The ODS Library Documentation
Version 2.16

Christopher Redder and Arlindo da Silva
Data Assimilation Office, NASA/GSFC, Greenbelt, MD 20771

February 2001

Contents

1 Package Overview

This document describes the Observation Data Stream (ODS) format used for pre- and post-analysis station data (non-gridded). These HDF compliant files are intended to serve as input as well as output for the Goddard EOS Data Assimilation System (GEOS/DAS). This document describes the concept of an ODS file and the application program interface (API) software. It is geared to the producers and users of such data sets.

This document deals with the Observation Data Stream (ODS) concept and its FORTRAN implementation using MFHDF, the NetCDF interface to the Hierarchical Data Format (HDF[1]) developed at the National Center for Supercomputer Applications (NCSA). No prior exposure to NetCDF or HDF is assumed, but familiarity with the general structure of Goddard EOS Data Assimilation System (GEOS/DAS) would be helpful.

1.1 Design Considerations

Originally, all observational data received at DAO were first archived with a consistent home-grown format known as UNPACK. UNPACK files held all observational data received, including data that is never included in the assimilation process. In preparation for assimilation, a pre-processing system referred to as REPACK selects data from UNPACK files, performs a few simple quality checks and corrections (such as the hydrostatic check), producing as output a REPACK file which is read by GEOS/DAS. REPACK files are usually much smaller than UNPACK files as they are tailored for the needs of the assimilation system. Details of the UNPACK/REPACK systems and file formats can be found in Lamich *et al.* (1996).

Useful by-products of the assimilation system were the innovation files (known internally as *del-files*) which contain the difference between the observations and the first guess (a 6 hour forecast) interpolated to the observation location, along with the quality control marks assigned by GEOS/DAS. These innovation files were the primary input for our covariance modeling effort, as well as for the monitoring and validation systems that were in development. The main disadvantages of the *del-files* were: 1) the actual observation values are not stored in the file, only the difference from the first guess, 2) the analyzed value interpolated to observation location is not present, and 3) the file structure is designed around the original upper air analysis system obtained from NCAR in the early 1980's and does not provide a natural framework for the wealth of new data types coming with the EOS era. Resolving these deficiencies would be helpful to DAO staff which rely on the observation value and the interpolated analysis as well as the innovation.

The main design goals of ODS are therefore:

1. To unify the REPACK, innovation files and other type of files with observation data into a single consistent format. (ODS files are not meant to be a replacement for UNPACK files.) This unification is especially helpful if observation data sets originate from different operation weather centers such as NCEP and ECMWP. Using HDF also ensures portability of the software and data files between a diverse set of hardware platforms.

[1]The conventions adopted in ODS do not strictly adhere to the EOSDIS standard for HDF files.

2. To develop simple data structures, with easy to use access routines available for many purposes. The access routines should be callable from commercial application software such as Matlab and IDL.

3. To make ODS files sufficiently self-describing and provide several additional pieces of information about the observation such as sounding identifiers and time stamp.

4. To accommodate not only data to be used directly by the GEOS/DAS but also *passive* data types which are useful for validation, tuning or diagnostic studies. This setup would allow bias corrections and random error characterization to be developed off-line for new data types before being used directly in the GEOS/DAS. Typical examples of *passive* data types are significant level data which are currently not used by the GEOS/DAS but provide an excellent validation tool.

5. To enable the user to acquire additional information required for each data type. With the increasing sophistication of the assimilation methodology, additional information (*meta-data*) about the data type will undoubtedly be required. Because this *meta-data* can vary widely with data type and is of concern only for the assimilation specialist, this data should be in a separate file. ODS files should then include only a pointer to this file. Guidelines for the preparation of meta-data files will be presented elsewhere.

6. To vary the ODS file format between the types termed *pre-analysis* and *post-analysis*. A *pre-analysis* file should be produced by the pre-processing system and does not contain those attributes which can only be provided by the assimilation system (e.g., the 6 hour forecast, the interpolated analysis value). A *post-analysis* should be produced by GEOS/DAS, having the same information in the pre-analysis ODS files complemented by the assimilation specific information. Nevertheless, a post-analysis ODS file should also be a valid input for GEOS/DAS as it provides the same information as the pre-analysis ODS file. As pointed out by David Lamich, GEOS/DAS could be configured to read data from a post-analysis ODS file and perform an assimilation using the same quality-control marks of the original assimilation. This capability is extremely useful to isolate the effects of the quality control subsystem when testing the impact of modifications to the system.

The manual provides a description of how the ODS concept and design is implemented. The remainder of this section describes the data stored in each ODS file and the access software. Section 2 discusses the installation, usage and testing. Appendix A gives an overview of the file structure, and Appendix B contain the prologues and thus the details of the interfaces of each callable ODS routine.

1.2 Header Information

Every ODS file contains header information about the observation reports and the file itself. Much of the information names the observation attributes, sets the valid ranges, and provides the units. Some of the information is used internally by the ODS library to locate, check and process the data. Most of these parameters are given in Sections 1.3 and 1.5.5. In the rest of this section, two types of header information is discussed: global attributes and lists of valid values of certain observation attributes.

1.2.1 Global attributes

Global attributes are primarily ODS file parameters and do not directly provide information about the observation reports. Table 1 contains the list of global attributes. Except for the *type* and *history*, all attributes are hardwired to the settings as indicated in Table 1.

Table 1: The global attributes

Global Attr Name	*Description*	*Present Setting*
source	Source of ODS file	Data Assimilation Office, Code 910.3, NASA/GSFC
title	Title of ODS file	GEOS DAS Observational Data Stream (ODS) File
type	ODS file type	
version	ODS version tag	2.16
data_info	Point of contact	Contact data@dao.gsfc.nasa.gov
history	History tag of ODS data	

The global attributes listed in Table 1 and defined by the application program are as follows:

type The ODS file type which is currently either pre-analysis or post-analysis and is set when the ODS file is created by the routine, `ODS_Create`, as discussed in Section 1.5.3.

history The history tag which is designed to provide information about the changes to the ODS file. This attribute can be updated when the file is closed by the routine `ODS_Close` as discussed in Section 1.5.3.

1.2.2 Lists

These lists contain information about the valid indices for the observation attributes, data type index (kt), data source index (kx), and the quality control history flag ($qcexcl$) which are described in Section 1.3.

Table 2: The set of lists containing information about the valid indices for the observation attributes, data type index (kt), data source index (kx), and the quality control history flag ($qcexcl$) which are described in Section 1.3. The first column provides the list names that are valid as input parameters to the routine, `ODS_GetList` (See Table 9). The second column contain the dimension names for input to the ODS query routine, `ODS_IGet` as (See Table 9 and discussed in Section 1.5.5.)

List Name	*Name of List Size*	*Description*	*For Sample List...*
kt_names	*nkt*	Names of GEOS/DAS data types	See Table 4
kt_units	*nkt*	Units of GEOS/DAS data types	See Table 4
kx_names	*nkx*	Names of GEOS/DAS data sources	See Table 5
kx_meta	*nkx*	kx specific meta-data information	See Table 5
qcx_names	*nqcx*	Meaning of each possible value of *qcexcl*	See Table 6

1.3 Observation attributes

Most of an ODS file consists of observation reports and is arranged in segments each of which contains all of the reports for a given synoptic data and time. Within each report, the observed value must be accompanied by a set of attributes describing the measurement[2] A list of these attributes is given in Table 3. All attributes listed in Table 3 must be present in any ODS file except for the attributes, omf, *oma*, and $xvec$, which apply only to post-analysis files. All attributes must be assigned; *missing values* are not allowed in an ODS file except for the attributes, xm, obs, omf, *oma* and $xvec$. (see Table 3). Any missing attributes should be set to 10^{15} (DAO standard).

Table 3: List of observation attributes. The FORTRAN interface refers to how this attribute is written/read to file using the software described in Sections 1.5.3 and 1.5.4

Attr. Name	*Description*	*FORTRAN Interface*	*Units*	*Valid Range*	*Storage (bytes)*
lat	Latitude	Real	degrees north	$[-90, +90]$	2
lon	Longitude	Real	degrees east	$[-180, +180]$	2
lev	Observation level or channel	Real	hPa,m or none		4
$time$	minutes since first julian day	Integer	minutes	$[-180, 65354]$	2
kt	Data type index	Integer		$[1, 255]$	1
kx	Data source index	Integer		$[1, 65535]$	2
ks	Sounding index	Integer		$[1, 2^{30}]$	4
xm	Meta-data	Real			4
$qcexcl$	Quality control exclusion flag	Integer		$[0, 255]$	1
$qchist$	Quality control history mark	Integer		$[0, 65534]$	2
obs	Observation value	Real	depends on kt		4
omf	Observation minus forecast	Real	depends on kt		4
oma	Observation minus analysis	Real	depends on kt		4
$xvec$	PSAS CG solution vector	Real	depends on kt		4

Each observation attribute is further described below.

1.3.1 Latitude (lat)

The latitude of the observation, in degrees, negative in the southern hemisphere, positive in the northern hemisphere. The latitude is encoded internally with 2 bytes at a resolution of 0.01°.

1.3.2 Longitude (lon)

The longitude of the observation, in degrees, from 180W to 180E. West longitudes should be encoded as negative numbers, *e.g.*, 90W = -90, 90E = +90. The longitude is encoded

[2]Technically some of these attributes should be referred to as *meta-data*. However, the term *meta-data* shall be defined in this document for additional information not present in the ODS files.

internally with 2 bytes at a resolution of 0.01°.

1.3.3 Level (*lev*)

In the case of conventional observations such as temperature soundings, this attribute refers to the vertical level, say 500 hPa. The concept of level is extended here to include the channels of radiance measurements. To allow greater flexibility, the level is recorded internally as a 4 byte float. For surface observations, including ships, buoys and other sea platforms, *lev* should be set to the height of the measuring instrument above the surface. For GEOS versions 3 and prior, *lev* has been set to 2000.0 for surface observations.

1.3.4 Sampling time (*time*)

This attribute gives the elapsed time in minutes since 0 GMT of the first day on file. It should refer to the actual time of measurement and not to the synoptic time. Negative numbers should indicate the time before 0 GMT of the first day on file. Julian days extend from midnight to midnight GMT rather than from noon to noon where julian day 2,440,000 is May 23, 1968. To convert from time since the first synoptic data and on file to calendar data and time (and vice-versa), use the ODS utility routines `ODS_Time2Cal` and `ODS_Cal2Time` (See Table 9). Also note in Table 3 that *time* is stored as a two byte integer and that the maximum value is 65354. As a result, the latest synoptic date can only be 45 days (i.e. 64800 minutes) after the first date on file before the sampling time exceeds the maximum allowed. In the f90 interface routines, `ODS_Put` and `ODS_Get`, the time is automatically converted to the number of minutes since the corresponding synoptic date and time.

1.3.5 Data type index (*kt*)

This index identifies the observable. For example, an observation of sea level pressure has $kt = 3$. A partial list of data types currently in use at DAO is given in Table 4. By retrieving the data type information from each ODS file the user will be able to identify each data type and its units.

1.3.6 Data source index (*kx*)

This index is an integer which identifies the origin of the measurement (see Table 5). For example, a measurement with $kx = 7$ was made by a radiosonde. The observation attribute *xm* (meta-data index, described below) can be used to identify the particular station that provided the report.

1.3.7 Sounding index (*ks*)

This index is needed to allow the assembly of observations coming from the same sounding during a given synoptic time. For example, a radiosonde reporting profiles of winds, temperature and humidity should have the same sounding index associated with each piece

Table 4: Sample list of data types used in GEOS/DAS (*kt* index). A current version of this list must be included in each ODS file for completeness.

kt	*Variable*	*Units*	*Description*
1	u_s	m/sec	Surface zonal wind
2	v_s	m/sec	Surface meridional wind
3	slp	hPa	Sea level pressure
4	u	m/sec	Upper-air zonal wind
5	v	m/sec	Upper-air meridional wind
6	h	m	Upper-air geopotential height
7	w	g/kg	Upper-air water vapor mixing ratio
8	T	Kelvin	Upper-air temperature
9	T_d	Kelvin	Upper-air dew-point temperature
10	rh	%	Upper-air relative humidity
	etc		

of data. In the case of satellite retrievals, each sounding should have their own index ks, although the retrievals originates from the same *instrument*. The concept of sounding is extended here to include also radiance measurements: satellite measured radiances at several channels (loosely equivalent to vertical levels) for a given horizontal location. Notice that sounding indices only refer to a given synoptic time. For example, if at 0Z the sounding from station LAX was assigned the index 5217, at the next synoptic time a sounding from the same station could have an index of 27. The use of ks can unify single level observation reports from such sources as aircraft which measure several quantities including pressure, temperature and wind speed and direction. As currently implemented, there is a maximum of 2^{30} (= 1,073,741,824) soundings allowed per synoptic time.

1.3.8 Meta-data index (xm)

This index is intended to point to meta-data information stored in a separate file tailored to the particular data source. For example, one could maintain a radiosonde master file which tabulates instrumentation, black-list and bias correction information for each reporting radiosonde. In this case, the meta-data information could point to the record in this master file corresponding to the reporting station. In the case of satellite data, the meta-data index could point to a record on a separate meta-data file which contains the state dependent error covariance for a given sounding. As for sounding indices, meta-data indices are not necessarily tied to station ID's. The meta-data index, xm, could also be used to store an attribute not supported by ODS such as the bias correction applied to an observation.

1.3.9 Quality control exclusion flag ($qcexcl$)

The exclusion mark is an integer code indicating whether an observation is to be excluded from the analysis, and, if so, for what reason. Only data with $qcexcl = 0$ are included in the analysis, those with $qcexcl > 0$ are excluded. A description of each exclusion code is contained in Table 6 but note that this list is subject to change. The current list of flags is

Table 5: Partial list of data sources used in GEOS/DAS (kx index). A current version of this list must be included in each ODS file for completeness.

kx	*Description*	*Meta-data*
1	Surface Land Obs - 1	
2	Surface Land Obs - 2	
3	Surface Ship Obs - 1	
4	Surface Ship Obs - 2	
5	Environment Buoy	
6	Drifting Buoy	
7	Rawinsonde	QCHum
8	Pilot Wind	
9	Ship Released Rawinsonde	
10	Dropwinsonde	
	etc.	

defined in the file, `m_odsmeta.f90`.

1.3.10 Quality control history mark (*qchist*)

The history mark is used to encode the history of the quality control process for each observation. See Table 7 for a list of history marks that are currently used, but note that this list is subject to change. In the current implementation, the history mark is an integer code, but in future it will most likely involve bitwise encoding. For this reason, the history mark names are not included in the ODS file header. They will be separately documented.

1.3.11 Observation (*obs*)

This attribute contains the actual value of the observation. As defined here, the u and v components of a wind field constitute two different observations. Also, a temperature profile at 18 mandatory levels is composed of 18 separate observations. The indices (kt,ks) can be used to group the data as profiles of a given quantity. Units are specified in the kt table such as the one given in Table 4.

1.3.12 Observation minus forecast (omf)

This value is the difference between the observation and the forecast interpolated to the sampling location. Units are specified in the kt table such as the one given in Table 4.

1.3.13 Observation minus analysis (oma)

This value is the difference between the observation and the analysis interpolated to the sampling location. Units are specified in the kt table such as the one given in Table 4.

Table 6: A samples list of quality control exclusion flags (*qcexcl*) used in GEOS/DAS.

qcexcl	*Description*
0	clear
1	unspecified preprocessing flag
2	impossible location
3	gcm deep underground
4	observation value undefined
5	forecast value undefined
6	observation level too high
7	passive data type
8	outside active time window
9	not an analysis variable
17	failed buddy check
18	incomplete wind vector
20	incomplete vertical profile
21	extreme outlier wrt background
22	no relative humidity information
23	extreme rh outlier wrt backgrd
24	extreme rhTf outlier wrt backgrd
25	failed rh buddy check
26	failed rhTf buddy check
27	invalid saturation mixing ratio

1.3.14 PSAS CG solution vector (*xvec*)

This value is the conjugate gradient (CG) solution vector returned from the Physical-Space Statistical Analysis System (PSAS) in the GEOS/DAS. Units are specified in the *kt* table such as the one given in Table 4 and are the same as that for the observation attribute omf. This attribute was added for Version 2.16.

1.4 ODS File Types

Each ODS file is either one of two type: pre-analysis or post-analysis. As discussed in Section 1.1, a pre-analysis file is produced by the pre-processing system and provides input to the analysis module. The post-analysis file is an extension to its pre-analysis counterpart and contains additional information from the analysis subsystem.

1.4.1 Content

For every observation report, a pre-analysis contains information for all attributes listed in Table 3 except omf (observation minus forecast), *oma* (observation minus analysis) and *xvec* (PSAS CG solution vector). The post-analysis contains all of the observation attributes. However, in no circumstance should the application program create a post-analysis ODS file and modify the the original observation attributes from the pre-analysis

Table 7: List of quality control history flags (*qchist*) used in GEOS/DAS. The current version of this list is subject to change.

qchist	*Description*
0	clear
1	unspecified preprocessing flag
2	OBSOLETE
3	gcm slightly underground
17	outlier wrt background
18	rh outlier wrt background
19	rhTf outlier wrt background

file (except possibly *xm*, *qcexcl* and *qchist*). If super-obing or other device is performed by the assimilation system, a new observation should be created, and the original observation should receive an appropriate quality control exclusion mark.

1.4.2 Storage requirements

Excluding overhead, a pre-analysis ODS file requires 28 bytes of storage for each observation as can be determined from Table 3. Therefore, a pre-analysis ODS file with an average of 200,000 observations for each of the 4 synoptic times will require $\approx$ 22.5 Mbytes of storage/day. A post-analysis file requires 40 bytes of storage for each observation so that, given the same parameters, a post-analysis ODS file will require $\approx$ 32 Mbytes of storage/day. However, the file for both types can be compressed upto 80 percent by applying the `gzip` utility.

1.5 Overview of the application program interface (API)

1.5.1 FORTRAN 90 interface

As will be evident in the next subsection, dozens of calls to the FORTRAN 77 interface routines may be required to read the entire data set for a synoptic date and time. Error handling can actually complicate the code even further. To remedy this problem, the FORTRAN 90 interface library was developed. The routines in this library have simple interfaces and can read and/or write all the data for a single synoptic time with few routine calls. The library routines call the FORTRAN 77 interface routines to open or create files, extract the file parameters (e.g. file type, number of observation reports), extract the observation data and then close the file. Furthermore, the routines perform system related tasks including memory allocation for space and exception handling for errors. Additional utility routines and modules exist to allocate memory space for input arguments for the library routines, define useful constants, print a summary to standard output and complete other useful tasks. Table 8 lists each routine, and Appendix B describes the input/output parameters in some detail.

The observation data is passed into and out of these routines via the FORTRAN 90 data structure (or record), `ods_vect`, as defined in the module `m_ods` (Section B.1). This struc-

Table 8: User callable FORTRAN 90 subroutines to read/write ODS files. For a complete description of the input/output parameters, see the appropriate section given between the parenthesis next to the routine name in the first column.

Routine Name	*Function*
`ODS_Init` (B.1.1)	Allocates memory for ODS vector
`ODS_Clean` (B.1.2)	Deallocates ODS vector
`ODS_Get` (generic)	Reads data from one (B.1.4) or more (B.1.3) files
`ODS_Put` (B.1.5)	Writes data to file
`ODS_Merge` (B.1.6)	Merges 2 or more ODS vectors
`ODS_MaskOut` (B.1.7)	Mask out observations
`ODS_Tally` (B.1.8)	Prints out ODS summary

ture consists of two components both of which are also structures. The first is `ods_meta` whose components refer to the global meta-data lists in Table 2. The names of the components are given in the first and second column of the table. An extra component, `nsyn`, provides the number of synoptic times per day in the given ODS file. The second component is `obs_vect` which stores the atomic attributes of each observation report for a given synoptic date and time. The component names are given in the first column in Table 3. Two additional components, `nobs` and `nvect`, provide the number of observation reports and allocated vector size where `nobs` $\leq$ `nvect`.

All base-level components are pointers to arrays except those given in the previous paragraph and the list sizes in the second column of Table 2. Since they are pointers, space must be created prior to storing the data. If data are to be written, then the routine `ODS_Init` must first be called to create the space. To read the data, a call to `ODS_Init` is not necessary (nor allowed), since allocation is automatically performed by the routine, `ODS_Get`. Once space has been allocated, the data can be accessed by referencing the components. For example, `ods_vect%obs_vect%lat(:n)` will access the latitude values for all observations reports where `n` is the number of observation reports (as stored in the component `ods_vect%obs_vect%nobs`).

Data for an entire synoptic time can be written by calling the routine, `ODS_Put`, as shown in the following sample section of FORTRAN 90 code:

```
use m_ods
type (odsvect) :: ods
integer :: ierr
nobs = nobsmax  ! arbitrarily set elsewhere such as a calling routine
call ODS_Init ( ods, NObs, ierr )

...
...
call ODS_Put ( 'outfile', 'post_anal', 19980521, 60000,
.                ods, ierr )
...
...
call ODS_Put ( 'outfile', 'post_anal', 19980521, 60000,
```

```
.                ods, ierr, append = .true. )

 call ODS_Clean ( ods, ierr )   ! deallocate (optional)
```

Note that the code includes a call to the routine, `ODS_Init`, which allocates space before writing to the file. The call to the routine `ODS_Clean` for deallocation is optional. Also, many ODS related parameters such as the maximum value for the data type, `ktmax`, are defined in the module, `m_odsmeta`. Finally, observation data can be added by implementing the option `append=.true.` if data already exists for a given synoptic date and time.

The following code will read an ODS file for a given synoptic date and time:

```
use m_ods
type (odsvect) :: ods
character (len=30) :: type
integer :: ierr

call ODS_Get ( 'infile', 19980521, 60000, type, ods, ierr )
```

The name of the called routine is generic for two routines which are given in the entry `ODS_Get` in Table 8. The interfaces differ only in the rank (0 or 1) of the array of the input filename(s). The right routine would be selected by the FORTRAN compiler.

Another set of ODS related utility routines exist in the module `m_ods_structure`. These routines are similar to some of those in the module `m_ods` but will soon be obsolete. The module and routine prologues are given in Appendix B.

1.5.2 FORTRAN 77 interface

The ODS library consists of user callable FORTRAN 77 subroutines which allow users to create, read and write ODS files. Table 9 lists each routine, and Appendix B describes the input/output parameters in some detail. For the sake of completeness, Appendix A describes the structure of the ODS file from the NetCDF interface perspective.

For most of the callable routines, the input parameter `id` is either returned or is an input parameter which should be regarded as unit number (or file handle) for access to the desired file. In addition, the status code parameter, `ierr`, is returned. It is advisable that the `ierr` parameter be check on returned from all ODS interface routines. A zero for `ierr` indicates no error. For all other values, see Table 10. In Section 1.5.3, the main steps involved in writing/reading an ODS file are outlined.

1.5.3 FORTRAN 77 write routines

To write data to an ODS file, one needs to either create a new file or open a pre-existing file. To open a pre-existing file, the routine, `ODS_Open()`, must be called with the mode set for writing data to the file. On return, `ODS_Open()` provides the unit number, `id`, for subsequent access to the file.

Table 9: User callable FORTRAN 77 subroutines to read/write ODS files. For a complete description of the input/output parameters, see the appropriate section given between the parenthesis next to the routine name in the first column.

Routine Name	*Function*
`ODS_Create` (B.4)	Creates an ODS file
`ODS_Open` (B.5)	Opens a pre-existing ODS file
`ODS_Append` (B.6)	Sets up routines to append data to current synoptic time
`ODS_Close` (B.7)	Closes an ODS file
`ODS_PutI` (B.8)	Writes an INTEGER observation attribute for a synoptic time
`ODS_PutR` (B.9)	Writes an REAL observation attribute for a synoptic time
`ODS_GetI` (B.10)	Reads an INTEGER observation attribute for a synoptic time
`ODS_GetR` (B.11)	Reads a REAL observation attribute for a synoptic time
`ODS_IGet` (B.12)	Gets an ODS parameter in integer format
`ODS_RGet` (B.13)	Gets an ODS parameter in floating point format
`ODS_CGet` (B.14)	Gets an ODS parameter in character format
`ODS_NGet` (B.15)	Gets the number of ob reports for a synoptic date and time
`ODS_GetList` (B.16)	Gets a list in character string format
`ODS_Julian` (B.17)	Converts calendar data to Julian day
`ODS_CalDat` (B.18)	Converts Julian day to calendar date
`ODS_Time2Cal` (B.19)	Converts ODS to calendar "time" format.
`ODS_Cal2Time` (B.20)	Converts calendar to ODS "time" format.
`ODS_Min2Cal` (B.21)	Converts minutes since a given reference to calendar "time" format
`ODS_Cal2Min` (B.22)	Converts calendar "time" format to minutes since a given reference
`ODS_ParmI` (B.27)	Gets an ODS integer parameter for creating subsequent files
`ODS_tParmR` (B.28)	Gets an ODS real parameter for creating subsequent files
`ODS_ParmC` (B.26)	Gets an ODS character parameter for creating subsequent files
`ODS_SetParmI` (B.23)	Sets an ODS integer parameter for creating subsequent files
`ODS_SetParmR` (B.24)	Sets an ODS real parameter for creating subsequent files
`ODS_SetParmC` (B.25)	Sets an ODS character parameter for creating subsequent files

To create a brand new ODS file the application program must acquire the current lists. (For the set of lists, see Section 1.2.2.) These lists will likely be maintained by DAO's production group. The first step is to call the routine `ODS_Create()` to setup the file data structures and save the lists. The first Julian day to be stored on file must also be passed into the routine. The ODS utility routines `ODS_Julian` and `ODS_CalDat` can be used to convert from calendar date to Julian day and vice-versa. When creating an ODS file the application program must specify whether a *pre-analysis* or *post-analysis* file is desired. On return, `ODS_Create()` returns the unit number, `id`, for subsequent access to the file.

All data for a given observation attribute and synoptic time is written to an ODS file with one call to the appropriate interface routine (unless the library is set to "append" mode using the interface routine `ODS_Append`. See Section B.6). Currently, analyses are performed 4 times a day at synoptic times 0, 6, 12 and 18 GMT. The current practice is that data collected at a windows of ± 3 hours around the synoptic time are assigned to the synoptic time. (Recall in Section 1.3.4 that ODS files now include time stamp information specifying

Table 10: Values of the `ierr` parameter returned by ODS routines described in Appendix B. The positive error codes are issued by the NetCDF routines; negative ones by the ODS package (except for `ierr = -1` which is issued by the NetCDF package). For more details consult the NetCDF manual (Rew *et al.* 1993).

ierr	*Description*
0	No Error
1	Not a netcdf id
2	Too many netcdfs open
3	netcdf file exists but user selected no clobber
4	Invalid Argument
5	Write to read only
6	Operation not allowed in data mode
7	Operation not allowed in define mode
8	Coordinates out of Domain
9	MAXNCDIMS exceeded
10	String match to name in use
11	Attribute not found
12	MAXNCATTRS exceeded
13	Not a netcdf data type
14	Invalid dimension id
15	NCUNLIMITED in the wrong index
16	MAXNCVARS exceeded
17	Variable not found
18	Action prohibited on NCGLOBAL varid
19	Not a netcdf file
-1	System error
-3	Invalid dimension size
-4	Invalid interface for variable

the actual time the measurement was made.) If the application program is to call the ODS interface routines to store a total `nobs` observations for the synoptic time 12 GMT, then vectors of length `nobs` for all the relevant observation attributes (see Section 1.3) must be defined. Once the vectors are defined, the following calls can then be executed to write the data to an ODS file:

```
integer  syn_jul, syn_time
real     lat   (nobs), lon(nobs), lev(nobs)
integer  time  (nobs), kt (nobs), kx (nobs), ks(nobs)
real     xm    (nobs)
integer  qcexcl(nobs), qchist(nobs)
real     obs   (nobs)

call ODS_PutR ( id, 'lat',    syn_jul, syn_time, nobs, lat,    ierr )
call ODS_PutR ( id, 'lon',    syn_jul, syn_time, nobs, lon,    ierr )
call ODS_PutR ( id, 'lev',    syn_jul, syn_time, nobs, lev,    ierr )
```

```
call ODS_PutI ( id, 'time',   syn_jul, syn_time, nobs, time,   ierr )
call ODS_PutI ( id, 'kt',     syn_jul, syn_time, nobs, kt,     ierr )
call ODS_PutI ( id, 'kx',     syn_jul, syn_time, nobs, kx,     ierr )
call ODS_PutI ( id, 'ks',     syn_jul, syn_time, nobs, ks,     ierr )
call ODS_PutR ( id, 'xm',     syn_jul, syn_time, nobs, xm,     ierr )
call ODS_PutI ( id, 'qcexcl', syn_jul, syn_time, nobs, qcexcl, ierr )
call ODS_PutI ( id, 'qchist', syn_jul, syn_time, nobs, qchist, ierr )
call ODS_PutR ( id, 'obs',    syn_jul, syn_time, nobs, obs,    ierr )
```

Refer to Appendix B for a complete description of I/O parameters for `ODS_PutI()` and `ODS_PutR()`. Notice that the particular order in which the data attributes are written is not relevant. However, if a given attribute is not saved, no values will be stored on the ODS file for that attribute. In addition, if an attempt is made to retrieve the attribute, then the NetCDF or HDF and thus the ODS interface routines will return a non-zero value for the returned status parameter `ierr`. The package performs a test to ensure that the data being stored are within the ranges specified in Table 3 and returns the appropriate error status parameter, `ierr`.

Notice that `syn_jul` and `syn_time` are the *synoptic* Julian day and *synoptic* time of the segment of observations. These are not necessarily the same as the *time* attribute listed in Table 3. For example, a segment of data corresponding to synoptic time 0Z will include observations that were measured in the previous day, between 23Z and midnight.

Many synoptic times can be stored on a single ODS file. However, for reasons given at the end of Section 1.3.4, the maximum is 45 days. In practice, ODS files will probably have data for only one day (4 synoptic times). After all desired synoptic data have been written to the file, the routine `ODS_Close()` should be called to close the file, providing an `event` string, probably containing the name and version of the program which created the file and the data of creation. If the routine `ODS_Close()` is not called, then the updated internal pointer data is not written to the file, effectively preventing the observation data from being saved.

1.5.4 FORTRAN 77 read routines

The first step is to call routine `ODS_Open()` to open the desired file with the mode set for reading or writing data to the file. On return, `ODS_Open()` returns the integer variable `id` for subsequent access of the file. Once the file is opened, all data for a given observation attribute and synoptic time can be retrieved with one call to the appropriate ODS library routine, *viz.*

```
integer  syn_jul, syn_time
real     lat(nobsmax), lon(nobsmax), lev(nobsmax)
integer  kt (nobsmax), kx (nobsmax)
real     obs(nobsmax)

nobs = nobsmax
call ODS_GetR ( id, 'lat', syn_jul, syn_time, nobs, lat, ierr )
```

```
call ODS_GetR ( id, 'lon', syn_jul, syn_time, nobs, lon, ierr )
call ODS_GetR ( id, 'lev', syn_jul, syn_time, nobs, lev, ierr )
call ODS_GetI ( id, 'kt',  syn_jul, syn_time, nobs, kt,  ierr )
call ODS_GetI ( id, 'kx',  syn_jul, syn_time, nobs, kx,  ierr )
call ODS_GetR ( id, 'obs', syn_jul, syn_time, nobs, obs, ierr )
```

Note that the `nobs` must first be defined at or higher than the actual number of observation reports. The routines `ODS_IGet` and `ODS_RGet` assume that the input value for `nobs` is the maximum allowed (i.e the available storage space). The parameter `ierr` should always be checked to make sure the data have been properly read. Also, the routine `ODS_Close()` should always be called to close the file especially if the observation data have been modified since the file was last opened. No event tag is necessary if the file is opened for reading only and is not modified.

1.5.5 Query routines

To retrieve parameters other than the number of observation reports and those listed in the first column of Table 2, the name of the parameter must first be derived using the following rules. To retrieve an ODS dimension such as the list sizes contained in the second column of Table 2, simply use the dimension name. For the global attributes, the name is given in the first column in Table 1 but must be preceded by a colon (:) with no intervening spaces.

For all other parameters, the parameter name contains two parts which are separated by a colon with no intervening spaces. The first part is a NetCDF variable name which in most cases corresponds to the name listed in the first column of Tables 2 and 3. However, there are two additional NetCDF variables which are named *syn_beg* and *syn_len* and are not included in the table. These variables correspond to pointers for defining the blocks each of which contain the all the observation reports for a synoptic date and time. Some of the NetCDF attributes associated with these variables are useful as will be discussed later in this section.

The second part of the parameter name is the NetCDF variable attribute name. Some of the more common attribute names in an ODS file are

name Name of the NetCDF variable. These names correspond to the table entries in the column labeled "Descriptions" in Tables 2 and 3. (A couple entries were abbreviated to save table space.)

units Units for the NetCDF variable. The units correspond to the table entries in the column labeled "Unit" in Table 3. (Blank entries denote that the attribute does not exist and is therefore invalid for the variable)

valid_range The valid minimum and maximum value for the NetCDF variable. The ranges correspond to the table entry in the column labeled "Valid Range" in Table 3. (Blank entries denote that the attribute does not exist and is therefore invalid for the variable)

missing_value The missing value which is currently set to 10^{15} (DAO standard). As discussed in Section 1.3, the only observation attributes that can be set to missing are xm, obs, omf, oma and $xvec$.

Other common attributes such as *add_offset* and *scale_factor* exist but are usually important only for the internal processing of ODS library routines. For additional information, see Appendix A. Note that not all NetCDF variables (and thus ODS observation attributes) have these attributes. Any attempt to retrieve the NetCDF attribute that does not exist for a given variable will return in an error status from the ODS query library routine. A few special but useful attributes associated with the NetCDF variable *syn_beg* are listed as follows:

first_julian_day The first julian on file. A julian day extends from midnight to midnight GMT rather than from noon to noon where julian day 2,440,000 is May 23, 1968.

latest_julian_day The latest julian day on file.

latest_synoptic_hour The latest synoptic hour (in GMT) during the latest julian day on file.

Once the two parts of the parameter name are known, an NetCDF attribute name can be derived. For example, if the ODS observation attribute is a data type index (kx) and the valid range ($valid\_range$) is desired then the derived name would be be $kx : valid\_range$.

The next step to retrieving the header information is to call routine `ODS_Open()` to open the desired file with the mode set for reading or writing data to the file. On return, `ODS_Open()` returns the integer variable `id` for subsequent access of the file. Once the file is opened, the parameter can be extracted viz,

```
integer         id, ierr
character * (*) parm_name
parameter     ( parm_name = 'kx:valid_range')
integer     (2) valid_range

call ODS_IGet ( id, parm_name, valid_range, ierr )
```

Note that the an array is used as an output argument to the query routine since the range actually consists of two values, the minimum and maximum. For the case when only one value is to be retrieved, the value can be passed via a scaler FORTRAN argument. The output argument from the routine `ODS_CGet` should not be an array.

To retrieve a list named in Table 2, the ODS routine, `ODS_GetList` should be used. The following sample code retrieves the list, *kt_names*:

```
integer         id, ierr
character * (*) dim_name
parameter     ( dim_name   = 'nkt')
character * (*) list_name
parameter     ( list_name  = 'kt_names')
integer         nkt
integer         kt_names * ( 150 ) ( nkt_max )
```

```
call ODS_IGet    ( id, dim_name,  nkt,           ierr )
call ODS_GetList ( id, list_name, nkt, kt_names, ierr )
```

Note that the `list_name` contains no colon since a NetCDF variable instead of a NetCDF attribute is being retrieved. Also note that `nkt` must first be defined at or higher than the actual list size since the routine `ODS_GetList` assumes that the input value for the list size, `nkt`, is the maximum allowed (i.e the available storage space).

To obtain the number of observation reports for a specified synoptic date and time, the integer function, `ODS_NGet` should be called. The following sample code retrieves the number of reports:

```
integer  ODS_NGet, ODS_Julian
integer  id, ierr, nobs, julian_day

jul_day  = ODS_Julian ( 19680523 )
nobs     = ODS_NGet   ( id, julian_day, 6, ierr )
```

Always remember to check the parameter `ierr` to make sure the data have been properly read. Also, use routine `ODS_Close()` to close the file when you are done especially if the data has been modified since the file was last opened. No event tag is necessary if the file is opened for reading only or is not modified..

1.6 Compatibility with different versions

Since the ODS library was first created, modifications have been implemented to the data structure. In anticipation of these changes, the ODS software was designed to minimize the changes to the routine interface. Thus, the software was developed to internally adjust to many variations in the data structure. For example, to conserve storage, the previous implementations stored the data source index (kx) as a one byte integer with a range from 1 to less than 255. Recently, the need arose to extend this range beyond 255 so that two bytes were required. In this case, the ODS routine, `ODS_Create`, was internally modified so that new files would store kx as a two byte rather than an one byte integer. To preserve compatibility with the original ODS files when opening an ODS file, the access software calls the NetCDF routine NCVINQ to inquire the type of the variable kx (i.e., one byte `byte` or 2 byte `short`) and take the appropriate action when reading kx from the file. These changes occur internally in the access software so that no changes are necessary in the calling interface or application program.

However, some of the observation attributes (i.e. NetCDF variables) were either deleted, added or simply renamed to implement ODS Version 2.00. Thus, modifications in the calling routines may be necessary to properly identify the requested information or perhaps add exception handling. These modifications would include resetting the attribute names that are part of the required arguments. Table 11 is provided below to list the observation

attributes used in ODS Versions 1.01 and 1.02. For the attribute names implemented for the current version, see Table 3. Comparison of Tables 3 and 11 reveals the following changes

Table 11: List of observation attributes in Versions 1.01 and 1.02. The FORTRAN interface refers to how this attribute is written/read to file using the software described in Sections 1.5.3 and 1.5.4

Attr. Name	*Description*	*FORTRAN Interface*	*Units*	*Valid Range*	*Storage (bytes)*
lat	Latitude	Real	degrees north	$[-90, +90]$	2
lon	Longitude	Real	degrees east	$[-180, +180]$	2
level	Pressure Level or channel	Real	hPa or none		4
julian	Julian day	Integer			1
time	Time stamp	Integer	minutes	$[0, 1439]$	2
kt	Data type index	Integer		$[1, 255]$	1
kx	Data source index	Integer		$[1, 65535]$	2
ks	Sounding index	Integer		$[1, 65535]$	2
km	Meta-data index	Integer		$[0, 2^{31} - 1]$	4
qc_flag	Quality control flag	Integer		$[0, 65534]$	2
mod_flag	Modification	Integer		$[0, 255]$	1
obs	Observation value	Real	depends on *kt*		4
omf	Observation minus forecast	Real	depends on *kt*		4
oma	Observation minus analysis	Real	depends on *kt*		4

from Version 1.02 to 2.00:

1. The attribute, *level*, in Version 1.02 was renamed to *lev*.
2. The attribute, *julian*, in Version 1.02 was deleted while the attribute, *time*, was redefined to the number of minutes since 0:00 GMT of the *first* Julian day on file.
3. The attribute, *km*, in Version 1.02 originally defined as an integer index was redefined to *xm* as a floating point number. This changed allows an unsupported attribute as well as an index pointer to be stored.
4. The attributes, *qc_flag* and *mod_flag* in Version 1.02 were renamed to *qcexcl* and *qchist*. The meaning of each flag value in Version 1.02 was largely undefined except for zero.
5. The attribute, *xvec* was added for Version 2.16.
6. The storage requirement per observation report (excluding overhead) in Version 1.02 has increased slightly by one byte for both pre-analysis and post-analysis files for Version 2.00. This increase results in larger data storage by ≈ 0.8 Mbytes/day assuming that 200,000 observations exist for each of the 4 synoptic times.

The header section in Version 1.02 remains the same except for the lists in Table 2. The following changes were implemented for Version 2.00

1. The lists, *kx_meta* and *qcx_names*, were added.
2. The names of the NetCDF dimensions, *ktmax* amd *kxmax*, were changed to *nkt* and *nkx*. The dimension, *nqcx*, was added.

2 Software Installation and Usage

2.1 Availability

A copy of this software can be obtained by virtue of the CVS command,

```
cvs -d ${UserID}@molotov.gsfc.nasa.gov:/CM/baseline/GEOS_DAS \
       checkout -r ${TAG} -d ${Out_SubDir} geosdas/Core/src/util/ODS
```

where the C-shell variable, `UserID`, is a valid user identification tag, `TAG` is the CVS software version tag and `Out_SubDir` is the destination subdirectory on the local platform.

For example, a valid string for `TAG` is

```
GEOS_2_9_CORE_DEV_BRANCH
```

If the user identification name is `dao_ops` and the local destination directory is `ods`, a valid CVS command is,

```
cvs -d dao_ops@molotov.gsfc.nasa.gov:/CM/baseline/GEOS_DAS \
       checkout -r GEOS_2_9_CORE_DEV_BRANCH \
                -d ods geosdas/Core/src/util/ODS
```

If the above CVS command is implemented in the home directory, then the source code and other files would be copied into the directory `~dao_ops/ods`.

Once the software is copied to the local output directory all valid tags can be listed for a given ODS file via the command,

```
cvs status -v $(FILE)
```

where `FILE` is a file extracted from CVS. To get all versions of the ODS library, the most logical choice for `FILE` is `Makefile`. The software can be updated to the desired version by entering the command,

```
cvs update -r ${TAG}
```

2.2 Dependencies and System Requirements

To install the complete ODS library, the host platform must have a FORTRAN 90 compiler. A FORTRAN 77 compiler can be used if the FORTRAN 90 modules are to be omitted, but must allow the usual extensions to the ANSI standard such as "include statements". There are a few declarations in the header file, ods_worksp.h, that are marked with the inline comment, "`(system dependent)`". These declarations for integer and real numbers explicitly set the storage size (e.g. 2 bytes) rather than use the machine default. The NetCDF routines assume that some of the input arguments have storage sizes that are predefined by the source code. However, the compilers on all machines, used thus far at the DAO, do support these extensions currently implemented in the ODS software. Even so, the compilers on almost all platforms allow some means of setting the storage size via the declaration statements so that only a few statements in the header file would need to be modified.

The ODS library calls HDF library routines with the NetCDF interface (named MFHDF in the NCSA distribution) to perform the low level I/O. The public domain HDF software can be obtained by anonymous ftp from `ftp://.ncsa.uiuc.edu/HDF`. NCSA usually provides pre-compiled libraries for most Unix workstations and the Cray supercomputer. If the system on which the ODS library is being installed has the `netcdf` library instead of the libraries, `mfhdf`, `df`, `jpeg` and `z`, then Unidata's version of the NetCDF has been implementated. Any program link to the Unidata's library will compile all right, but an error message will result if an attempt is made to read the standard ODS files produced at the DAO. However, the HDF library can read files created by the Unidata's library.

2.3 Installation Instructions

To build the ODS library, the following files are required:

Makefile This file is used to compile using the make utility

Makefile.conf.*ARCH* where *ARCH* is the result of the C-Shell UNIX command, `uname -s` or `uname -n`. This include file for the file, Makefile, contains all the machine dependent configuration parameters necessary to build the software on the desired platform.

Makefile.depend This file contains the list of object files with each entry containing the corresponding source code and the header files that are referenced by the source code.

configure Script to link the make file to the appropriate configuration file, Makefile.conf.*ARCH* (where ARCH is the result of the C-Shell UNIX command, `uname -s`). The command,

```
chmod +x configure
```

should be performed so that permission is granted to execute the script

ods_hdf.h Header file for the upper level routines in the ODS library

ods_worksp.h Header file for the lower level routines in the ODS library.

stdio.h Header file defining the FORTRAN IO unit numbers.

ods.xxx Source code file for the FORTRAN 90 routines. This file is not required if the library is to be built without the FORTRAN 90 modules. Also, the FORTRAN 90 modules are not required for testing the ODS library.

***.f** Source code files for the FORTRAN 77 routines. A list of all files is contained in the make file, Makefile.depend. The name of most files corresponds to the name of the one library routine that the file contains. The exceptions are the file, ods_file.f which contains the routines, `ODS_Create`, `ODS_Open`, `ODS_Close`, and `ODS_Append` and all of the block data modules and the file ods_parm.f which contains the routines `ODS_ParmI`, `ODS_ParmR` and `ODS_ParmC`.

***.f90** Source code module files for the FORTRAN 90 routines.

In order to build, the following instructions must be performed

1. Execute the configure script. If your machine is not supported, copy one the Makefile.conf.* for the machine closest to yours, naming the file Makefile.conf.*ARCH*, where *ARCH* is the result of

   ```
   uname -n
   ```

   or

   ```
   uname -s
   ```

   Then, edit the configuration file and redefined the make file macros so that correct commands and compiler options are used. A explanation of the macros required to install and test is given in the file, Makefile.

2. Enter:

   ```
   make lib
   ```

   The library file, libods.a, will be created if the build is successful. To omit the FORTRAN 90 modules, enter

   ```
   make lib MODOBJS=
   ```

   The file will be slightly smaller than if the FORTRAN 90 modules are included. For a list of additional targets, enter

   ```
   make help
   ```

2.4 Library testing

The following files are required in addition to the ones listed in Section 2.3:

ods_test.f Source code for testing the ODS library routines. This program requires no interactive input.

ods_test.dao The output that should be generated by running the test program.

scratch_import.cdl An ASCII file containing the dump of an ODS file created on an SGI 32-bit machine (IRIX). This file will be used to test the portability of both the library routines and the ODS file created on different platforms.

scratch_v1_02.cdl An ASCII file containing the dump of an ODS file created from Version 1.02 of the library. This file will be used to test the ability of the library routines to read the ODS file created by a previous version of the software.

To perform the test, enter:

```
make test
```

If all goes well the last 6 lines of output should read:

```
ncgen -b -o test1_import.nc sample_import.cdl
ncgen -b -o test1_v1_02.nc sample_v1_02.cdl
./ods_test.x > ods_test.out
/bin/rm test1_import.nc test1_v1_02.nc test1.nc
diff ods_test.out ods_test.dao
***** ODS test successful ***
```

If not, visually check the differences between the files `ods_test.out` and `ods_test.dao` as formatted output may differ slightly. In addition, the printed version tag may also differ.

2.5 Compiling an ODS application program

The first step is to make sure is that the HDF and ODS library is installed on your system as described in Sections 2.2 and 2.3. If your application program is in a file named `my_program.f`, then the likely Unix command to compile and link this program is

```
f90 -o my_program.x my_program.f           \
                   -L$(OLibDir) -lods.a \
                   -L$(HLibDir) -lmfhdf -ldf -ljpeg -lz
```

where the make file macros `OLibDir` and `HLibDir` are the directories containing the ODS and HDF libraries. Please notice that the ODS files are written using the NetCDF Application Program Interface (API) but relies on the HDF library. *Therefore, as discussed in Section 2.2 a standard ODS/HDF file produced at the DAO cannot be read by a program linked with Unidata's NetCDF library.*

2.6 Tools for manipulating ODS files

In addition to the ODS library, some tools were developed to process observation data. A list of the UNIX command-line tools is as follows:

UNIX command-line tools:

deltoods Converts data from "del" to ODS format. The software and documentation can be found on hera in the following directories,

```
/production/ods/tools/deltoods/src    # ... the source codes
/production/ods/tools/deltoods/doc/ps # ... documentation in
                                      #     postscript
```

grisas Produces a time-series of area and/or short-time averages. The data in the resulting output file can be viewed and further manipulated with grisasview. Both tools are available only on NAS machines in the directory,

```
~dasilva/bin
```

gritas Calculates the time-means and root-time-mean-squares at grid boxes, globally or regionally. The data in the resulting output file can be viewed and further manipulated with gritasview. Both tools are available only on NAS machines in the directory,

```
~dasilva/bin
```

ncdc2ods Converts data from the National Climatic Data Center (NCDC) to ODS format. Enter ncdc2ods.x without any arguments for a summary of command-line interface. The software and documentation can be found on hera in the following directories,

```
/production/ods/tools/ncdc2ods/v1.00/src    # ... the source codes
/production/ods/tools/ncdc2ods/v1.00/bin    # ... pre-compiled executables
/production/ods/tools/ncdc2ods/v1.00/doc/ps # ... documentation in
                                            #     postscript
```

odslist Lists sorted contents of an ODS file to an ASCII file. In combination with diff or xdiff, this tool can be used to compare two ODS files. Enter odslist without any arguments for a for a summary of command-line interface. This program is compiled with the GEOS core system and is currently available on the NAS machines only, located at

```
~todling/GEOS-2.9.0/geosdas/Core/bin
```

ods_scan.x is a utility developed for the GEOS-3 operational system to check and inspect ODS files for correctness. The utility scans the ODS and prints out information including an observation count by synoptic hour and/or data type index (kx). The software has a simple command line interface (enter ods_scan.x for the usage) and is available in the GEOS-3.x bin directory, i.e.,

```
/u/dao_ops/GEOS-3.2.5/bin.
```

odsselect Selects data from ODS files by data type (kt), data source (kx), level (lev), and quality (qcexcl). Output consists of one or more ODS files containing the selected data. Enter odsselect without any arguments for a summary of command-line interface. This program is compiled with the GEOS core system and is currently available on the NAS machines only, located at

```
~todling/GEOS-2.9.0/geosdas/Core/bin
```

z2t Converts height data to thickness or mean-layer temperatures. Enter z2t.x without any arguments for a summary of command-line interface. The software and documentation can be found on hera in the following directories,

```
/production/ods/tools/z2t/v1.00/src    # ... the source codes
/production/ods/tools/z2t/v1.00/bin    # ... pre-compiled executables
/production/ods/tools/z2t/v1.00/doc/ps # ... documentation in
                                       #     postscript
```

MATLAB-based tools:

odsview Point-and-click graphical ODS file browser. Documentation at URL is as follows:

```
file:/ford1/local/matlab5.3/toolbox/dao/ods/doc/odsview.html
file:/ford1/local/matlab5.3/toolbox/dao/help/ods/odsview.html
```

odsload Load data from an ODS file into MATLAB. Documentation at URL is as follows:

```
file:/ford1/local/matlab5.3/toolbox/dao/help/ods/odsload.html
```

Installation: These programs are installed on /ford1/local. For setup instructions, see URL,

```
file:/ford1/local/matlab5.3/toolbox/dao/setup.html
```

Libraries using ODS routines:

obs_io Read and write all ODS data for a given synoptic time. The software can be obtained by following the procedures given in Section 2.1 with `TAG` and the CVS subdirectory begin set to

```
OBS_IO_2_1_1
```

and

```
geosdas/Core/src/obs_io
```

.

References

Lamich, D., M. Hall, Y. Kondratyeva, R. Govindaraju and E. Hartnett, 1996: Documentation of Observational Data Preparation for the Goddard Earth Observing System (GEOS) Data Assimilation System. *NASA Tech. Memorandum 104606, in preparation.*

Pfaendtner, J., S. Bloom, D. Lamich, M. Seablom, M. Sienkiewicz, J. Stobie, A. da Silva, 1995: Documentation of the Goddard Earth Observing System (GEOS) Data Assimilation System–Version 1. *NASA Tech. Memo. No. 104606*, Vol. **4**, Goddard Space Flight Center, Greenbelt, MD 20771. Available electronically on the World Wide Web as `ftp://dao.gsfc.nasa.gov/pub/tech_memos/volume_4.ps.Z`

Rew, R., G. Davis and S. Emmerson, 1993: *NetCDF User's Guide, an Interface for Data Access.* Unidata Program Center, University Corporation for Atmospheric Research, Boulder, CO, 186pp.

Revision History

Version Number	**Version Date**	**Pages Affected/ Extent of Changes**	**Aproval Authority**
Version 1	November 10, 1999	Derived from DAO ON 1995-01	TBD

A ODS File Structure

This appendix describes the structure of the ODS file using the NetCDF interface to HDF(MFHDF). Most users will rely on the FORTRAN subroutines (described in Section 1.5 and Appendix B) to access the data without the need of knowing the details of the file structure. Such information is important for those users desiring to access the data using the HDF C language interface or from application programs such as IDL. It is assumed that the reader has familiarity with basic NetCDF concepts. For an introduction to NetCDF consult Rew *et al.* (1993).

Table 12 is the dump of the NetCDF header information for an ODS file. In NetCDF jargon, this dump presents the CDL description of the file. Most of the NetCDF dimension, variable and attribute names have already been discussed and listed in Sections 1.2, 1.3 and 1.5.5 and in Tables 1, 2 and 3. Notice that all synoptic times are stored contiguously in a long one-dimensional vector partitioned in batches, each of which has a constant length given by the dimension `batchlen`. To gain access to a segment for a particular synoptic time and Julian day, the "pointer", `syn_beg` and `syn_len` can be used. Given a Julian day and a synoptic time, `syn_beg` provide the location of the beginning of the data segment, and `syn_len` the number of observation available for the particular synoptic time. The locations are given in observation index number. From the index number, the batch number and the location within the batch can be determined. The pointer variables are stored as two dimensional variables; the first and second dimension correspond to the Julian day and synoptic hour, respectively. An index of 1 for the first dimension corresponds to the first Julian day on file. For the second dimension, an index of 1 corresponds to 0 GMT and an index of 2 corresponds to the next synoptic hour which is usually 6 GMT..

Some NetCDF variables such as `lat` and `time`, have the attributes `scale_factor` and `add_offset`. The presence of these variable implies that the ODS software performs a linear transformation before writing and reading the variable, viz,

$$X = aX_s + b \tag{1}$$

where X_s is the value written to the file and a and b are constants given by `scale_factor` and `add_offset`, respectively. The variable X is the observation attribute value given in an observation report and passed to or returned from ODS interface routines. If `scale_factor` is missing, then $a = 1$, and if `add_offset` is missing, then $b = 0$. If X_s is an integer, then X_s is converted to a real number, and then the nearest integer of aX_s is determined. For the case when X_s is to be calculated from the integer X, then X - b is converted to real, and then the nearest integer of $(X$ - $b)/a$ is determined. If X_s is a real number but stored on file as an integer then the type conversion (i.e. real to integer and vice versa), then the type conversion is performed immediately prior to writing the data to the file or immediately after reading the data from file. The same rule applies if X_s is integer but stored on file as a real number.

Table 12: Annotated header of an ODS file produced by the MFHDF utility `ncdump -h`.

```
netcdf SELECT {
dimensions:
```

```
 nbatches = UNLIMITED ; // (1162 currently)
 batchlen = 1000 ;
 nkt = 38 ;
 nkx = 512 ;
 nqcx = 33 ;
 ndays = 255 ;
 nsyn = 4 ;
 strlen = 32 ;

variables:
 char kt_names(nkt, strlen) ;
 kt_names:name = "Name of GEOS/DAS data types" ;
 char kt_units(nkt, strlen) ;
 kt_units:name = "Units for each GEOS/DAS data type" ;
 char kx_names(nkx, strlen) ;
 kx_names:name = "Name of GEOS/DAS data sources" ;
 char kx_meta(nkx, strlen) ;
 kx_meta:name = "kx specific metadata information" ;
 char qcx_names(nqcx, strlen) ;
 qcx_names:name = "Meaning of each possible value of the quality control exclusion mark" ;
 short lat(nbatches, batchlen) ;
 lat:name = "Latitude" ;
 lat:units = "degrees north" ;
 lat:valid_range = -90.f, 90.f ;
 lat:scale_factor = 0.0099999998f ;
 short lon(nbatches, batchlen) ;
 lon:name = "Longitude" ;
 lon:units = "degrees east" ;
 lon:value_at_90E = 90.f ;
 lon:value_at_90W = -90.f ;
 lon:valid_range = -180.f, 180.f ;
 lon:scale_factor = 0.0099999998f ;
 float lev(nbatches, batchlen) ;
 lev:name = "Pressure level or channel" ;
 lev:units = "hPa or none" ;
 short time(nbatches, batchlen) ;
 time:name = "minutes since 0:00 GMT of first julian day on file" ;
 time:units = "minutes since 1999-08-01 00:00:00.0" ;
 time:add_offset = 32587 ;
 time:valid_range = -180, 65354 ;
 byte kt(nbatches, batchlen) ;
 kt:name = "Data type index" ;
 kt:add_offset = 1 ;
 kt:valid_range = 1, 38 ;
 short kx(nbatches, batchlen) ;
 kx:name = "Data source index" ;
 kx:add_offset = 32768 ;
 kx:valid_range = 1, 512 ;
```

```
 long ks(nbatches, batchlen) ;
 ks:name = "Sounding index" ;
 ks:add_offset = 0 ;
 ks:valid_range = 1, 1073741824 ;
 float xm(nbatches, batchlen) ;
 xm:name = "Metadata" ;
 xm:missing_value = 9.9999999e+14f ;
 float obs(nbatches, batchlen) ;
 obs:name = "Observation value" ;
 obs:missing_value = 9.9999999e+14f ;
 float omf(nbatches, batchlen) ;
 omf:name = "Observation minus 6h forecast" ;
 omf:missing_value = 9.9999999e+14f ;
 float oma(nbatches, batchlen) ;
 oma:name = "Observation minus analysis" ;
 oma:missing_value = 9.9999999e+14f ;
 float xvec(nbatches, batchlen) ;
 xvec:name = "PSAS CG solution vector" ;
 xvec:missing_value = 9.9999999e+14f ;
 byte qcexcl(nbatches, batchlen) ;
 qcexcl:name = "Quality control exclusion mark" ;
 qcexcl:valid_range = 0, 255 ;
 short qchist(nbatches, batchlen) ;
 qchist:name = "Quality control history mark" ;
 qchist:add_offset = 32767 ;
 qchist:valid_range = 0, 65534 ;
 long days(ndays) ;
 long syn_beg(ndays, nsyn) ;
 syn_beg:name = "Begining of synoptic hour for each day" ;
 syn_beg:reference_date = "1968-05-23" ;
 syn_beg:value_at_reference_date = 2440000 ;
 syn_beg:first_julian_day = 2451392 ;
 syn_beg:latest_julian_day = 2451393 ;
 syn_beg:latest_synoptic_hour = 18 ;
 long syn_len(ndays, nsyn) ;
 syn_len:name = "Number of observations for syn. time" ;

// global attributes:
 :source = "Data Assimilation Office, Code 910.3, NASA/GSFC" ;
 :title = "GEOS DAS Observational Data Stream (ODS) File" ;
 :type = "post_analysis" ;
 :version = "2.16" ;
 :data_info = "Contact data@dao.gsfc.nasa.gov" ;
 :history = "test1

}
```

B Routine/Function Prologues

B.1 Module m_ods — Implements observation data stream vector class.

INTERFACE:

```
module  m_ods
```

USES:

```
use m_odsmeta
implicit none
```

PUBLIC TYPES:

```
PRIVATE
PUBLIC  ods_meta        ! ODS global metadata (kx names, etc.)
PUBLIC  obs_vect        ! atomic attributes (lat, lon, etc.)
PUBLIC  ods_vect        ! ODS vector consisting of ods_meta and obs_vect
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  ods_init
PUBLIC  ods_clean
PUBLIC  ods_put
PUBLIC  ods_get
PUBLIC  ods_Merge
PUBLIC  ods_MaskOut
PUBLIC  ods_Tally
PUBLIC  ods_Select
```

PUBLIC DATA MEMBERS:

```
PUBLIC  obs_missing      ! observation missing value
```

DESCRIPTION:

This module defines the observation data stream vector class. It relies on the ODS library and HDF.

REVISION HISTORY:

```
02Sep1999 da Silva  Created from Ricardo's m_ods_structure, with several
                    changes to remove circular dependencies on future
                    analysis modules, making it to reflect the data
                    structures of an actual ODS file.
15Nov1999 da Silva  Added ktRH.
03Dec1999 da Silva  Added ODS_Merge()
05Jan2000 da Silva  Removed definition of kt???
13Dec2000 C. Redder Add the component, nsyn, to ods_meta, and the
                    parameter constant, NSYN_DEF.
22Jan2001 da Silva  Added Dick's ods_select()
16Feb2001 Todling   Added Xvec as an attribute of ODS.
```

B.1.1 ODS_Init — Allocate memory for ODS vector.

INTERFACE:

```
subroutine ODS_Init ( ods, nobs, rc,       &
                      nkt, nkx,  nqc, ncr, nsyn ) ! Optional
```

USES:

```
implicit none
```

INPUT/OUTPUT PARAMETERS:

```
type(ods_vect), intent(inout) ::  ods    ! ODS vector to be allocated

integer, intent(inout)        :: nobs    ! number of observations; if
                                         ! nobs<0  nobs will be reset to
                                         !         a large internal value.

                                         ! Size of tables and atributes:
integer, intent(inout), optional :: nkt  !  number of KT's
integer, intent(inout), optional :: nkx  !  number of KX's
integer, intent(inout), optional :: nqc  !  number of QC flags
integer, intent(inout), optional :: ncr  !  number of chars in ODS tables
integer, intent(inout), optional :: nsyn !  number of synoptic times per
                                         !    day
```

OUTPUT PARAMETERS:

```
integer, intent(out)          ::  rc     ! Error return code:
```

```
                                        !  0 - all is well
                                        !  1 - error allocating metadata
                                        !  2 - error allocating attributes
```

DESCRIPTION:

Allocates memory for an ODS vector.

REVISION HISTORY:

16Nov1998	Todling	Initial code
02Sep1999	da Silva	Revised names, removed debris.
13Sep2000	da Silva	Fixed bug in optional parameters
13Dec2000	Redder	Added optional input argument, nsyn
16Feb2001	Todling	Changed to account for Xvec addition

B.1.2 ODS_Clean - Deallocates ODS vector

INTERFACE:

```
      subroutine ODS_Clean ( ods, rc )
```

USES:

```
      implicit none
```

INPUT/OUTPUT PARAMETERS:

```
  type(ods_vect), intent(inout) ::  ods    ! ODS vector to be allocated
```

OUTPUT PARAMETERS:

```
  integer, intent(out)          ::  rc     ! Error return code:
                                        !  0 - all is well
                                        !  1 - error deallocating metadata
                                        !  2 - error deallocating attributes
```

DESCRIPTION:

Frees memory used by an ODS vector.

REVISION HISTORY:

```
07sep1999   da Silva   Initial code
16Feb2001   Todling    Changed to account for Xvec addition
```

B.1.3 ODS_GetM_ — Reads data from an ODS file

INTERFACE:

```
subroutine ODS_GetM_ ( nfiles, fnames, nymd, nhms, ftypes, ods, rc )

!USES
  Implicit NONE
```

INPUT PARAMETERS:

```
  integer, intent(in)            :: nfiles  ! number of files
  character(len=*), intent(in)   :: fnames(nfiles) ! ODS file names
  integer, intent(in)            :: nymd    ! year-month-day, e.g., 19990701
  integer, intent(in)            :: nhms    ! hour-min-sec,   e.g., 120000
```

OUTPUT PARAMETERS:

```
                                             ! ODS file type: 'pre_anal'
                                             !  or 'post_anal'. If 'unknown'
                                             !  the could not be read
  character(len=*), intent(out)   :: ftypes(nfiles)

  type(ods_vect), intent(out)    ::  ods    ! ODS vector

  integer, intent(out)           ::  rc     ! Error return code:
                                             ! = 0    - all is well
                                             ! <1000  - could not read file # rc
                                             ! > 999  - rc-999 is rc from ODS_Merge
```

DESCRIPTION:

This routine reads data from 1 or more ODS files, allocates the necessary memory for the ODS vector, and loads the data for the synoptic time (nymd,nhms). Check the contents of `ftypes` to check whether individual files could be read sucessfully.

REVISION HISTORY:

```
03Dec1999  da Silva  Initial code.
09Mar2000  da Silva  Now is OK if some of the files could not be read,
                     the only requiment is that at least one file can
                     read.
12Dec2999  Redder    Revised prologue to reflect the added flexibility
                     of arbitrarily setting the ODS file parameter,
                     nsyn.
```

B.1.4 ODS_Get1_ — Reads data from a single ODS file

INTERFACE:

```
subroutine ODS_Get1_ ( fname, nymd, nhms, ftype, ods, rc, &
                       ncid )                    ! Optional

!USES
  Implicit NONE
```

INPUT PARAMETERS:

```
  character(len=*), intent(in)   :: fname   ! ODS file name
  integer, intent(in)            :: nymd    ! year-month-day, e.g., 19990701
  integer, intent(in)            :: nhms    ! hour-min-sec,   e.g., 120000

  integer, intent(in), optional  :: ncid    ! ODS file id as returned by
                                            ! ODS_Open(); in this case
                                            ! fname is not used
```

OUTPUT PARAMETERS:

```
  character(len=*), intent(out)  :: ftype   ! ODS file type: 'pre_anal'
                                            !  or 'post_anal'
  type(ods_vect), intent(out)    :: ods     ! ODS vector

  integer, intent(out)           :: rc      ! Error return code:
                                            !  0 - all is well
                                            !  1 - could not open file
```

```
!  2 - could not get table sizes
!  3 - could not get no. obs
!  4 - could not allocate memory
!  5 - could not read tables
!  6 - could not read attributes (pre)
!  7 - could not read attributes (post)
!  8 - could not obtain first julian day on file
```

DESCRIPTION:

This routine reads data from an ODS file, allocates the necessary memory for the ODS vector, and loads the data for the synoptic time (nymd,nhms). Usually the ODS file is opened, the data is read in and the file is closed. However, if the optional parameter `ncid` is specified, the file is assumed to have been externally opened with routine `ODS_Open()`.

REVISION HISTORY:

```
07Sep1999  da Silva   Initial code.
09Mar2000  da Silva   Now it checks if file exists before calling
                      ODS_Open().
12Dec2000  C. Redder  Removed the optional input parameter, whms.
                      Modified error handling procedures to clean
                      up (that is close the file if ncid is not
                      present) if an error occurs.  Added code to
                      translate output sampling time (in minutes
                      since the current synoptic date and time)
                      from ODS sampling time (in minutes since
                      0:00 GMT of the first day on file).  Added
                      to extract and save the number of synoptic
                      time per day.
16Feb2001   Todling   Changed to account for Xvec addition
19Feb2001   Todling   Bug fix; GetList(qcx_n) arg nkx instead of nqc
27Feb2001   Todling   Changed close to pass blank string.
```

B.1.5 ODS_Put — Writes to an ODS file

INTERFACE:

```
subroutine ODS_Put ( fname, ftype,  nymd, nhms, ods, rc, &
                     ncid,  append, new )         ! Optional
!USES
  Implicit NONE
```

INPUT PARAMETERS:

```
character(len=*), intent(in)   :: fname   ! ODS file name
character(len=*), intent(in)   :: ftype   ! ODS file type: 'pre_anal'
                                          !  or 'post_anal'
integer, intent(in)            :: nymd    ! year-month-day, e.g., 19990701
integer, intent(in)            :: nhms    ! hour-min-sec,   e.g., 120000

integer, intent(in), optional  :: ncid    ! ODS file id as returned by
                                          ! ODS_Open(); in this case
                                          ! fname is not used

type(ods_vect), intent(in)     :: ods     ! ODS vector

logical, intent(in), optional  :: append  ! If true, the data is appended
                                          ! to the ODS file at this time.
                                          ! Default: append = .false.

logical, intent(in), optional  :: new     ! If true, file will be created
                                          ! whether it exists or not
                                          ! Default: new    = .false.
```

OUTPUT PARAMETERS:

```
integer, intent(out)           :: rc      ! Error return code:
                                          !  0 - all is well
                                          !  1 - could not open file
                                          !  2 - could not create file
                                          !  3 - could not append file
                                          !  4 - could not write a
                                          !     'pre_anal' observation
                                          !      attribute
                                          !  5 - could not write a
                                          !     'post_anal' observation
                                          !      attribute
                                          !  6 - error in allocating
                                          !      scratch space.
                                          !  7 - could not obtain the
                                          !      number of observations
                                          !      already on file
                                          !  8 - could not obtain the
                                          !      the first Julian day
                                          !      on file.
```

DESCRIPTION:

This routine writes an ODS vector to an ODS file at synoptic time (`nymd,nhms`). Usually the ODS file is opened, the data is written to and the file is closed. However, if the optional

parameter `ncid` is specified, the file is assumed to have been externally opened with routine `ODS_Open()`.

REVISION HISTORY:

```
    07Sep1999  da Silva  Initial code.
    15Nov1999  da Silva  Added optinal parameter "new".
    13Dec2000  C. Redder Revisions in prologue, Modified error
                         handling procedures to clean up (that is
                         close the file if ncid is not present)
                         if an error occurs.   Added code to
                         translate input sampling time (in minutes
                         since the current synoptic date and time)
                         to the ODS sampling time (in minutes since
                         0:00 GMT of the first day on file).  Added
                         code to call ODS_Append only if there are
                         already observations stored on file for
                         the given synoptic day and time.  Added
                         code to set the number of synoptic times
                         based on the value stored in the record
                         ods%meta in the component nsyn.
    16Feb2001   Todling  Changed to account for Xvec addition
    27Feb2001   Todling   Changed close to pass blank string.
```

B.1.6 ODS_Merge — Merges 2 or more ODS vectors

INTERFACE:

```
      subroutine ODS_Merge ( ods_in, nods, ods_out, rc )
```

USES:

```
      implicit none
```

INPUT PARAMETERS:

```
   integer,        intent(in)   ::  nods          ! number of input ODS vectors

    type(ods_vect), intent(in)  ::  ods_in(nods)  ! Many ODS vectors
```

OUTPUT PARAMETERS:

```
type(ods_vect), intent(out)  ::  ods_out       ! Merge ODS vector
integer,        intent(out)  ::  rc            ! Error return code:
                                               !  0 - all is well
                                               !  1 - could not allocate mem
```

DESCRIPTION:

Merges 2 or more ODS vectors.

BUGS:

```
   The merged vector has the same metadata as the first vector, that is
the metadata is not really merged.
```

REVISION HISTORY:

```
03dec1999   da Silva   Initial code
23dec1999   da Silva   Fixed metadata size bug.
26jul2000   da  Silva  Removed kid from merging since kid is set
                       to 1:nvct during ods_init().
16Feb2001   Todling    Changed to account for Xvec addition
```

B.1.7 ODS_MaskOut - Mask out observations given

INTERFACE:

```
subroutine ODS_MaskOut ( y, nobs, boxes, nboxes, X_FLAG, qcx, nset, &
                         outside )   ! optional
```

USES:

```
implicit none
```

INPUT PARAMETERS:

```
type(obs_vect), intent(inout)  :: y       ! obs vector
integer,     intent(in)        :: nobs    ! no. of obs
integer,     intent(in)        :: nboxes  ! no. of boxes

                                          ! Hyper cubes:
real,        intent(in)        :: boxes(2,6,*)
```

```
                                              ! First dimention:
                                              !   1 - beginning
                                              !   2 - end
                                              ! Second dimension:
                                              !   1 - kt range
                                              !   2 - kx range
                                              !   3 - latitude  range
                                              !   4 - longitude range
                                              !   5 - level range
                                              !   6 - time stamp range


      integer,     intent(in)        :: X_FLAG  ! QC flag value to set

      logical, intent(in), OPTIONAL  :: outside ! If true, observations
                                              !  outside boxes will be
                                              !  masked out
```

OUTPUT PARAMETERS:

```
   integer,        intent(out)  ::  qcx(nobs)     ! Typically, qc exclusion
                                                  !  flag

   integer,        intent(out)  ::  nset          ! no. of obs set with X_FLAG
```

DESCRIPTION:

Maskout observations given attribute ranges. By default, the output vector is set with `X_FLAG` if the observation lies INSIDE the boxes. When this routine is called with `outside = .true.`, the output vector is set with `X_FLAG` if the observation lies OUTSIDE the boxes.

REVISION HISTORY:

```
    03dec1999   da Silva   Initial code
    13dec1999   da Silva   Removed timestamp check for now.
```

B.1.8 ODS_Tally - Prints out ODS summary

INTERFACE:

```
subroutine ODS_Tally ( lu, ods, nobs, rc, &
                       kt_only )  ! optional
```

USES:

```
implicit none
```

INPUT PARAMETERS:

```
integer,     intent(in)        :: lu      ! unit number for ASCII
                                           !  OUTPUT
type(ods_vect), intent(in)     :: ods     ! ODS vector
integer,     intent(in)        :: nobs    ! no. of obs

logical, intent(in), OPTIONAL  :: kt_only ! only summary by kt
```

OUTPUT PARAMETERS:

```
integer,     intent(out)       :: rc      ! error code:
                                           !  0   - all is well
                                           ! ...  - abnormal exit
```

DESCRIPTION:

Prints out a summary of an ODS vector.

REVISION HISTORY:

```
13dec1999   da Silva   Initial code based on OBSMRY() from the PSAS
                       library.
```

B.1.9 ODS_Select - Select observations by attribute values

INTERFACE:

```
subroutine ODS_Select ( ods, nobs, nsel, rc,                   &
                        odss,                                  & ! optional
                        complement,                            & ! optional
                           kid      ,   kx      ,     kt      , & ! optional
```

```
                                 ks      , time      , qcexcl      , & ! optional
                              qchist     ,  lat      ,    lon      , & ! optional
                                 lev     ,   xm      ,    obs      , & ! optional
                                 OmF     ,  OmA      ,   Xvec      , & ! optional
                                 kid_list ,   kx_list ,     kt_list , & ! optional
                                  ks_list , time_list , qcexcl_list , & ! optional
                              qchist_list ,  lat_list ,    lon_list , & ! optional
                                 lev_list ,   xm_list ,    obs_list , & ! optional
                                 OmF_list ,  OmA_list ,   Xvec_list , & ! optional
                                 kid_range,   kx_range,     kt_range, & ! optional
                                  ks_range, time_range, qcexcl_range, & ! optional
                              qchist_range,  lat_range,    lon_range, & ! optional
                                 lev_range,   xm_range,    obs_range, & ! optional
                                 OmF_range,  OmA_range,   Xvec_range  ) ! optional
```

USES:

```
   implicit none
```

INPUT PARAMETERS:

```
   type(ods_vect), intent(inout) :: ods        ! obs vector
   integer, intent(in)           :: nobs       ! number of obs
   logical, intent(in), optional :: complement ! apply complement of selection criteria

                                               ! selection criteria:
                                               ! ------------------

   integer, intent(in), optional :: kid          !  identification index
   real,    intent(in), optional :: lat          !  latitute
   real,    intent(in), optional :: lon          !  longitude
   real,    intent(in), optional :: lev          !  level
   integer, intent(in), optional :: kx           !  data source index
   integer, intent(in), optional :: kt           !  data type index
   integer, intent(in), optional :: ks           !  sounding index
   real,    intent(in), optional :: xm           !  metadata
   integer, intent(in), optional :: time         !  time
   real,    intent(in), optional :: obs          !  obs value
   real,    intent(in), optional :: OmF          !  obs minus forecast
   real,    intent(in), optional :: OmA          !  obs minus analysis
   real,    intent(in), optional :: Xvec         !  PSAS CG solution vector
   integer, intent(in), optional :: qcexcl       !  QC exclusion flag
   integer, intent(in), optional :: qchist       !  QC history flag

   integer, intent(in), optional :: kid_list(:)    !  list of identification indices
   real,    intent(in), optional :: lat_list(:)    !  list of latitutes
```

```
real,    intent(in), optional :: lon_list(:)    !  list of longitudes
real,    intent(in), optional :: lev_list(:)    !  list of levels
integer, intent(in), optional :: kx_list(:)     !  list of data source indices
integer, intent(in), optional :: kt_list(:)     !  list of data type indices
integer, intent(in), optional :: ks_list(:)     !  list of sounding indices
real,    intent(in), optional :: xm_list(:)     !  list of metadata
integer, intent(in), optional :: time_list(:)   !  list of times
real,    intent(in), optional :: obs_list(:)    !  list of obs values
real,    intent(in), optional :: OmF_list(:)    !  list of obs minus forecasts
real,    intent(in), optional :: OmA_list(:)    !  list of obs minus analyses
real,    intent(in), optional :: Xvec_list(:)   !  list of PSAS CG sol vec
integer, intent(in), optional :: qcexcl_list(:) !  list of QC exclusion flags
integer, intent(in), optional :: qchist_list(:) !  list of QC history flags

integer, intent(in), optional :: kid_range(2)     !  range of identification indices
real,    intent(in), optional :: lat_range(2)     !  range of latitutes
real,    intent(in), optional :: lon_range(2)     !  range of longitudes
real,    intent(in), optional :: lev_range(2)     !  range of levels
integer, intent(in), optional :: kx_range(2)      !  range of data source indices
integer, intent(in), optional :: kt_range(2)      !  range of data type indices
integer, intent(in), optional :: ks_range(2)      !  range of sounding indices
real,    intent(in), optional :: xm_range(2)      !  range of metadata
integer, intent(in), optional :: time_range(2)    !  range of times
real,    intent(in), optional :: obs_range(2)     !  range of obs values
real,    intent(in), optional :: OmF_range(2)     !  range of obs minus forecasts
real,    intent(in), optional :: OmA_range(2)     !  range of obs minus analyses
real,    intent(in), optional :: Xvec_range(2)    !  range of PSAS CG sol vecs
integer, intent(in), optional :: qcexcl_range(2)  !  range of QC exclusion flags
integer, intent(in), optional :: qchist_range(2)  !  range of QC history flags
```

OUTPUT PARAMETERS:

```
type(ods_vect), intent(out), optional :: odss     ! selected obs vector
integer, intent(out)                  :: nsel     ! number of selected obs
integer, intent(out)                  :: rc       ! return code:
                                                  !  0: all ok
                                                  !  1: allocate error
```

DESCRIPTION:

Selects from the input observation vector ods according to a given set of conditions, specified in terms of values and/or enumerated lists and/or ranges of any of the atomic attributes. For example,

```
call ODS_Select ( ods, nobs, nsel, rc,       &
```

```
                        qcexcl=0, kx_list=(/14,16/), lev_range=(/150.,250./) )
```

selects observations that simultaneously satisfy all specified criteria. Alternatively,

```
call ODS_Select ( ods, nobs, nsel, rc,       &
                  complement=.true.          &
                  qcexcl=0, kx_list=(/14,16/), lev_range=(/150.,250./) )
```

selects observations that do not satisfy any of the specified conditions.

This procedure only sees the first nobs observations in the input observation vector ods; nobs may be less than the actual number of observations in ods.

On return, nsel is the number of selected observations, and the first nobs observations in ods are reordered such that the selected observations are located at the front. Thus this procedure can be applied successively in order to combine selection criteria, for example as in

```
call ODS_Select ( ods, nobs, n   , rc, kt=6 )
call ODS_Select ( ods, n   , nsel, rc, complement=.true., qcexcl=0 )
```

which will select all height observations that did not pass quality control.

An output obs vector can be specified for the selected observations:

```
call ODS_Select ( ods, nobs, nsel, rc,       &
                  odss,                      &
                  qcexcl=0, kx_list=(/14,16/), lev_range=(/150.,250./) )
```

in which case the original ods will remain unchanged.

Notes:

- This routine must be called with keyword arguments for the selection criteria
- Each range includes the endpoints, BUT
- It is up to the user to deal with possible effects of floating point arithmetic on equality tests for real attributes. For example, use

  ```
  lev_range=(/500.-epsilon(1.),500.+epsilon(1.)/)
  ```

- There is no check for inconsistent conditions; the result will be `nsel=0, rc=0`
- Memory for output obs vector is allocated by this routine

REVISION HISTORY:

```
01Nov2000   Dee        Initial code
16Feb2001   Todling    Changed to account for Xvec addition
23Mar2001   Redder     Correct latex bugs in prologue
```

B.2 Module m_odsmeta — Defines the ODS metadata conventions

INTERFACE:

```
MODULE m_odsmeta
```

USES:

```
Implicit none
```

DESCRIPTION:

Defines ODS metadata conventions.

TO DO:

```
1. Define history and exclusion marks for superobbing
2. Implement bitwise encoding of history mark
3. Add kx/kt parameters
```

REVISION HISTORY:

```
01Feb99 (Dee)      See v1.1 revision history for previous revisions
13Dec99 (RT)       Added preprocessing flags defined by Meta and Yelena
20Dec99 (Dee/RT)   Removed restriction on preprocessing and online
                    flag values; made it into a module.
23dec99  Todling/
         da Silva  Created from qc_flag.h, added GEOS-4 specific flags.
28Dec99  Todling   Adding kt info (use to live in kt_max.h).
09Feb00  Dee       Added X_NOSTATS, X_PSAS
09Feb00  Todling   Defined kxmax here.
06Jun00  Todling   Updated kt list.
04Oct00  Todling   Increased kxmax to 512.
```

CONTENTS:

```
!                              -------------------------
!                              Data Type (kt) Parameters
!                              -------------------------
!
!       10Apr93 - Jing G. - Initial code (kt_max.h)
! 24Mar95 - Jing G. - New version for text based data table.
! 09Mar98 - Todling  - Added all kt definition from ON-95.01
!       10Apr98 - Todling - Updated kt according to Y.Kondratyeva
!       29Dec99 - Todling - Embedded in ODSmeta module.
!
  integer, parameter  :: ktmax  = 38

!.....................................................................

! surface variables
integer, parameter  :: ktus    =  1
integer, parameter  :: ktvs    =  2
integer, parameter  :: ktslp   =  3
! upper-air variables
integer, parameter  :: ktuu    =  4
integer, parameter  :: ktvv    =  5
integer, parameter  :: ktHH    =  6
integer, parameter  :: ktww    =  7  ! CAUTION: used to be ktqq
!   won't work w/ PSAS
integer, parameter  :: ktTT    =  8
integer, parameter  :: ktTd    =  9
integer, parameter  :: ktrh    = 10
integer, parameter  :: ktqq    = 11  ! CAUTION: see ktww above
! More surface variables
integer, parameter  :: ktus10  = 12
integer, parameter  :: ktTs10  = 13
integer, parameter  :: ktTds   = 14
integer, parameter  :: ktrhs   = 15
integer, parameter  :: ktqs10  = 16

        integer, parameter  :: ktpr    = 17
        integer, parameter  :: kttpw   = 18
        integer, parameter  :: kttlw   = 19
        integer, parameter  :: ktfcc   = 20
! Chemistry
        integer, parameter  :: kttco3  = 21
        integer, parameter  :: kto3    = 22
        integer, parameter  :: ktuthk  = 23

        integer, parameter  :: ktspd2m   = 24
        integer, parameter  :: ktmxspd2m = 25
        integer, parameter  :: ktwgust2m = 26
```

```
        integer, parameter  :: ktt2m     = 27
        integer, parameter  :: ktmxt2m   = 28
        integer, parameter  :: ktmmnt2m  = 29
        integer, parameter  :: ktdewt2m  = 30
        integer, parameter  :: ktrh2m    = 31
        integer, parameter  :: ktsphu2m  = 32
        integer, parameter  :: ktps2m    = 33
        integer, parameter  :: ktvis     = 34
        integer, parameter  :: ktsdepth  = 35
        integer, parameter  :: ktwcond   = 36
        integer, parameter  :: ktth_UppA = 37
        integer, parameter  :: ktskint   = 38

        integer, dimension(21), parameter :: ktSurfAll = (/ ktus, ktvs, ktslp,
     &                                                      ktus10, ktTs10, ktTds, ktrhs, ktqs
     &                                                      ktspd2m, ktmxspd2m, ktwgust2m, ktt
     &                                                      ktmxt2m, ktmmnt2m, ktdewt2m, ktrh2
     &                                                      ktsphu2m, ktps2m, ktskint, ktpr, kt

        integer, dimension(10), parameter :: ktUppaAll = (/ ktuu, ktvv, ktHH, ktww,
     &                                                      ktTT, ktTd, ktrh, ktqq,
     &                                                      ktuthk, ktth_UppA /)

        integer, dimension( 2), parameter :: ktChemAll = (/ kttco3, kto3 /)

        integer, dimension( 5), parameter :: ktOthrAll = (/ kttlw, ktfcc, ktvis,
     &                                                      ktsdepth, ktwcond /)

!.

!                             ---------------------------
!                             Data Source (kx) Parameters
!                             ---------------------------
        integer, parameter ::   kxmax = 512

!                             ------------------
!                             QC Flag parameters
!                             ------------------
!
!     History flags
!     -------------

      integer, parameter :: H_PRE_SUSP   =   1  ! unspecified preprocessing history flag
      integer, parameter :: H_UNDERG     =   3  ! suspect due to underground check

        ! NCEP quality marks that we treat as suspect
      integer, parameter :: H_NCEP1      =   4  !   NCEP CQC - modified value
      integer, parameter :: H_NCEP3      =   5  !   NCEP CQC - suspect value
```

```
      integer, parameter :: H_NCEP89     =   6  !   NCEP PREVENTS - P suspect
                                                !     (underground or sfcP too high)
        !      or suspect specific humid.
      integer, parameter :: H_NCEP14     =   7  !   NCEP SDM purged
      integer, parameter :: H_NCEP_OTHER =   8  !   NCEP missing or unknown QM
                                                !     (QM != 0,1,2,3,8,9,13,14,15)
                                                !  Suspect mark derived from
        !     pressure value (obs maybe OK,
        !     pressure is suspect)
      integer, parameter :: H_NCEP_PRES1 =   9  ! NCEP CQC modified P value
      integer, parameter :: H_NCEP_PRES3 =  10  ! NCEP CQC suspect P value
      integer, parameter :: H_NCEP_PRES14=  11  ! NCEP SDM purged P
                                                !  Suspect marks for moisture
        !  vars calcfrom suspect input values

      integer, parameter :: H_SUSP_TEMP    = 12 ! moisture from suspect temp
      integer, parameter :: H_SUSP_DEWTEMP = 13 ! moisture from suspect dewpt

      integer, parameter :: H_BACKG   = 17      ! background check
      integer, parameter :: H_BACRH   = 18      ! rh backgrd check
      integer, parameter :: H_BACRHTF = 19      ! rhTf backgrd check

      integer, parameter :: H_YELLOW  = 20      ! obs marked as suspect by "Yellow List"
      integer, parameter :: H_SIMUL   = 21      ! obs confidence level less than 1


!     Exclusion flags
!     ---------------
      integer, parameter :: X_PRE_BAD     =   1 ! preprocessing exclusion flag
      integer, parameter :: X_BAD_LOC     =   2 ! bad location flag
      integer, parameter :: X_UNDERG      =   3 ! underground flag
      integer, parameter :: X_OBS_FILL    =   4 ! observation fill flag
      integer, parameter :: X_FCS_FILL    =   5 ! forecast at obs location fill flag
      integer, parameter :: X_TOO_HIGH    =   6 ! obs above certain desired level
      integer, parameter :: X_PASSIVE     =   7 ! passive data type exclusion flag
      integer, parameter :: X_TIME_ACT    =   8 ! data outside active time window
      integer, parameter :: X_NOT_ANAVAR  =   9 ! not an analysis variable


                                                ! NCEP quality marks treated as exclusion
      integer, parameter :: X_NCEP13      =  10 ! NCEP CQC bad observation
      integer, parameter :: X_NCEP15      =  11 ! NCEP PREPDATA bad observation
      integer, parameter :: X_NCEP_PRES13 =  12 ! NCEP CQC bad pressure
      integer, parameter :: X_NCEP_PRES15 =  13 ! NCEP PREPDATA bad pressure

                                                ! DAO preprocessing quality marks
      integer, parameter :: X_RANGE       =  14 ! DAO range check failed
      integer, parameter :: X_DUP         =  15 ! DAO duplicate obs. (>1 in 6 hr)
```

```
      integer, parameter :: X_HYDRO       =  16 ! DAO failed hydrostatic check
                                                !  Rejection marks for moisture values
!     calculated from rejected/dubious input


      integer, parameter :: X_BUDDY   = 17      ! buddy check
      integer, parameter :: X_WIND    = 18      ! wind check
      integer, parameter :: X_NOSTATS = 19      ! no error statistics
      integer, parameter :: X_PROFILE = 20      ! profile check
      integer, parameter :: X_BACKG   = 21      ! background check
      integer, parameter :: X_NORH    = 22      ! no matching rh
      integer, parameter :: X_BACRH   = 23      ! rh background check
      integer, parameter :: X_BACRHTF = 24      ! rhTf background check
      integer, parameter :: X_BUDRH   = 25      ! rh buddy check
      integer, parameter :: X_BUDRHTF = 26      ! rhTf buddy check
      integer, parameter :: X_QSAT    = 27      ! invalid saturation mixr

      integer, parameter :: X_BAD_TEMP    =  28 ! moisture from bad temperature
      integer, parameter :: X_THIN        =  29 ! observation excl. by thinner

      integer, parameter :: X_UNPHYSICAL  = 30  ! obs with unphysical value
      integer, parameter :: X_RED         = 31  ! obs excluded by "Red List"
      integer, parameter :: X_SIMUL       = 32  ! obs could not be simulated
      integer, parameter :: X_PSAS        = 33  ! excluded by PSAS


!     Descriptions of history flags:
!     ------------------------------

! number of history flags in use
      integer, parameter :: nqcHmax = 21

      character(len=32), parameter :: qcHnames(nqcHmax)=(/
     1                 'unspecified preprocessing flag  ',
     2                 '                                ',
     3                 'gcm slightly underground        ',
     4                 'NCEP CQC - modified value       ',
     5                 'NCEP CQC - suspect value        ',
     6                 'NCEP PREVENTS undergnd/bad P, q ',
     7                 'NCEP SDM purged                 ',
     8                 'NCEP unknown or missing QM      ',
     9                 'NCEP CQC modified P             ',
     +                 'NCEP CQC suspect P value        ',
     1                 'NCEP SDM purged P               ',
     2                 'moisture from suspect temp.     ',
     3                 'moisture from suspect dewpt.    ',
     4                 '                                ',
     5                 '                                ',
```

```
     6                 '                                ',
     7                 'outlier wrt background          ',
     8                 'rh outlier wrt background       ',
     9                 'rhTf outlier wrt background     ',
     +                 'marked suspect by Yellow list   ',
     1                 'obs simulated w/ confidence < 1 '/)


!     Descriptions of exclusion flags:
!     --------------------------------

! number of exclusion flags in use
      integer, parameter :: nqcXmax = 33

      character(len=33), parameter :: qcXnames(nqcXmax)=(/
     1                 'unspecified preprocessing flag  ',
     2                 'impossible location             ',
     3                 'gcm deep underground            ',
     4                 'observation value undefined     ',
     5                 'forecast value undefined        ',
     6                 'observation level too high      ',
     7                 'passive data type               ',
     8                 'outside active time window      ',
     9                 'not an analysis variable        ',
     +                 'NCEP CQC bad observation        ',
     1                 'NCEP PREPDATA bad observation   ',
     2                 'NCEP CQC bad pressure           ',
     3                 'NCEP PREPDATA bad pressure      ',
     4                 'DAO range check failed          ',
     5                 'DAO duplicate obs. (>1 in 6 hr) ',
     6                 'DAO failed hydrostatic check    ',
     7                 'failed buddy check              ',
     8                 'incomplete wind vector          ',
     9                 'improper error statistics       ',
     +                 'incomplete vertical profile     ',
     1                 'extreme outlier wrt background  ',
     2                 'no relative humidity information',
     3                 'extreme rh outlier wrt backgrd  ',
     4                 'extreme rhTf outlier wrt backgrd',
     5                 'failed rh buddy check           ',
     6                 'failed rhTf buddy check         ',
     7                 'invalid saturation mixing ratio ',
     8                 'moisture from bad temp.         ',
     9                 'observation removed by thinner  ',
     +                 'unphysical value                ',
     1                 'excluded by "Red List"          ',
     2                 'obs could not be simulated      ',
     3                 'excluded by PSAS                '/)
```

B.3 Module ODS_Structure — Defines the ODS Structure

INTERFACE:

```
module  m_ods_structure
```

USES:

```
use  m_odsmeta

implicit none
```

PUBLIC TYPES:

```
PRIVATE
PUBLIC  ods_attributes_type
PUBLIC  ods_description_type
PUBLIC  ods_structure
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  Create_ODS_Structure
PUBLIC  Destroy_ODS_Structure
PUBLIC  Create_ODS_Attributes
PUBLIC  Destroy_ODS_Attributes
PUBLIC  Create_ODS_Description
PUBLIC  Destroy_ODS_Description
PUBLIC  ODS_GEOS4to2
PUBLIC  ODS_GEOS2to4
PUBLIC  Tally
```

DESCRIPTION:

Defines all ODS attributes as a f90 structure.

REVISION HISTORY:

```
16Nov98   Todling   Initial code.
15Jan99   Todling   Added description data type.
28Dec99   Todling   Added odsmeta and removed  kt_max.
13Apr00   Todling   Made Desc public.
07Feb01   Todling   Added Xvec to structure.
```

B.3.1 Create_ODS_Structure — Allocate space for all ODS structure

INTERFACE:

```
subroutine Create_ODS_Structure ( ods, nobs_max )
```

USES:

```
implicit none
include 'ods_stdio.h'
```

INPUT PARAMETERS:

```
integer nobs_max
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_structure  ) ods
```

DESCRIPTION:

Allocates space for all ODS structure.

REVISION HISTORY:

```
16Nov98   Todling   Initial code
16Feb00   Todling   rename stdio.h to ods_stdio.h
```

B.3.2 Create_ODS_Attributes — Allocate space for ODS attributes

INTERFACE:

```
subroutine Create_ODS_Attributes ( attr, nobs_max )
```

USES:

```
implicit none
include 'ods_stdio.h'
```

INPUT PARAMETERS:

```
integer nobs_max
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_attributes_type  ) attr
```

DESCRIPTION:

Allocates space for ODS attributes.

REVISION HISTORY:

```
16Nov98   Todling   Initial code
16Feb00   Todling   rename stdio.h to ods_stdio.h
```

B.3.3 Create_ODS_Description — Allocates and defines ODS desc vars

INTERFACE:

```
subroutine Create_ODS_Description ( desc )
```

USES:

```
implicit none

include 'ods_stdio.h'
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_description_type  ) desc
```

DESCRIPTION:

Allocates space for ODS attributes.

REVISION HISTORY:

```
16Nov98   Todling   Initial code
16Feb00   Todling   rename stdio.h to ods_stdio.h
```

B.3.4 ODS_GEOS4to2 - Convert GEOS-4 obs vector to GEOS-2 equivalent

INTERFACE:

```
subroutine ODS_GEOS4to2 ( y, ods, null )
```

USES:

```
use m_ods

implicit none
```

INPUT PARAMETERS:

```
type(obs_vect), intent(in)             ::  y  ! New (GEOS-4) obs vector

logical, intent(in), optional          :: null ! Set to .t. to nullify
```

OUTPUT PARAMETERS:

```
type(ods_attributes_type), intent(out) :: ods ! Old (GEOS-2) obs vector
```

DESCRIPTION:

Converts a new (GEOS-4) observation vector to the old (GEOS-4) observation vector type (`ods_attributes_type in GEOS-2 parlance`).

REVISION HISTORY:

```
    12oct1999  da Silva   Initial code.
    07Feb2001  Todling    Added Xvec.
    17Feb2001  Todling    Added optional parameter to nullify pointers
```

B.3.5 ODS_GEOS2to4 - Convert GEOS-2 obs vector to GEOS-4 equivalent

INTERFACE:

```
       subroutine ODS_GEOS2to4 ( ods, y )
```

USES:

```
       use m_ods

       implicit none
```

INPUT PARAMETERS:

```
       type(ods_attributes_type), intent(in) :: ods ! Old (GEOS-2) obs vector
```

OUTPUT PARAMETERS:

```
       type(obs_vect), intent(out)           ::  y  ! New (GEOS-4) obs vector
```

DESCRIPTION:

```
Converts a old (GEOS-2/3) observation vector to the new (GEOS-4) observation
vector type.
```

REVISION HISTORY:

```
    15Feb2001  Todling    Created based on GEOS4to2
    09Mar2001  Todling    Bug fix in xvec pointer.
```

B.3.6 Tally — Summarizes information on observations

INTERFACE:

```
      subroutine Tally ( ods, nobs, nymd, nhms, label )
```

USES:

```
      Implicit NONE


      include 'ods_stdio.h'
```

INPUT PARAMETERS:

```
      type ( ods_structure ) ods

      integer   nobs                ! total no. of observations

                                    ! Synoptic time:
      integer   nymd                !  Year-month-day, e.g.,  19971012
      integer   nhms                !  hour-minute-sec, e.g.,   120000

      character*(*) label           ! label for tally table
```

DESCRIPTION:

```
Summarizes information on observations.  Mimics OI tally routine.
```

REVISION HISTORY:

```
    15Aug98   Todling    Initial code.
    14Dec98   Todling    Passing ods structure.
    16Feb00   Todling   rename stdio.h to ods_stdio.h
```

B.3.7 Destroy_ODS_Structure - Deallocated all ODS structure

INTERFACE:

```
      subroutine destroy_ods_structure ( ods )
```

USES:

```
implicit none
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_structure ) ods
```

DESCRIPTION:

Wipe out ODS structure memory allocation.

REVISION HISTORY:

```
15Jan99   Todling   Initial code
```

B.3.8 Destroy_ODS_Attributes - Deallocated ODS attribute arrays

INTERFACE:

```
subroutine Destroy_ODS_Attributes ( vars )
```

USES:

```
implicit none
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_attributes_type ) vars
```

DESCRIPTION:

Wipe out ODS attributes memory allocation.

REVISION HISTORY:

```
16Nov98   Todling   Initial code
```

B.3.9 Destroy_ODS_Description - Deallocated ODS description arrays

INTERFACE:

```
subroutine Destroy_ODS_Description ( vars )
```

USES:

```
implicit none
```

INPUT/OUTPUT PARAMETERS:

```
type ( ods_description_type ) vars
```

DESCRIPTION:

Wipe out ODS description memory allocation.

REVISION HISTORY:

```
15Jan99   Todling   Initial code
```

B.4 ODS_Create() — Creates an ODS file

Creates an ODS file (including setting up the data structure and contents), sets some internal file parameters and saves text information associated with GEOS/DAS standard data types, data sources and quality control exclusion marks.

Two types of ODS can be created:

Pre-analysis: this file contains the observed value, along with space-time coordinates and a meta-data indicator. At DAO this file is usually produced by the REPACK pre-processing system.

Post-analysis: In addition to the same information present in the pre-analysis files, additional information produced during the assimilation process is included. Examples are the difference between 6 hour forecast and the observed values, and quality control flags assigned to the observations.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Create ( id,    filename, ods_type,
     .                        FirstJDay,
     .                        n_kt,  kt_names, kt_units,
     .                        n_kx,  kx_names, kx_meta,
     .                        n_qcx, qcx_names,
     .                        ierr )
```

INPUT PARAMETERS:

```
      implicit         NONE
      character * (*)  filename             ! name of ODS file
      character * (*)  ods_type             ! = 'pre_anal', if a pre-analysis
                                            !   ODS file is to be created.
                                            ! = 'post_anal', if a post-analysis
                                            !   ODS file is to be created.
      integer          FirstJDay            ! first julian day to be written
                                            !  to the file.  Use the
                                            !  function ODS_Julian to
                                            !  obtain this number.
      integer          n_kt                 ! maximum number of GEOS/DAS
                                            !   data types
      character * (*)  kt_names  ( n_kt )   ! names for each data type
      character * (*)  kt_units  ( n_kt )   ! units for each data type
      integer          n_kx                 ! maximum number of GEOS/DAS
                                            !   data sources
      character * (*)  kx_names  ( n_kx )   ! names fore each data source
      character * (*)  kx_meta   ( n_kx )   ! kx specific meta-data
                                            !   information Use this to specify
                                            !   the meaning of the parameter
                                            !   "xm" or to specify the name of
                                            !    the external OMS file name.
                                            !   e.g. "oms_file:myinstr.oms"
      integer          n_qcx                ! maximum number of possible
                                            !   values for the quality
                                            !   control exclusion mark
                                            !   (qcexcl)
      character * (*)  qcx_names ( n_qcx )  ! information about the meaning
                                            !   of the variable, "qcexcl".
```

OUTPUT PARAMETERS:

```
      integer          id                   ! ODS file handle (or file id);
                                            !  use this to refer to this
                                            !  file later on.
      integer          ierr                 ! Error code. If non-zero, an
                                            !  error has occurred. For a list
                                            !  of possible values, see the
```

```
                                        !  description section of this
                                        !  prologue.  If an error has
                                        !  occurred, then file was either
                                        !  closed or not created
      NOTE: No more than id_max ( as defined in header file ods_hdf.h )
            files can be opened at any one time.
```

SEE ALSO:

```
      ODS_Open()    opens a pre-existing ODS file and returns the
                    dimensions and text data for kt_names, kt_units
                    and kx_names.
      ODS_Close()   closes the ODS file
      ODS_Julian()  convert the "calendar" date to the Julian day
```

REVISION HISTORY:

```
      02Oct1995   da Silva  Specification.
      17May1996   Redder    Original code.
      13Apr1998   Redder    Changed the variable names from kt_max and
                            kx_max to nkt and nkx.  Added code to
                            process according to version number.
                            Added code to retrieve the tables kx_meta
                            and qcx_names and netcdf dimension variable
                            nqcx.  These changes were made to create ODS
                            version 2.00
      19Nov1999   Redder    Added a latex label in and moved the
                            subroutine statement into the prologue.
                            Modified the comments for the return status
                            code.
      06Dec1999   Redder    Corrections to the documentation in the
                            prologue.
```

B.5 ODS_Open() — Opens an ODS file

```
Opens an existing ODS file for reading or writing.

Note:  For a list of error codes, see Table 10.
```

INTERFACE:

```
      subroutine ODS_Open ( id, filename, mode, ierr )
```

INPUT PARAMETERS:

```
      implicit         NONE
      character * (*)  filename       ! name of ODS file
      character * (*)  mode           ! mode = 'w' open for writing
                                      ! mode = 'r' open for reading
```

OUTPUT PARAMETERS:

```
      integer          id             ! ODS file handle (file id)
                                      !   use this to refer to this
                                      !   file later on.
      integer          ierr           ! Error code. If non-zero, an
                                      !  error has occurred. For a list
                                      !  of possible values, see the
                                      !  description section of this
                                      !  prologue.  If an error has
                                      !  occurred, then file was closed.
      NOTE: No more than id_max ( as defined in header file ods_hdf.h )
            files can be opened at any one time.
```

SEE ALSO:

```
      ODS_Create()   creates the ODS file, sets the dimensions
                     and saves the text data for kt_names, kt_units,
                     kx_names, kx_meta, and qcx_names.
```

REVISION HISTORY:

```
      02Oct1995   da Silva  Specification.
      17May1996   Redder    Original code.
      13Apr1998   Redder    Changed the variable names from ktmax and
                            kxmax to nkt and nkx.  Added code to process
                            according to version number.  Added code to
                            retrieve the tables kx_meta and qcx_names
                            and netcdf dimension variable nqcx.  Removed
                            input and output arguments to create a
                            minimalistic version of this routine.  These
                            changes were made to create ODS version 2.00
      19Nov1999   Redder    Added a latex label in and moved the
                            subroutine statement into the prologue.
                            Modified the comments for the return status
                            code.
```

B.6 ODS_Append() — Sets up software to append data to file

```
Sets up the software package to add additional observation reports to the data
segment specified by the latest julian day and synoptic time for which data was
```

written (stored in common). The set up does not affect the subsequent execution of the routines, ODS_GetI and ODS_GetR.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Append ( id, nval, ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id              ! ODS file handle as returned
                                      !  from ODS_Create() or
                                      !  ODS_Open().
                                      !
      integer         nval            ! the number of values to be
                                      !   added.
```

OUTPUT PARAMETERS:

```
      integer          ierr           ! Error code. If non-zero, an
                                      !  error has occurred. For a list
                                      !  of possible values, see the
                                      !  description section of this
                                      !  prologue.
```

SEE ALSO:

```
      ODS_PutI()    write a list of integer values to   an ODS file
      ODS_GetI()    get   a list of integer values from an ODS file
      ODS_PutR()    write a list of real    values to   an ODS file
      ODS_GetR()    get   a list of real    values to   an ODS file
      ODS_Julian()  convert the "calendar" date to the Julian day
      ods_hdf.h     include file defining internal constants
                     and global variables, and setting up data
                     structures
```

REVISION HISTORY:

```
      02Oct1995   da Silva    Specification.
      17May1996   C. Redder   Original code.
      13Apr1998   C. Redder   Updated documentation to reflection
                             changes made in other routines to
                             create ODS version 2.00
      19Nov1999   C. Redder   Added a latex label in and moved the
                             subroutine statement into the prologue.
```

```
                        Modified the comments for the return status
                        code.
      06Dec1999   C. Redder   Corrections to the documentation in the
                        prologue.
      16Feb2000   Todling    Rename stdio.h to ods_stdio.h
```

B.7 ODS_Close() — Closes an ODS file

Closes an ODS file and saves the pointer data necessary to observation data stored in segments according Julian day and synoptic time. If the file has been opened for writing the event tag and the updated pointer data are saved

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Close ( id, event, ierr )
```

INPUT PARAMETERS:

```
      implicit          NONE
      integer           id         ! ODS file handle as returned from
                                   !  ODS_Create() or ODS_Open().
      character * ( * ) event      ! program event tag - usually the
                                   !  name of the program which modified
                                   !  the file and a date.  The
                                   !  length of the string should be
                                   !  as small as possible with no
                                   !  trailing blanks
```

OUTPUT PARAMETERS:

```
      integer           ierr       ! Error code. If non-zero, an
                                   !  error has occurred. For a list
                                   !  of possible values, see the
                                   !  description section of this
                                   !  prologue.  If an error has
                                   !  occurred, then the pointer data
                                   !  and thus the observation data will
                                   !  likely not be updated.
```

SEE ALSO:

```
      ODS_Create()   creates an ODS file
```

```
ODS_Open()     opens an existing ODS file
ods_hdf.h      an include file defining internal constants
                and global variables, and setting up data
                structures
ods_stdio.h        an include file defining standard input/output
                unit numbers
```

SYSTEM ROUTINES:

```
The NetCDF API is used to access the HDF/ODS file.
```

REVISION HISTORY:

02Oct1995	da Silva	Specification.
17May1996	Redder	Original code.
07Jul1998	Redder	Fixed bug that truncates the last character in the history tag when an event tag is appended.
19Nov1999	Redder	Added a latex label in and moved the subroutine statement into the prologue. Modified the comments for the return status code.
06Dec1999	Redder	Corrections to the documentation in the prologue.
16Feb2000	Todling	Rename stdio.h to ods_stdio.h

B.8 ODS_PutI() — Writes data in native integers to file

Stores the contents of an integer variable for a full synoptic hour on an opened ODS file.

Notes:

1. For a list of error codes, see Table 10.
2. This routine does perform checks to verify whether the range of values on input is consistent with the internal variable type in the ODS file. For example, a real variable here could be stored as a two byte variable on file. Therefore, all elements in the array to be stored as two byte integers should have values between -32,767 and 32,767. If the software does detect a number inconsistent with the variable type, then the routine returns with an error message without saving at least some of the values. This check prevents overflows from occurring which would produce unexpected results and probably abnormal program termination.

3. The input values should be within the range specified in Table 3 or by the ODS file annotated header of the ODS file produced with the MFHDF utility, ncdump. If no range is specified, then the maximum and minimum values for the internal variable types are as follows:

type	minimum value	maximum value
byte	0	255
short (2 byte) integer	-32,767	32,767
long (4 byte) integer	-2,147,483,643	2,147,483,643
real (4 bytes)	-3.40e38	3.40e38

4. Once data from the specified date and hour has been written, the routine will allow overwriting of data from a previous date and hour. The only restriction is that the size of the input data block does not exceed the space already allocated in the ODS file. Only the first nval values will be overwritten.

INTERFACE:

```
      subroutine ODS_PutI ( id, VarName, julian_day, syn_hour,
     .                       nval,    values,     ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id              ! ODS file handle as returned
                                      !  from ODS_Create() or
                                      !  ODS_Open().
      character * (*) VarName         ! name of variable
      integer         julian_day      ! Julian Day.  Use the function
                                      !  ODS_Julian to obtain this
                                      !  number.
      integer         syn_hour        ! hour of synoptic time since
                                      !  0:00 GMT (e.g., 0, 6, 12, 18)
      integer         nval            ! number of values for this
                                      !  synoptic time.
      integer         values ( nval ) ! values to be written
```

OUTPUT PARAMETERS:

```
      integer         ierr            ! Error code. If non-zero, an
                                      !  error has occurred. For a list
                                      !  of possible values, see the
                                      !  description section of this
                                      !  prologue.
```

SEE ALSO:

```
ODS_GetI()    get   a list of integer values from an ODS file
ODS_PutR()    write a list of real    values to   an ODS file
ODS_GetR()    get   a list of real    values from an ODS file
ODS_Julian()  convert the "calendar" date to the Julian day
ods_hdf.h     include file defining internal constants
                and global variables, and setting up data
                structures
```

REVISION HISTORY:

02Oct1995	da Silva	Specification.
17May1996	Redder	Original code.
13Apr1998	C. Redder	Updated documentation to reflect changes made in other routines to create ODS version 2.00
26May1999	C. Redder	Fixed bug in handling the case when nval is zero or less.
19Nov1999	C. Redder	Added a latex label in and moved the subroutine statement into the prologue. Modified the comments for the return status code. Removed references to the old version of the ODS documentation.
06Dec1999	C. Redder	Corrections to the documentation in the prologue.

B.9 ODS_PutR() — Writes data in native real format to file

Stores the contents of a real variable for a full synoptic hour on an opened ODS file.

Notes:

1. For a list of error codes, see Table 10.
2. This routine does perform checks to verify whether the range of values on input is consistent with the internal variable type in the ODS file. For example, a real variable here could be stored as a two byte variable on file. Therefore, all elements in the array to be stored as two byte integers should have values between -32,767 and 32,767. If the software does detect a number inconsistent with the variable type, then the routine returns with an error message without saving at least some of the values. This check prevents overflows from occurring which would produce unexpected results and probably abnormal program termination.

3. The input values should be within the range specified in Table 3 or by the ODS file annotated header of the ODS file produced with the MFHDF utility, ncdump. If no range is specified, then the maximum and minimum values for the internal variable types are as follows:

type	minimum value	maximum value
byte	0	255
short (2 byte) integer	-32,767	32,767
long (4 byte) integer	-2,147,483,643	2,147,483,643
real (4 bytes)	-3.40e38	3.40e38

4. Once data from the specified date and hour have been written, the routine will allow overwriting of data from a previous date and hour. The only restriction is that the number to be written does not exceed the space already allocated in the ODS file. Only the first nval values will be overwritten.

INTERFACE:

```
      subroutine ODS_PutR ( id, VarName, julian_day, syn_hour,
     .                       nval,    values,     ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id              ! ODS file handle as returned
                                      !  from ODS_Create() or
                                      !  ODS_Open().
      character * (*) VarName         ! name of variable
      integer         julian_day      ! Julian Day.  Use the function
                                      !  ODS_Julian to obtain this
                                      !  number.
      integer         syn_hour        ! hour of synoptic time since
                                      !  0:00 GMT (e.g., 0, 6, 12, 18)
      integer         nval            ! number of values for this
                                      !  synoptic time
      real            values ( nval ) ! values to be written
```

OUTPUT PARAMETERS:

```
      integer         ierr            ! Error code. If non-zero, an
                                      !  error has occurred. For a list
                                      !  of possible values, see the
                                      !  description section of this
                                      !  prologue.
```

SEE ALSO:

```
ODS_PutI()    write a list of integer values to   an ODS file
ODS_GetI()    get   a list of integer values from an ODS file
ODS_GetR()    get   a list of real    values from an ODS file
ODS_Julian()  convert the "calendar" date to the Julian day
ods_hdf.h     include file defining internal constants
               and global variables, and setting up data
               structures
```

REVISION HISTORY:

02Oct1995	da Silva	Specification.
17May1996	Redder	Original code
13Apr1998	C. Redder	Updated documentation to reflect changes made in other routines to create ODS version 2.00
26May1999	C. Redder	Fixed bug in handling the case when nval is zero or less.
19Nov1999	C. Redder	Added a latex label in and moved the subroutine statement into the prologue. Modified the comments for the return status code. Removed references to the old version of the ODS documentation.
06Dec1999	C. Redder	Corrections to the documentation in the prologue.

B.10 ODS_GetI() — Reads data from file to native integers

Reads the contents of an integer variable for a full synoptic time from an opened ODS file.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
subroutine ODS_GetI ( id, VarName, julian_day, syn_hour,
.                         nval,    values,     ierr )
```

INPUT PARAMETERS:

```
implicit        NONE
integer         id              ! ODS file handle as returned
                                !  from ODS_Create() or
                                !  ODS_Open().
```

```
character * (*) VarName         ! name of variable
integer         julian_day      ! Julian Day.  Use the function
                                !  ODS_Julian to obtain this
                                !  number.
integer         syn_hour        ! hour of synoptic time since
                                !  0:00 GMT (e.g., 0, 6, 12, 18)
```

INPUT/OUTPUT PARAMETERS:

```
integer         nval            ! on input: maximum number of
                                !  values (usually determined
                                !  by the space allocated for
                                !  storage)
                                ! on output: the number of
                                !  values obtained from the
                                !  file
                                ! note: If the input value is
                                !  smaller than the number
                                !  of values to be read, then
                                !  the routine exits with a
                                !  system error message and
                                !  code of ODS_DimErr ( = -3 ).
                                !  Also, nval is set to the
                                !  minimum value required in
                                !  order for the routine to
                                !  execute successfully.
```

OUTPUT PARAMETERS:

```
integer         values ( nval ) ! values to be written
integer         ierr            ! Error code. If non-zero, an
                                !  error has occurred. For a list
                                !  of possible values, see the
                                !  description section of this
                                !  prologue.
```

SEE ALSO:

```
ODS_PutI()    write a list of integer values to   an ODS file
ODS_PutR()    write a list of real    values to   an ODS file
ODS_GetR()    get   a list of real    values from an ODS file
ODS_NGet()    get the number of values for a given julian day
               and synoptic hour
ODS_Julian()  convert the "calendar" date to the Julian day
ods_hdf.h     include file defining internal constants
               and global variables, and setting up data
               structures
```

REVISION HISTORY:

```
02Oct1995   da Silva    Specification.
16May1996   C. Redder   Original code
13Apr1998   C. Redder   Updated documentation to reflect changes
                        made in other routines to create ODS
                        version 2.00
19Nov1999   C. Redder   Added a latex label in and moved the
                        subroutine statement into the prologue.
                        Modified the comments for the return status
                        code.
16Feb2000   Todling     Rename stdio.h to ods_stdio.h
```

B.11 ODS_GetR() — Read data from file to native real format

Reads the contents of a real variable for a full synoptic time from an opened ODS file.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_GetR ( id, VarName, julian_day, syn_hour,
     .                           nval,    values,     ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id              ! ODS file handle as returned
                                      !  from ODS_Create() or
                                      !  ODS_Open().
      character * (*) VarName         ! name of variable
      integer         julian_day      ! Julian Day.  Use the function
                                      !  ODS_Julian to obtain this
                                      !  number.
      integer         syn_hour        ! hour of synoptic time since
                                      !  0:00 GMT (e.g., 0, 6, 12, 18)
```

INPUT/OUTPUT PARAMETERS:

```
      integer         nval            ! on input: maximum number of
                                      !  values (usually determined
                                      !  by the space allocated for
                                      !  storage)
                                      ! on output: the number of
                                      !  values obtained from the
```

```
                                    !  file
                                    ! note: If the input value is
                                    !  smaller than the number
                                    !  of values to be read, then
                                    !  the routine exits with a
                                    !  system error message and
                                    !  code of ODS_DimErr ( = -3 ).
                                    !  Also, nval is set to the
                                    !  minimum value required in
                                    !  order for the routine to
                                    !  execute successfully.
```

OUTPUT PARAMETERS:

```
      real            values ( nval ) ! values to be written
      integer         ierr            ! Error code. If non-zero, an
                                      !  error has occurred. For a list
                                      !  of possible values, see the
                                      !  description section of this
                                      !  prologue.
```

SEE ALSO:

```
      ODS_PutI()    write a list of integer values to   an ODS file
      ODS_GetI()    get   a list of integer values from an ODS file
      ODS_PutR()    write a list of real    values to   an ODS file
      ODS_NGet()    get the number of values for a given julian day
                      and synoptic hour
      ODS_Julian()  convert the "calendar" date to the Julian day
      ods_hdf.h     include file defining internal constants
                      and global variables, and setting up data
                      structures
```

REVISION HISTORY:

```
      02Oct1995   da Silva   Specification.
      17May1996   C. Redder  Original code.
      30May1997   C. Redder  Correction for output of an error message.
      13Apr1998   C. Redder  Updated documentation to reflect changes
                             made in other routines to create ODS
                             version 2.00
      19Nov1999   C. Redder  Added a latex label in and moved the
                             subroutine statement into the prologue.
                             Modified the comments for the return status
                             code.
      16Feb2000   R. Todling Rename stdio.h to ods_stdio.h
```

B.12 ODS_IGet — Returns an integer ODS file parameter

Returns the integer value of the user-selected parameter of a NetCDF file (identified by the character string, parm_name, and the file handle, id). The file parameters are the NetCDF dimensions and the NetCDF attributes as defined in the routine, ODS_Create. These parameters are identified by the parameter name, parm_name, and the ODS file handle, id. The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is an NetCDF attribute, then parm_name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in parm_name must be the delimiter. If the parameter is a NetCDF dimension, then parm_name must not contain any delimiters. Examples for parm_name:

```
nkt           is the dimension, 'nkt'
kt:valid_max  is the attribute, 'valid_max', for
              the variable, 'kt'
:type         is the global attribute, type
```

Note: For a list of error codes, see Table 10.

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
subroutine ODS_IGet ( id, parm_name, parm_values, ierr )
```

INPUT PARAMETERS:

```
implicit   NONE
integer    id              ! ODS file handle
character  parm_name * (*) ! The name of the NetCDF parameter
                           !   to be obtained.  The case of
                           !   each letter is significant.
```

OUTPUT PARAMETERS:

```
integer    parm_values (*) ! The parameter values
integer    ierr            ! Error code. If non-zero, an
```

```
                              !  error has occurred. For a list
                              !  of possible values, see the
                              !  description section of this
                              !  prologue.
      NOTES: This routine is designed to extract NetCDF
             attributes of limited length and assumes that
             the calling routine has already predetermined
             the number the values that this routine will
             extract.  The names of some useful parameters:
             nkt                         maximum number of GEOS/DAS
                                          data types  This dimension
                                          is named ktmax for versions
                                          earlier than 2.00
             nkx                         maximum number of GEOS/DAS
                                          data sources.  This dimension
                                          is named kxmax for versions
                                          earlier than 2.00
             nqcx                        maximum number of possible
                                          values for the quality control
                                          exclusion mark (qcexcl).  This
                                          variable is available only for
                                         versions 2.00 or later
             syn_beg:first_julian_day    first julian day on file
                                          to the file.
             syn_beg:latest_julian_day latest julian day for which
                                          data exists
             syn_beg:latest_synoptic_hour
                                         latest julian hour for which
                                          data exists
```

SEE ALSO:

```
      ODS_RGet() gets the floating point value of a user-selected
                 NetCDF file parameter
      ODS_CGet() gets the character string of a user-selected
                 NetCDF file parameter
      ODS_NGet() get the number of observation reports for a given
                 julian day and synoptic hour
```

REVISION HISTORY:

```
      15Apr1996   Redder   Original version
      13Apr1998   Redder   Made routine to be an interface routine
                           in ODS version 2.00.  Added checks made
                           to input arguments.
      01Nov1999   Redder   Revised code to prevent subscript errors
                           in character strings
      19Nov1999   Redder   Added a latex label in and moved the
```

```
                        subroutine statement into the prologue.
                        Modified the comments for the return status
                        code.
      06Dec1999   Redder   Corrections to the documentation in the
                        prologue.
      16Feb2000   Todling  Rename stdio.h to ods_stdio.h
```

B.13 ODS_RGet — Returns a floating point ODS file parameter

Returns the floating point value of the user-selected parameter of a NetCDF file (identified by the character string, parm_name, and the file handle, id). The file parameters are are the NetCDF dimensions and the NetCDF attributes as defined in the routine, ODS_Create. These parameters are identified by the parameter name, parm_name, and the ODS file handle, id. The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is a NetCDF attribute, then parm_name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in parm_name must be the delimiter. If the parameter is a NetCDF dimension, then parm_name must not contain any delimiters. Examples for parm_name:

```
            nkt           is the dimension, 'nkt'
            kt:valid_max  is the attribute, 'valid_max', for
                          the variable, 'kt'
            :type         is the global attribute, type
```

Note: For a list of error codes, see Table 10.

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
      subroutine ODS_RGet ( id, parm_name, parm_values, ierr )
```

INPUT PARAMETERS:

```
implicit   NONE
integer    id              ! ODS file handle
character  parm_name * (*) ! The name of the NetCDF
                           !   parameter to be obtained.
                           !   The case of each letter
                           !   is significant.
```

OUTPUT PARAMETERS:

```
real       parm_values (*) ! The parameter values
integer    ierr            ! Error code. If non-zero, an
                           !  error has occurred. For a list
                           !  of possible values, see the
                           !  description section of this
                           !  prologue.
NOTE: This routine is designed to extract NetCDF attributes
      of limited length and assumes that the calling
      routine has already predetermined the number the
      values that this routine will extract.
```

SEE ALSO:

```
ODS_IGet() gets the integer of a user-selected
           NetCDF file parameter
ODS_CGet() gets the character string of a user-selected
           NetCDF file parameter
ODS_NGet() get the number of observation reports for a given
           julian day and synoptic hour
```

REVISION HISTORY:

```
05Apr1996   Redder   Original version
13Apr1998   Redder   Made routine to be an interface routine
                     in ODS version 2.00.  Added checks made
                     to input arguments.
01Nov1999   Redder   Revised code to prevent subscript errors
                     in character strings
19Nov1999   Redder   Added a latex label in and moved the
                     subroutine statement into the prologue.
                     Modified the comments for the return status
                     code.
06Dec1999   Redder   Corrections to the documentation in the
                     prologue.
```

B.14 ODS_CGet — Returns a character ODS file parameter

Returns the character string of the user-selected parameter of a NetCDF file (identified by the character string, parm_name, and the file handle, id). If the input string is larger than is necessary, then trailing blanks is inserted into the unused portion of the string. The file parameters are the NetCDF dimensions and the NetCDF attributes as defined in the routine, ODS_Create. These parameters are identified by the parameter name, parm_name, and the ODS file handle, id. The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is a NetCDF attribute, then parm_name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in parm_name must be the delimiter. Examples for parm_name:

```
kt:valid_max  is the attribute, 'valid_max', for
              variable, 'kt'
:type         is the global attribute, type
```

Note: For a list of error codes, see Table 10.

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
subroutine ODS_CGet ( id, parm_name, parm_string, ierr )
```

INPUT PARAMETERS:

```
implicit   NONE
integer    id                ! ODS file handle
character  parm_name   * (*) ! The name of the parameter
                             !   to be obtained.  The case of
                             !   each letter is significant.
```

OUTPUT PARAMETERS:

```
character  parm_string * (*) ! The parameter values
integer    ierr              ! Error code. If non-zero, an
                             !  error has occurred. For a list
```

```
                              !  of possible values, see the
                              !  description section of this
                              !  prologue.
      NOTES: If insufficient space is allocated for parm_string,
             the parameter string is truncated. This routine is
             not appropriate for extracting the dimension sizes
             To obtain the dimension sizes, use the routine,
             ODS_IGet.  Some of the parameters including
            ':filename' and ':mode' (read or write mode) are
             stored in common rather than the netcdf file.
             The names of some useful parameters:
               :filename      name of ODS file
               :mode          mode = 'w' open for write and read
                              mode = 'r' open for read only
               :version       version tag
               :history       history tag
               :type          file type ("pre-analysis" or
                                        "post-analysis")
```

SEE ALSO:

```
      ODS_IGet() Gets the integer of a user-selected NetCDF file
                 parameter
      ODS_RGet() Gets the floating point value of a user-selected
                 NetCDF file parameter
      ODS_NGet() Gets the number of observation reports for a given
                 julian day and synoptic hour
```

REVISION HISTORY:

```
      05Apr1996   Redder   Original version
      13Apr1998   Redder   Made routine to be an interface routine
                           in ODS version 2.00.  Added checks made
                           to input arguments.
      15Mar1999   Redder   Added code to retrieve the ODS type
      01Nov1999   Redder   Revised code to prevent subscript errors
                           in character strings
      19Nov1999   Redder   Added a latex label in and moved the
                           subroutine statement into the prologue.
                           Modified the comments for the return status
                           code.
      06Dec1999   Redder   Corrections to the documentation in the
                           prologue.
      16Feb2000   Todling  Rename stdio.h to ods_stdio.h
```

B.15 ODS_NGet — Returns the number of observation reports

Returns the number of observation reports for a given julian day and synoptic hour

Note: For a list of error codes, see Table 10.

INTERFACE:

```
integer function ODS_NGet ( id, julian_day, syn_hour, ierr )
```

INPUT PARAMETERS:

```
implicit   NONE
integer    id                ! ODS file handle
integer    julian_day        ! Julian day.  Use the function
                             !   ODS_Julian to obtain this
                             !   number.
integer    syn_hour          ! hour of synoptic time since
                             !   0:00 GMT (e.g., 0, 6, 12,
                             !   18)
```

OUTPUT PARAMETERS:

```
integer    NObs              ! Number of observation report
integer    ierr              ! Error code. If non-zero, an
                             !  error has occurred. For a list
                             !  of possible values, see the
                             !  description section of this
                             !  prologue.
```

SEE ALSO:

```
ODS_IGet()    gets the integer of a user-selected
              NetCDF file parameter
ODS_RGet()    gets the floating point value of a user-selected
              NetCDF file parameter
ODS_CGet()    gets the character string of a user-selected
              NetCDF file parameter
ODS_Julian()  converts the "calendar" date to the Julian day
```

REVISION HISTORY:

```
13Apr1996   Redder   Original version.  Routine developed to
                     create ODS version 2.00
20Apr1998   Redder   Correct bug in error message
19Nov1999   Redder   Added a latex label in and moved the
                     subroutine statement into the prologue.
```

```
                        Modified the comments for the return status
                        code.
      06Dec1999   Redder  Corrections to the documentation in the
                        prologue.
```

B.16 ODS_GetList — Reads a list from an ODS file

Reads the user-defined list or array of strings (identified by the character string, ListName) from a NetCDF data file (identified by the ODS file handle id).

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_GetList ( id, ListName, ListSz, List, ierr )
```

INPUT PARAMETERS:

```
      implicit         NONE
      integer          id          ! ODS file handle
      character  * (*) ListName    ! The name of the list to be
                                   !   obtained.  The case of each
                                   !   letter is significant.
```

INPUT/OUTPUT PARAMETERS:

```
      integer         ListSz       ! on input: maximum number of
                                   !   values (usually determined
                                   !   by the space allocated for
                                   !   storage)
                                   ! on output: the number of
                                   !   values obtained from the
                                   !   file
                                   ! note: If the input value is
                                   !   smaller than the number
                                   !   of values to be read, then
                                   !   the routine exits with a
                                   !   system error message and
                                   !   code of ODS_DimErr ( = -3 ).
                                   !   Also, nval is set to the
                                   !   minimum value required in
                                   !   order for the routine to
                                   !   execute successfully.
```

OUTPUT PARAMETERS:

```
      character  * (*) List  ( * ) ! The entries of the list.
      integer    ierr              ! Error code. If non-zero, an
                                   !  error has occurred. For a list
                                   !  of possible values, see the
                                   !  description section of this
                                   !  prologue.
      NOTE:  Each file in the present version of ODS ( version 2.00 )
             contains the lists with the following names
             kt_names    Name of GEOS/DAS data types
             kt_units    Units for each GEOS/DAS data type
             kx_names    Name of GEOS/DAS data sources
             kx_meta     kx specific metadata information.  This
                         information is used to specify the meaning
                         of the ODS variable "xm" or to specify the
                         name of the external OMS file name.  This
                         list is available only in versions 2.00
                         or later.
             qcx_names   information about the meaning of the ODS
                         variable, "qcexcl".  This list is available
                         only in versions 2.00 or later.
```

SEE ALSO:

```
      ODS_PutList ( Put the user-defined list string or array
                    of strings to the NetCDF data file. )
```

FILES USED:

```
      netcdf.inc, a header file, for defining NetCDF library
             parameters
      ods_hdf.h, a header file, for defining hardwired constants
             and defining global variables and setting up data
             structures
      ods_stdio.h, a header file, for defining standard input/output
             unit numbers
   !LIBRARIES ACCESSED:
      NetCDF
```

REVISION HISTORY:

```
      08Mar1996   C. Redder   Original version
      13Apr1998   C. Redder   Made routine to be an interface routine
                              in ODS version 2.00.  Added checks made
                              to input arguments.
      20Apr1998   C. Redder   Fixed bug in bounds check
      02Nov1999   C. Redder   Fixed bug in storing the list entries
                              in the last block of data.
      19Nov1999   C. Redder   Added a latex label in and moved the
```

```
                        subroutine statement into the prologue.
                        Modified the comments for the return
                        status code.
      06Dec1999   C. Redder   Corrections to the documentation in the
                        prologue.
      16Feb2000   R. Todling  Rename stdio.h to ods_stdio.h
```

B.17 ODS_Julian() — Returns the Julian day

The routine returns the Julian day number based on the calendar date. The algorithm was adopted from Press et al. The Julian day number on May 23, 1968 is 2,440,000. The year zero is treated as the year 1 since the year 0 does not exist.

Reference:

Press, William H., Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery, 1992: *Numerical Recipes in Fortran*, 2nd Ed. Cambridge University Press, New York, NY, 963pp.

INTERFACE:

```
      integer function ODS_Julian ( CalDate )
```

INPUT PARAMETERS:

```
      implicit NONE
      integer  CalDate  ! Calendar date in the format YYYYMMDD
                        !   where YYYY is the year, MM is the
                        !   month and DD is the day.  A negative
                        !   number implies that the year is B.C.
```

OUTPUT PARAMETERS:

```
      integer  JulDay     Julian day number
```

SEE ALSO:

```
      ODS_Min2Cal()   - Convert minutes since a given reference to
                        "calendar" date and time
      ODS_Cal2Min()   - Convert "calendar" date and time to minutes
                        since a given reference
      ODS_Time2Cal()  - converts ODS "time" attribute to "calendar"
                        date and time
      ODS_Cal2Time()  - converts "calendar" date and time to ODS
                        time attribute
      ODS_CalDat()    - calculates the "calendar" date and time
                        from the Julian day
```

REVISION HISTORY:

```
13Apr1998  C. Redder   Original code.  Routine was developed to
                       create ODS version 2.00
19Nov1999  C. Redder   Added a latex label in and moved the
                       subroutine statement into the prologue.
06Dec1999  C. Redder   Corrections to the documentation in the
                       prologue.
10May2000  C. Redder   Updated prologue to include the routines
                       ODS_Min2Cal and ODS_Cal2Min in the
                       "See Also" list
```

B.18 ODS_CalDat() — Convert Julian day to calendar date

The routine returns the calendar day based on the Julian day number. The algorithm was adopted from Press et al. The Julian day number on May 23, 1968 is 2,440,000. The return date is in the format YYYYMMDD where YYYY is the year, MM is the month and DD is the day. A negative number implies that the year is B.C.

Reference:

Press, William H., Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery, 1992: *Numerical Recipes in Fortran*, 2nd Ed. Cambridge University Press, New York, NY, 963pp.

INTERFACE:

```
integer function ODS_CalDat ( JulDay )
```

INPUT PARAMETERS:

```
implicit NONE
integer  JulDay   ! Julian day number
```

OUTPUT PARAMETERS:

```
integer  CalDat    Date in "calendar" format
```

SEE ALSO:

```
ODS_Min2Cal()  - Convert minutes since a given reference to
                 "calendar" date and time
ODS_Cal2Min()  - Convert "calendar" date and time to minutes
                 since a given reference
```

```
      ODS_Time2Cal() - converts ODS "time" attribute to "calendar"
                       date and time
      ODS_Cal2Time() - converts "calendar" date and time to ODS
                       time attribute
      ODS_Julian()   - calculates the Julian day from date and
                       time in "calendar" format
```

REVISION HISTORY:

```
      13Apr1998  C. Redder  Original code.  Routine was developed to
                           create ODS version 2.00
      19Nov1999  C. Redder  Added a latex label in and moved the
                           subroutine statement into the prologue.
      10May2000  C. Redder  Updated prologue to include the routines
                           ODS_Min2Cal and ODS_Cal2Min in the
                           "See Also" list
```

B.19 ODS_Time2Cal() — Convert ODS to calendar date and time

The routine converts the ODS "time" to calendar format. The ODS time is the number of minutes since 0:00 GMT of the first Julian day (stored in common) on file. The Julian day in this routine is assumed to begin at 0:00 GMT. The format for the calendar date and time is year 2000 compliant.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Time2Cal ( id, nval, ODSTime,
     .                          CalDate, CalTime, ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id               ! ODS file handle as returned
                                       !  from ODS_Create() or
                                       !  ODS_Open().
      integer         nval             ! number of values to be
                                       !  converted
      integer         ODSTime  ( nval ) ! ODS time
```

OUTPUT PARAMETERS:

```
      integer         CalDate  ( nval ) ! Calendar date in the format
                                        !  YYYYMMDD where YYYY is the
```

```
                                    !  year, MM is the month and
                                    !  DD is the day.  A negative
                                    !  number implies that the
                                    !  year is B.C.
integer         CalTime  ( nval ) ! Time in the format HHMMSS
                                    !  where HH is the hour, MM
                                    !  is the minute and SS is
                                    !  the second.
integer         ierr              ! Error code. If non-zero, an
                                    !  error has occurred. For a list
                                    !  of possible values, see the
                                    !  description section of this
                                    !  prologue.
```

SEE ALSO:

```
ODS_Min2Cal()   - Convert minutes since a given reference to
                  "calendar" date and time
ODS_Cal2Min()   - Convert "calendar" date and time to minutes
                  since a given reference
ODS_Cal2Time()  - converts "calendar" date and time to ODS
                  time attribute
ODS_Julian()    - calculates the Julian day from date and
                  time in "calendar" format
ODS_CalDat()    - calculates the "calendar" date and time
                  from the Julian day
```

REVISION HISTORY:

```
13Apr1998 C. Redder  Original code.  Routine was developed to
                     create ODS version 2.00
19Nov1999 C. Redder  Added a latex label in and moved the
                     subroutine statement into the prologue.
                     Modified the comments for the return status
                     code.
10May2000 C. Redder  Updated prologue to include the routines
                     ODS_Min2Cal and ODS_Cal2Min in the
                     "See Also" list.  Fixed bug for handling
                     negative values for the ODS time
```

B.20 ODS_Cal2Time() — Convert calendar to ODS ”time” format

```
The routine converts data and time in "calendar" format to the ODS "time" format.
The ODS time is the number of minutes since 0:00 GMT of the first Julian day
(stored in common) on file.  The Julian day in this routine is assumed to begin
at 0:00 GMT. The format for the calendar date and time must be year 2000 compliant.
```

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Cal2Time ( id, nval, CalDate, CalTime,
     .                          ODSTime, ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         id               ! ODS file handle as returned
                                       !  from ODS_Create() or
                                       !  ODS_Open().
      integer         nval             ! number of values to be
                                       !  converted
      integer         CalDate  ( nval ) ! Calendar date in the format
                                       !  YYYYMMDD where YYYY is the
                                       !  year, MM is the month and
                                       !  DD is the day.  A negative
                                       !  number implies that the
                                       !  year is B.C.
      integer         CalTime  ( nval ) ! Time in the format HHMMSS
                                       !  where HH is the hour, MM
                                       !  is the minute and SS is
                                       !  the second.
```

OUTPUT PARAMETERS:

```
      integer         ODSTime  ( nval ) ! ODS time
      integer         ierr             ! Error code. If non-zero, an
                                       !  error has occurred. For a
                                       !  list of possible values
                                       !  see Table 8 of da Silva
                                       !  and Redder (1995).
```

SEE ALSO:

```
      ODS_Min2Cal()   - Convert minutes since a given reference to
                        "calendar" date and time
      ODS_Cal2Min()   - Convert "calendar" date and time to minutes
                        since a given reference
      ODS_Time2Cal()  - converts ODS "time" attribute to "calendar"
                        date and time
      ODS_Julian()    - calculates the Julian day from date and
                        time in "calendar" format
      ODS_CalDat()    - calculates the "calendar" date and time
                        from the Julian day
```

REVISION HISTORY:

```
13Apr1998  C. Redder  Original code.  Routine was developed to
                      create ODS version 2.00
19Nov1999  C. Redder  Added a latex label in and moved the
                      subroutine statement into the prologue.
                      Modified the comments for the return status
                      code.
06Dec1999  C. Redder  Minor change in case of variable name.
16Feb2000  R. Todling Rename stdio.h to ods_stdio.h
10May2000  C. Redder  Updated prologue to include the routines
                      ODS_Min2Cal and ODS_Cal2Min in the
                      "See Also" list
```

B.21 ODS_Min2Cal() — Convert minutes since a given reference to "calendar" date and time

The routine converts the minutes since a given reference date and time to calendar format. The reference date and time is also in calendar format. The format for the calendar date and time is year 2000 compliant.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Min2Cal ( NVal,    RefDate, RefTime,
     .                         Minutes, CalDate, CalTime )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         NVal              ! number of values to be
                                        !  converted
      integer         RefDate           ! Reference date in the format
                                        !  YYYYMMDD where YYYY is the
                                        !  year, MM is the month and
                                        !  DD is the day.  A negative
                                        !  number implies that the
                                        !  year is B.C.
      integer         RefTime           ! Reference time in the format
                                        !  HHMMSS where HH is the hour,
                                        !  MM is the minute and SS is
                                        !  the second.
      integer         Minutes  ( NVal ) ! Minutes since given reference
                                        !  date and time
```

OUTPUT PARAMETERS:

```
      integer         CalDate  ( NVal ) ! Calendar date in the format
                                       !  YYYYMMDD where YYYY is the
                                       !  year, MM is the month and
                                       !  DD is the day.  A negative
                                       !  number implies that the
                                       !  year is B.C.
      integer         CalTime  ( NVal ) ! Time in the format HHMMSS
                                       !  where HH is the hour, MM
                                       !  is the minute and SS is
                                       !  the second.
```

SEE ALSO:

```
      ODS_Cal2Min()   - Convert "calendar" date and time to minutes
                        since a given reference
      ODS_Time2Cal()  - converts ODS "time" attribute to "calendar"
                        date and time
      ODS_Cal2Time()  - converts "calendar" date and time to ODS
                        time attribute
      ODS_Julian()    - calculates the Julian day from date and
                        time in "calendar" format
      ODS_CalDat()    - calculates the "calendar" date and time
                        from the Julian day
```

REVISION HISTORY:

```
      10May2000 C. Redder  Original code.
```

B.22 ODS_Cal2Min() — Convert "calendar" date and time to minutes since a given reference

The routine converts date and time in calendar format to minutes since a given reference date and time (also in calendar format). The format for the calendar date and time is year 2000 compliant.

Note: For a list of error codes, see Table 10.

INTERFACE:

```
      subroutine ODS_Cal2Min ( NVal,
     .                         CalDate, CalTime,
     .                         RefDate, RefTime, Minutes )
```

INPUT PARAMETERS:

```
      implicit        NONE
      integer         NVal              ! number of values to be
                                        !  converted
      integer         CalDate  ( NVal ) ! Calendar date in the format
                                        !  YYYYMMDD where YYYY is the
                                        !  year, MM is the month and
                                        !  DD is the day.  A negative
                                        !  number implies that the
                                        !  year is B.C.
      integer         CalTime  ( NVal ) ! Time in the format HHMMSS
                                        !  where HH is the hour, MM
                                        !  is the minute and SS is
                                        !  the second.
```

OUTPUT PARAMETERS:

```
      integer         RefDate           ! Reference date in the format
                                        !  YYYYMMDD where YYYY is the
                                        !  year, MM is the month and
                                        !  DD is the day.  A negative
                                        !  number implies that the
                                        !  year is B.C.
      integer         RefTime           ! Reference time in the format
                                        !  HHMMSS where HH is the hour,
                                        !  MM is the minute and SS is
                                        !  the second.
      integer         Minutes  ( NVal ) ! Minutes since given reference
                                        !  date and time
```

SEE ALSO:

```
      ODS_Min2Cal()   - Convert minutes since a given reference to
                        "calendar" date and time
      ODS_Time2Cal()  - converts ODS "time" attribute to "calendar"
                        date and time
      ODS_Cal2Time()  - converts "calendar" date and time to ODS
                        time attribute
      ODS_Julian()    - calculates the Julian day from date and
                        time in "calendar" format
      ODS_CalDat()    - calculates the "calendar" date and time
                        from the Julian day
```

REVISION HISTORY:

```
      10May2000 C. Redder  Original code.
```

B.23 ODS_SetParmI — Sets a NetCDF integer parameter for creating files

Sets an ODS parameter for creating files by subsequent calls to the interface routine, ODS_Create. The new value will be in affect until this parameter is changed by this routine (or the routine ODS_Create for the parameters, 'nkt', 'nkx' and 'nqcx'). The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is an NetCDF attribute, then the parameter name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in the name must be the delimiter. If the parameter is a NetCDF dimension, then the name must not contain any delimters. Some usefule parameters are as follows:

```
ndays - maximum number of days stored in file
nsyn  - number of synoptic times per day
```

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
subroutine ODS_SetParmI ( ParmName, ParmVal, ierr )
```

INPUT PARAMETERS:

```
implicit        NONE
character * (*) ParmName   ! The name of the NetCDF file
                           !   parameter to be set.  The
                           !   case of each letter is
                           !   significant.
integer         ParmVal    ! The value corresponding to the
                           !   parameter name
!OUTPUT PARAMETER:
integer         ierr       ! The returned error code
NOTE: If the parameter name is not valid, then this routine
      will neither print an error message nor return an
      error status.  In addition, the routine ODS_Create
      will ignore the value associated with the invalid
      name.
```

SEE ALSO:

```
ODS_SetParmR ( Saves the floating point value of a NetCDF
               file parameter )
ODS_SetParmC ( Saves the character string of a NetCDF file parameter )
```

FILES USED:

```
netcdf.inc, a header file, for defining NetCDF library
       parameters
ods_hdf.h, a header file, for defining hardwired constants
       and defining global variables and setting up data
       structures
ods_stdio.h, a header file, for defining standard input/output
       unit numbers
```

REVISION HISTORY:

```
15Apr1996   Redder   Origional version
06Oct2000   Redder   Made routine callable.
```

B.24 ODS_SetParmR — Sets a NetCDF real parameter for creating files

Sets an ODS parameter for creating files by subsequent calls to the interface routine, ODS_Create. The new value will be in affect until this parameter is changed by this routine (or the routine ODS_Create for the parameters, 'nkt', 'nkx' and 'nqcx'). The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is an NetCDF attribute, then the parameter name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in the name must be the delimiter. If the parameter is a NetCDF dimension, then the name must not contain any delimters.

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
subroutine ODS_SetParmR ( ParmName, ParmVal, ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      character * (*) ParmName   ! The name of the NetCDF file
                                 !   parameter to be set.  The
                                 !   case of each letter is
                                 !   significant.
      real            ParmVal    ! The value corresponding to the
                                 !   parameter name
  !OUTPUT PARAMETER:
      integer         ierr       ! The returned error code
      NOTE: If the parameter name is not valid, then this routine
            will neither print an error message nor return an
            error status.  In addition, the routine ODS_Create
            will ignore the value associated with the invalid
            name.
```

SEE ALSO:

```
      ODS_SetParmI ( Saves the integer of a NetCDF file parameter )
      ODS_SetParmC ( Saves the character string of a NetCDF file
                     parameter )
```

FILES USED:

```
      netcdf.inc, a header file, for defining NetCDF library
             parameters
      ods_hdf.h, a header file, for defining hardwired constants
             and defining global variables and setting up data
             structures
      ods_stdio.h, a header file, for defining standard input/output
             unit numbers
```

REVISION HISTORY:

```
      15Apr1996   Redder   Origional version
      06Oct2000   Redder   Made routine callable.
```

B.25 ODS_SetParmC — Sets an ODS character parameter for creating files

Sets an ODS parameter for creating files by subsequent calls to the interface routine, ODS_Create. The new value will be in affect until this parameter is changed by this routine (or the routine ODS_Create for the parameters, 'nkt', 'nkx' and 'nqcx'). The parameter name must be consistent with the notation of

network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is an NetCDF attribute, then the parameter name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in the name must be the delimiter. If the parameter is a NetCDF dimension, then the name must not contain any delimters.

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
      subroutine ODS_SetParmC ( ParmName, ParmString, ierr )
```

INPUT PARAMETERS:

```
      implicit        NONE
      character * (*) ParmName   ! The name of the NetCDF file
                                 !   parameter to be set.  The
                                 !   case of each letter is
                                 !   significant.
      character * (*) ParmString ! The string corresponding to the
                                 !   parameter name
  !OUTPUT PARAMETER:
      integer         ierr       ! The returned error code
      NOTE: If the parameter name is not valid, then this routine
            will neither print an error message nor return an
            error status.  In addition, the routine ODS_Create
            will ignore the value associated with the invalid
            name.
```

SEE ALSO:

```
      ODS_SetParmI ( Saves the integer of a NetCDF file parameter )
      ODS_SetParmR ( Saves the floating point value of a NetCDF
                     file parameter )
```

FILES USED:

```
      netcdf.inc, a header file, for defining NetCDF library
             parameters
      ods_hdf.h, a header file, for defining hardwired constants
             and defining global variables and setting up data
             structures
      ods_stdio.h, a header file, for defining standard input/output
             unit numbers
```

REVISION HISTORY:

```
       15Apr1996   Redder   Origional version
       06Oct2000   Redder   Made routine callable.
```

B.26 ODS_ParmC — Gets a NetCDF string parameter for creating files

```
Returns an ODS parameter for creating files by subsequent calls to the interface
routine, ODS_Create.  The returned value is determined by a default value or
by a call to the companion routine ODS_SetParmC (or the routine ODS_Create for
the parameters, 'nkt', 'nkx' and 'nqcx').  The parameter name must be consistent
with the notation of network Common Data form Language (CDL) as described in
the Rew et al.  (1993).  For this routine, if the parameter is an NetCDF attribute,
then the parameter name must consist of the NetCDF variable name followed by
the delimiter character, ':', and then the attribute name with no blanks.  If
the attribute is global, then the first character in the name must be the delimiter.
If the parameter is a NetCDF dimension, then the name must not contain any delimters.
```

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
       character * (*) function ODS_ParmC ( ParmName,
      .                                      DFault_String,
      .                                      ierr )

       implicit NONE
```

INPUT PARAMETERS:

```
       character * (*) ParmName      ! The name of the NetCDF file
                                     !   parameter to be set.  The
                                     !   case of each letter is
                                     !   significant.
       character * (*) DFault_String ! The default string in case the
                                     !   parameter name is not on the
                                     !   list of specified integers
   !OUTPUT PARAMETER:
       integer         ierr          ! The returned error code
```

SEE ALSO:

```
ODS_ParmI    ( Sets the integer value of a NetCDF file parameter )
ODS_ParmR    ( Sets the floating point value of a NetCDF file
               parameter )
ODS_SetParmC ( Saves the character string of a NetCDF file parameter )
```

FILES USED:

```
netcdf.inc, a header file, for defining NetCDF library
       parameters
ods_hdf.h, a header file, for defining hardwired constants
       and defining global variables and setting up data
       structures
```

REVISION HISTORY:

```
15Apr1996   Redder   Origional version
25Oct2000   Redder   Made routine callable.
```

B.27 ODS_ParmI — Gets a NetCDF integer parameter for creating files

Returns an ODS parameter for creating files by subsequent calls to the interface routine, ODS_Create. The returned value is determined by a default value or by a call to the companion routine ODS_SetParmI (or the routine ODS_Create for the parameters, 'nkt', 'nkx' and 'nqcx'). The parameter name must be consistent with the notation of network Common Data form Language (CDL) as described in the Rew et al. (1993). For this routine, if the parameter is an NetCDF attribute, then the parameter name must consist of the NetCDF variable name followed by the delimiter character, ':', and then the attribute name with no blanks. If the attribute is global, then the first character in the name must be the delimiter. If the parameter is a NetCDF dimension, then the name must not contain any delimters. Some useful parameters are as follows:

```
ndays - maximum number of days stored in file
nsyn  - number of synoptic times per day
```

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
      integer function ODS_ParmI ( ParmName, DFault_Val, ierr )
```

INPUT PARAMETERS:

```
      character * (*) ParmName   ! The name of the NetCDF file
                                 !   parameter to be set.  The
                                 !   case of each letter is
                                 !   significant.
      integer         DFault_Val ! The default value in case the
                                 !   parameter name is not on the
                                 !   list of specified integers
   !OUTPUT PARAMETER:
      integer         ierr       ! The returned error code
```

SEE ALSO:

```
      ODS_ParmR    ( Sets the floating point value of a NetCDF file
                     parameter )
      ODS_ParmC    ( Sets the character string of a NetCDF file parameter )
      ODS_SetParmC ( Saves the character string of a NetCDF file parameter )
```

FILES USED:

```
      netcdf.inc, a header file, for defining NetCDF library
             parameters
      ods_hdf.h, a header file, for defining hardwired constants
             and defining global variables and setting up data
             structures
```

REVISION HISTORY:

```
      04Sep1996   Redder   Origional version
      25Oct2000   Redder   Made routine callable.
```

B.28 ODS_ParmR — Gets a NetCDF real parameter for creating files

```
Returns an ODS parameter for creating files by subsequent calls to the interface
routine, ODS_Create.  The returned value is determined by a default value or
by a call to the companion routine ODS_SetParmR (or the routine ODS_Create for
the parameters, 'nkt', 'nkx' and 'nqcx').  The parameter name must be consistent
with the notation of network Common Data form Language (CDL) as described in
the Rew et al.  (1993).  For this routine, if the parameter is an NetCDF attribute,
then the parameter name must consist of the NetCDF variable name followed by
the delimiter character, ':', and then the attribute name with no blanks.  If
the attribute is global, then the first character in the name must be the delimiter.
If the parameter is a NetCDF dimension, then the name must not contain any delimters.
Some useful parameters are as follows:
```

```
          ndays - maximum number of days stored in file
          nsyn  - number of synoptic times per day
```

Reference:

Rew, Russ, Glenn Davis and Steve Emmerson, 1993: *NetCDF User's Guide*, Unidata Program Center, University Corporation for Atmospheric Research, National Science Foundation, Boulder, CO.

INTERFACE:

```
       real function ODS_ParmR ( ParmName, DFault_Val, ierr )
```

INPUT PARAMETERS:

```
       character * (*) ParmName   ! The name of the NetCDF file
                                  !   parameter to be set.  The case
                                  !   of each letter is significant.
       real            DFault_Val ! The default value in case the
                                  !   parameter name is not on the
                                  !   list of specified integers
   !OUTPUT PARAMETER:
       integer         ierr       ! The returned error code
```

SEE ALSO:

```
       ODS_ParmI    ( Sets the integer value of a NetCDF file parameter )
       ODS_ParmC    ( Sets the character string of a NetCDF file parameter )
       ODS_SetParmC ( Saves the character string of a NetCDF file parameter )
```

FILES USED:

```
       netcdf.inc, a header file, for defining NetCDF library
              parameters
       ods_hdf.h, a header file, for defining hardwired constants
              and defining global variables and setting up data
              structures
```

REVISION HISTORY:

```
       04Sep1996   Redder   Origional version
       25Oct2000   Redder   Made routine callable.
```