ICESAT/GLAS
Science Computing Facility

Product Tools Guide

Kristine Barbieri

Version 200712.0
November 2007

NASA/Goddard Space Flight Center
Cryospheric Sciences Branch/Code 614.1
Greenbelt, MD 20771

Section 1: Introduction

This document is intended to be a reference for the programmers at the GLAS remote SCF sites. It contains information about tools available to read the products and corresponding Data Management System files. These tools are in addition to the GLAS visualization GUI that displays all parameters on all products. Please note that these tools were in most cases written to be diagnostic tools for the GLAS SCF programmers, therefore, they may not do everything that a user may want. If this is the case, this reference guide should give enough information so that users may create their own programs using these tools as templates. DO NOT modify any of the codes themselves, always copy to your own directory, rename and modify.

Section 2: Table of Contents

Section 1: Introduction 2
Section 2: Table of Contents 3
Section 3: Reading the GLAS Standard Data Products 4
Section 3.1: Fortran 90 Tools 4
Section 3.1.1: F90 Product Structures.....4
Section 3.1.2: READ_PRODUCT_FILE.F90 5
Section 3.1.3: OUTPUT_LATLONELV.F90.....6
Section 3.1.4: OUTPUT_LATLONELV_1HZ.F90 7
Section 3.1.5: OUTPUT_HI_DEM.F90.....8
Section 3.1.6: PRINT_TIME_GLA06.F90 9
Section 3.1.7: READ_GLAS_FILE.F90 10
Section 3.1.8: READ_GLA01_WF.F90.....12
Section 3.1.9: HIGH_DEM_RET2.F90.....13
Section 3.2: IDL Tools.....16
Section 3.2.1: IDL Product Structures 16
Section 3.2.2: READGLADATA.PRO.....17
Section 3.2.3: DEMO1_GLA06ELEV.PRO.....18
Section 3.2.4: READ_GLAS_FILE.PRO.....19
Section 3.2.5: READ_GLA01_WF.PRO 21
Section 3.2.6: HIGH_DEM_RET.PRO 23
Section 4.1: BN Files.....26
Section 4.2: GR Files.....27
Section 4.3: UR Files.....28
Section 4.4: PS Files 29
Section 5: Reading the Rev File.....30
Section 6: Converting Between Date and J2000 Seconds.....31

Section 3: Reading the GLAS Standard Data Products

Section 3.1: Fortran 90 Tools

Section 3.1.1: F90 Product Structures

All of the Fortran 90 code at the mSCF uses the same product structures stored in a library to read the GLAS product files. This is to ensure consistency and decrease the amount of code maintenance. Different library versions correspond to different product formats, therefore it is important to use the correct version for a given file. As a default, it is usually best to use the latest version unless you know the version that corresponds to the file.

Before executing the programs in this section, the shared library path needs to be defined. As a convenience, the libraries and modules in the latest-versioned-subdirectory are now linked to an "ops" subdirectory. Therefore, /SCF/lib/ops may be used when exporting the shared library path:

i.e. export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)

The Fortran 90 product libraries used at the mSCF are in the directory:
- /SCF/lib/*latest_lib_version* → /SCF/lib/ops

The Fortran 90 modules for this library are in the directory:
- /SCF/src/modules/*latest_lib_version* → /SCF/src/modules/ops

Copies of the Fortran 90 product structure source codes are in the directory:
- /SCF/libsrc/*latest_lib_version*

In the source codes, the prod_mod.f90 files contain the structures of the products that are in integer format. The alg_mod.f90 files contain the structures of the algorithm variables that correspond to the product parameters. The algorithm variables are used in the scientific algorithms and are generally floating-point numbers. The scal_mod.f90 files contain scaling values that scale the floating-point algorithm values to integer product values and back again.

Section 3.1.2: READ_PRODUCT_FILE.F90

This program reads any product file and outputs to the screen: time, unique index, latitude, longitude, elevation, and GLA01 mode at once per second. For parameters at 40 Hz, only the first element is output. This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

To run, type:

- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/read_product_file <*product_file_name*>

The source code is:

- /SCF/src/directories/*latest_version*/Makefile_read_product
- /SCF/src/directories/*latest_version*/read_product_file.f90
- Any other codes or libraries are listed in the makefile

Section 3.1.3: OUTPUT_LATLONELV.F90

This program reads a GLA05 or GLA06 product file and outputs to a binary file and to the screen: latitude, longitude, and elevation at 40 Hz. The binary file is called latlonelv.dat. This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

Note: The output file, latlonelv.dat, cannot be overwritten. If you run the program twice without renaming or removing the output file, you will get the error:

I/O Error opening output file

To run, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/output_latlonelv <*product_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/Makefile_output_latlonelv
- /SCF/src/directories/*latest_version*/output_latlonelv.f90
- Any other codes or libraries are listed in the makefile

Section 3.1.4: OUTPUT_LATLONELV_1HZ.F90

This program reads a GLA05 or GLA06 product file and outputs to a binary file and to the screen: latitude, longitude, and elevation at 1 sec. The binary file is called latlonelv_1hz.dat. This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

Note: The output file, latlonelv_1hz.dat, cannot be overwritten. If you run the program twice without renaming or removing the output file, you will get the error:

I/O Error opening output file

To run, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/output_latlonelv_1hz <*product_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/Makefile_output_latlonelv_1hz
- /SCF/src/directories/*latest_version*/output_latlonelv_1hz.f90
- Any other codes or libraries are listed in the makefile

Section 3.1.5: OUTPUT_HI_DEM.F90

This program reads a GLA06 product file and outputs to a binary file and to the screen: latitude, longitude, elevation and SRTM elevation (DEM_hires_elv) at 40 Hz. The binary file is called output_hi_dem.dat. This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

Note: The output file, output_hi_dem.dat, cannot be overwritten. If you run the program twice without renaming or removing the output file, you will get the error:

I/O Error opening output file

To run, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/output_hi_dem <*product_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/Makefile_output_hi_dem
- /SCF/src/directories/*latest_version*/output_hi_dem.f90
- Any other codes or libraries are listed in the makefile

Section 3.1.6: PRINT_TIME_GLA06.F90

This program reads a GLA05 or GLA06 product file and outputs to the screen: shot, time1 (sec), time2 (msec), latitude, longitude, and elevation at 40 Hz. This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the “F90 Product Structures” section above).

To run, type:

- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)

- /SCF/bin/ops/print_time_gla06 <*product_file_name*>

The source code is:

- /SCF/src/directories/*latest_version*/Makefile_print_time_gla06
- /SCF/src/directories/*latest_version*/print_time_gla06.f90
- Any other codes or libraries are listed in the makefile

Section 3.1.7: READ_GLAS_FILE.F90

This program reads the data products and prints the parameters in integer format except for the flags that are printed in hex. These are sample codes to help users develop their own applications. All the source codes, makefile, and executable are in the directory, /SCF/src/readGLAS/*latest_version*. The programs are written in Fortan90 and developed in a HP/UX 11.x runtime environment. A relatively modern Fortran90 compiler is required. Information on how to run and compile is in the README_GLAS file.

Running the Program

To run the program, type:

read_glas_file <file=file_name> <start_rec=recStart> <n_of_rec=recN> <start_time=timeStart> <delta_time=timeD> <gla04_mod=gla04_mod>

where:

- file_name: The full path product file name.
- recStart: The beginning record to print the data. The record count starts after the headers. For example, recStart=1 means that the program will print the first data record after the headers. When using this keyword must also use recN.
- recN: Number of records to print. When using this keyword must also use recStart.
    - recStart=0 recN=0 will print all data records.
    - recStart=1, recN=1 is the default (prints the first data record) if recStart and recN are not specified
- timeStart: The start time to print the data in J2000 seconds. J2000 seconds start with second 0 on January 1, 2000 at 12 pm UTC. When using this keyword must also use timeD.
- timeD: Number of seconds to print the data. When using this keyword must also use timeStart.
    - Must set recStart=0 and recN=0 when using timeStart and timeD
- gla04_mod: For GLA04 only, indicates file type:
    - gla04_mod=1 for LPA
    - gla04_mod=2 for LRS
    - gla04_mod=3 for GYO
    - gla04_mod=4 for IST
    - gla04_mod=5 for BST
    - gla04_mod=6 for SCP

Notes about running:

- Keywords must always be included even for file_name.
- The file keyword is required.
- Either start_rec/n_of_rec or start_time/delta_time keywords are required. If both sets are used, then start_time/delta_time will be used.
- The gla04_mod keyword is optional and only used for GLA04 files.
- The product name must start with GLAxx where xx is the product number.

- The file scf70_000_00_00.dat must be in the same directory as the executable.

Modifying and Recompiling the Program

The program is: read_glas_file.f90
The executable is: read_glas_file
The makefile is: Makefile_read_glas

Each program prints out all the product variables. To print only part of these variables:

- Copy the source codes to your own directory. Include the make_defs and make_depends files and the scf70_000_00_00.dat file. The file scf70_000_00_00.dat needs to be in the same directory where the executable file is.
- The print statements are in GLAxx_print_mod.f90, where xx is the product number.
- In each GLAxx_print_mod.f90 there is a subroutine GLAxx_print(unit) where the unit is assigned to be 6 (standard out). The GLAxx_print routine prints all the variables in the product structure. You can comment out the variables that you are not interested in.
- Recompile by typing "make -f Makefile_read_glas".

Product structures are available for viewing from the SCF website, http://glas-scfweb.gsfc.nasa.gov by clicking on the link "GLAS standard data products" in the text.

Section 3.1.8: READ_GLA01_WF.F90

This program reads from GLA01 and outputs the uncompressed waveform values in volts to the screen. Forty shots are output for each data record. The output for each shot is: unique index number, shot time, number of samples in the shot, shot number, and the uncompressed waveform values in volts. The source code, makefile, and executable are in the directory, /SCF/src/readGLAS/*latest_version*. The programs are written in Fortan90 and developed in a HP/UX 11.x runtime environment. A relatively modern Fortran90 compiler is required. Information on how to run and compile is in the README_WF file.

Running the Program

To run a program, type:

read_gla01_wf <file=file_name> <start_rec=recStart> <n_of_rec=recN> <start_time=timeStart> <delta_time=timeD>

where:
- file_name: The full path product file name.
- recStart: The beginning record to print the data. The record count starts after the headers. For example, recStart=1 means that the program will print the first data record after the headers. When using this keyword must also use recN.
- recN: Number of records to print. When using this keyword must also use recStart.
    - recStart=0 recN=0 will print all data records.
    - recStart=1, recN=1 is the default (prints the first data record) if recStart and recN are not specified
- timeStart: The start time to print the data in J2000 seconds. J2000 seconds start with second 0 on January 1, 2000 at 12 pm UTC. When using this keyword must also use timeD.
- timeD: Number of seconds to print the data. When using this keyword must also use timeStart.
    - Must set recStart=0 and recN=0 when using timeStart and timeD

Notes about running:
- All arguments, excluding file_name, are optional.
- Keywords must always be included even for file_name.
- When using recStart, recN, timeStart, and timeD, the program will print the most restrictive condition.
- The file scf70_000_00_00.dat must be in the same directory as the executable.

Recompiling the Program

The program is: read_gla01_wf.f90
The executable is: read_gla01_wf
The makefile is: Makefile_read_wf

Section 3.1.9: HIGH_DEM_RET2.F90

This program reads from GLA06 and outputs the elevation per shot along with high resolution DEM values as derived from the SRTM (Shuttle Radar Topography Mission) database. The SRTM values form a 3x3 box around the latitude and longitude. Output is printed to an output file, GLA06_DEM.dat. The source code, makefile, and executable are in the directory, /SCF/src/readGLAS/latest_version. The programs are written in Fortan90 and developed in a HP/UX 11.x runtime environment. A relatively modern Fortran90 compiler is required. Information on how to run and compile is in the README_DEM file.

Notes about the SRTM Data

- Data only available between - 60 to 60 degrees of latitude.
- No ocean data available.
- 1 x 1 degree boxes of data are divided into 1201 x 1201 points.
- Track files were created from the Finished SRTM DTED® Level 1 data.
- Each track file covers a swath that has an E-W dimension of 4.2 km centered on the ICESat reference orbit ground track, subject to the latitude and ocean restrictions

Important Note about the SRTM Output Values

- GLA06 geoid values have been added to the SRTM elevation values to ease comparison with the GLAS elevations.

More information is available in the file SRTM_INFO.txt.

Running the Program

To run a program, type:

high_dem_ret <file_name>

where:
- file_name: The GLA06 product file name.

Notes about Running

The following files need to be in the directory where you invoke the program:

- the GLA06 file
- scf70_000_00_00.dat
- necessary Track file(s)

The track files are located in /SCF/product_sets/SRTM_tracks. If the directory is not there, please notify the SCF manager. The files are large and may not be transferred to a rSCF unless specifically requested. The zbulk8 and zbulk91 subdirectories contain the tracks for the 8-day

and 91-day repeat cycles, respectively. If the user is unaware of which track(s) are required, simply execute the following unix command:

```
strings <gla06_file_name> | grep Track
```

Note: when using a segment 4 file, you will need two track files: one for the current track and one for the next track.

Output

Output is written to the file "GLA06_DEM.dat".

The following parameters are output:

- Shot counter
- Unique record index
- UTC time in J2000 sec
- Delta time from equator crossing of track in sec
- Surface type
- Latitude in degrees
- Longitude in degrees
- Elevation from GLA06 in meters
- Error return code
- SRTM DEM in meters (9 values depicting 3x3 box)

Values for the error return code are listed below:

Error Values: 0 = None
1 = Read Error on Track Record
2 = Track Mismatch of Internal Track Number & Input Value
3 = Time Exceeds Bound of Reference Orbit
4 = Position is Outside EW Track Bounds
5 = Mismatch of Time/Coords relative to Reference Orbit
6 = Input Coordinates Outside Accessible Bounds
7 = DEM Data Not Available at this Location
8 = Averaged Value Returned

Notes about the SRTM Output Values

The nature of the SRTM's 3 arc-second quantization in latitude and longitude makes it much like a large Cartesian grid of nodes, with an elevation, latitude, and longitude associated with each node. SRTM data retrieval is performed relative to an input set of GSAS latitude/longitude coordinates. These coordinates are used to find the closest node. This will not be an exact match, but it is used as the center-point to retrieve a 3 x 3 matrix of elevations surrounding the user's input position. This matrix is written to the output in the order: A, B, C, D, E, F, G, H, I.

It follows the row/column coordinate organization:

```
A = (1,1)   B = (1,2)    C = (1,3)    South
D = (2,1)   E = (2,2)    F = (2,3)      |
G = (3,1)   H = (3,2)    I = (3,3)    North

        West -> East
```

A user may then use this matrix to interpolate any near-by elevation or slope of the terrain as a function of the coordinates. Increments between node-points are 1/1200 of a degree (or 0.000833 degrees). The center point can be retrieved by rounding each of the input latitude and longitude points to the nearest 1/1200 of a degree.

Recompiling the Program

The program is:	High_DEM_Ret2.f90
The executable is:	high_dem_ret
The makefile is:	Makefile_high_dem_ret

Section 3.2: IDL Tools

Section 3.2.1: IDL Product Structures

All of the IDL code at the mSCF uses the same product structures created for the visualization software. This software is periodically stored in versioned “snapshot” directories to preserve updated work. Files in the latest snapshot directory are also stored in /SCF/IDL/lib/scf_visualizer/ops. Read_glas_file.pro is slightly different in that it does not access the product structures in these directories, but instead contains copies of them its own directory to make it inclusive.

Section 3.2.2: READGLADATA.PRO

This program reads data from any product file. Users may specify unique index range, time range, product parameters, and product version. The program needs the accompanying unique index (UR) file. Description and examples are at the top of the program. This program uses the mSCF IDL product structures and needs to have their path added to the IDL path (refer to the "IDL Product Structures" section above).

To run, type:
- cd /SCF/IDL/lib/readgladata
- idl
- GLAS_AddReadersToPath – adds /SCF/IDL/lib/scf_visualizer/ops to path
- Follow the directions in /SCF/IDL/lib/readgladata/readgladata.pro
- Notes on printing:
  - Printing routines are located in /SCF/IDL/lib/scf_visualizer/ops:
    - i.e. gla06_algv4_0_init.pro contains gla06_algv4_0_print for printing GLA06 version 4.0 algorithm data
    - i.e. gla06_pv4_0__define.pro contains gla06_pv4_0_print for printing GLA06 version 4.0 raw data
  - If output data array contains all values then use appropriate printing routine
    (i.e. readgladata, file, data, version=[4,0]
    gla06_algv4_0_print, data)
  - If output data array contains all raw values then use appropriate printing routine
    (i.e. readgladata, file, data, version=[4,0], /RAW
    gla06_pv4_0_print, data)
  - If output data array contains only certain values, may need to format printout
    (i.e. readgladata, file, data, version=[4,0], components=['d_utctime', 'l_surftype']
    print, format='(f25.6,4i3)', data)
  - To print header array with keywords listed, use help rather than print
    (i.e. readgladata, file, data, version=[4,0], HEADER=Header
    help, /struct, header)

The program getbylatlon.pro is an add-on to readgladata.pro and allows the user to input latitude and longitude in addition to the input parameters accepted by readgladata.pro.

To run, type:
- cd /SCF/IDL/lib/readgladata
- idl
- GLAS_AddReadersToPath – adds /SCF/IDL/lib/scf_visualizer/ops to path
- Follow the directions in /SCF/IDL/lib/readgladata/getbylatlon.pro

Section 3.2.3: DEMO1_GLA06ELEV.PRO

This program was written as a template to create your own IDL codes for reading products and plotting parameters. It reads GLA06 and plots elevation vs. time. The program needs the accompanying unique index (UR) file. Description and examples are at the top of the program. This program uses the mSCF IDL product structures and needs to have their path added to the IDL path (refer to the “IDL Product Structures” section above).

To run:

- Follow the directions in /SCF/IDL/lib/scf_visualizer/examples/demo1_gla06elev.pro

Section 3.2.4: READ_GLAS_FILE.PRO

This program reads from any product except for GLA03 and non-LPA GLA04. However, only the product variables are printed, not the algorithm variables. Users need to specify the input file, product, release, first record to print, and number of records to print in the control file, read_glas_ctrl.dat. Do not change the control file name, it is expected by the program. An optional output file name may also be specified. If present, the output will be written to the file; if not, the output will be written to the screen. Information on how to run and an example of the control file is in the README_GLAS file. This program may be run at the IDL prompt or using the IDL Virtual Machine:

To run at the IDL prompt, type:
- cd /SCF/IDL/IDLreadGLAS
- Edit the control file, read_glas_ctrl.dat, and modify as necessary
- Idl
- !path='/SCF/IDL/IDLreadGLAS:'+!path (if not in /SCF/IDL/IDLreadGLAS)
- .run read_glas_file
- read_glas_file
- If an output file was specified, click on the done screen

Example of control file:

```
INPUT_FILE =
/SCF/product_sets/current/L3_2109/GLA05_022_2109_001_1275_3_01_0001.P0621
# Product Number corresponding to input file (i.e. 1 for GLA01)
I_PRODUCT = 5
# Data release number of input file
RELEASE = 22
# Beginning time to print
START_TIME = 140781825.9
# Number of seconds to print
DELTA_TIME = 2
# First data record to print
REC1 = 1
# Number of records to print starting with first record above
# NREC = 0 will print to the end of the file
NREC = 2
# Output File (Optional)
OUTPUT_FILE = out.dat
# Unpack Flags? (1=yes) (Optional)
UNPACK_FLAGS = 1
=END
```

Notes on keywords:
- Use START_TIME/DELTA_TIME or REC1/NREC to specify records displayed. If both sets are specified, then START_TIME/DELTA_TIME will be used.
- UNPACK_FLAGS: Flags are usually packed on the GLAS products so that several flags of one to several bits each can be stored together in a byte or more. By default, the flags are displayed as they are on the product. If the keyword, "UNPACK_FLAGS = 1" is specified,

however, then the flags are unpacked in a special section at the end of each record.
Note: This option is only available for releases 22 (V4.3) and above

To run using the IDL virtual machine, type:
- cd /SCF/IDL/IDLreadGLAS
- Edit the control file, read_glas_ctrl.dat, and modify as necessary
- idl -vm='read_glas_file.sav'
- Click on the flash screen
- If an output file was specified, click on the done screen

To run on the PC, type:
- Edit the control file, read_glas_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Bring up IDL
- Go to File/Preferences and set IDL “Startup” and “Path” accordingly
- .run read_glas_file
- read_glas_file
- Click on the done screen

To run on the PC using the IDL virtual machine:
- Edit the control file read_glas_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Click on read_glas_file.sav
- Click on the flash screen
- Click on the done screen

Section 3.2.5: READ_GLA01_WF.PRO

This program reads from GLA01 and outputs the uncompressed waveform values in volts. Users need to specify the input file, release, first record to print, and number of records to print in the control file, read_gla01_ctrl.dat. Do not change the control file name, it is expected by the program. An optional output file name may also be specified. If present, the output will be written to the file, if not the output will be written to the screen. Forty shots are output for each data record. The output for each shot is: unique index number, shot time, number of samples in the shot, shot number, and the uncompressed waveform values in volts. Information on how to run and an example of the control file is in the README_WF file. This program may be run at the IDL prompt or using the IDL Virtual Machine:

To run at the IDL prompt, type:
- cd /SCF/IDL/IDLreadGLAS
- Edit the control file, read_gla01_ctrl.dat, and modify as necessary
- Idl
- !path='/SCF/IDL/IDLreadGLAS:'+!path (if not in /SCF/IDL/IDLreadGLAS)
- .run read_gla01_wf
- read_gla01_wf

Example of control file:

```
INPUT_FILE =
/SCF/product_sets/current/L3_2111/GLA01_019_2111_002_0424_3_01_0001.P0770
# Data release number of input file
RELEASE = 19
# Beginning time to print
START_TIME = 140781825.9
# Number of seconds to print
DELTA_TIME = 2
# First data record to print
REC1 = 1
# Number of records to print starting with first record above
# NREC = 0 will print to the end of the file
NREC = 1
# Output File (Optional)
OUTPUT_FILE = out_wf.dat
=END
```

Notes on keywords:
- Use START_TIME/DELTA_TIME or REC1/NREC to specify records displayed. If both sets are specified, then START_TIME/DELTA_TIME will be used.

To run using the IDL virtual machine, type:
- cd /SCF/IDL/IDLreadGLAS
- Edit the control file, read_gla01_ctrl.dat, and modify as necessary
- idl -vm=’read_gla01_wf.sav’
- Click on the flash screen

To run on the PC, type:

- Edit the control file, read_gla01_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Bring up IDL
- Go to File/Preferences and set IDL “Startup” and “Path” accordingly
- .run read_gla01_wf
- read_gla01_wf

To run on the PC using the IDL virtual machine:
- Edit the control file read_gla01_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Click on read_gla01_wf.sav
- Click on the flash screen

Section 3.2.6: HIGH_DEM_RET.PRO

Reads data from a GLA06 file and prints the elevation per shot along with high resolution DEM values as derived from the SRTM (Shuttle Radar Topography Mission) database. The SRTM values form a 3x3 box around the latitude and longitude. Output is printed to an output file. Information on how to run and an example of the control file is in the README_DEM file.

NOTES ABOUT THE SRTM DATA:

- Data only available between - 60 to 60 degrees of latitude.
- No ocean data available.
- 1 x 1 degree boxes of data are divided into 1201 x 1201 points.
- Track files were created from the Finished SRTM DTED® Level 1 data.
- Each track file covers a swath that has an E-W dimension of 4.2 km centered on the ICESat reference orbit ground track, subject to the latitude and ocean restrictions.

Important Note about the SRTM Output Values

- GLA06 geoid values have been added to the SRTM elevation values to ease comparison with the GLAS elevations.

More information is available in the file SRTM_INFO.txt.

SYNTAX:

high_dem_ret

NEEDS A CONTROL FILE TO RUN:

Assumes that the name of the control file is high_dem_ret_ctrl.dat and it resides in the directory that high_dem_ret is run from.

EXAMPLE OF A CONTROL FILE:

```
# Input File (GLA06)
INPUT_FILE = GLA06_024_2103_001_1289_1_01_0001.DAT
=END
```

NOTES ABOUT RUNNING:

The following files need to be in the directory where you invoke the program:

- the GLA06 file
- scf70_000_00_00.dat
- necessary Track file(s)

The track files are located in /SCF/product_sets/SRTM_tracks. If the directory is not there, please notify the SCF manager. The files are large and may not be transferred to a rSCF unless specifically requested. The zbulk8 and zbulk91 subdirectories contain the tracks for the 8-day

and 91-day repeat cycles, respectively. If the user is unaware of which track(s) are required, simply execute the following unix command:

```
strings <gla06_file_name> | grep Track
```

Note: when using a segment 4 file, you will need two track files: one for the current track and one for the next track.

OUTPUT:

Output is written to the file "GLA06_IDL_DEM.dat".

The following parameters are output:

- Shot counter
- Unique record index
- UTC time in J2000 sec
- Delta time from equator crossing of track in sec
- Surface type
- Latitude in degrees
- Longitude in degrees
- Elevation from GLA06 in meters
- Error return code
- SRTM DEM in meters (9 values depicting 3x3 box)

Values for the error return code are listed below:

Error Values: 0 = None
1 = Read Error on Track Record
2 = Track Mismatch of Internal Track Number & Input Value
3 = Time Exceeds Bound of Reference Orbit
4 = Position is Outside EW Track Bounds
5 = Mismatch of Time/Coords relative to Reference Orbit
6 = Input Coordinates Outside Accessible Bounds
7 = DEM Data Not Available at this Location
8 = Averaged Value Returned

NOTES ABOUT THE SRTM OUTPUT VALUES:

The nature of the SRTM's 3 arc-second quantization in latitude and longitude makes it much like a large Cartesian grid of nodes, with an elevation, latitude, and longitude associated with each node. SRTM data retrieval is performed relative to an input set of GSAS latitude/longitude coordinates. These coordinates are used to find the closest node. This will not be an exact match, but it is used as the center-point to retrieve a 3 x 3 matrix of elevations surrounding the user's input position. This matrix is written to the output in the order: A, B, C, D, E, F, G, H, I.

It follows the row/column coordinate organization:

A = (1,1)	B = (1,2)	C = (1,3)	South
D = (2,1)	E = (2,2)	F = (2,3)	|
G = (3,1)	H = (3,2)	I = (3,3)	North

West -> East

A user may then use this matrix to interpolate any near-by elevation or slope of the terrain as a function of the coordinates. Increments between node-points are 1/1200 of a degree (or 0.000833 degrees). The center point can be retrieved by rounding each of the input latitude and longitude points to the nearest 1/1200 of a degree.

REQUIRED PATH:
If the user wants to run the program from a directory other than the directory that the source files are in, xx, the following command needs to be applied to include the source code directory:

```
!path=+"xx:"+!path
```

Example to run the program:

```
idl
!path="/SCF/IDL/IDLreadGLAS:"+"/usr/local/bin:"+!path
.run high_dem_ret.pro
high_dem_ret
```

OTHER MODES OF RUNNING:

This software also runs under the IDL virtual machine. To do so:

- Edit the read_glas_ctrl.dat control file as desired
- Type at the Unix command line: idl -vm='high_dem_ret.sav'
- Click on the flash screen when it comes up
- Click on the done screen when it comes up

To run on the PC:
- Edit the control file, high_dem_ret_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Bring up IDL
- Go to File/Preferences and set IDL "Startup" and "Path" accordingly
- Type at the IDL prompt: .run high_dem_ret
- Type at the IDL prompt: high_dem_ret

To run on the PC using the IDL virtual machine:
- Edit the control file, high_dem_ret_ctrl.dat, and modify as necessary (an output file needs to be specified)
- Click on high_dem_ret.sav
- Click on the flash screen

Section 4: Creating and Reading Data Management System files

Section 4.1: BN Files

The bin file lists the geographic bins that the product set traverses and the unique index numbers within the product set that go through that bin. One bin file is required for each 14 rev product set. Refer to the *SCF Data Management System File Format Document* for the format of this file. This file is necessary for viewing the products with the "Select from Data Base" option of the visualization GUI.

The bin file is created using the Fortran 90 program, mkbinrev. This program uses a control file as its argument. An example of the control file is as follows:

```
=MKBINREV Control file SCF data selection
INPUT_FILE=GLA05_002_1101_001_0029_1_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_2_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_3_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_4_01_0001.P0003
TIME_PASS_FILE=/SCF/ancillary_data/rev_files/total_rev_file.dat
OUTPUT_FILE=BNA_1101_001_0029.P0003_00
ERR_FILE=/SCF/ancillary_data/scf70_000_00_00.dat
=END of control file
```

This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

To create the BN file, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/mkbinrev <*control_file_name*>

To read the BN file, type:
- /SCF/bin/ops/readbinrev <*BN_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/readbinrev.f90

Section 4.2: GR Files

The georeference file allows direct access to the start of each bin in the bin file. One georeference file is required for each bin file. Refer to the *SCF Data Management System File Format Document* for the format of this file. This file is necessary for viewing the products with the "Select from Data Base" option of the visualization GUI.

The georeference file is created using the Fortran 90 program, mkgeo. This program uses a control file as its argument. An example of the control file is as follows:

```
=MKGEO Control file SCF data selection
INPUT_FILE= BNA_1101_001_0029.P0003_00
OUTPUT_PATH=
ERR_FILE=/SCF/ancillary_data/scf70_000_00_00.dat
=END of control file
```

This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

To create the GR file, type:

- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/mkgeo <*control_file_name*>

To read the GR file, type:

- /SCF/bin/ops/readgeo <*GR_file_name*>

The source code is:

- /SCF/src/directories/*latest_version*/readgeo.f90

Section 4.3: UR Files

The unique index file correlates the physical data record numbers with the unique index numbers in a product file. One unique index file is required for each product file. Refer to the *SCF Data Management System File Format Document* for the format of this file. This file is necessary for viewing the products with the "Select from Data Base" or "Select from Specific Data Set" option of the visualization GUI.

The unique index file is created using the Fortran 90 program, mkunique_index. This program uses a control file as its argument. An example of the control file is as follows:

```
=GLAS_INDEX Control file SCF data selection
INPUT_FILE=GLA05_002_1101_001_0029_1_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_2_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_3_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_4_01_0001.P0003
OUTPUT_FILE_PATH=
ERR_FILE=/SCF/ancillary_data/scf70_000_00_00.dat
=END of control file
```

This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

To create the UR file, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/mkunique_index <*control_file_name*>

To read the UR file, type:
- /SCF/bin/ops/readunique <*UR_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/readunique.f90

Section 4.4: PS Files

The pass file lists the passes present in the product file. One pass file is required for each product file. Refer to the *SCF Data Management System File Format Document* for the format of this file. This file is necessary for viewing the products with the "Select from Data Base" option of the visualization GUI.

The pass file is created using the Fortran 90 program, mkpass. This program uses a control file as its argument. An example of the control file is as follows:

```
=MKPASS Control file SCF data selection
INPUT_FILE=GLA05_002_1101_001_0029_1_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_2_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_3_01_0001.P0003
INPUT_FILE=GLA05_002_1101_001_0029_4_01_0001.P0003
TIME_PASS_FILE=/SCF/ancillary_data/rev_files/total_rev_file.dat
OUTPUT_PATH=
ERR_FILE=/SCF/ancillary_data/scf70_000_00_00.dat
=END of control file
```

This program uses the mSCF Fortran 90 libraries and needs to have the library path defined or else a core dump will occur (refer to the "F90 Product Structures" section above).

To create the PS file, type:
- export SHLIB_PATH=/SCF/lib/ops (for ksh) OR
  setenv SHLIB_PATH /SCF/lib/ops (for csh, tcsh) OR
  export LD_LIBRARY_PATH=/SCF/lib/ops (for linux ksh)
- /SCF/bin/ops/mkpass <*control_file_name*>

To read the PS file, type:
- /SCF/bin/ops/readpass <*PS_file_name*>

The source code is:
- /SCF/src/directories/*latest_version*/readpass.f90

Section 5: Reading the Rev File

The rev file lists the starting time of each pass. The rev file is constantly appended as the satellite makes new passes. It is created at the I-SIPS and sent to the mSCF. Refer to the *SCF Data Management System File Format Document* for the format of this file. This file is necessary for viewing the products with the "Select from Data Base" option of the visualization GUI. It is also necessary for creating bin files and pass files.

The rev file currently used at the mSCF is:
- /SCF/ancillary_data/rev_files/total_rev_file.dat

To read the rev file, type:
- /SCF/bin/ops/readrev <*rev_file_name*>

In each record, the first number is the start time of the rev in J2000 seconds UTC. The last number is the rev number.

The source code is:
- /SCF/src/directories/*latest_version*/readrev.f90

Section 6: Converting Between Date and J2000 Seconds

A utility has been created to convert from date (year-month-day, hour-minutes-seconds) to J2000 seconds and visa versa. This utility is /SCF/bin/ops/datecon2003.

To see a list of options for using datecon2003, type:
- /SCF/bin/ops/datecon2003

The last two items listed are for converting date to J2000 seconds and back.

To convert from J2000 seconds to date, type:
- /SCF/bin/ops/datecon2003 j2000sectoyymd <J2000sec>

To convert from date to J2000 seconds, type:
- /SCF/bin/ops/datecon2003 yymdtoj2000sec <yyyymmdd hhmmss>

Remember that for J2000sec=0 the date is January 1, 2000 at 12:00 pm UTC.