GEOCAT MANUAL

Geocat Version 0.5

geocat_manual.doc
Revised 2007-12-16
Graeme Martin
graemem@ssec.wisc.edu

Introduction

The Geostationary Cloud Algorithm Testbed (Geocat) was initially developed by the GOES-R Algorithm Working Group (AWG) Cloud Application Team to serve as a flexible cloud retrieval algorithm development tool. Since its inception Geocat's capabilities have grown such that all sorts of algorithms (fire detection, soundings, aerosol, etc.) can be accommodated. Geocat provides a convenient interface to measured radiances, ancillary data (NWP profiles, surface emissivity, surface type, snow, etc.), fast model generated clear sky radiance profiles, and measured and/or generated data from previous image time steps. Geocat also provides a common algorithm output structure, whose definition is transparent to the algorithm developer. Geocat is currently capable of processing GOES (current generation), MSG, MTSAT, or simulated GOES-R ABI data.

Geocat is distributed as source code and typically built per-user in the home directory. Ancillary and input data can be shared among users. The user can select which science algorithms will be incorporated into a given build of Geocat, and can then select which algorithms to run in a given program invocation. This design is intended to facilitate rapid algorithm development and to allow easy comparison of output from different algorithms or different versions of the same algorithm. Geocat's memory footprint can be manipulated at program invocation by specifying the segment size, i.e. the number of image lines to process at a time.

Adding an Algorithm to Geocat

This set of instructions assumes you have installed and successfully tested Geocat by following the installation instructions. The 'Geocat' directory referred to throughout this document is the installation directory. In the Geocat directory there is a subdirectory called 'src' that contains the core Geocat source files. You should not modify the files in this directory. To add an algorithm to Geocat, you will run a script that will create a new directory (the 'algorithm' directory) containing a Fortran module source file and an XML file that describes the algorithm to Geocat. After you have successfully compiled and run the 'stub' algorithm, you will be able to modify the Fortran file and the XML file.

Generate a 'stub' module and XML algorithm description file

Choose a directory name, algorithm name, keyword, module name and main subroutine name for your algorithm, as described below. Any of these values can be changed later.

Table 1: create_module.sh arguments

directory	name of the directory to be created to hold source files for this algorithm and any related algorithms, e.g. clouds
algorithm	unique long name for your algorithm in quotes, e.g. 'Baseline GOES IM Cloud Mask'
keyword	unique keyword for your algorithm that will be used as a prefix for output variables, e.g. baseline_cmask_goes_im
module	the name of the Fortran module (also the source filename will be *module*.f90), e.g. baseline_cloud_mask
subroutine	the name of the main (entry) subroutine in the Fortran module, e.g. baseline_cloud_mask_main

In your main Geocat directory, type:

```
./create_module.sh directory algorithm keyword module subroutine
```

The script will create the specified directory with the files algorithm_list.xml and *module*.f90, where *module* is the value you specified.

Generate and compile the source

1) In your main Geocat directory, run the python script to generate the source files 'algorithm_mod.f90' and 'algorithm_module_names.f90', and the makefile.

```
python algomodgen.py directory
```

where 'directory' is the name of the directory containing your algorithm. The generated files will be located in the directory 'src'.

2) In the directory 'src', build the application in debug mode. On a 32-bit machine, type:

```
make intel32
```

On other machines, type:

```
make intel
```

The Intel Fortran compiler is currently the only compiler that is known to work. If successful, an executable named 'geocat' will have been created in your main Geocat directory.

Run the algorithm in Geocat

1) Run Geocat with the '-alist' option to print algorithm information:

```
./geocat -alist
```

The output should list a single algorithm with index '1' and the name you specified earlier.

2) Test the stub algorithm by processing a MET-8 file from the data package, reading pre-computed navigation information from a file:

```
./geocat -a 1 -verbose -read_nav -f
met08_disk_1_2006_213_1200.area.gz -time_report
```

or

```
./geocat -akey algorithm_keyword -verbose -read_nav –f
met08_disk_1_2006_213_1200.area.gz -time_report
```

If successful, there will be a file called 'geocatL2.Meteosat-8.2006213.120000.hdf' in the directory 'l2_output' that contains navigation and ecosystem data.

In the second example above, 'algorithm_keyword' is the keyword that you specified when you invoked created_module.sh. Although the two invocations are equivalent, In practice it is best to use the '-akey' option instead of '-a' since the algorithm index is auto-generated by the python script algomodgen.py, and may change when other algorithms are added. The keyword will not change unless the user changes it in the XML file.

Now that you have successfully created a new algorithm module, you can add algorithm-specific source code to the module file and modify the XML to describe the inputs and outputs required by your algorithm.

The XML Algorithm Description File

Each algorithm directory must contain an XML file that describes the algorithm and defines the interface between the algorithm module and Geocat. It is used to generate Fortran source code that is then compiled into the Geocat executable. Any optional data requirements are specified in the XML file (cloud mask, NWP, RTM, etc.), as well as any required output or scratch variables. The file must be named 'algorithm_list.xml'.

Whenever the XML file is modified, the algorithm interface source files and makefile must be re-generated and Geocat must be recompiled, as described in the section "Building Geocat".

The basic format of the XML file is shown in Figure 1 in the Appendix. An example XML file from the cloud mask algorithm is shown in Figure 2 in the Appendix.

The 'algorithm' XML element describes a single algorithm; if there are multiple algorithm modules in the same directory, each is represented by a single algorithm element in the XML file. Just keep in mind that no two algorithms can have the same algorithm keyword. The python script will warn you if it encounters multiple instances of any of those parameters.

It is important to note that multiple algorithm directories can be included in a single build of Geocat by specifying multiple directory arguments to the python script (see the section "Building Geocat").

Informational tags and attributes

The 'algorithm' element contains the informational tags and attributes shown below. The values that have not been set by the 'create_module.sh' script should be assigned meaningful values (see Table 2).

```
<algorithm name="name" keyword="keyword">
        <comment value="fortran header comment" />
        <module value="geocat_module_name" />
        <function value="geocat_function_name" />
        <reference value="reference" />
        <ancil_subdir value="ancil_subdir" />
```

Table 2: XML algorithm description file -- algorithm element

tag	attribute	description
algorithm	name	unique long name for the algorithm.
algorithm	keyword	unique short name for the algorithm, used as a prefix for algorithm-specific variables written to the output file

comment	value	(optional) string that describes the algorithm
version	value	(optional) algorithm version number
module	value	name of the Fortran module that will contain the algorithm (also used in name of source file)
function	value	name of the entry subroutine for your algorithm to be called by Geocat
reference	value	a string describing the origins of the algorithm science (e.g. Johnson et al., 2005)
ancil_subdir	value	name of the directory that will contain ancillary data for the algorithm, located in 'data/algorithms'
inactive		(optional) if present, the algorithm will not be compiled into Geocat
cmask_needed		(optional) if present, indicates cloud mask is required by the algorithm
ctype_needed		(optional) if present, indicates cloud type is required by the algorithm
cldz_needed		(optional) if present, indicates cloud top height is required by the algorithm
amask_needed		(optional) if present, indicates aerosol mask is required by the algorithm (experimental)
nwp_needed		(optional) if present, indicates NWP data is required by the algorithm
rtm_needed		(optional) if present, indicates RTM data is required by the algorithm
active_needed		(optional) if present, indicates active sensor data is required by the algorithm (experimental)
sds_list		list of sds_item elements specifying the Geocat output and scratch variables that will be used by the algorithm (see sds_item description in Table 3)
channels	values	comma separated list of Geocat channels used by the algorithm (see Geocat / instrument channel mappings in Table 4)

Specifying input data

If the tags "rtm_needed" and "nwp_needed" are included in the XML file, Geocat will make RTM and NWP data available as input to the algorithm.

The tags "cmask_needed", "ctype_needed" and "cldz_needed" indicate that the inputs cloud mask, cloud type and cloud height, respectively, are required by the algorithm. If one of these tags is present, the user must specify a default cloud mask, cloud type or cloud height algorithm at program invocation, either with the command line option "-cmask", "-ctype" or "-cldz", or in the file "geocat.default". See the section "Running Geocat" for more information. Since Geocat is not distributed with any cloud algorithms, the user is responsible for either writing an algorithm that produces the required product, or obtaining a

version from the clouds team (Contact Andrew Heidinger, heidinger@ssec.wisc.edu, or Mike Pavolonis, mpav@ssec.wisc.edu)

Level 1B calibrated and navigated data and ancillary data are provided automatically as input to the algorithm and do not need to be specified in the XML file.

Specifying input channels

The channels element contains a comma-separated list of input channels required by the algorithm:

```
<channels values="comma,separated,list,of,channels" />
```

In order to use channel-specific input variables, the channel must be specified here. For example, if an algorithm requires ref2 (calibrated channel 2 reflectance), the channels list must contain the number '2'.

Specifying output and scratch variables

Geocat can write data produced by algorithms at pixel resolution to a Level 2 output file. Any output variable that is required by an algorithm must be specified in the XML file by an 'sds_item' in the 'sds_list' element. Memory will only be allocated for the variables in this list. Table 17 contains a list of product-specific output variables that are available to algorithms at pixel resolution.

The format of the 'sds_item' is shown below.

```
<sds_item name="sds_name">
        <dim3_size value="dimen3_size" /> (optional)
        <stride_factor value="stride_factor" /> (optional)
        <produces_output /> (optional)
</sds_item>
```

The attribute 'sds_name' is the name of the variable, e.g. 'fire_mask'. If the variable is 3-dimensional, the size of the third dimension is specified by 'dimen3_size'. The xy stride factor can be specified with 'stride_factor'. The variable will only be written to the output file if the 'produces_output' tag is present; if this tag is not present the variable can still be used as a scratch space at pixel resolution.

Most of the variables listed in Table 17 are associated with a specific product, but there are several generic variables that are available for scratch space and debugging, with names like 'i1_generic1' and 'r4_generic1'.

Table 3: XML algorithm description file -- sds_item element

tag	attribute	description
sds_item	name	name of Geocat output variable
dim3_size	value	(optional) positive integer, specifies the 3rd dimension, for 3-dimensional variables only
stride_factor	value	(optional) positive integer, specifies the xy stride factor (warning: this feature is experimental!)
produces_output		(optional) if present, data will be written to the Level 2 output file

Modifying the algorithm source code

The algorithm source code is contained in a Fortran file called '*module*.f90' located in the algorithm directory, where 'module' is the case-sensitive value specified in the XML algorithm file.

The Geocat processing model

Geocat operates on chunks of data called segments, specified at runtime as a fixed number of image lines. A single segment is read in, processed, and output is written to file before the next segment is read in. The memory footprint of Geocat can be controlled by changing the segment size, either at the command line or in the file 'geocat.default'.

The main algorithm subroutine is invoked once per segment, after calibration, navigation and any user-specified optional processing like NWP or RTM calculation. Typically an algorithm loops through the lines and elements in the current segment, processing a single pixel at a time. An algorithm should never assume fixed-size data arrays, because segment size is not known until runtime, and the last segment is typically smaller than the others. The number of elements and lines in the current segment are available in the variables 'sat%nx' and 'sat%ny'. These variables represent the upper limit on indices into input and output arrays that are at pixel-resolution.

If an algorithm encounters bad data or other error conditions, it should not terminate the application by calling a "stop" statement or by any other means. Preferably the algorithm will set an error flag or display an error message. This is to ensure robustness when applications are being run together and to avoid the undesirable condition that "bad" data as determined by an algorithm can terminate the entire program.

A limitation of the segment processing model as currently implemented is that only pixels in the same segment can be used to perform spatially dependent calculations, e.g. spatial uniformity. A future version will allow for user-controlled scan line segment overlap.

The algorithm module source file

The 'stub' algorithm implementation that is generated by the script 'create_module.sh' contains a Fortran module with a single subroutine and placeholder comments indicating where blocks of code belong. Once you have successfully compiled and tested the stub algorithm, you can edit the source file to integrate actual algorithm code.

Module-level declarations

Declarations at the module level (those that occur before the 'contains' statement) are scoped to the entire module. For example, a variable declared at the module-level can be used in the subroutines and functions in the module.

Preferably variables and constants should be declared using the named constant 'kind' parameters defined in 'constant.f90' (e.g. 'real4') rather than explicitly stating the precision. This enforces consistency of precision throughout the program, and allows the precision to be changed by modifying a single line, thus improving portability.

```
INTEGER (kind=int4) :: sfc_level
```

The statement 'use ALGORITHM_MODULE_USAGE' allows access to Geocat input and output structs and utility procedures throughout the module.

Main subroutine

The main subroutine is called by Geocat and must have a single input-type argument, the Geocat algorithm index, which is used to index into the output struct. The suggested structure of the main subroutine is:

- declaration section
- pre-processing steps
- the main per-pixel processing loop: for each pixel, do something
- post-processing steps

The generated code includes a per-pixel loop. Within the loop, input data for the current pixel can be accessed with the line and element indices as in the example below:

```
if (sat%space_mask(ielem,iline) == sym%SPACE) then
          cycle
endif
```

NOTE: The code snippet above checks whether the current pixel is a 'space' pixel, as determined by Geocat, and if so, skips to the next loop iteration. This is important because in order to save processing time *Geocat does not provide algorithm input for space pixels*. This means that if the current pixel is flagged as a space pixel in the space_mask array (e.g. the pixel is not geolocated on the earth or it has a viewing angle greater than the maximum allowable viewing angle defined by the user at run time), none of the input values are valid. So it is very important to perform this check before processing a pixel.

Within the pixel loop, you can access data from other pixels in the segment (spatial dependence), but be careful not to index outside the bounds of the arrays that hold data for the current segment. The number of lines and elements in the current segment are available in the 'sat' struct variables 'ny' and 'nx':

```
if ((ielem + 1) <= sat%nx) then
      if (sat%space_mask(ielem+1,iline) == sym%SPACE) then
            write (*,*) 'living on the edge'
      endif
endif
```

Accessing input data

Channel-specific input data (e.g. channel 2 reflectance) is only available if the channel has been specified in the XML algorithm description file. Similarly, availability of NWP and RTM data is controlled with the flags 'nwp_needed' and 'rtm_needed' in the XML description file. See Table 2 for more information.

The 'sat' structure contains navigation information, level 1b data, level 2 clouds and aerosol data, ancillary data, RTM top-of-atmosphere clear sky radiance and brightness temperature data at pixel resolution, information about the program invocation and the current segment, and general utility information. Descriptions of the variables in the 'sat' structure, including types, sizes and units, are in Table 5, Table 6, Table 7, Table 8, Table 9, Table 10 and Table 11. Variables with dimensions (Nx, Ny) are at pixel resolution, where 'Nx' is the number of elements in the segment and 'Ny' is the number of lines.

The 'nwp' structure contains NWP data from the input source specified (either GDAS or GFS at this point). The native spatial resolution is 0.5 or 1.0 degrees, and the time resolution is 6 hours. The vertical profile variables are interpolated to the standard 101 AIRS levels.

Descriptions of the variables in the 'nwp' structure are in Table 12 and Table 13. The 'dat' member is a 2-dimensional array at the NWP spatial resolution, where each element of the array is itself a structure containing NWP data. Conversion between pixel resolution and NWP resolution is accomplished by retrieving the NWP array indices for a given pixel from the 'sat' structure:

```
xnwp = sat%x_nwp(ielem,iline)
ynwp = sat%y_nwp(ielem,iline)
tpw = nwp%dat(xnwp, ynwp)%tpw
```

Some of the variables in the 'dat' structure are arrays with dimension 'Nprof', corresponding to the number of profiles, which is available in the variable 'nwp%nlevels'.

```
plev = nwp%dat(xnwp, ynwp)%plev(some_level)
```

In case you are wondering, the NWP structure is set up in this manner so that memory allocation for the 101-level profiles can be easily controlled.

The 'rtm' array contains RTM data at the NWP spatial resolution (generated by the default RTM solver, PFAST at this point). Conversion between pixel resolution and NWP resolution is accomplished as described in the 'nwp'

structure section. Each element of the 'rtm' array is a structure that contains an array named 'd'. Each element of 'd' corresponds to a different viewing zenith angle bin. The index into 'd' for a given pixel is obtained from the 'ivza' variable in the 'sat' structure as shown below. Each element of 'd' is a structure that contains RTM data, described in Table 14.

```
xnwp = sat%x_nwp(ielem,iline)
ynwp = sat%y_nwp(ielem,iline)
ivza = sat%ivza(ielem,iline)
bt_clr7 = rtm(xnwp,ynwp)%d(ivza)%bt_clr7
```

Some of the variables in the 'd' structure are arrays with dimension 'Nprof', corresponding to the number of profiles, which is available in the variable 'nwp%nlevels'. Note that the 'rtm' structure array cannot contain data without the 'nwp' structure containing data. Geocat will check to make sure that if the rtm_needed tag is set, the nwp_needed tag is also set. The 'nwp' structure can exist without the 'rtm' structure, however. An example reference to a rtm profile variable is shown below.

```
cld_prof10 = rtm(xnwp,ynwp)%d(ivza)%cld_prof10(some_level)
```

Note that some RTM data (top of atmosphere clear sky radiances and brightness temperatures) is available at pixel resolution in the 'sat' structure.

In addition to the structures described above, which contain primarily data, some other input structures are available to the algorithm modules:

The 'TIMEstr' structure contains the time of the current image (see Table 15).

The 'AREAstr' structure contains the AREA directory of the input file. See the definition of 'area_struct' in the source file 'area_read.f90'.

Accessing other Geocat data

The 'sym' structure contains symbols used throughout Geocat including mask codes and error codes. See the source file 'constant.f90' for a list of available symbols.

The variable 'scinfo' is an array of 'sc_params' structures containing information about the satellite platforms supported by Geocat. The index of the satellite platform of the current image is stored in the variable 'sc_ind'. Thus the 'sc_params' structure for the current image is accessed as 'scinfo(sc_ind)'. See the definition of 'sc_params' in the source file 'pixel_common.f90'.

Some **fundamental physical constants** are available to algorithms via Geocat. See the file 'fundamental_constants_geocat.f90' for their definitions.

Writing output data

Output is written to the Level 2 file via **the 'out2' array** of structures. Each element in the 'out2' array corresponds to a different algorithm. Access the correct array element with the variable 'ialgo':

```
out2(ialgo)%firemask(ielem, iline) = 60
```

Memory is allocated only for variables in the 'out2' structure that are specified in the XML algorithm description file (see Table 3). Therefore, any 'out2' variable that is used in an algorithm module must be specified in the XML file. If the tag 'produces_output' is included in the XML specification, Geocat will write the output to the Level 2 file. If the tag is not present, no output will be written, but the variable can still be used for scratch space.

See Table 16 and Table 17 for a description of the variables in the 'out2' structure. Most of the variables are product-specific, but there are also some generic variables available for scratch and debug output, with names like 'i2_generic1'.

Pointers and convenience aliasing

To simplify per-pixel data access syntax and potentially avoid bugs, you can set a pointer to point to the current array element, and thereafter use the pointer variable instead of indexing into the array. For example, declare the pointer like this:

```
INTEGER(kind=int1), pointer :: firemask
```

and set the pointer in the main pixel loop:

```
firemask => out2(ialgo)%firemask(ielem, iline)
```

Later you can read or write directly to the pointer variable:

```
firemask = 60
```

Utility functions and subroutines

Geocat utility functions and subroutines that are callable from the algorithm modules are listed in Table 18.

Building Geocat

If an algorithm description XML file has been modified since the last compile, the python script must be run to re-generate the algorithm interface source files and makefile, then the source must be recompiled. The arguments to the python script determine which algorithms are included in a given build. The calling syntax is:

```
python algomodgen.py directory_1 [directory_2……directory_N]
```

where each of the specified directories contains an XML file that describes the algorithms in that directory. The algorithm indices will be assigned in the order the algorithms are processed, i.e. in the order the directories are specified, and within each directory in the order algorithms are specified in the XML file.

To build Geocat, in the main Geocat source directory ('src'), type:

```
make target
```

where 'target' is one of the following:

intel32	32-bit Intel architectures, debug
intel32_opt	32-bit Intel architectures, optimized
intel	64-bit Intel architectures, debug
intel_opt	64-bit Intel architectures, optimized

It is recommended that the debug targets are used during algorithm development; optimized builds can be used later for data processing and performance testing.

Running Geocat

Command line options

Command line options control input and output files and locations, and program execution. To see a list of available command line options and their descriptions, run Geocat with the '-h' option:

```
./geocat –h
```

The file 'geocat.default', located in the Geocat directory, provides an alternate method of specifying file locations and some other options. The file can be edited directly, and will be read in by Geocat at runtime. Command line options take precedence over default values. The file 'geocat.default.template' contains a description of each line in the file 'geocat.default'.

Informational options

`-alist`
List information about each available algorithm and exit.

`-help`
`-h`
List all available command line options and exit.

`-pblock`
Output a summary of the directory, navigation, and calibration block and exit. Used with option '-f'.

`-scinfo`
Output information on available satellite platforms and exit.

`-version`
`–v`
Display the code version number and exit.

Input options

Geocat reads Level 0 input from band-separated AREA files, specified with the '-f' option and located in the directory specified by the '-area_dir' option. Input AREA files can optionally be compressed with gzip ('.gz' appended to filename). Geocat extracts the channel number from the filename, as determined by the '-aformat' specification.

Because navigation calculations are expensive, processing times can be substantially reduced when reprocessing the same input file or when working with geostationary satellite data by writing navigation data to file with the '-

dumpnav' option, and reading it in with the '-read_nav' option in subsequent program invocations. If navigation data is not available for the current Level 0 file, the '-read_nav' option will be ignored.

`-active_dir` *`directory`*
Specify the directory where all active sensor data are located (experimental).

`-aformat` *`char_occur1 char`*
Specify the location of the channel number in the AREA filename. By default the channel number is located between the second and third underscore (arguments '2 _') .

`-area_dir` *`directory`*
Specify the location of the input AREA files.

`-f` *`filename`*
Specify an input AREA file to process. Can be specified multiple times to process multiple files.

`-nav_dir` *`directory`*
Specify the directory where the navigation output file(s) are located.

`-nwp {gfs, ncep, ecmwf}`
Specify the NWP data source.

`-nwp_dir` *`directory`*
Specify the directory where the NWP file(s) are located.

`-read_nav`
Set this flag if imager navigation is to be read from file.

`-snow_dir` *`directory`*
Specify the directory where the 4-km snow map(s) are located.

Output options

Level 1, Level 2 and RTM output files are written to locations specified by the options below. Level 1 and RTM data will only be written if channels are specified with the '-dumpch' and '-dumprtm' options. Level 2 output is turned on and of with the '-l2' and '-nol2' options.

`-dumpch` *`channel [channel] ...`*
Specify which channels are to be dumped to a Level 1 file.

`-dumpnav`
Set this flag to create a navigation file.

`-dumprtm` *`channel [channel] ...`*
Specify which channels are to be dumped to a RTM file.

-l1_dir *directory*
Specify the directory where the Level 1 output file(s) are to be written.

-l2
Set this flag to create Level 2 data.

-l2_dir *directory*
Specify the directory where the Level 2 output file(s) are to be written.

-nav_dir *directory*
Specify the directory where the navigation output file(s) are located.

-nol2
Set this flag to not create Level 2 data.

-rtm {plod, crtm}
Specify the fast radiative transfer model to use.

-rtm_dir *directory*
Specify the directory where the RTM output file(s) are to be written.

Processing options

Specify the algorithms to run by index with the '-a' option. Algorithms will be processed in the order they are specified.

-a *alg_index [alg_index] ...*
Specify which algorithms are executed by index (see -alist to list algorithm indices). Indices are auto-assigned and likely to change across program builds. Consider using option -akey as a more robust alternative.

-akey *keyword [keyword] ...*
Specify which algorithms are executed by keyword (see -alist to list algorithm keywords).

-amask *keyword*
Specify the default aerosol mask algorithm, by keyword (experimental, see -alist to list algorithm keywords).

-cldz *keyword*
Specify the default cloud height algorithm, by keyword (see -alist to list algorithm keywords).

-cmask *keyword*
Specify the default cloud mask algorithm, by keyword (see -alist to list algorithm keywords).

-ctype *keyword*
Specify the default cloud type algorithm, by keyword (see -alist to list algorithm keywords).

-fast_planck
Set this flag to indicate Planck Function look-up tables are to be used when converting measured radiances to brightness temperature.

-maxsatzen *angle_in_degrees*
Set the maximum viewing angle to be considered valid. Pixels with a viewing angle greater than the value specified will not be calibrated and navigated by Geocat, and will be flagged as space pixels in the space mask.

-nscans *nscans_per_cycle*
Set the segment size (the number of scans to process per calling cycle).

-use_albedo
Set if the MODIS surface albedo database is to be used.

-use_seebor
Set if the Seebor surface emissivity database is to be used.

-use_snow
Set if the 4-km snow maps are available to be used.

-x *xstart xend xstride*
Specify the first and last elements to be processed along with the interval between values in the x or element dimension.

-y *xstart xend*
Specify the first and last lines to be processed in the y or line dimension.

Standard output options

-time_report
Output the total amount of time each algorithm took to process.

-verbose
Output detailed processing information to the standard output device.

APPENDIX

Figure 1: XML algorithm description file format

```
<geocat>
        <algorithms>
                <algorithm name="name" keyword="keyword" index="idx">
                        <comment value="fortran header comment" />
                        <module value="geocat_module_name" />
                        <function value="geocat_function_name" />
                        <reference value="reference" />
                        <ancil_subdir value="ancil_subdir" />

                        <inactive /> (optional)

                        <cmask_needed /> (optional)
                        <ctype_needed /> (optional)
                        <cldz_needed /> (optional)
                        <amask_needed /> (optional)
                        <nwp_needed /> (optional)
                        <rtm_needed /> (optional)
                        <active_needed /> (optional)

                        <sds_list>
                                <sds_item name="sds_name">
                                        <dim3_size value="dimen3_size" /> (optional)
                                        <stride_factor value="stride_factor" />
                                                                        (optional)
                                        <produces_output /> (optional)
                                </sds_item>
                        </sds_list>

                        <channels values="comma,separated,list,of,channels" />
                </algorithm>
        </algorithms>
</geocat>
```

Figure 2: XML algorithm description file -- an example algorithm element from the cloud mask algorithm

```
<algorithm name="Baseline SEVIRI Cloud Mask" keyword="baseline_cmask_seviri"
index="1">
        <comment value="Baseline GOES-R CAT ABI Cloud Mask for SEVIRI" />
        <module value="baseline_cloud_mask" />
        <function value="baseline_cloud_mask_main" />
        <reference value="GOES-R AWG Cloud Team" />
        <ancil_subdir value="baseline_cloud_mask" />
```

```
        <nwp_needed />
        <rtm_needed />

        <sds_list>
                <sds_item name="cloud_mask">
                        <produces_output />
                </sds_item>
                <sds_item name="cloud_mask_packed">
                        <dim3_size value="4" />
                        <produces_output />
                </sds_item>
                <sds_item name="quality_flags1">
                        <dim3_size value="24" />
                </sds_item>
                <sds_item name="r4_generic1">
                        <produces_output />
                </sds_item>
                <sds_item name="r4_generic2">
                        <produces_output />
                </sds_item>
                <sds_item name="i1_generic1">
                </sds_item>
        </sds_list>

        <channels values="2,7,14,15" />
</algorithm>
```

Table 4: Geocat / instrument channel mappings

GEOCAT Channel	GOES I-M Channel	GOES N-P Channel	MSG Channel	MTSAT Channel	ABI Channel
1					1
2	1	1	1	1	2
3			2		3
4					4
5			3		5
6					6
7	2	2	4	5	7
8			5		8
9	3	3		4	9
10			6		10
11			7		11
12			8		12
13					13
14	4	4	9	2	14
15	5		10	3	15
16		6	11		16

Table 5: ‘sat’ structure -- GEOCAT setup information

variable name	data type/size	units	description
ancil_path	string	none	The main ancillary data directory
iseg	Int32	none	The index of the current scan line segment (zero-based)
nscans_per_segment	Int32	none	The number of scan lines per segment
nseg	Int32	none	The total number of scan line segments

nx	Int32	none	The number of pixel elements currently in memory
nx0	Int32	none	The number of pixel elements prior to subsampling
ny	Int32	none	The number of scan lines currently in memory
ny_all	Int32	none	The total number of scan lines prior to parsing into scan line segments and prior to subsampling
ny0	Int32	none	The number of scan lines prior to subsampling. Not currently active.
sfc_albedo_source	Int8	none	The surface reflectance data source (see sym structure)
sfc_emiss_source	Int8	none	The surface emissivity data source (see sym structure)
snow_mask_source	Int8	none	The snow mask data source (see sym structure)
xstart	Int32	none	The starting pixel element index
xstride	Int32	none	The subsampling factor in the pixel element direction (1: every pixel, 2: every other pixel, etc…)
ystart	Int32	none	The starting scan line index
ystride	Int32	none	The subsampling factor in the scan line direction (1: every line, 2: every other line, etc…). Not currently active.

Table 6: 'sat' structure -- navigation-related data

variable name	data type/size	units	description
cos_satzen	Float32(Nx, Ny)	none	Cosine of the satellite zenith angle
cos_solzen	Float32(Nx, Ny)	none	Cosine of the solar zenith angle
glintzen	Float32(Nx, Ny)	degrees	Glint angle
lat	Float32(Nx, Ny)	degrees	Pixel latitude
lon	Float32(Nx, Ny)	degrees	Pixel longitude
relaz	Float32(Nx, Ny)	degrees	Relative azimuth
sataz	Float32(Nx, Ny)	degrees	Satellite azimuth
satzen	Float32(Nx, Ny)	degrees	Satellite zenith angle
scatzen	Float32(Nx, Ny)	degrees	Scattering angle
solaz	Float32(Nx, Ny)	degrees	Solar azimuth
solzen	Float32(Nx, Ny)	degrees	Solar zenith angle
sp_lat	Float32	degrees	Sub-satellite latitude
sp_lon	Float32	degrees	Sub-satellite longitude
space_mask	Int8(Nx, Ny)	none	Indicates whether a pixel is geolocated on the earth's surface (NO: pixel is geolocated, YES: pixel is not geolocated)
sun_earth_distance	Float32	AU	The sun earth distance
x_nwp	Int32(Nx, Ny)	none	The corresponding NWP grid cell index for each satellite pixel for the x-direction
y_nwp	Int32(Nx, Ny)	none	The corresponding NWP grid cell index for each satellite pixel for the y-direction

Table 7: 'sat' structure -- Level 1B calibrated & navigated data

variable name	data type/size	units	description
bad_pixel_mask	Int8(Nchan+1, Nx, Ny)	none	Basic indicator of spectral data quality (NO: data is within expected range, YES: data is outside of spectral range) for each possible spectral channel. The nchan+1 index is set to YES if any of the individual channels had out of range data values for a given pixel.
bt10	Float32(Nx, Ny)	K	Calibrated channel 10 brightness temperature
bt11	Float32(Nx, Ny)	K	Calibrated channel 11 brightness temperature
bt12	Float32(Nx, Ny)	K	Calibrated channel 12 brightness temperature
bt13	Float32(Nx, Ny)	K	Calibrated channel 13 brightness temperature
bt14	Float32(Nx, Ny)	K	Calibrated channel 14 brightness temperature
bt15	Float32(Nx, Ny)	K	Calibrated channel 15 brightness temperature
bt16	Float32(Nx, Ny)	K	Calibrated channel 16 brightness temperature
bt7	Float32(Nx, Ny)	K	Calibrated channel 7 brightness temperature
bt8	Float32(Nx, Ny)	K	Calibrated channel 8 brightness temperature
bt9	Float32(Nx, Ny)	K	Calibrated channel 9 brightness temperature
ems7	Float32(Nx, Ny)	none	Calibrated channel 7 pseudo-emissivity
rad10	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 10 radiance
rad11	Float32(Nx, Ny)	$mWm^{-2}sr^{-}$	Calibrated channel 11 radiance

		$^{1}(cm^{-1})^{-1}$	
rad12	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 12 radiance
rad13	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 13 radiance
rad14	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 14 radiance
rad15	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 15 radiance
rad16	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 16 radiance
rad7	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 7 radiance
rad8	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 8 radiance
rad9	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calibrated channel 9 radiance
ref1	Float32(Nx, Ny)	%	Calibrated channel 1 reflectance (solar zenith angle corrected)
ref2	Float32(Nx, Ny)	%	Calibrated channel 2 reflectance (solar zenith angle corrected)
ref3	Float32(Nx, Ny)	%	Calibrated channel 3 reflectance (solar zenith angle corrected)
ref4	Float32(Nx, Ny)	%	Calibrated channel 4 reflectance (solar zenith angle corrected)
ref5	Float32(Nx, Ny)	%	Calibrated channel 5 reflectance (solar zenith angle corrected)
ref6	Float32(Nx, Ny)	%	Calibrated channel 6 reflectance (solar zenith angle corrected)
ref7	Float32(Nx, Ny)	%	Calibrated channel 7 pseudo-reflectance (solar zenith angle corrected)

Table 8: ‘sat’ structure -- ancillary data

variable name	data type/size	units	description
coast_mask	Int8(Nx, Ny)	none	The pixel-level indicator of proximity to a coast line (see sym structure)
desert_mask	Int8(Nx, Ny)	none	The pixel level desert classification (see sym structure)
land_mask	Int8(Nx, Ny)	none	The pixel-level land/water mask (see sym structure)
sfc_emiss10	Float32(Nx, Ny)	none	Channel 10 surface emissivity
sfc_emiss11	Float32(Nx, Ny)	none	Channel 11 surface emissivity
sfc_emiss12	Float32(Nx, Ny)	none	Channel 12 surface emissivity
sfc_emiss13	Float32(Nx, Ny)	none	Channel 13 surface emissivity
sfc_emiss14	Float32(Nx, Ny)	none	Channel 14 surface emissivity
sfc_emiss15	Float32(Nx, Ny)	none	Channel 15 surface emissivity
sfc_emiss16	Float32(Nx, Ny)	none	Channel 16 surface emissivity
sfc_emiss7	Float32(Nx, Ny)	none	Channel 7 surface emissivity
sfc_emiss8	Float32(Nx, Ny)	none	Channel 8 surface emissivity
sfc_emiss9	Float32(Nx, Ny)	none	Channel 9 surface emissivity
sfc_type	Int8(Nx, Ny)	none	The pixel-level surface type classification (see sym structure)
snow_mask	Int8(Nx, Ny)	none	The pixel-level snow/ice mask (see sym structure)
sst_clim	Float32(Nx, Ny)	K	Monthly mean climatological SST
sst_clim_uni	Float32(Nx, Ny)	K	Monthly mean climatological ST uniformity parameter (e.g. spatial standard deviation)
volcano_mask	Int8(Nx, Ny)	none	The pixel-level indicator of proximity to a volcano (see sym structure)
zsfc	Float32(Nx, Ny)	meters	Surface elevation

Table 9: ‘sat’ structure -- RTM data at pixel resolution

variable name	data type/size	units	description
bt_clr10	Float32(Nx, Ny)	K	Calculated channel 10 clear sky TOA brightness temperature
bt_clr11	Float32(Nx, Ny)	K	Calculated channel 11 clear sky TOA brightness temperature

bt_clr12	Float32(Nx, Ny)	K	Calculated channel 12 clear sky TOA brightness temperature
bt_clr13	Float32(Nx, Ny)	K	Calculated channel 13 clear sky TOA brightness temperature
bt_clr14	Float32(Nx, Ny)	K	Calculated channel 14 clear sky TOA brightness temperature
bt_clr15	Float32(Nx, Ny)	K	Calculated channel 15 clear sky TOA brightness temperature
bt_clr16	Float32(Nx, Ny)	K	Calculated channel 16 clear sky TOA brightness temperature
bt_clr7	Float32(Nx, Ny)	K	Calculated channel 7 clear sky TOA brightness temperature
bt_clr8	Float32(Nx, Ny)	K	Calculated channel 8 clear sky TOA brightness temperature
bt_clr9	Float32(Nx, Ny)	K	Calculated channel 9 clear sky TOA brightness temperature
isfc	Int32(Nx, Ny)	none	The index in the atmospheric profile variables corresponding to the level that is closest to and above the surface
ivza	Int32(Nx, Ny)	none	The viewing zenith angle index associated with the clear sky radiance calculations
rad_clr10	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 10 clear sky TOA radiance
rad_clr11	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 11 clear sky TOA radiance
rad_clr12	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 12 clear sky TOA radiance
rad_clr13	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 13 clear sky TOA radiance
rad_clr14	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 14 clear sky TOA radiance
rad_clr15	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 15 clear sky TOA radiance
rad_clr16	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 16 clear sky TOA radiance
rad_clr7	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 7 clear sky TOA radiance
rad_clr8	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 8 clear sky TOA radiance
rad_clr9	Float32(Nx, Ny)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Calculated channel 9 clear sky TOA radiance

Table 10: 'sat' structure -- Level 2 clouds and aerosol data

variable name	data type/size	units	description
aeromask	Int8(Nx, Ny)	none	The unpacked results of the default aerosol mask (experimental)
cldmask	Int8(Nx, Ny)	none	The unpacked results of the default cloud mask
cldphase	Int8(Nx, Ny)	none	The unpacked results of the default cloud phase
cldtype	Int8(Nx, Ny)	none	The unpacked results of the default cloud type

Table 11: 'sat' structure -- general utility

variable name	data type/size	units	description
area_ch_dump_flg	Int8(Nchan)	none	Array indicating which channels are to be written to a Level 1 file
area_ch_flg	Int8(Nchan)	none	Array indicating whether a given channel is in memory
area_ch_rtm_flg	Int8(Nchan)	none	Array indicating which channels clear sky RTM calculations are to be performed for
area_rtm_dump_flg	Int8(Nchan)	none	Array indicating which channels are to be written to a RTM file
buf_bt	Float32(Nx, Ny)	variable	Calibration utility parameter
buf_ems	Float32(Nx, Ny)	variable	Calibration utility parameter
buf_rad	Float32(Nx, Ny)	variable	Calibration utility parameter
buf_ref	Float32(Nx, Ny)	variable	Calibration utility parameter
filename	string	none	The name of the file from which the navigation is read
idet	Int32	none	Radiometric detector index
line_prefix	Int16(variable size)	none	Utility pointer used for reading and calibrating data
machine_byte_ordering	Int32	none	LITTLE_ENDIAN or BIG_ENDIAN
nalgo	Int32	none	The total number of valid algorithms available in GEOCAT. This information is needed for output purposes so that attributes for non-algorithm output are assigned structures indices that do not interfere with algorithm output (e.g. sat%nalgo + 1, 2, 3, etc…).
scaled_int16	Int16(variable size)	none	Utility pointer used for reading and calibrating data
scaled_int8	Int8(variable size)	none	Utility pointer used for reading and calibrating data

swap_bytes	Int8	none	Swap integer bytes read from binary files (NO or YES)

Table 12: 'nwp' structure -- NWP data at NWP spatial resolution

variable name	data type/size	units	description
dat	profile_params(Nx_nwp, Ny_nwp)	none	Array of structures containing NWP data. See 'dat' element description.
dlat	Float32	degrees	latitude spacing
dlon	Float32	degrees	longitude spacing
first_lat	Float32	degrees	first latitude in the equal angle grid
first_lon	Float32	degrees	first longitude in the equal angle grid
nlat	Int32	none	number of latitudes
nlevels	Int32	none	number of vertical levels in the NWP data
nlevels_nointerp	Int32	none	original number of vertical levels in the NWP data before interpolation
nlon	Int32	none	number of longitudes
rtm_nvzen	Int32	none	number of viewing angle bins used in the RTM calculations

Table 13: 'nwp%dat' array element -- NWP data

variable name	data type/size	units	description
a	Float32	none	An interpolation weight utility parameter
flag	Int8	none	A flag indicating whether the profile for the current NWP grid cell has been interpolated to 101 levels
inversion_lev	Int32(Nprof)	none	Profile of flags that indicates whether a given tropospheric level is within a temperature inversion (NO or YES)
lat	Float32	degrees	Central latitude of NWP grid cell
lon	Float32	degrees	Central longitude of NWP grid cell
ninversion	Int32	none	The number of tropospheric temperature inversions in a given profile
o3col	Float32	DU	The total column ozone amount
o3lev	Float32(Nprof)	g/kg	The atmospheric ozone profile
plev	Float32(Nprof)	hPa	The atmospheric pressure profile
plev_nointerp	Float32(Nprof)	hPa	original pressure level representation
pmsl	Float32	hPa	The mean sea level pressure
psfc	Float32	hPa	The surface pressure
ptropo	Float32	hPa	The pressure at the tropopause
rh2m	Float32	%	The relative humidity at 2 meters
sfc_level	Int32	none	The profile index roughly corresponding to the lowest atmospheric level that is above the surface
snowsfc	Float32	cm	The liquid equivalent snow depth
strato_level	Int32	none	The profile index that best describes the level at which the stratosphere ends and the mesosphere begins
t2m	Float32	K	The temperature at 2 meters
tlev	Float32(Nprof)	K	The atmospheric temperature profile
tpw	Float32	cm	The total column precipitable water
tpwlev	Float32(Nprof)	g/cm^{-2}	The atmospheric precipitable water profile (e.g. integrated water path)
tropo_level	Int32	none	The profile index that best describes the level at which the troposphere ends and the stratosphere begins
tsfc	Float32	K	The surface temperature
tsfc_uni	Float32	K	The standard deviation of the surface temperature within a predefined spatial region that is used to help determine the proximity to coast lines and mountainous areas
ttropo	Float32	K	The temperature of the tropopause
ulev_nointerp	Float32(Nprof)	hPa	East / West component of the wind profile, uninterpolated in the vertical
vlev_nointerp	Float32(Nprof)	hPa	North / South component of the wind profile, uninterpolated in the vertical
wlev	Float32(Nprof)	g/kg	The atmospheric water vapor profile
zlev	Float32(Nprof)	m	The atmospheric height profile
zsfc	Float32	m	The surface height
ztropo	Float32	m	The height of the tropopause

Table 14: 'rtm%d' array element -- RTM data

variable name	data type/size	units	description
bt_clr10	Float32	K	Clear sky TOA brightness temperature for channel 10 at the NWP spatial resolution
bt_clr11	Float32	K	Clear sky TOA brightness temperature for channel 11 at the NWP spatial resolution
bt_clr12	Float32	K	Clear sky TOA brightness temperature for channel 12 at the NWP spatial resolution
bt_clr13	Float32	K	Clear sky TOA brightness temperature for channel 13 at the NWP spatial resolution
bt_clr14	Float32	K	Clear sky TOA brightness temperature for channel 14 at the NWP spatial resolution
bt_clr15	Float32	K	Clear sky TOA brightness temperature for channel 15 at the NWP spatial resolution
bt_clr16	Float32	K	Clear sky TOA brightness temperature for channel 16 at the NWP spatial resolution
bt_clr7	Float32	K	Clear sky TOA brightness temperature for channel 7 at the NWP spatial resolution
bt_clr8	Float32	K	Clear sky TOA brightness temperature for channel 8 at the NWP spatial resolution
bt_clr9	Float32	K	Clear sky TOA brightness temperature for channel 9 at the NWP spatial resolution
cloud_prof10	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 10
cloud_prof11	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 11
cloud_prof12	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 12
cloud_prof13	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 13
cloud_prof14	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 14
cloud_prof15	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 15
cloud_prof16	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 16
cloud_prof7	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 7
cloud_prof8	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 8
cloud_prof9	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Black cloud radiance profile for channel 9
flag	Int8	none	A flag indicating whether calculations for a given viewing zenith angle have been performed
rad_atm_clr10	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 10
rad_atm_clr11	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 11
rad_atm_clr12	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 12
rad_atm_clr13	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 13
rad_atm_clr14	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 14
rad_atm_clr15	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 15
rad_atm_clr16	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 16
rad_atm_clr7	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 7
rad_atm_clr8	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 8
rad_atm_clr9	Float32(Nprof)	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky atmospheric radiance profile for channel 9
rad_clr10	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 10 at the NWP spatial resolution
rad_clr11	Float32	$mWm^{-2}sr^{-}$	Clear sky TOA radiance for channel 11 at the NWP spatial

		$^{1}(cm^{-1})^{-1}$	resolution
rad_clr12	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 12 at the NWP spatial resolution
rad_clr13	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 13 at the NWP spatial resolution
rad_clr14	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 14 at the NWP spatial resolution
rad_clr15	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 15 at the NWP spatial resolution
rad_clr16	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 16 at the NWP spatial resolution
rad_clr7	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 7 at the NWP spatial resolution
rad_clr8	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 8 at the NWP spatial resolution
rad_clr9	Float32	$mWm^{-2}sr^{-1}(cm^{-1})^{-1}$	Clear sky TOA radiance for channel 9 at the NWP spatial resolution
rtm_util	Float32(Nprof)	variable	Generic rtm profile utility variable
satzen	Float32	degrees	The mid point of the viewing angle bin
trans_atm_clr10	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 10
trans_atm_clr11	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 11
trans_atm_clr12	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 12
trans_atm_clr13	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 13
trans_atm_clr14	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 14
trans_atm_clr15	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 15
trans_atm_clr16	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 16
trans_atm_clr7	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 7
trans_atm_clr8	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 8
trans_atm_clr9	Float32(Nprof)	none	Clear sky atmospheric transmittance profile for channel 9

Table 15: 'TIMEstr' structure -- time of the current image

variable name	data type/size	units	description
day	Int32	days	day of current image
hour	Int32	hours	hour of cuurent image
hour_frac	Float32	hours	fractional hour
ileap	Int32	none	0 = no leap year, 1 = leap year
jday	Int32	days	julian day of current image
minute	Int32	minutes	minute of current image
month	Int32	months	month of current image
second	Int32	seconds	second of current image
year	Int32	years	year of current image

Table 16: 'out2' array element -- algorithm output metadata

variable name	data type/size	units	description
ancil_subdir	string	none	The algorithm ancillary data directory
ch_flg	Int8(Nchan)	none	Flag array indicating which channels are available to current algorithm based on input to algorithm_mod.f90
total_processing_time	Float32	seconds	Counter to keep a running total of time elapsed in each algorithm module

Table 17: 'out2' array element -- algorithm output data variables

variable name	data type/size	units	description
aero_col	Float32(Nx, Ny)	$tons/m^2$	Total column aerosol loading
aerobotp	Float32(Nx, Ny)	hPa	Aerosol base pressure
aerobott	Float32(Nx, Ny)	K	Aerosol base temperature
aerobotz	Float32(Nx, Ny)	m	Aerosol base height
aerodeff	Float32(Nx, Ny)	microns	Aerosol effective particle diameter
aeroemiss	Float32(Nx, Ny)	none	Infrared 11-micron aerosol emissivity
aerofrac	Float32(Nx, Ny)	none	Aerosol fraction
aeromask	Int8(Nx, Ny)	none	Final result aerosol mask output

aeromask_packed	Int8(Nbytes, Nx, Ny)	none	Byte-packed aerosol mask output
aerop	Float32(Nx, Ny)	hPa	Aerosol top pressure
aeroreff	Float32(Nx, Ny)	microns	Aerosol effective particle radius
aerot	Float32(Nx, Ny)	K	Aerosol top temperature
aerothick	Float32(Nx, Ny)	m	Aerosol geometrical thickness
aeroz	Float32(Nx, Ny)	m	Aerosol top height
amv_p_high	Float32(Nx, Ny)	hPa	Atmospheric motion vector pressure for high layer
amv_p_low	Float32(Nx, Ny)	hPa	Atmospheric motion vector pressure for low layer
amv_p_mid	Float32(Nx, Ny)	hPa	Atmospheric motion vector pressure for middle layer
amv_t_high	Float32(Nx, Ny)	K	Atmospheric motion vector temperature for high layer
amv_t_low	Float32(Nx, Ny)	K	Atmospheric motion vector temperature for low layer
amv_t_mid	Float32(Nx, Ny)	K	Atmospheric motion vector temperature for middle layer
amv_u_high	Float32(Nx, Ny)	m/s	Atmospheric motion vector u-component for high layer
amv_u_low	Float32(Nx, Ny)	m/s	Atmospheric motion vector u-component for low layer
amv_u_mid	Float32(Nx, Ny)	m/s	Atmospheric motion vector u-component for middle layer
amv_v_high	Float32(Nx, Ny)	m/s	Atmospheric motion vector v-component for high layer
amv_v_low	Float32(Nx, Ny)	m/s	Atmospheric motion vector v-component for low layer
amv_v_mid	Float32(Nx, Ny)	m/s	Atmospheric motion vector v-component for middle layer
amv_z_high	Float32(Nx, Ny)	m	Atmospheric motion vector height for high layer
amv_z_low	Float32(Nx, Ny)	m	Atmospheric motion vector height for low layer
amv_z_mid	Float32(Nx, Ny)	m	Atmospheric motion vector height for middle layer
aod_ir	Float32(Nx, Ny)	none	Infrared 11-micron aerosol optical depth
aod_vis	Float32(Nx, Ny)	none	Visible aerosol optical depth
cape	Float32(Nx, Ny)	J/kg	Convective Available Potential Energy
cldbeta1011	Float32(Nx, Ny)	none	Cloud effective absorption ratio [10/11]
cldbeta1011_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [10/11]
cldbeta1011_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [10/11]
cldbeta1011_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [10/11]
cldbeta1112	Float32(Nx, Ny)	none	Cloud effective absorption ratio [12/11]
cldbeta1112_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [12/11]
cldbeta1112_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [12/11]
cldbeta1112_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [12/11]
cldbeta1113	Float32(Nx, Ny)	none	Cloud effective absorption ratio [13/11]
cldbeta1113_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [13/11]
cldbeta1113_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [13/11]
cldbeta1113_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [13/11]
cldbeta3911	Float32(Nx, Ny)	none	Cloud effective absorption ratio [3.9/11]
cldbeta3911_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [3.9/11]
cldbeta3911_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [3.9/11]
cldbeta3911_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [3.9/11]
cldbeta8511	Float32(Nx, Ny)	none	Cloud effective absorption ratio [8.5/11]
cldbeta8511_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [8.5/11]

cldbeta8511_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [8.5/11]
cldbeta8511_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [8.5/11]
cldbetaxxxx_high	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen high layer (independent of whether a cloud is present in that layer) [any channel/any channel]
cldbetaxxxx_low	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen low layer (independent of whether a cloud is present in that layer) [any channel/any channel]
cldbetaxxxx_mid	Float32(Nx, Ny)	none	Effective absorption ratio of a user chosen middle layer (independent of whether a cloud is present in that layer) [any channel/any channel]
cldbotp	Float32(Nx, Ny)	hPa	Cloud base pressure
cldbotp_high	Float32(Nx, Ny)	hPa	Cloud base pressure for the highest cloud layer
cldbotp_low	Float32(Nx, Ny)	hPa	Cloud base pressure for the second highest cloud layer
cldbott	Float32(Nx, Ny)	K	Cloud base temperature
cldbott_high	Float32(Nx, Ny)	K	Cloud base temperature for the highest cloud layer
cldbott_low	Float32(Nx, Ny)	K	Cloud base temperature for the second highest cloud layer
cldbotz	Float32(Nx, Ny)	m	Cloud base height
cldbotz_high	Float32(Nx, Ny)	m	Cloud base height for the highest cloud layer
cldbotz_low	Float32(Nx, Ny)	m	Cloud base height for the second highest cloud layer
clddeff	Float32(Nx, Ny)	microns	Cloud effective particle diameter
clddeff_high	Float32(Nx, Ny)	microns	Cloud effective particle diameter of the highest cloud layer
clddeff_low	Float32(Nx, Ny)	microns	Cloud effective particle diameter of the lowest cloud layer
cldemiss	Float32(Nx, Ny)	none	Infrared 11-micron cloud emissivity
cldemiss_high	Float32(Nx, Ny)	none	Infrared 11-micron emissivity of the highest cloud layer
cldemiss_low	Float32(Nx, Ny)	none	Infrared 11-micron emissivity of the lowest cloud layer
cldfrac	Float32(Nx, Ny)	none	Cloud fraction
cldiwc	Float32(Nx, Ny)	g/m^3	Cloud ice water content
cldiwc_high	Float32(Nx, Ny)	g/m^3	Cloud ice water content for the highest cloud layer
cldiwc_low	Float32(Nx, Ny)	g/m^3	Cloud ice water content for the second highest cloud layer
cldiwp	Float32(Nx, Ny)	g/m^2	Cloud ice water path
cldiwp_high	Float32(Nx, Ny)	g/m^2	Cloud ice water path for the highest cloud layer
cldiwp_low	Float32(Nx, Ny)	g/m^2	Cloud ice water path for the second highest cloud layer
cldlwc	Float32(Nx, Ny)	g/m^3	Cloud liquid water content
cldlwc_high	Float32(Nx, Ny)	g/m^3	Cloud liquid water content for the highest cloud layer
cldlwc_low	Float32(Nx, Ny)	g/m^3	Cloud liquid water content for the second highest cloud layer
cldlwp	Float32(Nx, Ny)	g/m^2	Cloud liquid water path
cldlwp_high	Float32(Nx, Ny)	g/m^2	Cloud liquid water path for the highest cloud layer
cldlwp_low	Float32(Nx, Ny)	g/m^2	Cloud liquid water path for the second highest cloud layer
cldmask	Int8(Nx, Ny)	none	Final result cloud mask output
cldmask_packed	Int8(Nbytes, Nx, Ny)	none	Byte-packed cloud mask output
cldp	Float32(Nx, Ny)	hPa	Cloud top pressure
cldp_high	Float32(Nx, Ny)	hPa	Cloud top pressure for the highest cloud layer
cldp_low	Float32(Nx, Ny)	hPa	Cloud top pressure for the second highest cloud layer
cldphase	Int8(Nx, Ny)	none	Final result cloud phase output
cldphase_high	Int8(Nx, Ny)	none	Final result cloud phase output for the highest cloud layer
cldphase_low	Int8(Nx, Ny)	none	Final result cloud phase output for the second highest cloud layer
cldphase_packed	Int8(Nbytes, Nx, Ny)	none	Byte-packed cloud phase output
cldreff	Float32(Nx, Ny)	microns	Cloud effective particle radius
cldreff_high	Float32(Nx, Ny)	microns	Cloud effective particle radius of the highest cloud layer
cldreff_low	Float32(Nx, Ny)	microns	Cloud effective particle radius of the lowest cloud layer
cldt	Float32(Nx, Ny)	K	Cloud top temperature
cldt_high	Float32(Nx, Ny)	K	Cloud top temperature for the highest cloud layer
cldt_low	Float32(Nx, Ny)	K	Cloud top temperature for the second highest cloud layer
cldthick	Float32(Nx, Ny)	m	Cloud geometrical thickness
cldthick_high	Float32(Nx, Ny)	m	Cloud geometrical thickness for the highest cloud layer
cldthick_low	Float32(Nx, Ny)	m	Cloud geometrical thickness for the second highest cloud layer
cldtype	Int8(Nx, Ny)	none	Final result cloud type output
cldtype_packed	Int8(Nbytes, Nx, Ny)	none	Byte-packed cloud type output
cldz	Float32(Nx, Ny)	m	Cloud top height
cldz_high	Float32(Nx, Ny)	m	Cloud top height for the highest cloud layer
cldz_low	Float32(Nx, Ny)	m	Cloud top height for the second highest cloud layer

cod_ir	Float32(Nx, Ny)	none	Infrared 11-micron cloud optical depth
cod_ir_high	Float32(Nx, Ny)	none	Infrared 11-micron cloud optical depth for the highest cloud layer
cod_ir_low	Float32(Nx, Ny)	none	Infrared 11-micron cloud optical depth for the second highest cloud layer
cod_vis	Float32(Nx, Ny)	none	Visible cloud optical depth
cod_vis_high	Float32(Nx, Ny)	none	Visible cloud optical depth for the highest cloud layer
cod_vis_low	Float32(Nx, Ny)	none	Visible cloud optical depth for the highest cloud layer
emiss_spectra_active	NA	NA	experimental
emiss11_high	Float32(Nx, Ny)	none	Infrared 11-micron emissivity of a user chosen high layer (independent of whether a cloud is present in that layer)
emiss11_low	Float32(Nx, Ny)	none	Infrared 11-micron emissivity of a user chosen low layer (independent of whether a cloud is present in that layer)
emiss11_mid	Float32(Nx, Ny)	none	Infrared 11-micron emissivity of a user chosen middle layer (independent of whether a cloud is present in that layer)
fire_radiative_power	Float32(Nx, Ny)	W	Fire radiative power
fire_size	Float32(Nx, Ny)	km2	Fire size
fire_temperature	Float32(Nx, Ny)	K	Fire temperature
fire_mask	Int8(Nx, Ny)	none	Final result fire mask output
fire_mask_packed	Int8(Nbytes, Nx, Ny)	none	Byte-packed fire mask output
i1_generic1	Int8(Nx, Ny)	variable	Int8 workspace
i1_generic2	Int8(Nx, Ny)	variable	Int8 workspace
i1_generic3	Int8(Nx, Ny)	variable	Int8 workspace
i1_generic4	Int8(Nx, Ny)	variable	Int8 workspace
i2_generic1	Int16(Nx, Ny)	variable	Int16 workspace
i2_generic2	Int16(Nx, Ny)	variable	Int16 workspace
i2_generic3	Int16(Nx, Ny)	variable	Int16 workspace
i2_generic4	Int16(Nx, Ny)	variable	Int16 workspace
i4_generic1	Int32(Nx, Ny)	variable	Int32 workspace
i4_generic2	Int32(Nx, Ny)	variable	Int32 workspace
i4_generic3	Int32(Nx, Ny)	variable	Int32 workspace
i4_generic4	Int32(Nx, Ny)	variable	Int32 workspace
inversion_dp	Float32(Nx, Ny)	hPa	Temperature inversion pressure depth
inversion_dz	Float32(Nx, Ny)	m	Temperature inversion geometrical depth
inversion_p	Float32(Nx, Ny)	hPa	Temperature inversion pressure
inversion_z	Float32(Nx, Ny)	m	Temperature inversion height
ist	Float32(Nx, Ny)	K	Ice surface temperature
landmask	Int8(Nx, Ny)	none	Final result land mask output
li	Float32(Nx, Ny)	none	Lifted Index
lst	Float32(Nx, Ny)	K	Land surface temperature
ndvi	Float32(Nx, Ny)	none	Normalized vegetation index
o3_col	Float32(Nx, Ny)	DU	Total column ozone loading
o3prof	Float32(Nprof, Nx, Ny)	g/kg	Atmospheric ozone profile
pprof	Float32(Nprof, Nx, Ny)	hPa	Atmospheric pressure profile
pprof_active	NA	NA	experimental
pprof_single	NA	NA	experimental
qcflg1	Int8(Nbytes, Nx, Ny)	none	Generic quality flag array
qcflg2	Int8(Nbytes, Nx, Ny)	none	Generic quality flag array
qcflg3	Int8(Nx, Ny, Nbytes)	none	Generic quality flag array
qcflg4	Int8(Nx, Ny, Nbytes)	none	Generic quality flag array
r4_generic1	Float32(Nx, Ny)	variable	Float32 workspace
r4_generic2	Float32(Nx, Ny)	variable	Float32 workspace
r4_generic3	Float32(Nx, Ny)	variable	Float32 workspace
r4_generic4	Float32(Nx, Ny)	variable	Float32 workspace
r8_generic1	Float64(Nx, Ny)	variable	Float64 workspace
r8_generic2	Float64(Nx, Ny)	variable	Float64 workspace
r8_generic3	Float64(Nx, Ny)	variable	Float64 workspace
r8_generic4	Float64(Nx, Ny)	variable	Float64 workspace
sfc_index_active	NA	NA	experimental
sfc_mask_active	NA	NA	experimental
so2_col	Float32(Nx, Ny)	DU	Total column SO_2 loading
sst	Float32(Nx, Ny)	K	Sea surface temperature
tprof	Float32(Nprof, Nx, Ny)	K	Atmospheric temperature profile
tprof_active	NA	NA	experimental
tpw	Float32(Nx, Ny)	cm	Total precipitable water vapor
tpw_high	Float32(Nx, Ny)	cm	Total precipitable water vapor of the high layer
tpw_low	Float32(Nx, Ny)	cm	Total precipitable water vapor of the low layer

tpw_mid	Float32(Nx, Ny)	cm	Total precipitable water vapor of the middle layer
tropo_index_active	NA	NA	experimental
tt	Float32(Nx, Ny)	none	Totals Totals Index
uprof	Float32(Nprof, Nx, Ny)	m/s	Atmospheric zonal wind profile
vprof	Float32(Nprof, Nx, Ny)	m/s	Atmospheric meridional wind profile
wprof	Float32(Nprof, Nx, Ny)	g/kg	Atmospheric water vapor profile
wprof_active	NA	NA	experimental
zprof	Float32(Nprof, Nx, Ny)	m	Atmospheric height profile
zprof_active	NA	NA	experimental
zsfc_active	NA	NA	experimental

Table 18: utility functions and subroutines available to algorithm modules

utility name	**location**	**description**
call_planck_rad_func	fct_pointers.c	Call the c-language planck function brightness temperature to radiance conversion routine
call_planck_temp_func	fct_pointers.c	Call the c-language planck function radiance to brightness temperature conversion routine
compute_day	num_mod.f90	Computes a calendar day given a julian day and a leap year flag
compute_month	num_mod.f90	Determines month given a julian day and leap year flag
compute_spatial_uniformity	num_mod.f90	Computes the mean, max, min, and standard deviation for each pixel of a pixel array given the number of surrounding pixels to include in the calculation as input
destroy_spatial_uniformity	num_mod.f90	This utility destroys the output arrays created by compute_spatial_uniformity
display_message	Message_Handler_geocat.f90	Display a message to screen
find_bounds	num_mod.f90	Determines the bounds of an array of pixels in lat/lon coordinates
gradient2d	num_mod.f90	Determines the direction of the gradient vector for each pixel given a 2D array of pixels
gradient2d_reverse	num_mod.f90	Determines the direction of the negative gradient vector for each pixel given a 2D array of pixels
icnvrt	num_mod.f90	Converts integers to characters and vice-versa
invert_2x2	num_mod.f90	Matrix inversion for a 2 x 2 matrix
invert_3x3	num_mod.f90	Matrix inversion for a 3 x 3 matrix
julian	num_mod.f90	Calculate the julian day given mm/dd/yyyy information
leap_year_fct	num_mod.f90	Determines if a given year is a leap year
load_temporal_data	temporal_utils.f90	This interface is used to quickly load L1, L2, or RTM data from previous or “future” time steps into memory
locate	num_mod.f90	Numerical recipes bisection search
median_filter	num_mod.f90	Filters an array of pixels using a median filter
pack_bytes	num_mod.f90	Takes an array of bytes and packs them into an output byte array according to the bit depth of the individual input bytes
planck_rad_fast	planck.f90	A fast look-up table based brightness temperature to radiance conversion routine
planck_temp_fast	planck.f90	A fast look-up table based radiance to brightness temperature conversion routine