetary

surface (integer\*4 number in arc-seconds \*1000).

See flag #06.

itchan Total number of channels in the spectrum 4

(integer\*4 value from 1 to 4852)

irmas The equivalent atmospheric thickness through 4

which the observation was obtained (=1.0

overhead scaled: airmass\*1000; integer\*4).

revs The number of independent spectral scans 4

which were added to make the spectrum

(integer\*4 number).

iband(2) The channel numbers which define the band 8

normalization (scaling to unity). (integers\*4)

irwav The record number within the file where the 4

wavelengths are found (integer\*4).

irespt The record pointer to where the resolution can 4

be found (or horizontal error bar) (integer\*4).

irecno The record number within the file where the 4

data is located (integer\*4 number).

itpntr Text data record pointer. This pointer points 4

to a data record where additional text describing

the data may be found. (32 bit integer)

ihist The program automatic 60 character history. 60

(ascii, character\*60)

mhist Manual history (4 lines of 74 characters 296

each. Program automatic for large history

requirements (ascii, character\*296).

nruns The number of independent spectral runs 4

which were summed or averaged to make this

spectrum (integer\*4).

siangl The angle of incidence of illuminating 4

radiation (Integer\*4 number, in arc-seconds\*6000).

(90 degrees=1944000000; -90 deg <= angle <= 90 deg)

integrating sphere = 2000000000

Geometric albedo = 2000000001

seangl The angle of emission of illuminating 4

radiation (Integer\*4 number, in arc-seconds\*6000).

(90 degrees=1944000000; -90 deg <= angle <= 90 deg)

integrating sphere = 2000000000

Geometric albedo = 2000000001

sphase The phase angle between iangl and eangl 4

(Integer\*4 number, in arc-seconds\*1500).

(180 degrees=972000000; -180 deg <= phase <= 180 deg)

integrating sphere = 2000000000

iwtrns Weighted number of runs (the number of runs 4

of the spectrum with the minimum runs which was

used in processing this spectrum, integer\*4).

itimch The time observed in the sample beam for 4

each half chop in milliseconds (for chopping

spectrometers only). (integer\*4)

xnrm The band normalization factor. For data scaled 4

to 1.0, multiply by this number to recover

photometric level (32 bit real number).

scatim The time it takes to make one scan of the 4

entire spectrum in seconds (32 bit real number).

timint Total integration time (usually=scatime \* nruns) 4

(32 bit real number).

tempd Temperature in degrees Kelvin 4

(32 bit real number).

data(256) The spectral data (256 channels of 32 bit 1024

real data numbers).

-----------------------------------------------------------------------------

case 1: total (bytes): 1536

==============================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* case 2: bit 00 = 1, bit 01 = 0 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

icflag Flags: see case 1 4

cdata(383) The continuation of the data values (383 channels 1532

of 32 bit real numbers).

------------------------------------------------------------------------------

case 2: total (bytes): 1536

==============================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* case 3: bit 00 = 0, bit 01 = 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

icflag Flags: see case 1 4

ititle 40 character title which describes the 40

data (ascii, character\*40).

usernm The name of the user that created the data record 8

(ascii, character\*8).

itxtpt Text data record pointer. This pointer points 4

to a data record where additional text may be

may be found. (32 bit integer)

itxtch The number of text characters (maximum= 19860). 4

itext 1476 characters of text. Text has embedded newlines 1476

so the number of lines available is limited only

by the number of characters available.

------------------------------------------------------------------------------

case 3: total (bytes): 1536

==============================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* case 4: bit 00 = 1, bit 01 = 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

icflag Flags: see case 1 4

tdata 1532 characters of text. 1532

------------------------------------------------------------------------------

case 4: total (bytes): 1536

==============================================================================