Chapter 15

The Z Routines

15.1 Overview

The AIPS system has a number of types of routines in which the details depend on the hardware and/or operating system upon which the code is running. These types of routines, which may vary from system to system, are denoted by the first letter of the name and include: (1) those which depend primarily on the operating system or CPU hardware (denoted by a "Z", thus the "Z" routines), (2) those which depend on the image display (TV) hardware and/or software (the "Y" routines) and (3) those which depend on array or vector hardware and/or software (the "Q" routines). This chapter discusses the "Z" routines; the "Y" and "Q" routines are discussed elsewhere in this manual.

In principle, all that is required to make AIPS work on a new machine is to develop a disk file structure and create a set of "Z", "Q" and "Y" routines to interface AIPS programs to the operating system, the file structure, the array or vector functions and the image display. If routines other than "Z" (or "Y" and "Q") routines are modified, then they will have to be modified every time the AIPS system is updated. For this reason, we recommend that *no routines other than "Z", "Y" or "Q" routines should be modified as part of the installation on a new system.*

This chapter will describe the functions of the upper layer of Z routines; in any implementation, there will probably be additional lower level machine-dependent routines. These Z routines form the basis of a virtual operating system under which the application code runs. Careful study of an existing implementation of AIPS is recommended before attempting a new installation.

The upper layer of the "Z" routines is defined by the contents of the directory with logical name APLGEN. Many of the routines in this directory are in fact generic and call second level routines to do the actual system-dependent operation. The second level "Z" routines called from first level "Z" routines in APLGEN are also included in APLGEN although many are in stubbed form. Many of the first level "Z" routines appear in APLGEN in stubbed form for completeness. Second (or lower) level "Z" routines should never be called from non "Z" routines.

For purposes of discussion, the Z routines will be divided up into a number of overlapping categories:

1. *System Functions* - These routines do various system functions, such as starting and stopping processes, inquiring what processes are running, and inquiring how much space is available on a given disk drive.

2. *Disk I/O and File Manipulation* - These routines create, destroy, expand, contract, open, close, read, and write disk files.

3. *Device I/O* - These routines talk to the terminals, the tape drive, graphics devices, image displays, etc. This area overlaps heavily with the disk I/O area.

4. *Data Manipulation Routines* - These routines convert data formats from external numbers and characters to local and vice versa, and move bits and bytes.

5. *Directory and Text File Routines* - These routines read the directories for, and read and/or write the contents of, text files.

6. *Television I/O routines.* These routines are discussed in the chapter on televisions and only a brief description is given in this chapter.

7. *Virtual Devices* - These routines communicate through network protocols to other processes, which may reside on another computer, which are connected to a physical device.

8. *Miscellaneous* - There are a number of routines, such as that which initializes the Device Characteristics Common, which do not easily fit in one of the other categories.

A detailed description of the call sequences to the non-TV specific first level "Z" routines and listings of the relevant INCLUDE files are at the end of this chapter.

15.1.1 Device Characteristics Common

Many of the parameters describing the host operating system and installation in AIPS programs are carried in the Device Characteristics Common, which is obtained using the include DDCH.INC. The text of this include file can be found at the end of this chapter.

The contents of the Device Characteristics common are initialized by a call to ZDCHIN. Details of the call sequence can be found at the end of this chapter. Many of the values in the Device Characteristics common are read from a disk file. The values in this file can be read and changed using the stand-alone utility program SETPAR. The constants kept in this common, the values in DEVTAB, and the use of logical unit numbers are described in the chapter on disk I/O.

15.1.2 FTAB

The FTAB array in the device characteristics common is used to keep AIPS and system I/O tables. The FTAB has separate areas for the three different kinds of I/O: (1) device I/O to devices which may not need I/O tables, (2) non-map or regular I/O, which is single buffered, wait mode I/O and 3) map I/O, which can be double buffered, non-wait mode I/O.

The FTAB has space for one system I/O table for non-map files and two system I/O tables for map files and space for 16 integer words for application routines to use for map I/O. The size of the entries in FTAB for the different types of I/O are carried in the Device Characteristics Common. The type of the I/O (map or non-map) is declared by the calling routine to the file/device open routine ZOPEN. In general, the FTAB is used to carry any system dependent information necessary for I/O to the device or file. Note: the size of FTAB is dimensioned in the DDCH.INC include for all applications.

The FTAB is divided up by ZDCHIN into three areas, one for each type of I/O. These areas are described in the following:

1. *Non-FTAB I/O* - This area has NTAB1 entries, each NBTB1 bytes long. The first integer word in each entry is zero, if that entry is not in use, and the LUN of the corresponding device, if the entry is in use.

2. *FTAB "non-map" I/O* - This area has NTAB2 entries, each NBTB2 bytes long. The first of these is zero, if that entry is not in use, and the LUN of the corresponding device, if the entry is in use. Following is space for one copy of whatever system I/O table is required for the host system.

3. *FTAB "map" I/O* - This area has NTAB3 entries, each NBTB3 bytes long. The first 16 integer words in each entry are reserved for application routines; the first of these is zero, if that entry is not in use, and the LUN of the corresponding device, if the entry is in use. Following these 16 integers is space for two copies of whatever system I/O table is required for the host system.

Note that a byte is defined in this manual as half a integer, except for most tape and other device I/O where it means 8 bits.

15.1.3 Logicals

The specification of directories inside of AIPS is almost always in terms of a logical directory; that is a variable whose value is the name of the directory or otherwise specifies the directory. Under VMS these are VMS logicals and under Unix they are environment variables. If AIPS is to be implemented in an operating system that does not have an equivalent facility then it must be invented for the AIPS environment. In file names these logicals are given in the form used in VMS, *i.e.* "logical:filename".

15.1.4 Disk Files

The AIPS system uses binary files for data and text files for source code and other information. The location and physical name of the various files depends very much on the host system and installation. The physical name of a file is derived by ZPHFIL and the location of a file is determined by ZPHFIL and ZOPEN (or ZTOPEN or ZTXOPN for text files).

Binary (data) files

Binary files are divided into two types, "map" and "non-map" files corresponding to the two types of I/O. Normally most AIPS binary files on a given disk are put in a single area or directory. Current implementations of AIPS use 8 characters for the basic physical name and 3 more if private catalogs are supported. Applications software will handle up to 48 characters in a name including logicals and/or directory names.

An example from a VAX system with private catalogs is "DA0n:ttrcccvv.uuu" ; where n is the one relative AIPS disk drive number, DA0n: is a logical variable which is assigned to a directory, tt is a two character file type (e.g., 'MA'), r is the data format version code ('A','B',...), ccc is the catalog slot number (hex), vv is the version (hex) (01 for "MA" and "UV" files), and uuu is the user's number in hexadecimal notation.

"Map" type files are files on which it should be possible to double buffer. It should be possible to expand or contract "map". These files should be as contigious as possible since contiguous files are more efficient, but they cause problems finding space for large files. These files should be capable of random access with I/O beginning on a disk sector boundary.

"Non-map" files should also be expandable and contractable. These files should be capable of random access with I/O beginning on a disk sector boundary. In practice, there is no longer any real difference between "map" and "non-map" files and the distinction in AIPS is purely for historical reasons.

Text files

Text files are used for storing source code and control information such as the RUN and HELP/INPUT files as well as a means of communicating information between AIPS and the external world. There are currently two systems of accessing text files; the older one which is read-only is for access to AIPS system text files. A second system will read or write arbitrary text files.

AIPS system text files are accessed using a combination of ZTOPEN, ZTREAD and ZTCLOS. The type of the text file is specified to ZPHFIL as one of several types; the directory may be further selected by the ZTOPEN argument VERSON which can specify the version (directory or area). The member (or file) name is specified to ZTOPEN and may contain up to eight characters. These types, and the files kept in each of the AIPS system text file directories, are described in the following:

1. HE - These are the HELP files which specify which AIPS adverbs are to be sent to tasks and contain the primary user documentation.

2. RU - The RUN files usually contain instructions for the AIPS program. Other types of text files may appear in this area as input for AIPS tasks.

General text files may be accessed using a combination of ZTXOPN, ZTXIO, and ZTXCLS. ZTXOPN will open a file whose specifications is up to 48 characters and must include a logical (environment variable under Unix) pointing to the directory. A file to be written must not previously exist unless the APPEND option is specified. ZTXIO can read or write lines to/from this text file and ZTXCLS will close it.

15.2 System Functions

There are a number of functions involving processes or system resources, which must be done in a system-dependent fashion. These include controlling processes (starting, killing, suspending and resuming) and determining the time, date, name of the current process, and the amount of CPU time used by the current task. Some of these may require special privileges.

The AIPS interactive program may start independent processes, called tasks, which do most of the computations. In order to start a task, AIPS first writes the task's adverbs (determined from the associated HELP file and the current POPS adverb values) together with an initial value of the task return code (-999) into the task data (TD) file, closes the TD file, and starts the task.

AIPS then loops with a fixed time delay (3 sec. for interactive, 8 sec. for interactive with POPS adverb DOWAIT=TRUE and 20 sec. for batch) until one of two conditions exist. These conditions are (1) the value in the TD file of the return code has changed or (2) the task is no longer running. In case 1, AIPS resumes normal operation; in case 2, if the value of the return code is unchanged, the task is assumed to have failed and any scratch files are destroyed. In case 1 or case 2, if the value of the return code is modified, AIPS continues and processes the return code. A non-zero return code indicates that the task failed.

The following list gives a short description of the first level system routines; complete descriptions of the call sequences can be found at the end of this chapter.

- ZABORT establishes or carries out (when appropriate) abort handling.
- ZACTV8 activate the requested program, returning process ID information.
- ZCPU return current process CPU time and IO count.
- ZDATE return the local date.
- ZDCHIN initialize message, device and Z-routine characteristics commons.
- ZDELAY delay current process a specified interval.
- ZFREE display available disk space.
- ZGNAME get name of current process.
- ZPRIO raise or lower the process priority.
- ZSETUP performs system-level operations after VERNAM, TSKNAM, NPOPS known.
- ZSTAIP does any system cleanup needed at the end of interactive AIPS session.
- ZTACTQ inquires if a task is currently active on the local computer.
- ZTIME return the local time of day.
- ZTKILL deletes (or kills) the specified process.
- ZTQSPY display AIPS account or all processes running on the system.
- ZTRLOG translate a logical name.
- ZWHOMI determines AIPSxn task name; sets NPOPS, assigns TV and TK devices.

The following items are second level system routines in APLGEN; others may be needed in a given implementation.

- ZABOR2 establishes or carries out (when appropriate) abort handling.
- ZDCHI2 initialize device and Z-routine characteristics commons - local vals.
- ZDCHIC set more system parameters; make them available to C routines.

- ZDELA2 delay current process a specified interval.
- ZFRE2 return AIPS data disk free space information.
- ZPRI2 raise or lower the process priority.
- ZSTAI2 does any system cleanup needed at the end of interactive AIPS session.
- ZTACT2 inquires if a task is currently active on the local computer.
- ZTQSP2 display AIPS account or all processes running on the system.

15.3 Disk I/O and File Manipulation Routines

This section describes the routines needed for manipulating disk data (binary) files. The physical names of disk data files are always constructed by ZPHFIL and these files are always opened by ZOPEN. There are separate routines for writing to the message file (ZMSGCL, ZMSGDK, and ZMSGOP) to avoid recursion when reporting an error message from one of the I/O routines.

Short descriptions of the first level disk file routines are given in the following list; detailed descriptions of the call sequences are given at the end of the chapter.

- ZCLOSE closes open devices: disk, line printer, terminal.
- ZCMPRS release space from the end of an open disk file.
- ZCREAT creates a disk file.
- ZDESTR destroy a closed disk file.
- ZEXIST return file size and, consequently, whether file exists.
- ZEXPND expand an open disk file — either map or non-map are allowed.
- ZFIO reads and writes single 256-integer records to non-map disk files.
- ZFULLN convert file name to full pathname with no logicals.
- ZMIO random-access, quick return (double buffer) disk I/O for large blocks.
- ZMKTMP convert a “temporary” file name into a unique name.
- ZMSGCL close Message file or terminal.
- ZMSGDK disk I/O to message file.
- ZMSGOP open a message file or message terminal.
- ZMSGXP expand the message file.
- ZOPEN open binary disk files and line printer and TTY devices.
- ZPHFIL construct a physical file or device name from AIPS logical parameters.
- ZPHOLV construct a physical file - version for UPDAT.
- ZRENAM rename a disk file.
- ZTFILL zero-fill, initialize a file I/O table (FTAB) entry.
- ZWAIT wait for asynchronous (“MAP”) I/O to finish.

The following items are second level disk I/O routines in APLGEN; others may be needed in a given implementation.

- ZCMPR2 truncate a disk file, returning blocks to the system.
- ZCREA2 create the specified disk file.
- ZDACLS close a disk file.
- ZDAOPN open the specified disk file.
- ZDEST2 destroy a closed disk file.
- ZEXIS2 return size of disk file and if it exists.
- ZEXPN2 expand an open disk file.
- ZFI2 read/write one 256-integer record from/to a non-map disk file.
- ZMI2 read/write large blocks of data from/to disk, quick return.
- ZMSGWR call MSGWRT based on call arguments - for C routines to call MSGWRT.
- ZPATH convert a file name 'Logical:file' to full path name.
- ZRENA2 rename a file.
- ZWAI2 wait for read/write large blocks of data from/to disk.

15.4 Device (non-disk) I/O Routines

Several of the routines discussed in the disk I/O section will also work on other devices. There are a number of special functions required for non-disk devices. One example of these is the routine to talk to a terminal; some operating systems don't allow Fortran I/O to a terminal, so this I/O is done through the routine ZTTYIO.

The following list gives a short description of these routines; complete descriptions of the call sequences can be found at the end of this chapter.

- ZBKLD1 initialize environment for BAKLD.
- ZBKLD2 does BACKUP operation: load images from tape to directory.
- ZBKLD3 clean up system things for BAKLD ending.
- ZBKTP1 initialize BACKUP to tape operation for BAKTP.
- ZBKTP2 write a cataloged file plus extensions to BACKUP tape in BAKTP.
- ZBKTP3 clean up host environment at end of BAKTP.
- ZDOPRT reads bit file and causes it to be plotted on printer/plotter.
- ZENDPG advance printer if needed to avoid electrostatic-printer "burn-out".
- ZLASCL close and spool a laser printer print/plot file.
- ZLASIO open, write to, close and spool a laser printer print/plot file.
- ZLASOP open a laser printer print/plot file.
- ZLPCLS close an open printer device.
- ZLPOPN open a line-printer text file.
- ZLWIO open, write to, close and spool a PostScript print/plot file.

- ZLWOP open a PostScript (LaserWriter) print/plot file.
- ZMOUNT mount or dismount magnetic tape device.
- ZPRMPT prompt user and read 80-characters from CRT screen.
- ZPRPAS prompt user and read 12-character password (invisible) from CRT.
- ZTAPE mount, dismount, position, write EOF, etc. for tapes.
- ZTAPIO tape operations for IMPFIT (compressed FITS transport tape).
- ZTKBUF flush TK buffer if needed, then store 8-bit byte in buffer.
- ZTKCLS close the TK device.
- ZTKFI2 read/write from/to a Tektronix device.
- ZTKOPN open a TK device.
- ZTPCLD close pseudo-tape disk file.
- ZTPCLS closes a tape device (real or pseudo-tape disk).
- ZTPMID pseudo-tape disk read/write for 2880-bytes records.
- ZTPMIO read/write tape devices with quick return I/O methods.
- ZTPOPD open a pseudo-tape, sequential disk file for FITS.
- ZTPOPN open tape or pseudo-tape device.
- ZTPWAD "wait" for IO operation to complete on pseudo-tape disk file (ZTPMID).
- ZTPWAT wait for asynchronous IO to finish on tape or pseudo-tape disk.
- ZTTBUF reads terminal input with no prompt or wait - simulates TV trackball.
- ZTTYIO read/write buffer to terminal.

The following items are second level system routines in APLGEN; others may be needed in a given implementation.

- ZLASC2 spool a closed laser printer print/plot file.
- ZLPCL2 queue a file to the line printer and delete.
- ZLPOP2 open a line-printer text file - actual OPEN call.
- ZMOUN2 mount or dismount magnetic tape device.
- ZTAP2 position (forward/back record/file), write EOF, etc. for tapes.
- ZTKCL2 close a Tektronix device.
- ZTKOP2 read/write from/to a Tektronix device.
- ZTPCL2 close a tape device.
- ZTPMI2 tape read/write.
- ZTPOP2 open a tape device for double-buffer, asymchronous IO.
- ZTPWA2 wait for read/write from/to a tape device.
- ZTTCLS close a terminal device.
- ZTTOP2 open a message terminal.
- ZTTOPN open a terminal device.

15.5 Data Manipulation Routines

The internal form in which characters, reals and integers are stored varies from computer to computer, but a given FITS data tape should be able to be read on any AIPS system. Thus it is necessary to be able to convert between the external (FITS or other) formats and the internal formats. The format of data on FITS tape files is discussed in another chapter.

The floating point formats, word size and the byte and word order etc. on the current machine are given in commons in the DDCH.INC INCLUDE.

- BYTFLP (I) Byte flip, 0=none, 1=bytes, 2=words, 3=both
- NBITWD (I) no. bits / word
- NBITCH (I) no. bits per character
- NWDPDP (I) no. words / double-precision floating point
- SPFRMT (I) Single precision floating-point format code: 0=> OTHER, 1=> IEEE, 2=> VAX F, 3=> VAX G, 4=> IBM (not supported yet).
- DPFRMT (I) Double precision floating-point format code (see codes for SPFRMT)

The following list gives the names and uses of the upper level data manipulation "Z" routines. Details of the call sequences are given later in this chapter.

- ZBYMOV move 8-bit bytes from in-buffer to out-buffer.
- ZBYTFL interchange bytes in buffer if needed to go between local and standard.
- ZC8CL convert packed ASCII buffer to local character string.
- ZCLC8 convert local character string to packed ASCII buffer.
- ZDHPRL convert 64-bit HP floating buffer to local DOUBLE PRECISION values.
- ZDM2DL convert Modcomp REAL*6 and REAL*8 to local double precision.
- ZGETCH get a character from a HOLLERITH word.
- ZGTBIT get array of bits from a word.
- ZI16IL convert FITS-standard 16-bit integers to local integers.
- ZI32IL convert FITS-standard 32-bit integers from buffer into local integers.
- ZI8IL convert 8-bit unsigned integers in buffer to local integers.
- ZILI16 convert local integers to 16-bit FITS integers in a buffer.
- ZILI32 convert local integer into FITS-standard 32-bit integers.
- ZMCACL convert Modcomp compressed ASCII to Hollerith characters (for FILLR).
- ZPTBIT put array of bits into a word.
- ZPUTCH inserts 8-bit "character" into a word.
- ZR32RL convert 32-bit IEEE floating buffer to local REAL values.
- ZR64RL convert 64-bit IEEE floating-point buffer to local "DOUBLE PRECISION".
- ZR8P4 converts pseudo I*4 to double precision - for tape handling only.
- ZRDMF convert DEC Magtape Format (36 bits data in 40 bits) to 2 integers.

- ZRHPRL convert 32-bit HP floating buffer to local REAL values.
- ZRLR32 converts buffer of local REAL values to IEEE 32-bit floating-point.
- ZRLR64 convert buffer of local double precision values to IEEE 64-bit float.
- ZRM2RL convert Modcomp to local single precision floating point.
- ZUVPAK Pack visibility data, 1 correlator per real with magic value blank.
- ZUVXPN Expands packed visibility data and adds weight.
- ZX8XL convert FITS table bit array to AIPS bit array.
- ZXLX8 convert AIPS bit array to FITS binary table bit array.

The following item is a second level binary data conversion routine; others may be needed in a particular implementation.

- ZBYTF2 interchange bytes in buffer if needed to go between local and standard.

15.6 Directory and Text File Routines

Text files are used for source code and other information and have been discussed previously in this chapter. There are currently two systems for access to text files one for AIPS system files and the other for general text files. The following list briefly describes the functions of the first level routines for text files; detailed descriptions of the call sequences are found at the end of this chapter.

- ZTCLOS close text file opened with ZTOPEN.
- ZTOPEN open text file - logical area, version, member name as arguments.
- ZTREAD read next 80-character record in sequential text file (ZTOPEN type).
- ZTXCLS close text file opened via ZTXOPN.
- ZTXIO read/write a line to a text file.
- ZTXMAT return list of files in specified area beginning with specified chars.
- ZTXOPN open a text file for read or write.

The following items are second level text file routines in APLGEN; others may be needed in a given implementation.

- ZDIR build a full path name to files in AIPS-standard areas (HE, RU, ...).
- ZTOPE2 open text file for ZTOPEN.
- ZTXMA2 find all file names matching a given wildcard specification.
- ZTXOP2 translate the file name and open a text file.

15.7 Television I/O

System dependent routines are usually needed to communicate with a physical display device. These routines were discussed in some detail in the chapter on displays and are only brielfy described here.

- ZARGCL close an ARGS TV device.
- ZARGMC issues a master clear to an ARGS TV.
- ZARGOP open ARGS TV device.
- ZARGXF translates IIS Model 70 commands into calls to ZARGS for ARGS TV.
- ZDEACL close DeAnza TV device.
- ZDEAMC issue a master clear to the TV - for DeAnzas this is a No-Op.
- ZDEAOP opens DeAnza TV device.
- ZDEAXF do I/O to DeAnza TV.
- ZIPACK pack/unpack long integers into short integer buffer.
- ZIVSOP opens IVAS TV device - using the IIS package.
- ZM70CL close an IIS Model 70 TV device, flushing any buffer.
- ZM70MC issues a master clear to an IIS Model 70 TV.
- ZM70OP open IIS Model 70 TV device.
- ZM70XF read/write data to IIS Model 70 TV with buffering.
- ZTTBUF reads terminal input with no prompt or wait - simulates TV trackball.
- ZV20CL close a Comtal Vision 1/20 TV device.
- ZV20MC issue a master clear to the TV - for Comtal this is a No-Op.
- ZV20OP open Comtal Vision 1/20 TV device.
- ZV20XF read/write data to Comtal Vision 1/20 TV device.

The following are second level TV I/O routines in APLGEN; in a given implementation there will be others.

- ZARGC2 close an ARGS TV device.
- ZARGO2 open ARGS TV device.
- ZARGS sends command to/from the ARGS TV device.
- ZDEAC2 close DeAnza TV device.
- ZDEAO2 opens DeAnza TV device.
- ZDEAX2 do actual read/write from/to DeAnza device.
- ZM70C2 close IIS Model 70/75 TV device.
- ZM70M2 issues a master clear to an IIS Model 70 TV.
- ZM70O2 opens IIS Model 70.75 TV device.
- ZM70X2 read/write from/to IIS Model 70/75 device.
- ZV20C2 close Comtal Vision 1/20 TV device.
- ZV20O2 opens Comtal Vision 1/20 TV device.
- ZV20X2 does I/O to Comtal Vision 1/20 TV device.

15.8 Virtual Devices

In many cases it is desirable to communicate with a physical device that is directly connected to a process that may reside on another computer. These devices are refered to here as virtual devices as the communication with them is through network protocols rather than directly. These routines communicate with a generic device as they may or may not know what physical device they are communicating with. Currently, these exist only for the display device (TV).

- ZVTVCL close connection in client (virtual-TV) to server (remote, real-TV).
- ZVTVGC close and reopen connection in server (real-TV) to client (virtual-tv).
- ZVTVOP opens connection from client (virtual-TV) to server (real-TV).
- ZVTVRC closes channel in server (real-TV) to client (virtual-TV).
- ZVTVRO open scket in server (real-TV) to any client (virtual-TV).
- ZVTVRX does IO for server (real TV) to client (Virtual-TV) incl close/reopen.
- ZVTVXF sends data from the client (virtual TV) to server (real TV).

The following are second level virtual device routines in APLGEN; in a given implementation there may be others.

- ZVTVC2 close virtual TV connection to remote, real-TV computer.
- ZVTVC3 close connection in real-TV computer to client, virtual-TV computer.
- ZVTVO2 open connection in client (virtual-TV) to server (remote, real-TV).
- ZVTVO3 open connection in server (real-TV) to client (virtual-TV).
- ZVTVX2 writes/reads to/from server for the client (virtual TV) machine.
- ZVTVX3 reads/writes from/to client (virtual TV) for the server (real TV).

15.9 Miscellaneous

Several Z routines don't fit naturally into any of the above categories. The following list gives a brief description of each; details of the call sequences and functions are given at the end of this chapter.

- ZADDR determine if 2 addresses inside computer are the same.
- ZERROR prints strings associated with system error codes for Z routines.
- ZHEX encode an integer into hexadecimal characters.
- ZKDUMP display portions of an array in various Fortran formats.
- ZMSGER prints strings associated with system error codes for ZMSG routines.
- ZMYVER returns OLD, NEW, or TST based on translation of logical AIPS_VERSION.

The following item is a second level miscellaneous routine; others may be needed in a particular implementation.

- ZERRO2 return system error message for given system error code.

15.10 INCLUDEs

There are several types of INCLUDE file which are distinguished by the first character of their name. Different INCLUDE file types contain different types of Fortran declaration statements as described in the following list.

- Pxxx.INC. These INCLUDE files contain declarations for parameters and the PARAMETER statements.
- Dxxx.INC. These INCLUDE files contain Fortran type (with dimension) declarations, COMMON and EQUIVALENCE statments.
- Vxxx.INC. These contain Fortran DATA statements.
- Zxxx.INC. These INCLUDE files contain declarations which may change from one computer or installation to another.

15.10.1 DDCH.INC

```
C                                                           Include DDCH.
C                                         AIPS system parameters
      CHARACTER SYSNAM*20, VERNAM*4, RLSNAM*8, DEVNAM(10)*48,
     *   NONNAM(8)*48, MAPNAM(12)*48, SYSTYP*4, SYSVER*8
      HOLLERITH HBLANK
      DOUBLE PRECISION DBLANK
      REAL   XPRDMM, XTKDMM, TIMEDA(15), TIMESG, TIMEMS, TIMESC, TIMECA,
     *   TIMEBA(4), TIMEAP(3), FBLANK, RFILIT(14)
      INTEGER   NVOL, NBPS, NSPG, NBTB1, NTAB1, NBTB2, NTAB2, NBTB3,
     *   NTAB3, NTAPED, CRTMAX, PRTMAX, NBATQS, MAXXPR(2), CSIZPR(2),
     *   NINTRN, KAPWRD, NWDPDP, NBITWD, NCHLIN, NTVDEV, NTKDEV, BLANKV,
     *   NTVACC, NTKACC, UCTSIZ, BYTFLP, USELIM, NBITCH, NCHPRT,
     *   KAP2WD, MAXXTK(2), CSIZTK(2), DASSGN(8,15), SPFRMT, DPFRMT,
     *   NSHORT, TTYCAR, DEVTAB(50), FTAB(1024)
      COMMON /DCHCHM/ SYSNAM, VERNAM, SYSTYP, SYSVER, RLSNAM,
     *   DEVNAM, NONNAM, MAPNAM
      COMMON /DCHCOM/ DBLANK, XPRDMM, XTKDMM, TIMEDA, TIMESG, TIMEMS,
     *   TIMESC, TIMECA, TIMEBA, TIMEAP, FBLANK, RFILIT, HBLANK,
     *   NVOL, NBPS, NSPG, NBTB1, NTAB1, NBTB2, NTAB2, NBTB3, NTAB3,
     *   NTAPED, CRTMAX, PRTMAX, NBATQS, MAXXPR, CSIZPR, NINTRN,
     *   KAPWRD, NWDPDP, NBITWD, NCHLIN, NTVDEV, NTKDEV, BLANKV,
     *   NTVACC, NTKACC, UCTSIZ, BYTFLP, USELIM, NBITCH, NCHPRT,
     *   KAP2WD, MAXXTK, CSIZTK, DASSGN, DEVTAB, SPFRMT, DPFRMT,
     *   NSHORT, TTYCAR
      COMMON /FTABCM/ FTAB
C                                                           End DDCH.
```

15.10.2 DMSG.INC

```
C                                                           Include DMSG.
C                                         AIPS system message common
      INTEGER   MSGCNT, NPOPS, NLUSER, NACOUN, MSGSUP, MSGREC,
     *   MSGKIL, ISBTCH, DBGAIP, MSGDM1, MSGDM2, MSGDM3
      CHARACTER MSGTXT*80, TSKNAM*6
      COMMON /MSGCOM/ MSGCNT, NPOPS, NLUSER, NACOUN, MSGSUP, MSGREC,
     *   MSGKIL, ISBTCH, DBGAIP, MSGDM1, MSGDM2, MSGDM3
      COMMON /MSGCHR/ MSGTXT, TSKNAM
C                                                           End DMSG.
```

15.11 Routines

15.11.1 SYSTEM

The following describes the first level System "Z" routines as documented in the APLGEN directory.

ZABORT

Will take one of two actions depending on the value of "action". If "action" is zero, it establishes abort handling to clean up for programs in the event of an ABORTASK or otherwise fatal signal. Hangup, interrupt (except for AIPS or BATER) and quit are ignored. If running under the control of a debugger, it simply returns (to avoid affecting the debugger signal handling). If "action" is non-zero, it will issue an illegal instruction in order to induce the abort handler. This is mostly to get a traceback for debugging purposes (e.g., when an invalid argument to a subroutine is detected).

```
ZABORT (TSKNAM, ACTION)
Inputs:
   TSKNAM   C*6   Program name
   ACTION   I     Action indicator code:
                     0 => establish abort handling
                     1 => invoke an illegal instruction
Output:
   none
Generic version - calls ZABOR2 after tests on TSKNAM.
```

ZACTV8

Activate the specified program and return the activated process identification information.

```
ZACTV8 (NAME, INPOPS, RVERSN, PID, IERR)
Inputs:
   NAME     C*6    Program name
   INPOPS   I      POPS # for the task to use
   RVERSN   C*48   Logical name or fully qualified name of the
                   directory from which to get the required
                   executable module
In/Out:
   PID      I(4)   Process identification information used directly
                   by subsequent calls to ZTACTQ where
                   In:  PID(1)   = user number for systems that use
                                   it (= 0 otherwise and on all
                                   AIPSB invocations)
                   Out: PID(2-4) = process ID number(s) as needed
Output:
   IERR     I      Error return code: 0 => no error
                      1 => program name too long
                      2 => activation error
Common: DMSG.INC
   DBGAIP   I      > 10 => start tasks in DEBUG mode if possible
Generic version = stub
```

ZCPU

Determines cumulative cpu usage in seconds for this process: i.e. each time a process calls ZCPU during an execution, TIME is larger.

```
ZCPU (TIME, IOCNT)
Output:
   TIME    R        Current CPU accumulation in seconds
   IOCNT   I        IO count accumulation
Generic version - stub
```

ZDATE

Returns local date

```
ZDATE (LDATE)
Output:
   LDATE   I(3)   local date where
                     LDATE(1)   year since 0.
                     LDATE(2)   month (1-12).
                     LDATE(3)   day   (1-31).
Generic version - stub
```

ZDCHIN

Initialize the device characteristics common and the FCB's (file control blocks) in FTAB(∗) for the maximum number of different file types that can be open at the same time. Initialize also other machine-dependent commons and the message common. Note that the task name is not set here.

```
ZDCHIN (DODISK, JOBLK)
Inputs:
   DODISK   L        Get SETPAR-controlled parameters from disk
Inputs from common: DMSG.INC
   TSKNAM   C*6      Task name if known - else ' ' (used in ABORT
                     handler mostly to separate standalones and
                     tasks)
Output:
   JOBLK    I(256)   I/O block - no longer used
Output in commons: DDCH.INC DMSG.INC
   all      ...      All values set to init except TSKNAM

ZDCHIN starts with hard-coded values.  Then, if DODISK is true,
resets those contained in the system parameter file.  The utility
program SETPAR is used to alter the system parameter file values.

Critical system constants (all "words" are local integers, all
"bytes" are AIPS-bytes, i.e., 1/2 a local integer and on 64 bit
architectures, double precision contructs should be preprocessed
into their single precision counterparts):
Generic version - calls ZDCHI2, ZDCHIC (init C codes).
```

ZDELAY

Cause an execution delay for a specified time interval.

```
ZDELAY (SECS, IERR)
Inputs:
   SECS   R   Number of seconds to delay
Output:
```

```
    IERR   I   Error return code: 0 => no error
                  1 => error
 Generic version - calls ZDELA2 protecting input argument.
```

ZFREE

Determine the number of free 256-integer blocks that are available on the disks used for AIPS user data and print the information on the user's terminal.

```
ZFREE (MSLEV, IERR)
Inputs:
   MSLEV   I   Message level to use (1 => to terminal only,
                  2-5 => also to message file)
Common /DCHCOM/
   NVOL    I   Number of AIPS disks
Output:
  IERR     I   Error return code: 0 => no error
                  1 => error
Generic version - uses ZFRE2 to get info.
```

ZGNAME

Get the name of the current process.

```
ZGNAME (NAME, IERR)
Output:
   NAME   C*6   Current process name
   IERR   I     Error return code: 0 => no error
                   1 => error
Generic version - stub
```

ZPRIO

Change the execution priority of the current process between that of batch and interactive processes.

```
ZPRIO (OP, IERR)
Inputs:
   OP     C*4   'UPPP' to interactive, 'DOWN' to batch
   IERR   I     Error return code: 0 => no error
                   1 => invalid opcode
                   2 => illegal request
                   3 => other
Generic version - uses ZPRI2 to do the real work.
```

ZSETUP

Performs any needed system level operations which can be performed only after VERNAM, TSKNAM, and NPOPS have been determined.

```
ZSETUP
(no call arguments)
Generic version  - only lowers priority of batch jobs from that of
interactive.  If batch are started at the interactive priority,
then they can usually go back to interactive when needed such as
while they possess a valuable device (e.g., a real AP).
```

ZSTAIP

Performs any operations needed to normalize the local operating system at the conclusion of an interactive AIPS session.

```
ZSTAIP (SCRTCH)
Outputs:
   SCRTCH   I(256)   Scratch buffer

Generic version  - calls ZSTAI2 in case there is something to do
in C or Macro, else a null routine.
```

ZTACTQ

Determines if a specified task is active.

```
ZTACTQ (NAME, PID, ACTIVE, IERR)
Inputs:
   NAME     C*6    Actual task name.
In/out:
   PID      I(4)   Process "ID" code
                   In:  PID(1) user number  0 => any
                        PID(2-4) process #  0 => unknown
                   Out: PID(1) not changed
                        PID(2-4) the ID determined here from NAME
Output:
   ACTIVE   L      Task active indicator (T => active)
   IERR     I      Error return code: 0 => ok.
                      2 => invalid task name.
                      3 => other
Generic version - calls ZTACT2.
```

ZTIME

returns the local time of day

```
ZTIME (ITIME)
Output:
   ITIME   I*2(3)   Local time where: ITIME(1) = hour (0-24)
                                      ITIME(2) = minute (0-60)
                                      ITIME(3) = second (0-60)
Generic version - stub
```

ZTKILL

Will delete/kill the process specified by NAME.

```
ZTKILL (NAME, PID, IERR)
Inputs:
   NAME   C*6    actual task name
   PID    I(4)   Process "ID" code:
                    PID(1) user number  0 => any user
                    PID(2-4) process #  0 => unknown
Output:
   IERR   I      Error number: 0 => ok.
                    1 => error.
Generic version - stubbed pending development.
```

ZTQSPY

Displays information on the user's terminal regarding AIPS account originated processes or all processes running on the system.

```
ZTQSPY (DOALL, TLIST)
Inputs:
   DOALL   R        > 0.0 => display all processes
Output:
   TLIST   I(256)   Scratch buffer (not used in generic)
Generic version - uses ZTQSP2.
```

ZTRLOG

Translate a logical name (i.e., environment variable). NOTE: This routine is ONLY for use by other Z-routines.

```
 ZTRLOG (LLEN, LOGNAM, XLEN, XLATED, XLNB, IERR)
Inputs:
   LLEN     I          Length of LOGNAM (1-relative)
   LOGNAM   C*(LLEN)   Logical name
   XLEN     I          Length of XLATED (1-relative)
Output:
   XLATED   C*(XLEN)   Translation (blank filled)
   XLNB     I          Position of last non-blank in XLATED
                       (1-relative)
   IERR     I          Error return code: 0 => no error
                           1 => error
Generic version - stub
```

ZWHOMI

Determines the actual task name under which the present version of AIPS is running. It uses this information to set the value of NPOPS in the common /MSGCOM/. It then assigns the TV and TK devices setting NTVDEV and NTKDEV in include DDCH.INC.

```
ZWHOMI (IERR)
Output:
   IERR   I   Error return code: 0 => no error
                 1 => process name is not AIPSx
                 2 => illegal POPS number
                 3 => other error
Generic version - uses ZGNAME, ZTRLOG
```

15.11.2 Disk I/O

The following describes the first level Disk I/O "Z" routines as documented in the APLGEN directory.

ZCLOSE

Close the file associated with LUN removing any exclusive use state and clear the FTAB entry for the LUN.

```
ZCLOSE (LUN, FIND, IERR)
Inputs:
   LUN      I           Logical unit number
   FIND     I           Index in FTAB to file control block for LUN
```

```
Output:
   IERR     I           Error return code: 0 => no error
                           1 => close error
                           2 => file already closed in FTAB
                           3 => both errors
                           4 => erroneous LUN
Generic version.
No longer closes TV devices as of the 15OCT87 release.
No longer closes tape devices as of the 15APR87 release.
No longer closes Tektronix devices as of the 15OCT87 release.
No longer closes text files as of the 15OCT87 release.
```

ZCMPRS

Releases unused disk space from the end of an open disk file. "Byte" defined as 1/2 of a integer.

```
ZCMPRS (IVOL, PNAME, LUN, LSIZE, SCRTCH, IERR)
Inputs:
   IVOL    I        volume number
   PNAME   C*48     physical file name
   LUN     I        logical unit number under which file is open.
In/Out:
   LSIZE   I        (In) desired final size in AIPS bytes
                    (Out) actual final size in AIPS bytes
Outputs:
   SCRTCH  I(256)   scratch buffer (not used under UNIX).
   IERR    I        error code: 0 => ok
                                1 => input data error
                                2 => compress error
Generic version - uses ZCMPR2.
```

ZCREAT

Create a disk file of a specified name and size reserving the disk space.

```
ZCREAT (IVOL, PNAME, RSIZE, MAP, ASIZE, SCRTCH, IERR)
Inputs:
   IVOL     I        Disk volume containing file
   PNAME    C*48     Physical file name
   RSIZE    I        Requested size of the file in AIPS-bytes (1/2
                     of a local integer)
   MAP      L        Is this a "map" file?
Output:
   ASIZE    I        Actual size of file in AIPS-bytes
   SCRTCH   I(256)   Scratch buffer
   IERR     I        Error return code: 0 => no error
                        1 => file already exists
                        2 => volume not found
                        3 => insufficient space
                        4 => other
                        5 => forbidden (reserved)
Generic version - uses ZCREA2
```

ZDESTR

Destroy (i.e., delete) a file. The file should already be closed.

```
ZDESTR (IVOL, PNAME, IERR)
Inputs:
   IVOL    I      Disk volume containing file, 1,2,3,...
   PNAME   C*48   Physical file name (left justified)
Output:
   IERR    I      Error return code: 0 => no error
                     1 => file not found (no message)
                     2 => device not found
                     3 => file in use
                     4 => other
Generic version.
```

ZEXIST

Determine if a file exists and if so, return its size in AIPS-bytes (1/2 of a local integer).

```
ZEXIST (IVOL, PNAME, ISIZE, SCRTCH, IERR)
Inputs:
   IVOL     I      Disk volume containing file, 1,2,3,...
   PNAME    C*48   Physical file name
Output:
   ISIZE    I      File size in AIPS-bytes (1/2 of a local integer)
   SCRTCH   I(*)   I/O buffer
   IERR     I      Error return code: 0 => file exists
                      1 => file does not exist
                      2 => inputs error
                      3 => other error
Generic version - SCRTCH not used, calls ZEXIS2.
```

ZEXPND

Increase the size of a disk file — it must be open

```
ZEXPND (LUN, IVOL, PNAME, NREC, IERR)
Inputs:
   LUN     I      LUN of file open file
   IVOL    I      Disk volume containing file, 1,2,3,...
   PNAME   C*48   Physical file name
In/Out:
   NREC    I      # 256-integer records requested/received
Output:
   IERR    I      Error return code: 0 => no error
                     1 => input error
                     2 => expansion error
                     3 => ZEXIST error
Generic version - uses ZEXPN2.
```

ZFIO

Transfer one logical record between an I/O buffer and device LUN. For disk devices, the record length is always 256 local integers and NREC is the random access record number. For non-disk devices, NREC is the number of 8-bit bytes.

```
ZFIO (OPER, LUN, FIND, NREC, BUFF, IERR)
Inputs:
   OPER   C*4      Operation code 'READ' or 'WRIT'
   LUN    I        Logical unit number
   FIND   I        Index in FTAB to file control block for LUN
   NREC   I        Random access record number (1-relative) for
                   disk transfers or number of 8-bit bytes for
                   sequential device transfers (e.g., Tektronix
                   terminals)
   BUFF   I(256)   I/O buffer
Output:
   IERR   I        Error return code: 0 => no error
                      1 => file not open
                      2 => input error
                      3 => I/O error
                      4 => end of file
Generic version.
No longer performs I/O to TV devices as of the 15MAR84 release.
No longer performs I/O to tape devices as of the 15APR87 release.
Only performs disk and Tektronix device I/O now.
```

ZFULLN

Convert file name from user specified name with or without a logical to a full path and file name with the logical translated.

```
ZFULLN (UNAME, DEFLOG, SUBR, FNAME, IERR)
Inputs:
   UNAME    C*(*)     User specified name - first a logical name,
                      then a colon, then a file name (with or
                      without a "." extension).  Logical optional.
                      ' ' => create temporary file (DEFLOG not ' ')
   DEFLOG   C*(*)     Default logical: ' ' => no default
                      '-1' => UNAME must provide a logical
   SUBR     C*6       Subroutine name to be used in creating a
                      "temporary" file name.  ' ' => no temporary
Output:
   FNAME    C*(*)     Full file name
   IERR     I         Error code: 0 => okay
                         1 => improper combination of inputs
                         2 => no translation for logical
                         3 => FNAME too short
                         4 => logical required, not provided
Generic version - may not need to make any machine-specific versions
unless there is something really special/odd about logical names.
```

ZMIO

Low level random access, large block, double buffered device I/O.

```
ZMIO (OPER, LUN, FIND, BLKNO, NBYTES, BUFF, IBUFF, IERR)
Inputs:
   OPER     C*4   Operation code 'READ' or 'WRIT'
   LUN      I     Logical unit number
```

```
   FIND     I     Index in FTAB to file control block for LUN
   BLKNO    I     Beginning virtual block number (1-relative).
                  Block size is given by NBPS in /DCHCOM/.
   NBYTES   I     Number of AIPS-bytes to transfer (an AIPS-byte is
                  1/2 a local integer).
   IBUFF    I     Buffer number to use (1 or 2)
In/out:
   BUFF     I(*)  I/O buffer
Output:
   IERR     I     Error return code: 0 => no error
                     1 => file not open
                     2 => input error
                     3 => I/O error
                     4 => end of file
Generic version - uses ZMI2.
No longer performs I/O to TV devices (15MAR84), to tape devices
(15APR87), and to Tektronix devices (15JUL87) release.
```

ZMKTMP

Form a unique, fully qualified, temporary file name.

```
ZMKTMP (FLEN, FILNAM, IERR)
Inputs:
   FLEN     I       Length of "filnam"
In/Out:
   FILNAM   C*(*)   File name with the extension .XXXXXX, (e.g.,
                    "ZXLPRT.XXXXXX").  The extension will be
                    transformed by 'mktemp' to make "filnam" a
                    unique file name.
Output:
   IERR     I       Error return code: 0 => no error
                       1 => inputs wrong
                       3 => filnam too short
Generic version - stub
```

ZMSGCL

Close a message file associated with LUN removing any exclusive use state and clear the FTAB entry for the LUN (much like ZCLOSE but does not call MSGWRT to avoid recursion).

```
ZMSGCL (LUN, FIND, IERR)
Inputs:
   LUN     I   Logical unit number
   FIND    I   Index in FTAB to file control block for LUN
Output:
   IERR    I   Error return code: 0 => no error
                  1 => close error
                  2 => file already closed in FTAB
                  3 => both errors
                  4 => erroneous LUN
Generic version - uses ZDACLS and ZTTCLS.
```

ZMSGDK

Transfer one 256-integer record to/from message disk file. Like ZFIO, but does not call MSGWRT to avoid recursion.

```
ZMSGDK (OPER, LUN, FIND, NREC, BUFF, IERR)
Inputs:
   OPER     C*4     Operation code 'READ' or 'WRIT'
   LUN      I       Logical unit number
   FIND     I       Index in FTAB to file control block for LUN
   NREC     I       Random access record number (1-relative)
                    In units of 256-integer records.
In/out:
   BUFF     I(256)   I/O buffer
Output:
   IERR     I        Error return code: 0 => no error
                        1 => file not open
                        2 => input error
                        3 => I/O error
                        4 => end of file
Generic version - calls ZFI2
```

ZMSGOP

Open a message file (much like ZOPEN, but does not call MSGWRT to avoid recursion). Non-map, exclusive, patient assumed on open.

```
ZMSGOP (LUN, FIND, IVOL, PNAME, IERR)
Inputs:
   LUN      I      Logical unit number
   IVOL     I      Disk volume containing file, 1,2,3,...
   PNAME    C*48   Physical file name
Output:
   FIND     I      Index in FTAB to file control block for LUN
   IERR     I      Error return code: 0 => no error
                      1 => LUN already in use
                      2 => file not found
                      3 => volume not found
                      4 => exclusive use denied
                      5 => no room for LUN in FTAB
                      6 => other open errors
Generic version - calls ZDAOPN, ZTTOPN, ZMSGER.
```

ZMSGXP

Increase the size of a message file (special version of ZEXPND that writes error messages to the terminal only in order to avoid recursion).

```
ZMSGXP (LUN, IVOL, PNAME, NREC, IERR)
Inputs:
   LUN     I      LUN of file open message file (must be 12)
   IVOL    I      Disk volume containing file, 1,2,3,...
   PNAME   C*48   Physical file name
In/Out:
   NREC    I      # 256-integer records requested/received
```

```
Output:
   IERR    I      Error return code: 0 => no error
                     1 => input error
                     2 => expansion error
                     3 => ZEXIS2 error
Generic version - calls ZEXPN2, ZEXIS2.
```

ZOPEN

Open a binary disk file, line printer or terminal Message files, text files, tape devices, Tektronix devices and TV devices are NOT opened using this routine (see ZMSGOP for message files, ZTOPEN for text files, ZTPOPN for tape devices, ZTKOPN for Tektronix devices and the device specific routine for TV devices, e.g., ZM70OP).

```
ZOPEN (LUN, FIND, IVOL, PNAME, MAP, EXCL, WAIT, IERR)
Inputs:
   LUN     I      Logical unit number
   IVOL    I      Disk volume containing file, 1,2,3,...
   PNAME   C*48   Physical file name (from ZPHFIL)
                  For line printers (LUN=1), the output text file
                  to be kept (' ' => use a scratch)
   MAP     L      Is this a "map" file?
   EXCL    L      Exclusive use requested?
   WAIT    L      Wait for exclusive use?
Output:
   FIND    I      Index in FTAB to file control block for LUN
   IERR    I      Error return code: 0 => no error
                     1 => LUN already in use
                     2 => file not found
                     3 => volume/logical not found
                     4 => exclusive use denied
                     5 => no room for LUN in FTAB
                     6 => other open errors
Generic version - uses ZDAOPN, ZLPOPN, ZTTOPN
No longer opens TV devices as of the 15MAR84 release.
No longer opens tape devices as of the 15APR87 release.
No longer opens Tektronix devices as of the 15OCT87 release.
No longer opens text files as of the 15OCT87 release.
```

ZPHFIL

Construct a physical file name in PNAM from TYPE, IVOL, NSEQ, and IVER - either for public data files or user-specific files.

```
ZPHFIL (TYPE, IVOL, NSEQ, IVER, PNAM, IERR)
Inputs:
   TYPE    C*2   Type of file: e.g. 'MA' for map file
   IVOL    I     Number of the disk volume to be used (1-15)
   NSEQ    I     Sequence number (000-4095)
   IVER    I     Version number (00-255)
Outputs:
   PNAM    C*48  physical file name, left justified
   IERR    I     Error return code: 0 = good return.   1 = error.
```

```
Example: If TYPE='MA', IVOL=7, AIPSVER='C', NSEQ=321, IVER=99,
         NLUSER=762 then
            PNAME='DA07:MAC14163;1'      for public data or
            PNAME='DA07:MAC14163.2FA;1'  for private data
         where 321 = 141 base 16, 99 = 63 base 16, 762 = 2FA base 16

TYPE = 'MT' leads to special name for tapes
TYPE = 'TK' leads to special name for TEK4012 plotter CRT
TYPE = 'TV' leads to special name for TV device
TYPE = 'ME' leads to special logical for POPS memory files

Generic version - the ZOPEN, ZTRLOG, etc. routines interpret the
resulting VMS-like names.
```

ZPHOLV

Construct a physical file name in PNAM from TYPE, IVOL, NSEQ, and IVER - either for public data files or user-specific files. Version of ZPHFIL for UPDAT, in which the Format version codes are passed as arguments.

```
ZPHOLV (VERDAT, VERSYS, TYPE, IVOL, NSEQ, IVER, PNAM, IERR)
Inputs:
   VERDAT  C*1   Data file version code letter
   VERSYS  C*1   System file version code letter
   TYPE    C*2   Type of file: e.g. 'MA' for map file
   IVOL    I     Number of the disk volume to be used (1-15)
   NSEQ    I     Sequence number (000-4095)
   IVER    I     Version number (00-255)
Outputs:
   PNAM    C*48  physical file name, left justified
   IERR    I     Error return code: 0 = good return.   1 = error.

Example: If TYPE='MA', IVOL=7, AIPSVER='C', NSEQ=321, IVER=99,
         NLUSER=762 then
            PNAME='DA07:MAC14163;1'      for public data or
            PNAME='DA07:MAC14163.2FA;1'  for private data
         where 321 = 141 base 16, 99 = 63 base 16, 762 = 2FA base 16

TYPE = 'MT' leads to special name for tapes
TYPE = 'TK' leads to special name for TEK4012 plotter CRT
TYPE = 'TV' leads to special name for TV device
TYPE = 'ME' leads to special logical for POPS memory files

Generic version - the ZOPEN, ZTRLOG, etc. routines interpret the
resulting VMS-like names.
```

ZRENAM

Rename a disk file.

```
ZRENAM (IVOL, OLDNAM, NEWNAM, IERR)
Inputs:
   IVOL     I      Disk volume containing files, 1,2,3,...
   OLDNAM   C*48   Old physical file name
```

```
   NEWNAM   C*48   New physical file name
Output:
   IERR     I      Error return code: 0 => ok
                      2 => old file not found
                      3 => volume not found
                      4 => old file busy
                      6 => new file name already exists
                      7 => other
Generic version.
```

ZTFILL

Fills in initial values in the FTAB file control block.

```
ZTFILL (FIND, MAP)
Inputs:
   FIND   I   Index in FTAB to file control block
   MAP    L   Map file indicator
Common: via DDCH.INC
   FTAB   I(*)   input:  FTAB(FIND) = LUN
                 output: FTAB(FIND+1...) = ? whatever is needed
Generic version - zeros the table.
```

ZWAIT

Wait until an asynchronous I/O operation completes.

```
ZWAIT (LUN, FIND, IBUFF, IERR)
Inputs:
   LUN     I   Logical unit number
   FIND    I   Index in FTAB to file control block for LUN
   IBUFF   I   Buffer # to wait for (1 or 2)
Output:
   IERR    I   Error return code: 0 => no error
                  1 => LUN not open in FTAB
                  2 => error in inputs
                  3 => I/O error
                  4 => end of file
                  7 => wait service error
Generic version: uses ZWAI2
```

15.11.3 Non-disk I/O routines

The following describes the first level non–disk I/O “Z” routines as documented in the APLGEN directory.

ZBKLD1

Routine to initialize the host environment in preparation for execution of ZBKLD2 under task BAKLD.

```
ZBKLD1 (IERR)
Output:
   IERR   I   error code : 0 => okay
                 1 => can't create listing subdirectory
                 2 => input error - translates fail
Generic version - stub
```

ZBKLD2

Host-dependent routine to process input tape for task BAKLD. The input tape is presumed to have been produced by task BAKTP executing on the same host/OS combination. Data format is the hosts 'backup' utility (BACKUP on VMS, 'tar' on Unix).

```
ZBKLD2 (OP, IERR)
Inputs:
   OP    C*4   'SKIP' skips over a saveset.
               'PRNT' moves over a saveset, listing directory info.
               'LOAD' loads a saveset.
Output:
   IERR  I     Error return
Generic version - stub.
```

ZBKLD3

Routine to clean up host environment after executing task BAKLD.

```
ZBKLD3 (IERR)
Output:
   IERR   I   Error code: 0 => okay
                 1 => can't open command file
Generic version - stub.
```

ZBKTP1

Initialize the host environment in preparation for ZBKTP2. If OPCODE = 'INIT', initializes the tape too. (used only by task BAKTP)

```
ZBKTP1 (ZLUN, ZBKNAM, BAKTXT, NTAPE, ZMTON, LZMTON, IERR)
Inputs:
   NTAPE    I       Tape drive number
   BAKTXT   C*(*)   lowest file name for file listing (not used
                    here, only in VMS now)
Outputs:
   ZLUN     I       LUN to use
   ZBKNAM   C*(*)   Command file name
   ZMTON    C*(*)   Tape name (translation of MT0n) n=NTAPE-1
   LZMTON   I       Actual length used in ZMTON
   IERR     I       Error code: 0 => okay
Generic stub
```

ZBKTP2

Host-dependent subroutine to write a file and its extensions to tape using the hosts "backup" utility.

```
ZBKTP2 (ZLUN, ZBKNAM, BAKTXT, BAKREC, IVOL, ICNO, NTAPE, ZMTON,
  *   LZMTON, IERR)
Inputs:
   ZLUN     I       LUN for command file
   ZBKNAM   C*(*)   Name of command file
   BAKTXT   C*(*)   lowest file name for file listing
   BAKREC   C*60    ?
   IVOL     I       Disk number
```

```
   ICNO     I       Catalog number
   NTAPE    I       Tape drive number
   ZMTON    C*(*)   Tape name (translation of MT0n) n=NTAPE-1
   LZMTON   I       Actual length used in ZMTON
Output:
   IERR     I       Error code: 0 => okay, else ZSHCMD
Generic version - stub
```

ZBKTP3

Clean up host environment after BAKTP execution.

```
ZBKTP3 (ZLUN, ZBKNAM, IERR)
Inputs:
   ZLUN     I       LUN for command file
   ZBKNAM   C*(*)   Name of command file
Output:
   IERR     I       Error code: 0 => okay, else ZSHCMD
Generic version - stub
```

ZDOPRT

Read a bit map such as produced by PRTDRW and convert it into a file that can be spooled to a Versatec printer/plotter.

```
ZDOPRT (IVOL, LUN, NCOPY, PNAME, ISIZE, INBLK, IERR)
Inputs:
   IVOL     I      Disk volume containing file, 1,2,3,...
   LUN      I      Logical unit number
   NCOPY    I      Number of copies of the plot to make
   PNAME    C*48   Physical file name (left justified)
   ISIZE    I      Size of INBLK in words
In/Out:
   INBLK    I(*)   Scratch buffer
Output:
   IERR     I      Error return code: 0 => no error
                      1 => error
Generic version - a stub
```

ZENDPG

Advance the line printer to avoid “burn-out” on electro-static type devices.

```
ZENDPG (LINE)
Inputs:
   LINE     I           # lines printed on page so far
Generic version, does a partial page.
```

ZLASCL

Close and spool a file for printing a plot on a laser printer.

```
ZLASCL (FILNAM, LUN, DELFIL, SYSERR, IERR)
Inputs:
   FILNAM   C*48   print/plot file name
   LUN      I      LUN under which file is open
   DELFIL   I      1 => delete the file after print, 0 => keep it
Output
   SYSERR   I      system error code (on IERR 2)
   IERR     I      error code: 0 => okay, 1 => error in queue,
                      2 => Fortran close error
Generic version - calls ZLASC2 for any special stuff after a Fortran
close.
```

ZLASIO

Open, write to or close and spool a file for printing a plot on a laser printer.

```
ZLASIO (OP, LUN, OUTFIL, NCHAR, CBUFF, IERR)
Inputs:
   OP       C*4    Operation code: 'OPEN', 'POPN', 'WRIT' or 'CLOS'
   LUN      I      Logical unit number for the laser printer
   OUTFIL   C*48   Output file name (used by opcode 'OPEN', 'POPN',
                   'CLOS')
   NCHAR    I      Number of characters to print in CBUFF
                   (used by opcode 'WRIT' only)
   CBUFF    C*(*)  I/O buffer (used by opcode 'WRIT' only)
Output:
   IERR     I      Error return code: 0 => no error
                      1 => invalid opcode
                      2 => trouble translating logical
                      3 => I/O error
Generic version - uses ZFULLN, ZLASOP, ZLASCL
```

ZLPCLS

Close a line printer file, spool it to a printer and delete it.

```
ZLPCLS (LUN, IERR)
Inputs:
   LUN      I   Logical unit number
Output:
   IERR     I   Error return code: 0 => no error
                   1 => close error
Generic version - closes file, calls ZLPCL2
```

ZLPOPN

Open a line printer file.

```
ZLPOPN (LPFILE, IERR)
Inputs:
   LPFIL   C*48   File name to use for line printer file
Output:
   IERR    I      Error return code: 0 => no error
                     1 => error, 2 => LPFILE already exists
Generic version - uses ZFULLN, ZLPOP2.
```

ZLWIO

Open a temporary file for printing a plot on a PostScript printer using the using the name ZLWIO.XXXXXX where XXXXXX is a unique extension (OP = 'OPEN'), write data to the temporary file (OP = 'WRIT'), or close, print and delete the file (OP='CLOS').

```
ZLWIO (OP, LUN, NCHAR, CBUFF, IERR)
Inputs:
   OP      C*4     Operation code ('OPEN', 'WRIT' or 'CLOS')
   LUN     I       Logical unit number
   NCHAR   I       Number of characters to print ('WRIT' only)
   CBUFF   C*(*)   I/O buffer ('WRIT' only)
Output:
   IERR    I       Error return code: 0 => no error
                      1 => input error
                      3 => no such logical device or
                           forming temporary file name
                      6 => I/O error
Generic version - calls ZFULLN, ZLWOP, ZLASCL
```

ZLWOP

Open a file for printing a plot on a QMS Lasergraphix device.

```
ZLWOP (OP, LUN, FILNAM, IERR)
Inputs:
   OP       C*4     Operation code: 'OPEN', 'POPN'
   LUN      I       Logical unit number for the QMS device
   FILNAM   C*(*)   File name
Output
   IERR     I       Error code: 0 => okay
Generic version - simple Fortran open.
```

ZMOUNT

Issue software mount or dismount for a given tape drive.

```
ZMOUNT (MOUNT, IDRIVE, IDENS, IERR)
Inputs:
   MOUNT    L   .TRUE. means mount, .FALSE. means dismount
   IDRIVE   I   Tape drive number
   IDENS    I   Density at which to mount tape (800, 1600 or 6250)
Output:
   IERR     I   Error return code: 0 => no error
                   1 => error
Generic version - calls ZMOUN2 to do real work
```

ZPRMPT

Prompts user on CRT screen and reads a line.

```
ZPRMPT (IPC, BUFF, IERR)
Input:
   IPC     I      prompt character.
Output:
```

```
      BUFF    C*80   line of user input.
      IERR    I      error code:   0 => ok.
                        1 => read/write error.
   Generic version - stub
```

ZPRPAS

Prompts the user on his terminal with the prompt string "Password:" and then reads back a 12-character "password" without anything being visible on the screen.

```
   ZPRPAS (PASS, BUFF, IERR)
   Outputs:
      PASS   C*12     Password - 12 unpacked characters: left
                                 justigied and blank filled.
      BUFF   I(256)   Scratch buffer (if needed)
      IERR   I        Error code: 0 => ok
                         ??? => I/O error of some sort
   Generic version - stub
```

ZTAPE

Performs standard tape manipulating functions.

```
   ZTAPE (OP, LUN, FIND, COUNT, IERR)
   Inputs:
       OP      C*4   Operation to be performed.
                     'ADVF' = advance file marks
                     'ADVR' = advance records
                     'BAKF' = backspace file marks.
                     'BAKR' = backspace records.
                     'DMNT' = dismount tape.
                     'MONT' = mount tape.
                     'REWI' = rewind the tape on unit LUN
                     'WEOF' = write end of file on unit LUN: writes 4
                              EOFs, positions tape after first one
                     'MEOF' = write 4 EOF marks on tape, position tape
                              before the first one
       LUN     I     logical unit number
       FIND    I     FTAB pointer.  Drive number for MOUNT/DISMOUNT.
       COUNT   I     Number of records or file marks to skip. On MOUNT
                     this value is the density.
   Outputs:
       IERR  I       Error return: 0 => ok
                        1 = File not open
                        2 = Input specification error.
                        3 = I/O error.
                        4 = End Of File
                        5 = Beginning Of Medium
                        6 = End Of Medium
   Generic version - uses ZTAP2 and ZMOUNT.
```

ZTAPIO

Tape operations for IMPFIT (compressed FITS transport tape).

```
ZTAPIO (OPER, LN, NAME, FD, BYTREQ, BYTRED, BUF,
  *   SYSERR)
Inputs:
   OPER     C*4    'OPEN', 'READ', 'CLOS'
   LN       I      length of name
   NAME     C*(*)  physical file name
   BYTREQ   I      bytes to be read
In/out:
   FD       I(*)   file descriptor (set on OPEN, else used)
Output:
   BYTRED   I      Bytes read on READ
   BUF      I(*)   Data buffer read
   SYSERR   I      System error code
Generic version - stub
```

ZTKBUF

Will flush the Tektronix output buffer TKBUFF, if necessary, then store the low order 8-bit byte of IN into the proper 8-bit byte of TKBUFF. This is a Z-routine to allow for any required local conversions.

```
ZTKBUF (IN, IT, FIND, IERR)
Inputs:
   IN       I           Word from which to extract the low order
                        byte and store in TKBUF
   IT       I           Data type indicator:
                           1 => control
                           2 => position
                           3 => char
   FIND     I           Index in FTAB to file control block for
                        Tektronix device LUN
Output:
   IERR     I           Error return code:
                           0 => no error
                         > 1 => write error (from TEKFLS)
Common: (DTKS.INC)
   TKSIZE   I           Size of TKBUFF in floating point words
   TKBUFF   R(TKBUFF)   Tektronix output buffer
   TKPOS    I           Current byte position in TKBUFF
Generic version.
```

ZTKCLS

Close a Tektronix device

```
ZTKCLS (LUN, FIND, IERR)
Inputs:
   LUN      I   Logical unit number
   FIND     I   Index in FTAB to file control block for LUN
Output:
   IERR     I   Error return code: 0 => no error
                   1 => Non-zero ZTKCL2 error
                   2 => Non-zero LSERCH error
                   3 => both 1 and 2
                   4 => invalid LUN
Generic version -  calls ZTKCL2 to perform the actual close.
```

ZTKFI2

Read/write "nbytes" of data from/to a Tektronix terminal.

```
ZTKFI2 (OPER, FCB, BUFF, NBYTES, IERR)
Inputs:
   OPER     C*4    Operation code "READ" or "WRIT"
   FCB      I(*)   File control block for opened Tektronix
   NBYTES   I      Number of 8-bit bytes to be transferred
In/out:
   BUFF     I(*)   I/O buffer
Output:
   IERR     I      Error return code: 0 => no error
                      2 => bad opcode
                      3 => I/O error
                      4 => end of file
Generic version - stub
```

ZTKOPN

Open a Tektronix device (calls ZTKOP2 to perform the actual open).

```
ZTKOPN (LUN, FIND, IERR)
Inputs:
   LUN    I   Logical unit number
Output:
   FIND   I   Index in FTAB to file control block for LUN
   IERR   I   Error return code: 0 => no error
                 1 => LUN already in use
                 2 => no such logical device
                 3 => device not found
                 4 => exclusive use denied
                 5 => no room for LUN in FTAB
                 6 => other open errors
Generic version.
```

ZTPCLS

Close the tape drive associated with LUN as well as its disk control file removing any exclusive use state and clear the corresponding FTAB entries. ZTPCL2 actually closes the tape drive and ZDACLS is called to close the disk control file. Also closes sequential type disk files via ZTPCLD.

```
ZTPCLS (LUN, FIND, IERR)
Inputs:
   LUN    I   Logical unit number
   FIND   I   Index in FTAB to file control block for LUN
Output:
   IERR   I   Error return code: 0 => no error
                 1 => close error
                 2 => non-zero LSERCH error
                 3 => both 1 and 2
                 4 => invalid LUN
Generic version.
```

ZTPMIO

Low level sequential access, large record, double buffered tape device I/O.

```
ZTPMIO (OPER, LUN, FIND, NBYTES, BUFF, IBUFF, IERR)
Inputs:
   OPER     C*4    Operation code 'READ' or 'WRIT'
   LUN      I      Logical unit number
   FIND     I      Index in FTAB to file control block for LUN
   NBYTES   I      Number of 8-bit bytes to transfer
   BUFF     I(*)   I/O buffer
   IBUFF    I      Buffer number to use (1 or 2)
Output:
   IERR     I      Error return code: 0 => no error
                      1 => file not open
                      2 => input error
                      3 => I/O error
                      4 => end of file (no messages)
Generic version.
```

ZTPOPN

Open a tape drive (as well as its corresponding disk control file) for sequential, "map" (double buffered, asynchronous) I/O or open a pseudo-tape sequential disk file. Exclusive use and wait to open are assumed. Uses a 'TP' disk "lock" file for real tapes.

```
ZTPOPN (LUN, FIND, IVOL, PNAME, OPER, IERR)
Inputs:
   LUN     I       Logical unit number (30 < LUN <= 30 + NTAPED
                      => tape, else disk)
   IVOL    I       Tape drive or disk volume containing file
   PNAME   C*48    tape disk physical file name
   OPER    C*4     'READ' => read only or 'WRIT' => read/write
Output:
   FIND    I       Index in FTAB to file control block for LUN
   IERR    I       Error return code: 0 => no error
                      1 => LUN already in use
                      2 => file not found
                      3 => volume not found
                      4 => exclusive use denied
                      5 => no room for LUN in FTAB
                      6 => other open errors
Generic version.
```

ZTPWAT

Wait until an asynchronous tape or sequential pseudo-tape disk file I/O operation completes.

```
ZTPWAT (LUN, FIND, IBUFF, LBYTES, IERR)
Inputs:
   LUN      I   Logical unit number
   FIND     I   Index in FTAB to file control block for LUN
   IBUFF    I   Buffer # to wait for (1 or 2)
Output:
   LBYTES   I   Number 8-bit bytes read/written (+1 if tape tape
```

```
                record longer than requested)
   IERR     I   Error return code: 0 => no error
                   1 => LUN not open in FTAB
                   3 => I/O error
                   4 => end of file
                   7 => wait service error
Generic version - calls ZTPWA2, ZTPWAD, ZERROR.
```

ZTTBUF

Dumps the contents (if any) of the input buffer into the array BUF without issuing a prompt or waiting for input. It does the terminal open, a call to some read with timeout set to 0, and the close.

```
ZTTBUF (BUF, BYTCNT, SYSERR, IERR)
Outputs:
   BUF      C*(*)   Input buffer
   BYTCNT   I       Number of bytes (characters) read
   IOSB     I       System error code
   IERR     I       Returned error code:  0 = OK
                       2 = no access to terminal
                       3 = IO error
                       8 = channel not deassigned
This routine was written to simulate trackball buttons on
the keyboard.    Andy Lubenow   22 Nov 1983
Generic version - stub
```

ZTTYIO

Perform I/O to a terminal.

```
ZTTYIO (OPER, LUN, FIND, NCHARS, BUFF, IERR)
Inputs:
   OPER     C*4   Operation code 'READ' or 'WRIT'
   LUN      I     Logical unit number
   FIND     I     Index in FTAB to file control block for LUN
   NCHARS   I     # characters to transfer (<= 132)
In/out:
   BUFF     C*(*) I/O buffer containing characters (1-256)
Output:
   IERR     I     Error return code: 0 => no error
                     1 => file not open
                     2 => input error
                     3 => I/O error
                     4 => end of file
Generic version: formatted IO does carriage control on output based
on parameter TTYCAR.
```

15.11.4 Data Manipulation

The following describes the first level data manipulation "Z" routines as documented in the APLGEN directory.

ZBYMOV

Moves a string of bytes from INB to OUTB. Byte = 8 bits (NOT an AIPS half-integer byte). This is used with tape input and output only.

```
ZBYMOV (NMOVE, INP, INB, OUTP, OUTB)
Inputs:
   NMOVE   I      Number of 8-bit bytes bytes
   INP     I      First value in INB to move (8-bit byte offset)
   INB     I(*)   Input buffer.
   OUTP    I      Location in OUTB to put first 8-bit byte  (8-bit
                  byte offset)
Output:
   OUTB    I(*)   Output buffer.
Generic version - stub.
```

ZBYTFL

Interchange the low order and high order bytes for all words in the input buffer and put the results in output buffer (which may be the same as the input buffer). For machines that are not byte flipped, the output buffer is identical to the input buffer (see BYTFLP in ZDCHIN). Any required byte swapping is performed via a call to ZBYTF2. NOTE a byte is 8 bits here.

```
ZBYTFL (NWORDS, INBUF, OUTBUF)
Inputs:
   NWORDS   I           Number of 16-BIT words to byte swap
   INBUF    I*2(*)      Input buffer
Output:
   OUTBUF   I*2(*)      Output buffer containing the byte swapped
                        words
Generic version - uses ZBYTF2 and ZADDR, works for 16, 32, 48, and
64-bit machines only
```

ZC8CL

Convert 8-bit ASCII standard characters in a buffer to local character form.

```
ZC8CL (NCHAR, NP, INBUF, OUTBUF)
Inputs:
   NCHAR    I       Number of characters to convert
   NP       I       Starting position in input buffer in units of
                    8-bit characters (1-relative)
   INBUF    R(*)    Input buffer containing 8-bit ASCII characters
Output:
   OUTBUF   C*(*)   Output buffer containing characters in local
                    form beginning in position 1
Generic version - assumes local characters are ASCII.
Requires local development if NBITWD not n*8.
```

ZCLC8

Convert local characters in a buffer to standard 8-bit ASCII character form.

```
ZCLC8 (NCHAR, INBUF, NP, OUTBUF)
Inputs:
   NCHAR    I       Number of characters to convert
   INBUF    C*(*)   Characters in local form
   NP       I       Starting position in output buffer in units of
                    8-bit characters (1-relative)
Output:
   OUTBUF   R(*)    Buffer containing characters in 8-bit ASCII form
Generic version - assumes local characters are ASCII.
Requires local development if NBITWD not n*8.
```

ZDHPRL

Converts from 64 bit Hewitt-Packard floating format to local double precision.

```
   HP R*8 format:
             1         2         3         4         5         6
   0123456789012345678901234567890123456789012345678901234567890123
   smmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmnnnnnnnnnnnnnnnnnnnnnnnneeeeeeez
The value can be determined as follows:
   man1 = smmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm is a two's complement
          signed integer.
   man2 = nnnnnnnnnnnnnnnnnnnnnnnn is a POSITIVE unsigned integer.
   exp = zeeeeeee is a two's complement signed integer.
   value = (man1 * 16777216 + man2) * 2.0D0 ** (exp - 55)

   ZDHPRL (NVAL, NP, INB, OUTB)
   Inputs:
      NVAL   I      Number of values to convert
      NP     I      First value in INB to convert
      INB    D(*)   64-bit HP format values
   Output:
      OUTB   D(*)   Local format values.
   Generic version; should work on any machine.
```

ZDM2DL

Convert Modcomp R*6 (zero padded to D) or R*8 data into local double precision.

```
ZDM2DL (NWORDS, INBUF, OUTBUF)
Inputs:
   NWORDS   I      Length of the input buffer in words
   INBUF    D(*)   Input buffer containing Modcomp R*6 (zero padded
                   to D) or D*8 data
Output:
   OUTBUF   D(*)  Output buffer containing local D.P. data
Notes:
(1) The INBUF should have its bytes flipped by a call to ZBYTFL
    which will leave the values split between 2 local integers.
    Do not call ZI32IL which may swith 16-bit words, or ZI16I1
    which will expand each 16-bit part to 32-bits.
(2) First 32 bits:
      Expects, after word flip on VMS only, a sign bit in bit 31
      (1=>negative), bits 22:30 are the  exponent biased by 512,
```

```
      bits 0:21 are the normalized fraction.  Negative values are
      obtained by 2's complement of the whole word.
    Second 32 bits:
      Just extended precision bits.
(3) Should work inplace.
Generic version --- stubbed.
```

ZGETCH

Extracts the character in position NCHAR of the HOLLERITH argument WORD and inserts it in the least significant bits of the INTEGER argument ICHAR with zero in the rest. It should also work for INTEGER WORD as long as NCHAR is valid. Characters are numbered from 1 in the order in which they would be printed by a Fortran "A" format specifier. NOTE - we actually get 8 bits here - so this routine works for bytes too.

```
ZGETCH (ICHAR, WORD, NCHAR)
Inputs:
   WORD   R   Word from which the character is to be extracted
   NCHAR  I   Position of character to extract
Output:
   ICHAR  I   Extracted character in LS bits, zero in the rest
Generic version - stub
```

ZGTBIT

Get the lowest order "nbits" bits of the bit pattern in "word" and return them in the array BITS with the lsb in bits[0]. For example, if

```
word  = 0 0 0 0 0 1 0 1 ... 0 0 0 1 0 0 1 1
          ^                                 ^

       MSB                                LSB
and nbits = 3 then bits[0] = 1 , bits[1] = 1 and bits[2] = 0
```

```
ZGTBIT (NBITS, WORD, BITS)
Inputs:
   NBITS   I      Number of bits
   WORD    I      Word from which to extract bits
Output:
   BITS    I(*)   Bit array (values 0 or 1)
Generic version - slow.
```

ZI16IL

Extract 16-bit, 2's complement integers from an input buffer and put them into an output buffer in local integer form. This must work even when the address of the input and output buffers is the same.

```
ZI16IL (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I         Number of 16-bit integers to extract
   NP     I         Starting position in the input buffer counting
                    from 1 in units of 16-bit integers
   INB    I*2(*)    Input buffer
Output:
   OUTB   I(NVAL)   Output buffer
Generic version - assumes that the local machine is 32-bit with a
valid INTEGER*2 Fortran data type
```

ZI32IL

Extract 32-bit, 2's complement integers from an input buffer and put them into an output buffer in local large integer form. This must work even when the address of the input and output buffers is the same. The IBM order applies to the input (i.e., the most significant part of the 32-bit integer is in the lower index of the input buffer and the least significant part is in the higher index.

```
ZI32IL (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I      # values to extract
   NP     I      Starting position in the input buffer (1-relative)
                 in units of 32-bit integers
   INB    I(*)   Input buffer
Output:
   OUTB   I(*)   Output buffer
Generic version  - does byte order flip in 16-bit words and a flip
of the order of the words.  This assumes local 32-bit integers, but
the capability of INTEGER*2 as well.
```

ZI8IL

Convert 8-bit unsigned binary numbers to local integers. This must work even when the input and output buffers are the same.

```
ZI8IL (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I         Number of 8-bit values to convert
   NP     I         Starting position in the input buffer counting
                    from 1 in units of 8-bit values
   INB    I(*)      Input buffer
Output:
   OUTB   I(NVAL)   Output buffer
Generic version - for NBITWD = n*8 uses ZGETCH, else is stubbed
```

ZILI16

Convert a buffer of local integers to a buffer of standard 16-bit, 2's complement integers.

```
ZILI16 (NINT, INB, NP, OUTB)
Inputs:
   NINT   I      Number of integers to convert
   INB    I(*)   Input buffer (start at index 1)
   NP     I      Starting index in the output buffer (1-relative)
                 in units of 16-bit integers
Output:
   OUTB   I(*)   Output buffer
Generic version - assumes that the local machine is 32-bit with a
valid INTEGER*2 Fortran data type
```

ZILI32

Convert a buffer of local large integers to a buffer of standard 32-bit, 2's complement integers. This must work even when the address of the input and output buffers is the same. The IBM order applies to the output (i.e., the most significant part of the 32-bit integer is in the lower index of the output buffer and the least significant part is in the higher index).

```
ZILI32 (NVAL, INB, NP, OUTB)
Inputs:
   NVAL   I         # integers to convert
   INB    I(*)      Input buffer (start at index 1)
   NP     I         Starting position in the output buffer
                    (1-relative) in units of 32-bit integers
Output:
   OUTB   I(NVAL)   Output buffer
Generic version  - does byte order flip in 16-bit words and a flip
of the order of the words.  This assumes local 32-bit integers, but
the capability of INTEGER*2 as well.
```

ZMCACL

Convert Modcomp compressed ASCII characters to local uncompressed characters. Successive calls will append data to the end of the output buffer. One compressed record is processed per call.

NOTE: this routine will not work in place.

Modcomp compressed ASCII format for each logical record:

```
BYTE             Use
 0     ASCII ETX (Hex 03)
 1     checksum (optional)
2-3    byte count, negative => end of file (not on tape?)
       (NOTE: may be bytes 1-2)
4-     Compressed ASCII characters.  A NULL (Hex 00) terminates.
       A negative value (most significant bit on) indicates a
       repetition of the previous character the number of times
       indicated by the absolute value of the negative number.
       Example: an ASCII 'C' followed by a byte with the HEX
       value FF (2's complement -1) indicates two 'C's.
ZMCACL (NBYTES, INBUF, OUTBUF, LASTCH)

Inputs:
   INBUF    I(*)        Input buffer of Modcomp compressed ASCII
                        packed characters
In/out:
   NBYTES   I           (In)  Maximum number of bytes to convert
                        (Out) Number of bytes added to the output
                              buffer
   LASTCH   I           Position of the next character in the output
                        buffer (zero before first call)
Output:
   OUTBUF   I(*)        Output buffer containing packed characters
                        (due to the expansion of the data, the size
                        of the output buffer is not strictly
                        predictable)
Generic version (stubbed).
```

ZPTBIT

Build WORD from NBITS bit values contained in the array BITS, where BITS[1] supplies the lsb, BITS[2] the next higher bit, etc. The rest of the bits in WORD are set to zero. For example, if

```
bits[1:*] = 0 1 1 1 0 1 0 1 ... 0 1 1 1 0 0 0 0
            ^                                 ^

          LSB                               MSB
and NBITS = 4, then WORD = 14 (decimal)

   ZPTBIT (NBITS, WORD, BITS)
   Inputs:
      NBITS   I      Number of bits to use from the array "bits"
      BITS    I(*)   Array of bit values (0 or 1)
   Output:
      WORD    I      Result containing bit pattern from the first
                     "nbits" values of the array "bits"
   Generic version - stub
```

ZPUTCH

Inserts the character contained in the least significant bits of the INTEGER argument ICHAR into the NCHAR position of the HOLLERITH argument WORD. It should also work for INTEGER WORD as long as NCHAR is valid. Characters are numbered from 1 in the order in which they would be printed by a Fortran "A" format specifier.

```
   ZPUTCH (ICHAR, WORD, NCHAR)
   Inputs:
      ICHAR   I   Character to insert in LS bits
      NCHAR   I   Position in WORD to store character
   Output:
      WORD    H   Word into which character is to be inserted
   Generic version - stub
```

ZR32RL

Converts from 32 bit IEEE floating format to local single precision.

```
The IEEE format is:

             1         2         3
   01234567890123456789012345678901
   seeeeeeeemmmmmmmmmmmmmmmmmmmmmmm

where sign = -1 ** s, exponent = eee..., mantissa = 1.mmmmm...

The value is given by:

   value = sign * 2 **(exponent-127) * mantissa
```

Note: these values have a "hidden" bit and must always be normalized The IEEE nan (not a number) values are used to indicate an invalid number; a value with sign and all exponent bits set is recognized as "nan".

The AIPS internal format for an invalid number is the value which has the same bit pattern as 'INDE'.

The IEEE special values (-0., +/- Infty) are not recognized.

A multiplication by a factor of 4.0 converts between VAX F and IEEE 32 bit formats.

```
   ZR32RL (NVAL, NP, INB, OUTB)
   Inputs:
```

```
   NVAL   I      Number of values to convert
   NP     I      First value in INB to convert
   INB    R(*)   32-bit IEEE format values
Output:
   OUTB   R(*)   Local format values ("nan" values are replaced
                 with AIPS' indefinite value = 'INDE')
Generic version - does IEEE and VAX F formats for 32-bit machines,
is stubbed with STOP for all others.
```

ZR64RL

Converts from 64 bit IEEE floating format data to local double precision (or corresponding 64 bit precision). The IEEE format is:

```
          1         2         3         4         5         6
0123456789012345678901234567890123456789012345678901234567890123
seeeeeeeeeeemmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
```

```
where sign = -1 ** s, exponent = eee..., mantissa = 1.mmmmm...

The value is given by:

   value = sign * 2 **(exponent-1023) * mantissa
```

Note: these values have a "hidden" bit and must always be normalized The IEEE nan (not a number) values are used to indicate an invalid number; a value with sign and all exponent bits set is recognized as a "nan". The AIPS internal format for an invalid number is the value which has the same bit pattern as 'INDE '. The IEEE special values (-0., +/- Infty) are not recognized. A multiplication by a factor of 4.0 converts between VAX G and IEEE 64 bit formats.

```
ZR64RL (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I      Number of values to convert
   NP     I      First value in INB to convert
   INB    D(*)   64-bit IEEE format values
Output:
   OUTB   D(*)   Local format values ("nan" values are replaced
                 with AIPS' D.P. blank = 'INDE    ')
Generic version  - does IEEE and VAX G formats for 32-bit machines,
is stubbed with STOP for all others.
```

ZR8P4

Converts between pseudo I*4 and Double Precision. The main use of this is to handle I*4 integers in buffers with mainly I*2 integers for tape IO - i.e. EXPORT and IBM format headers. The assumption is that the full buffer has been treated with (on read of tape) ZI16IL and needs to be repacked to 32 bits and then changed to local integer and then to local D.P. And the reverse on tape writing.

```
ZR8P4 (OP, INTG, DX)
Inputs:
   OP     C*4    '4IB8' IBM I*4 to D
                 '8IB4' D to IBM I*4
In/out:
```

```
   INTG   I(2)   the I
   DX     D      the D
IBM pseudo-I*4 has the form of a 2's complement, 32-bit integer
with the most significant 16 bits in the I word of lower index and
the least significant 16 bits in the I word of higher index.
Generic version - may work for all
```

ZRDMF

Convert data packed in DEC-Magtape format (DMF) to pairs of local 32-bit integers.

The DMF format is:

```
Track    1    2    3    4    5    6    7    8
Byte
  1      F0   F1   F2   F3   F4   F5   F6   F7
  2      F8   F9  F10  F11  F12  F13  F14  F15
  3      F16  F17  R0   R1   R2   R3   R4   R5
  4      R6   R7   R8   R9  R10  R11  R12  R13
  5       0    0    0    0  R14  R15  R16  R17
```

where Rn refers to the right halfword, Fn to the left halfword.

Since the purpose of this routine is to read Modcomp tapes written with this peculiar format, F16, F17, R16 and R17 (the high order bits) are zero for VLA data, but are used for the word count.

The first word (5 bytes) of a tape block contains the word count of the block. The word count is a 16-bit, twos-complement integer comprised of bits R2-R16. All other words are treated as pairs of 16-bit, twos-complement integers comprising bits F0-F15 and R0-R15.

Input data is assumed to be packed into 1 1/8 integers and output data will be returned in a pair of local integers per DEC-10 word. The first integer of each pair corresponds to the left DEC-10 halfword (zero for the word count) and the second to the right halfword. The sign of each halfword is preserved on expansion to a local integer.

```
ZRDMF (NWORDS, INBUF, OUTBUF, FLAG)

Inputs:
   NWORDS   I           Length of the input buffer in DEC-10
                        words
   INBUF    I(*)        Input buffer containing DMF format data
   FLAG     I           If > 0, the first word word is the
                        beginning of a tape block
Output:
   OUTBUF   I(*)        Output buffer containing two local
                        integers per input DEC-10 word
Generic version (stubbed).
```

ZRHPRL

Converts from 32 bit Hewitt-Packard floating format to local single precision.

```
   HP R*4 format:
             1         2         3
   01234567890123456789012345678901
   smmmmmmmmmmmmmmmmmmmmmmmeeeeeeez
The value can be determined as follows:
   man = smmmmmmmmmmmmmmmmmmmmmmm is a two's complement signed
```

```
      integer.
exp = zeeeeeee is a two's complement signed integer.
value = man * 2.0 ** (exp - 23)

ZRHPRL (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I      Number of values to convert
   NP     I      First value in INB to convert
   INB    R(*)   32-bit HP format values
Output:
   OUTB   R(*)   Local format values.
Generic version; should work on any machine.
```

ZRLR32

Converts from local single precision to 32 bit IEEE floating format. See ZR32RL for the description of IEEE 32 bit format. The AIPS internal format for an invalid number is the value which has the same bit pattern as 'INDE'.

A multiplication by a factor of 4.0 converts between VAX F and IEEE 32 bit formats.

```
ZRLR32 (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I      Number of values to convert
   NP     I      First value in OUTB for result
   INB    R(*)   Local format values
Output:
   OUTB   R(*)   32-bit IEEE FORMAT values ('INDE' values are
                 replaced with "nan")
Generic version - does IEEE and VAX F formats for 32-bit machines,
is stubbed with STOP for all others.
```

ZRLR64

Converts from local double precision (or corresponding 64 bit precision) to 64 bit IEEE floating format. See ZR64RL for a description of the IEEE 64 bit format.

The AIPS internal format for an invalid number is the value which has the same bit pattern as 'INDE '.
A multiplication by a factor of 4.0 converts between VAX G and IEEE 64 bit formats.

```
ZRLR64 (NVAL, NP, INB, OUTB)
Inputs:
   NVAL   I      Number of values to convert
   NP     I      First location in OUTB for results
   INB    D(*)   Local format values
Output:
   OUTB   D(*)   64-bit IEEE format values ('INDE    ' values are
                 replaced with "nan")
Generic version  - does IEEE and VAX G formats for 32-bit machines,
is stubbed with STOP for all others.
```

ZRM2RL

Convert Modcomp single precision floating point data into local single precision floating point.

```
ZRM2RL (NWORDS, INBUF, OUTBUF)
Inputs:
   NWORDS   I      Length of the input buffer in words
   INBUF    R(*)   Input buffer containing Modcomp R*4 data
Output:
   OUTBUF   R      Output buffer containing local REAL data
Notes:
   Before call, input buffer should have its bytes flipped via
   ZI32IL which will leave the values in one local 32-bit integer

   Expects, after word flip, sign bit in bit 31 (1=>negative),
   bits 22:30 are the  exponent biased by 256(?), bits 0:21 are
   the normalized fraction.  Negative values are obtained by 2's
   complement of the whole word.

   Should work inplace.

Generic version - stub
```

ZUVPAK

Routine to pack uv data with magic value blanking. One AIPS logical uv data record is processed at a time. Two values are packed into a single real value.

```
ZUVPAK (NCORR, VISIN, WTSCL, VISOUT)
Inputs:
   NCORR  I       Number of correlator values in data
   VISIN  R(3,*)  Unpacked uv data as real, imag and weight per
                  correlator.
Output:
   WTSCL  R(2)    "Weight" and "scale" random parameters for the
                  packed record.
   VISOUT R(*)    Packed visibility data with local magic value
                  blanking.
Version for machines with valid integers half the size of reals.
```

ZUVXPN

Routine to expand packed uv data to unpacked form. One AIPS logical uv data record is processed at a time.

```
ZUVXPN (NCORR, VISIN, WTSCL, VISOUT)
Inputs:
   NCORR  I        Number of correlator values in data
   VISIN  R(*)     Packed visibility data with local magic value
                   blanking.
   WTSCL  R(*)    "Weight" and "scale" random parameters for the
                   packed record.
Output:
   VISOUT  R(3,*)  Unpacked uv data as real, imag and weight per
                   correlator.
Version for machines with valid integers half the size of reals.
```

ZX8XL

Converts a FITS table bit array to an AIPS bit array. An AIPS bit array has the bits in locations of increasing signifigance NBITWD per integer. A FITS bit array has the bits in order of decreasing signifigance with 8 bits per real world byte, zero filled

```
ZX8XL (NBIT, INB, OUTB)
Inputs:
   NBIT   I      Number of bits
   INB    I(*)   Input buffer of FITS bit array data as 8-bit byte
                 stream.
Output:
   OUTB   I(*)   Out buffer of AIPS bit array data.
Generic version: works whenever local integer is a multiple of 16
bits and has no more than 64 bits.
```

ZXLX8

Converts an AIPS table bit array to an FITS bit array. An AIPS bit array has the bits in locations of increasing signifigance NBITWD per integer. A FITS bit array has the bits in order of decreasing signifigance with 8 bits per real world byte, zero filled.

```
ZXLX8 (NBIT, INB, OUTB)
Inputs:
   NBIT   I      Number of bits
   INB    I(*)   Input buffer of AIPS bit array data.
Output:
   OUTB   I(*)   Out buffer, note an integral number of 16-bit
                 integers in OUTB are modified.  OUTB is left in
                 the form of IEEE integers (i.e. ms byte first).
Generic version: works whenever local integer is a multiple of 16
bits and has no more than 64 bits.
```

15.11.5 Directory and Text File

The following describes the first level text file "Z" routines as documented in the APLGEN directory.

ZTCLOS

Close the text file and clear the FTAB entry associated with LUN.

```
ZTCLOS (LUN, FIND, IERR)
Inputs:
   LUN      I   Logical unit number
   FIND     I   Index in FTAB for LUN
Output:
   IERR     I   Error return code: 0 => no error
                   1 => close error
                   2 => file already closed in FTAB
                   3 => both errors
                   4 => erroneous LUN
Generic version.
```

ZTOPEN

Open a text file - logical area, version, member name as arguments

```
ZTOPEN (LUN, FIND, IVOL, PNAME, MNAME, VERSON, WAIT, IERR)
Inputs:
   LUN      I      Logical unit number
   IVOL     I      Disk volume containing file, (not used)
   PNAME    C*48   Physical file name, only used to determine file
                   type or logical area
   MNAME    C*8    Text file name
   VERSON   C*48   Logical name for directory or version of
                   directory to search (for file-specific
                   directories)
   WAIT     L      T => wait until file is available (not used)
Outputs:
   FIND     I      Index in FTAB for LUN
   IERR     I      Error return code: 0 => no error
                      1 => LUN already in use
                      2 => file not found
                      3 => volume not found
                      4 => file locked
                      5 => no room for LUN in FTAB
                      6 => other open errors
Generic version - uses ZDIR, ZFULLN, and Fortran functions INQUIRE
and then calls ZTOPE2.
```

ZTREAD

Read the next sequential 80-character card image from a text file.

```
ZTREAD (LUN, FIND, RBUFF, IERR)
Inputs:
   LUN     I      Logical unit number
   FIND    I      Index in FTAB for LUN
Output:
   RBUFF   C*80   I/O buffer for card image
   IERR    I      Error return code: 0 => no error
                     1 => file not open
                     2 => end of file
                     4 => other I/O error
Generic version - assumes simple Fortran IO.
```

ZTXCLS

Close the text file and clear the FTAB entry associated with LUN.

```
ZTXCLS (LUN, FIND, IERR)
Inputs:
   LUN    I   Logical unit number
   FIND   I   Index in FTAB for LUN
Output:
   IERR   I   Error return code: 0 => no error
                 1 => close error
                 2 => file already closed in FTAB
```

```
                  3 => both errors
                  4 => inputs error
Generic version: uses Fortran CLOSE
```

ZTXIO

Read/write the next sequential line from/to a text file.

```
ZTXIO (OPER, LUN, FIND, LINE, IERR)
Inputs:
   OPER   C*4     Operation code ('READ' or 'WRIT')
   LUN    I       Logical unit number
   FIND   I       Index in FTAB for LUN
Input/output:
   LINE   C*(*)   Line of text.  For WRIT, ZTXIO writes the full
                  string including any trailing blanks.  Use ITRIM
                  and substring notation in the call if you desire
                  only up to the last non-blank (which is usually
                  preferable!).  On READ, adequate size must be
                  declared in calling routine.
Output:
   IERR   I       Error return code: 0 => no error
                     1 => file not open
                     2 => end of file
                     3 => input error
                     4 => other I/O error
Generic version.
```

ZTXMAT

Open a directory and find a list of member file names whose first NCH characters match the first NCH characters of MNAME plus extension (based on file type).

```
ZTXMAT (IVOL, PNAME, MNAME, NCH, VERSON, NAMES, NNAM, IERR)
Inputs:
   IVOL     I         Disk volume containing file, (ignored)
   PNAME    C*48      Physical file name (only used to determine
                      file type)
   MNAME    C*8       Test file name
   NCH      I         Number of characters to compare (<= 8)
                         0 is okay -> get full directory
   VERSON   C*48      Logical name for directory or version of
                         directory to search (for file type
                         specific directories)
In/out:
   NNAM     I         Number of names in NAME: input = max,
                         output = actual used
Output:
   NAMES    H*8(*)    File names which match the given file spec
                      NOTE HOLLERITH for lower level routines
   IERR     I         Error return code: 0 => no error
                         1 => no matches found
                         2 => error in inputs
                         3 => error opening directory
Generic version - uses ZDIR, ZFULLN, ZTXMA2
```

ZTXOPN

Open a text file.

```
ZTXOPN (OPCODE, LUN, FIND, OUTFIL, APPEND, IERR)
Inputs:
   OPCODE   C*4    Open for 'READ' or 'WRIT'
   LUN      I      Logical unit number
   OUTFIL   C*48   Physical file name
   APPEND   L      If true append new text to end of old file.
                      (OPCODE='WRIT' only).
Outputs:
   FIND     I      Index in FTAB for LUN
   IERR     I      Error return code: 0 => no error
                      1 => error in inputs
                      2 => LUN already in use
                      3 => no room for LUN in FTAB
                      4 => trouble translating logical
                      5 => file already exists
                      6 => open error
Generic version --- uses ZTXOP2 for actual open.
```

15.11.6 Virtual Devices

The following describes the first level virtual device "Z" routines as documented in the APLGEN directory.

ZVTVCL

Close the virtual TV device channel in the client (VTV) machine to the server (remote, real-TV) machine - uses ZVTVC2.

```
ZVTVCL (LUN, FIND, IERR)
Inputs:
   LUN    I   Logical unit number
   FIND   I   Index in FTAB to file control block for LUN
Output:
   IERR   I   Error return code: 0 => no error
                 1 => close error or flush error
                 2 => file already closed in FTAB
                 3 => both errors
                 4 => erroneous LUN
Generic version - calls ZVTVC2.
```

ZVTVGC

Opens the connection in TVMON to the remote machine which is running the AIPS VTV (Virtual TV) code. This differs from ZVTVO3 in that the socket is already there and an old connection must be closed before a new one can be accepted. Called by ZVTVRX.

```
ZVTVGC (FCB, IERR)
Outputs:
   FCB   I(*)   File descriptor
   IERR  I      Error: 0 => okay
Generic version - stub
```

ZVTVOP

Does whatever is needed to open communication from the current program to a remote machine that has a real TV display attached. This is a generic upper level Z routine.

```
ZVTVOP (LUN, IND, IERR)
Output:
    IERR  I     Error code: 0 => ok
                   1 = LUN already in use
                   2 = file not found
                   3 = volume not found
                   4 = excl requested but not available
                   5 = no room for lun
                   6 = other open errors
Generic version - uses ZVTVO2 and ZPHFIL.
```

ZVTVRC

Close the virtual TV device channel: from receiver point of view

```
ZVTVRC (LUN, FIND, IERR)
Inputs:
   LUN    I   Logical unit number
   FIND   I   Index in FTAB to file control block for LUN
Output:
   IERR   I   Error return code: 0 => no error
                 1 => close error or flush error
                 2 => file already closed in FTAB
                 3 => both errors
                 4 => erroneous LUN
Generic version - uses ZVTVC3.
```

ZVTVRO

Does whatever is needed to enable communication from the current program (TVMON) on a machine that has a real TV display attached to any remote machine needing the display. This is a generic upper level Z routine.

```
ZVTVRO (LUN, IND, IERR)
Inputs:
   LUN    I    An LUN to use (not TVLUN or TVLUN2)
Output:
   IND    I    FTAB location opened
   IERR   I    Error code: 0 => ok
                  1 = LUN already in use
                  2 = file not found
                  3 = volume not found
                  4 = excl requested but not available
                  5 = no room for lun
                  6 = other open errors
```

ZVTVRX

Performs "I/O" over some communication mechanism to a cooperating program with the intention of driving a real TV device controlled by this program.

```
ZVTVRX (FIND, BUFSW, HBUF, IERR)
Inputs:
   FIND    I      FTAB location for socket to remote machine
   BUFSW   I      Number 16-bit integers total data to send
                  If BUFSW > 0, write to client; if <= 0, read
                  from client machine.
In/Out:
   HBUF    I(*)   Buffer: in FITS standard 16 bit
                     (1,2) : READ or WRIT
                     (3,5) : subroutine name packed ASCII
                     (6)   : BUFSW - 8
                     (7)   : BUFSR
                     (8)   : error returned to client
                     (9..) : extra data
Output:
   IERR    I      Error code: 0 => ok.
Generic version - does little except call ZVTVX3
```

ZVTVXF

Performs "I/O" over some communication mechanism to a cooperating program with the intention of driving a real TV device controlled by the cooperating program.

```
ZVTVXF (BUFSW, BUFSR, HBUF, IERR)
Inputs:
   BUFSW  I      Number 16-bit integers extra data to send
   BUFSR  I      # 16-bit integers of extra data to receive
In/Out:
   HBUF   I(*)   Buffer:  in FITS standard 16 bit
                    (1,2) :  READ or WRIT
                    (3,5) :  subroutine name packed ASCII
                    (6)   :  BUFSW
                    (7)   :  BUFSR (TV IERR returned)
                    (8)   : error return
                    (9..) :  extra data
Output:
   IERR   I      Error code:  0 => ok.
Generic version - does little except call ZVTVX2
```

15.11.7 Miscellaneous

The following describes the first level miscellaneous "Z" routines as documented in the APLGEN directory.

ZADDR

Determine if two addresses are the same.

```
ZADDR (ADDR1, ADDR2, IERR)
Inputs:
   addr1   I*2   Address 1
   addr2   I*2   Address 2
Output:
   ierr    I     Error return code:
                    0 => addresses are the same
                    1 => addresses are different
Generic version - stub
```

ZERROR

This routine will attempt to translate the system error code and if appropriate (i.e., FCB(1).NE.-999) print the name of the file or device on which the error occurred as well as the contents of the the file control block (blocks, if map I/O).

```
ZERROR (ZRTNAM, SYSERR, PNAME, FCB, MAP)
Inputs:
   ZRTNAM   C*6    Z-routine where the error occurred
   SYSERR   I      System error code (also stored in FCB)
   PNAME    C*48   Physical file name; ' ' => unknown (omit)
   FCB      I(*)   File control block in FTAB for the file (or
                   device) also containing the system error code
                   FCB(1) = -999 => omit
   MAP      L      Map or non-map I/O involved?
Common: DMSG.INC
   DBGAIP   I      MOD(DBGAIP,10) > 2 => force ZERRO2 call
                   MOD(DBGAIP,10) > 2 => dump FCB(s)
Generic version - calls ZERRO2.
```

ZHEX

Convert decimal to nc character hexidecimal 'string' - leading blanks are made into 0's, i.e. "illegal" Fortran format Znc.nc

```
 ZHEX (IVAL, NC, HVAL)
Input:
   IVAL   I       Decimal value to convert
   NC     I       Width of receiving field
Output:
   HVAL   C*(*)   String receiving hexidecimal conversion
Generic version - klunky but works
```

ZKDUMP

Dump portions of an array in a variety of formats (e.g., Fortran I, E, A and hexidecimal).

```
ZKDUMP (I1, I2, K, C)
Inputs:
   I1    I      First subscript in integer array to dump
   I2    I      Last subscript in integer array to dump
   K     I(*)   Integer array
   C     R(*)   Real array equivalenced to K in calling routine
Generic version - uses ZHEX, assumes 32-bit integers
```

ZMSGER

This routine will attempt to translate the system error code and if appropriate (i.e., FCB(1).NE.-999) print the name of the file or device on which the error occurred as well as the contents of the the file control block (blocks, if map I/O). For MSGWRT - to avoid recursions.

```
ZMSGER (ZRTNAM, SYSERR, PNAME, FCB, MAP)
Inputs:
   ZRTNAM   C*6    Z-routine where the error occurred
   SYSERR   I      System error code (also stored in FCB)
```

```
   PNAME    C*48   Physical file name; ' ' => unknown (omit)
   FCB      I(*)   File control block in FTAB for the file (or
                   device) also containing the system error code
                   FCB(1) = -999 => omit
   MAP      L      Map or non-map I/O involved?
Common: DMSG.INC
   DBGAIP   I      MOD(DBGAIP,10) > 2 => force ZERRO2 call
                   MOD(DBGAIP,10) > 2 => dump FCB(s)
Generic version - calls ZERRO2.
```

ZMYVER

Determine the default AIPS version (OLD or NEW or TST). Error messages are only written to the terminal since this routine is called before message handling via MSGWRT is established.

```
ZMYVER
 (no call arguments)
Output in DDCH.INC:
   VERNAM   C*4   String = 'OLD:', 'NEW:' or 'TST:'
Generic version.
```