CERES Software Bulletin 95-05

CERES Release 1 Data Product Module Guidelines, June 28, 1995

Erika Geier (e.b.geier@larc.nasa.gov)

Purpose:

There seems to be a great deal of confusion about the requirements, suggestions, and preferences for data product modules. Since this issue seems to be reemerging and several people have asked about guidelines, here is an attempt to put this issue to bed.

Introduction:

It is understood that in some cases these guidelines are late in coming and might require some software to be reworked. It is hoped that all data product modules, at a minimum, bear a family resemblance.

Data product module use should be two-fold. They should be used in the data processing system and also in the validation or visualization software needed to support Release 1. It is still unclear if this should be accomplished by maintaining two versions of the module, or two versions of the toolkit wrappers (one which uses toolkit and the other which uses F90).

What follows are three lists. The first list is one of requirements which all archival data product modules must meet. The second list contains recommendations which are highly recommended. The third and final list contains some ideas about what other things one might wish to include in such a module.

Module Requirements:

- All data product modules must contain a type definition of the data product.
- All data product modules must contain subroutines which open, close, read, and write the respective data product.
- All data product modules must contain formatted print routine(s).

Recommendations:

- Open Subroutine

  The open subroutine should be of the form:

  dp_open(filename, access, unit_number, open_success, additional parameters)

  where:

  dp is the name of the data product - SSF, CRS, TSI, SRBavg, MOA, etc.

  filename is the input filename or structure containing equivalent toolkit information

  access is an input read or write access flag. It is preferred you pass in and check for CERESlib read and write character parameter constants here (read_file and write_file)

  unit_number is an output integer unit number associated with the file

open_success is a success output. It is preferred you pass in and check for
CERESlib success integer parameter constants here (ok and bad)
additional parameters are whatever else you need

- Close Subroutine
  The close subroutine should be of the form:
  dp_close(unit_number, close_success, any additional parameters)
  where:
  dp is the name of the data product - SSF, CRS, TSI, SRBavg, MOA, et
  unit_number is an input integer unit number associated with the file to be closed
  close_success is a success output. It is preferred you pass in and check for
  CERESlib success integer parameter constants here (ok and bad)
  additional parameters are whatever else you need

- Read Subroutine
  The read subroutine should be of the form:
  dp_read(unit_number, data, read_success, any additional parameters)
  where:
  dp is the name of the data product - SSF, CRS, TSI, SRBavg, MOA, etc.
  unit_number is an input integer unit number associated with the file to be read
  data is one instance of the data product (if direct access, data is one record)
  read_success is a success output. It is preferred you pass in and check for
  CERESlib success integer parameter constants here (ok, eof, and bad)
  any additional parameters are whatever else you need

- Write Subroutine
  The write subroutine should be of the form:
  dp_write(unit_number, data, write_status, any additional parameters)
  where:
  dp is the name of the data product - SSF, CRS, TSI, SRBavg, MOA, etc.
  unit_number is an input integer unit number associated with the file to be written
  data is one instance of the data product (if direct access, data is one record)
  write_success is a success output. It is preferred you pass in and check for
  CERESlib success integer parameter constants here (ok and bad)
  any additional parameters are whatever else you need

- Headers
  Some subsystems require the header information to process the data. Therefore, subsystems should allow space on their Release1 data product for any header information they or the subsystem following them might need for the processing of Release 1 data. A minimal subset of metadata should be defined and written into the header for each subsystem. Header reads should be an optional part of the open subroutine. Similarly, header writes should be an optional part of the close subroutine.

- Modules which fall under the same science team/subsystem or have very close relationships with other modules should bear close resemblances to one another. For example: SSF and

CRS both operate on the footprint level and should be very similar. All TISA modules should, likewise, be very similar.

Options:

- Ranges/Bounds
  Upper and lower bounds may be specified within the module.
  Bound checking subroutines may be included in module.

- Any definitions that are required for the module to run or aid in understanding or defining the data product as a whole. Do not include definitions that are specific to the way one subsystem operates and do not lend themselves to anything else.

- Any functions or subroutines that perform general functions which may be of interest to data product readers or writers. Do not include functions or subroutines which are specific to the operation of one particular set of production code.