NOAA Technical Memorandum ERL GLERL-63

USER'S MANUAL FOR GLERL DATA ACCESS SYSTEM (GDAS)

David J. Schwab
Edward W. Lynn
Gary E. Spalding

Great Lakes Environmental Research Laboratory
Ann Arbor, Michigan 48104
March 1987

UNITED STATES DEPARTMENT OF COMMERCE

Malcolm Baldrige, Secretary

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION

Anthony J. Calio, Administrator

Environmental Research Laboratories

Vernon E. Derr, Director

NOTICE

Mention of a commercial company or product does not constitute an endorsement by NOAA/ERL. Use for publicity or advertising purposes, of information from this publication concerning proprietary products or the tests of such products, is not authorized.

For sale by the National Technical Information Service, 5285 Port Royal Road
Springfield, VA 22161

CONTENTS

FIGURES

USER'S MANUAL FOR GLERL DATA ACCESS SYSTEM (GDAS)*

David J. Schwab, Edward W. Lynn and Gary E. Spalding

ABSTRACT. The GLERL Data Access System (GDAS) is a series of FORTRAN subroutines designed to aid in the storage and retrieval of data from time-series data bases (collections of time-series with a common start date, start time, duration, and recording interval). Each time-series within the data base is uniquely identified by a station identifier, a parameter descriptor code identifying the type of data, and a fixed height or depth relative to the water surface. Information about the time-series is contained in a GDAS header file; the data are stored in a GDAS data file. This report describes the format of GDAS files and the GDAS subroutines used to access GDAS files, and gives an example of how to use the GDAS subroutines.

1. INTRODUCTION

GLERL has a large number of data sets comprising time series of data for various physical parameters. These data sets have been developed at various times by, or for, various scientists; the way in which data are stored and accessed differs among the data sets. The lack of uniformity of data formats sometimes makes' it difficult for scientists to gain access to needed data.

This report describes a uniform format for computerized storage of time-series data. The format is general enough to allow storage of a wide variety of data types. It is also intended that the format serve as self-documentation for archived data. Standardized access and processing programs can then be developed to facilitate analysis of diverse data types and reduce duplication of program development and coding.

The GLERL Data Access System (GDAS) has been developed to overcome problems stemming from the nonuniformity of current data sets. GDAS has these features:

- Data are stored in a standard format, independent of the content or size of a particular data set.
- Data can be stored in "time-series" or "time-synchronous" format.
- The data-recording interval can range from hundredths of seconds to years.
- The routines that create, read, and write data set contents are written in the new FORTRAN standard (FORTRAN 77) and use no non-standard code.
- There are routines that document data set contents.

'GLERL Contribution No. 522

- Any number of data sets can be accessed simultaneously.

- The number of parameter time series (PTS) in a single data set is limited only by the computer's disk capacity.

- Error conditions are reported by helpful error messages.

- Relocatable GDAS routines are in a library. The user need supply only the main program and any additional routines required in order to use GDAS.

GDAS is intended to consolidate a data set consisting of various types of time-series data into a single data file with a specified start date, start time, duration, and recording interval. It will generally be applied to data from an array of instruments, although it could just as easily accommodate a subset of the array, or a single instrument. However, all data within a single data set must have the same recording interval. Recording intervals ranging from hundredths of seconds to years can be accommodated by GDAS. Each time-series of data within the data set will have a unique station identifier, a parameter descriptor code identifying the type of data, and a fixed height or depth relative to the water surface.

2. GDAS FILE STRUCTURE

A GDAS data system consists of two files, a "header" file and a "data" file, both of which are direct access files (under the FORTRAN 77 definition). The header file contains information about the parameters in the data system. It can be thought of as a set of instructions about how data values are organized in the data file. It contains the start date and time of the data set, the time interval between successive data values, and a description of each different type of data in the data set. GDAS subroutines DBNEW or DBHED can be used to create a GDAS header file. The user supplies the information to be contained in the header file in arrays passed to the subroutines.

The data file contains only the data values. It can be thought of simply as a large matrix of numbers. The matrix has one row for each time series of data in the data set and one column for each point in time at which the data were recorded. Missing values in a time-series are flagged by a missing-data code. When the data file is created, all data files are usually initialized to the missing data value. GDAS subroutine DBNEW or DBDAT can be used for this purpose. Actual data values are then inserted into the data file by GDAS subroutine DBWRIT. Subroutine DBREAD is used to retrieve data from the data file.

For efficiency, when the numbers in the data set matrix are stored in the data file, each record (a record is the amount of data retrieved with a single FORTRAN 'READ' statement) may contain several data values. The records can be made up of either successive data values from along one row of the matrix or successive values from one column of the matrix, but the way records are made up must be the same for all records in the data file. The information about how the records in the data file are made up is contained in the header file.

If the records in the data file are made up of data values from along the rows of the data matrix, the data file format will be called "time-series" format. If the records in the data file are made up of data values from down the columns of the data matrix, the data file format will be called "time-synchronous" format (See Figs. 1 and 2).

In applications where the data are going to be used mainly as time-series of values from a single station, the data file should be organized in "time-series" format. In applications where data values from a large percentage of the stations in the data base at a single point in time are more likely to be required, the data file should be organized in "time-synchronous" format.

GDAS variable types follow the FORTRAN default naming conventions; i.e., variables whose first letter is in the range I-N are INTEGER*4, all others are REAL*4. Only the following character variables are used in calls to the GDAS subroutines:

NAME - CHARACTER*80	UNITS - CHARACTER*48
DSDES - CHARACTER*80	STA - CHARACTER*5
DESCR - CHARACTER*8	DES - CHARACTER*2

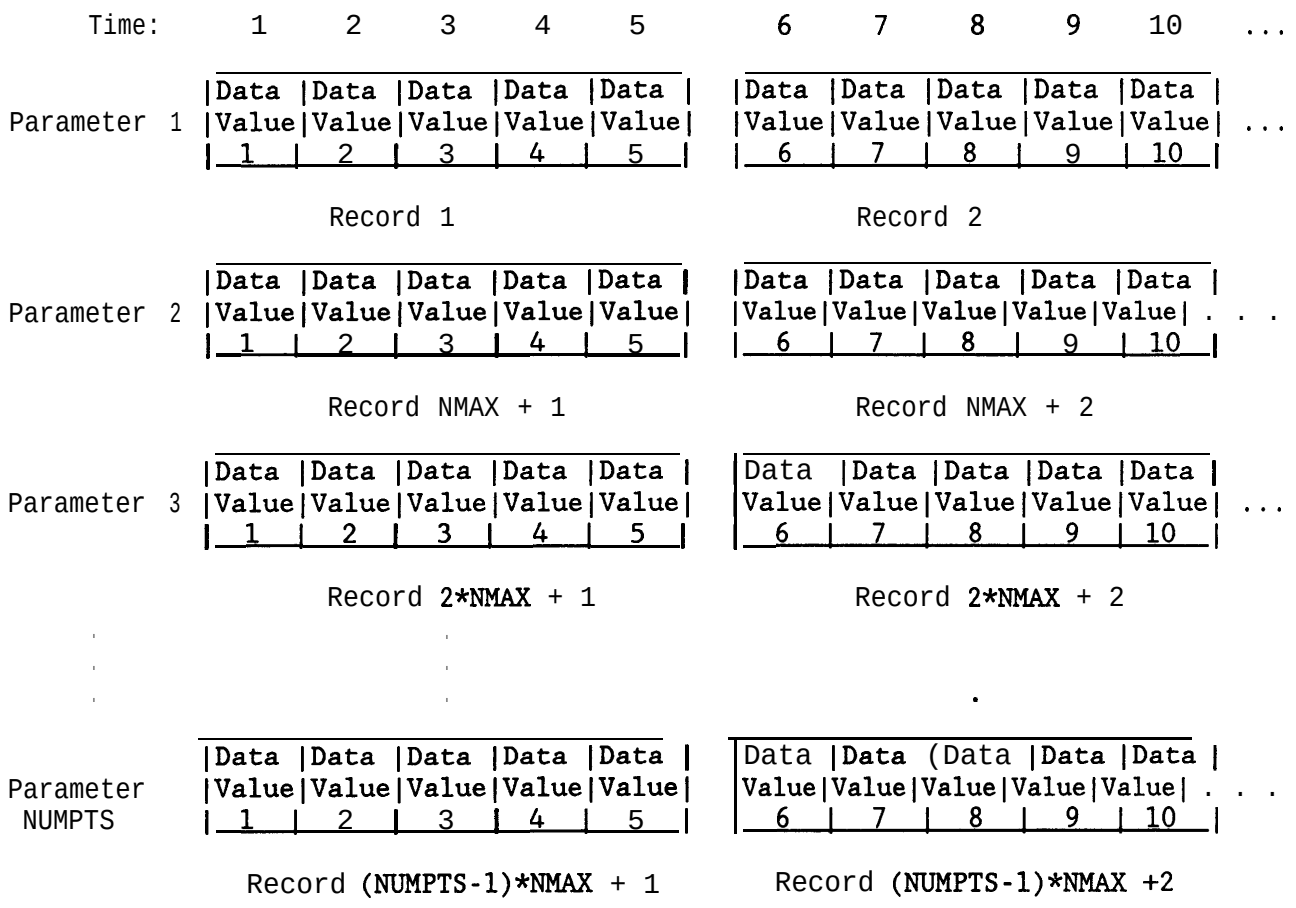

Figure 1. --Organization of a "time-series" GDAS data file with NCOL = 5. "Data Value" indicates order of data values in array DATA in subroutines DBREAD and DBWRIT.

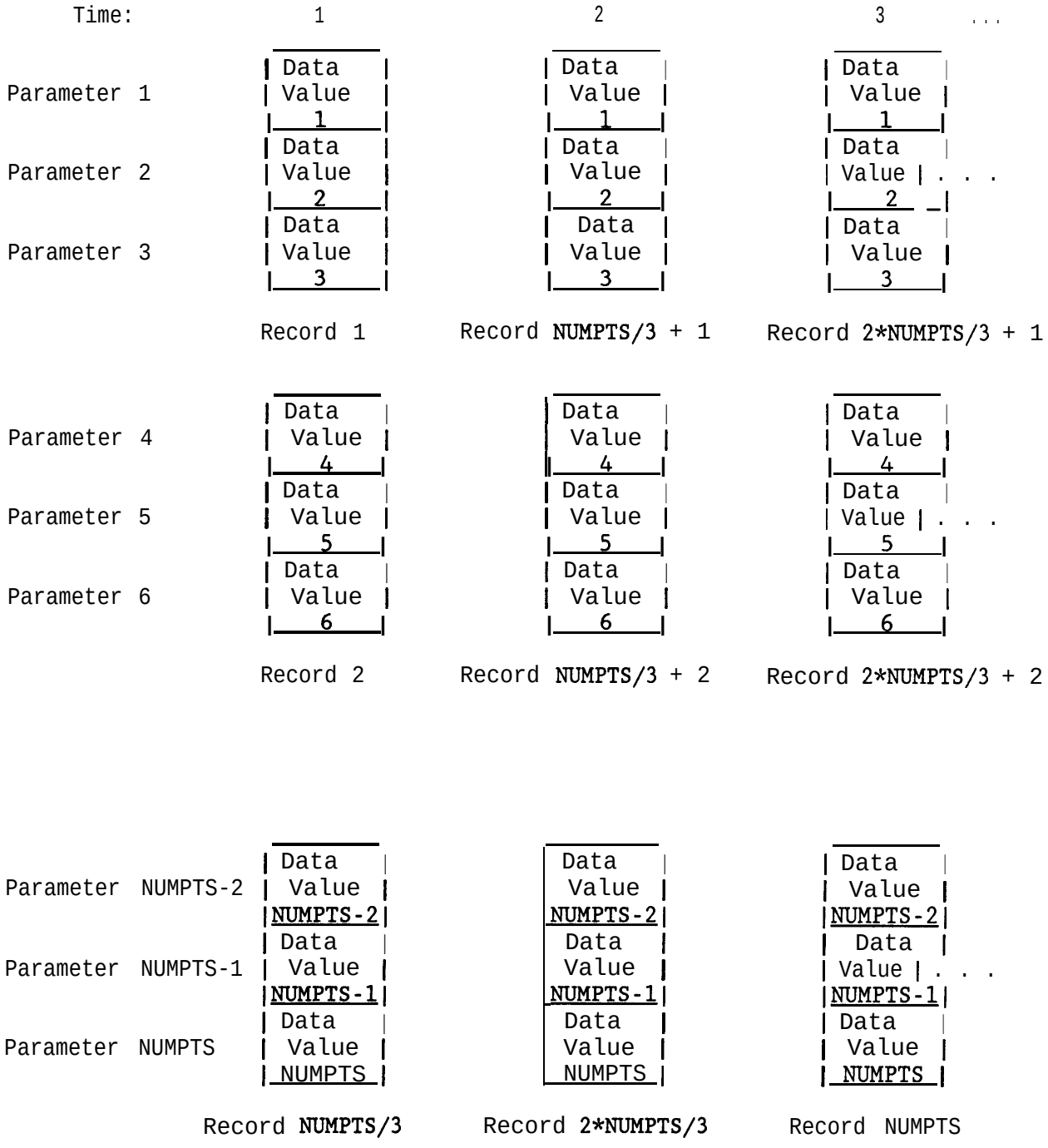

Figure 2. --Organization of a "time-synchronous" GDAS data file with NROW = 3. "Data Value" indicates order of data values in array DATA in subroutine DBREAD and DBWRIT.

One LOGICAL variable called DBERR is used as the last parameter in every GDAS subroutine to indicate if an error condition was detected by GDAS. If an error condition was detected, a message is printed to FORTRAN I/O unit 6 (by default this is the normal output for a FORTRAN program) and DBERR is set to .TRUE.. Control is then returned to the calling program. It is up to the calling program to check DBERR and take appropriate action.

The header file for each data set in GDAS "time-series" or "time-synchronous" format contains (1) a Header Record, (2) any number of Descriptor (text) Records, and (3) a Subheader Record for each parameter time series (PTS) in the data set. The record length of all records in the header file is 20 words (80 characters). GDAS subroutines DBNEW or DBHED can be used to create a GDAS header file. Information for the Header Record, Descriptor Records, and Subheader Records is passed to the subroutine in arrays. Information from an existing GDAS header file can be retrieved by calling subroutines DBOPEN and DBSUBH. Subroutine DBINFO will print a complete or partial summary of the information in a GDAS header file. Subroutine DBLOOK allows the user to scan the data base data file to determine data availability.

2.1 Header Record

Record 1 of the header file is the Header Record, which consists of 20 integer values describing the organization of the data file. The GDAS subroutines store these values in array IHEAD as follows:

IHEAD(1) = NCOL The number of data values per record for each parameter for a data file in "time-series" format, or 1 for a data file in "time-synchronous" format.

IHEAD(2) = NROW The number of data values per record for each time for a data file in "time-synchronous' format, or 1 for a data file in "time-series" format. NROW must be a submultiple of NUMPTS.

IHEAD(3) = NUMPTS The number of parameter time-series (PTS's) in the data set. A PTS is a time series of values of a particular parameter at a particular location (latitude, longitude, and height or depth from surface).

IHEAD(4) = NMAX Total number of data records in the data file for each PTS.

IHEAD(5) = INTERV Basic time interval indicator.

- 6 = Years
- 5 = Months
- 4 = Days
- 3 = Hours
- 2 = Minutes
- 1 = Seconds
- -1 = Tenths of Seconds
- -2 = Hundredths of Seconds

Figure 3 illustrates the effect a particular choice of INTERV has on the meaning of IYEARS, IJULS, and ITIMES.

IHEAD(6) = NINTRV Recording interval, i.e., the time interval between successive data values in units of the basic time interval (INTERV).

IHEAD(7) = IYEARS The year of the first record in the data file.

IHEAD(8) = IJULS The Julian day (or month if INTERV = 5) of the first record in the data file. Not used if INTERV = 6. See Figure 3 for definitions of IJULS for different values of INTERV.

IHEAD(9) = ITIMES The number of basic time intervals (INTERV) after 00:00 local time (see IHEAD(11)) at which the first record in the data file starts. Not used if INTERV > 3. See Figure 3 for definitions of ITIMES for different values of INTERV.

IHEAD(10) = NDSDES The number of data system descriptor records.

IHEAD(11) = ITZONE Number of hours between local time zone for this data base and GMT (i.e., 4 = EDT, 5 = EST, 6 = CST, 0 = GMT).

IHEAD(12-20) Unused (reserved for future use).

INTERV	BASIC TIME INTERVAL	IYEARS	IJULS	ITIMES
6	YEARS	YEAR	NOT USED	NOT USED
5	MONTHS	YEAR	MONTH	NOT USED
4	DAYS	YEAR	JULIAN DAY	NOT USED
3	HOURS	YEAR	JULIAN DAY	HOURS
2	MINUTES	YEAR	JULIAN DAY	MINUTES
1	SECONDS	YEAR	JULIAN DAY	SECONDS
-1	TENTHS OF SECONDS	YEAR	JULIAN DAY	TENTHS OF SECONDS
-2	HUNDREDTHS OF SECONDS	YEAR	JULIAN DAY	HUNDREDTHS OF SECONDS

Figure 3. --Date and time arguments relative to different values of INTERV. Note that IYEARS=IHEAD(7), IJULS=IHEAD(8), and ITIMES=IHEAD(9) may have different meanings depending on the value of INTERV=IHEAD(5).

2.2 Descriptor Records

Records 2 through (NDSDES + 1) of the header file consist of 80-character Descriptor Records:

DSDES(N) character*80 descriptors of data set. N = 1,2,...,NDSDES.

These records should contain a summary of general information about a particular data base, i.e., who collected the data, where they were collected, why they were collected, and reference to any relevant literature about the data base.

2.3 Subheader Records

Record (NDSDES + 2) through record (NDSDES + NUMPTS + 1) of the header file contain Subheader Records, one for each PTS in the data system. Each Subheader Record consists of the following fields:

DESCR Eight-character descriptor for this PTS.
* DESCR(1:5) is the station identifier or station number.
* DESCR(6:7) is the parameter descriptor ('AT' for air temp. etc.).
DESCR(8:8) is the parameter type ('R' for real, 'I' for integer).

* NVERT Height or depth relative to water level, in centimeters,

RLAT Latitude in degrees north (i.e., 43.887).

RLON Longitude in degrees west (i.e., 82.667).

BIAS Bias to be added to each data value, if required or IBIAS if DESCR(8:8) = 'I').

RMISS Code for missing data (or IMISS if DESCR(8:8) = 'I').

REDIT Code for edited data (or IEDIT if DESCR(8:8) = 'I').

UNITS 48-character description of units (i.e., 'CM/SEC' for currents) for this PTS and any other relevant information.

The parameter descriptor field should be used to identify the physical parameter being measured in this PTS. These parameter descriptors are suggested:

---

*These fields uniquely identify any PTS in the data file. No two PTS's in the same data file can have identical station identifier, parameter descriptor, and height. All other fields in the Subheader Record are not required by GDAS but should be included to fully document the contents of the data file. In addition, Descriptor Records can be used to describe the particular station identifiers, parameter descriptors, and height values used in the Subheader Records.

CS - current speed

CD - current direction

CU - eastward component of current

CV - northward component of current

WS - wind speed

WD - wind direction

AT - air temperature

WT - water temperature

BP - barometric pressure

RH - relative humidity

WL - water level

SW - significant waveheight

DP - dominant wave period

AP - average wave period

PF - peak energy wave frequency

Any nonstandard parameter descriptors used in a Subheader Record should be defined in the Descriptor Records.

When the header file for a data base is initially created, the user is required to specify values for all the parameters in the Header Record and Subheader Records as well as the Descriptor Records. These values are then fixed for that data base. In order to change any of the values, a new header file and a new data file should be created.

3. GDAS PROGRAMMING EXAMPLES

3.1 Creating a New GDAS Data Base (Header File and Initialized Data File)

First: Dimension header file arrays.

(1) Dimension Header Record array IHEAD(21).

(2) Dimension 80-character Descriptor Record array DSDES(N) to at least the number of descriptor records describing the data system.

(3) Dimension all Subheader Record array variables to the number of PTS's (parameter time-series) in the data system.

```
C
C ARRAYS CONTAINING HEADER RECORD AND DESCRIPTOR RECORDS
C
      DIMENSION IHEAD(21)
      CHARACTER DSDES(5)*80
C
C ARRAYS CONTAINING SUBHEADER RECORDS OF THE DATA BASE HEADER
C  FILE
C
      DIMENSION NVERT(500),RLAT(500),RLON(500)
      DIMENSION BIAS(500),RMISS(500),REDIT(500)
      CHARACTER DESCR(500)*8, UNITS(500)*48
```

Second: Fill the Header Record, Descriptor Record, and Subheader Record arrays.

(1) Fill Header Record array IHEAD with integer values that describe the data base. See Sec. 2.1 for a description of what the integer values in IHEAD mean.

(2) Fill 80-character Descriptor Record array DSDES(N) with text that describes the data system.

(3) Fill all Subheader Record arrays, one entry for each PTS in the data system.

```
C
C SET VALUES IN THE HEADER RECORD ARRAY FOR A DATA BASE IN
C TIME-SERIES FORMAT WITH A RECORDING INTERVAL OF ONE HOUR, A
C RECORDING LENGTH OF 24 HOURS, CONTAINING 500 PTS'S, STARTING
C AT 0000 GMT ON 1 JANUARY 1983, AND EXTENDING TO 2300 GMT ON
C 31 DECEMBER 1983.
C
      IHEAD(1) = 24
      IHEAD(2) = 1
      IHEAD(3) = 500
      IHEAD(4) = 365
      IHEAD(5) = 3
      IHEAD(6) = 1
      IHEAD(7) = 1983
      IHEAD(8) = 1
      IHEAD(9) = 0
      IHEAD(10)= 5
      IHEAD(11)= 0
C
C READ THE DATA SYSTEM DESCRIPTOR RECORDS FROM A FILE
C
      DO 10 I = 1, IHEAD(10)
      READ(1,'(A)') DSDES(I)
   10 CONTINUE
```

```
C
C READ THE SUBHEADER RECORDS FROM A FILE
C
      DO 20 I = 1, IHEAD(3)
      READ(1,*,ERR=99,END=99)  DESCR(I),NVERT(I),RLAT(I),RLON(I),
      1  BIAS(I),RMISS(I),REDIT(I),UNITS(I)
   20 CONTINUE
```

Third: Pass data base name, LUN (logical unit number), Header Record array, Descriptor Record array, and all Subheader Record arrays to GDAS subroutine DBNEW.

```
LUN = 7
CALL DBNEW('MYDATA',LUN,IHEAD,DSDES,DESCR,NVERT,RLAT,RLON,BIAS,
1  RMISS,REDIT,UNITS,DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WHILE CREATING DATA BASE'
```

If multiple GDAS data bases are used by a program at the same time, each data base will require its own IHEAD, DESCR, and NVERT arrays.

3.2 Reading From or Writing To an Existing GDAS Data Base

First: Dimension header file arrays.

(1) Dimension Header Record array IHEAD(21).

(2) Dimension Subheader Record character array DESCR(N) to the number of PTS's (parameter time-series) in the data system.

(3) Dimension Subheader Record array NVERT(N) to the number of PTS's in the data system.

(4) Dimension an array to send or receive actual data values

```
DIMENSION  IHEAD(21)
CHARACTER*8  DESCR(500)
DIMENSION  NVERT(500)
DIMENSION  DATA(120)
DATA IDIM/500/
```

Second: Open the data base with a call to DBOPEN.

```
LUN = 7     (or LUN = -7 for WRITE access to file)
CALL DBOPEN('MYDATA',LUN,IHEAD,DESCR,NVERT,IDIM,DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WHILE OPENING DATA BASE'
```

Third: Find the index corresponding to the particular station identifier parameter descriptor, and height you are interested in reading or writing.

```
C
C GET AIR TEMPERATURE DATA FOR STATION 45007 AT 5 METERS ABOVE
```

```
C WATER SURFACE
C
      STA = '45007'
      DES = 'AT'
      IHIGH = 500
      NPTS = NPTSDB(IHEAD,DESCR,NVERT,STA,DES,IHIGH)
```

Fourth: Assign the starting date for read or write and call DBREAD or DBWRITE to read or write data values.

```
C
C GET 120 DATA VALUES STARTING AT 0600 GMT ON 19 JULY 1983.
C
      IYR = 1983
      IJUL = 200
      ITIME = 6
      NDATA = 120
      CALL DBREAD(IHEAD,NPTS,IYR,IJUL,ITIME,NDATA,DATA,DBERR)
      IF(DBERR) PRINT *,' ERROR OCCURRED WHILE READING DATA BASE'

     OR
C
C READ DATA FROM A FILE
C
      DO 10 I = 1, 120
   10 READ(1,*) DATA(I)
      CALL DBWRIT(IHEAD,NPTS,IYR,JDAY,ITIM,NDATA,DATA,DBERR)
      IF(DBERR) PRINT *,
     1' ERROR OCCURRED WHILE WRITING TO DATA BASE'
```

3.3 Generating a Summary of an Existing Data Base Header File

First: Pass data base name and option parameters to GDAS subroutine DBINFO.

```
CALL DBINFO('MYDATA',1,1,1,1,1,1,DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WHILE GENERATING SUMMARY' .
```

3.4 Determining Data Availability

First: Dimension header file arrays.

(1) Dimension Header Record array IHEAD(21).

(2) Dimension all Subheader Record array variables to the number of PTS's (parameter time-series) in the data system.

```
C
C ARRAYS CONTAINING HEADER RECORD AND SUBHEADER RECORDS OF THE
C DATA BASE HEADER FILE
C
    DIMENSION IHEAD(21)
    DIMENSION NVERT(500),RLAT(500),RLON(500)
```

```
DIMENSION BIAS(500),RMISS(500),REDIT(500)
CHARACTER DESCR(500)*8, UNITS(500)*48
```

Second: Open the data base with a call to DBOPEN.

```
LUN = 7     (or LUN = -7 for WRITE access to file)
CALL DBOPEN('MYDATA',LUN,IHEAD,DESCR,NVERT,IDIM,DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WHILE OPENING DATA BASE'
```

Third: Retrieve subheader data with a call to DBSUBH.

```
CALL DBSUBH('MYDATA',IHEAD,RLAT,RLON,BIAS,RMISS,REDIT,UNITS, DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WHILE READING SUB HEADER'
```

Fourth: Call subroutine DBLOOK to scan data base data matrix

```
CALL DBLOOK(IHEAD,DESCR,NVERT,RMISS,DBERR)
IF(DBERR) PRINT *,' ERROR OCCURRED WITHIN SUBROUTINE DBLOOK'
```

4. GDAS SUBROUTINES

As more experience is gained with the GDAS system, additional useful subroutines will probably be developed by users of the sytem. It is hoped that the structure of the basic subroutines will serve as a model for the structure of any new subroutines. These subroutines are now available:

DBDAT - Creates a GDAS data file and initializes all data values in the data file file to the missing data flag for each PTS.

DBDATE - Adjusts time by incrementing the time arguments by the specified number of days.

DBHED - Creates a GDAS header file.

DBINDX - Returns the column index number of the data matrix corresponding to a particular date.

DBINFO - Prints different types of summaries of the information in the header file.

DBJDAT - Returns the Julian day given the year, month, and day.

DBLOOK - Allows the user to scan the data base data file to determine data availability.

DBMMDD - Converts a given year and Julian day to month and day of month.

DBNEW - Creates a GDAS header file and a GDAS data file and initializes all data values in the data file to the missing data flag for each PTS.

DBOPEN - Opens an existing GDAS data file and reads the GDAS header file.

DBREAD - Retrieves data values from a GDAS data file.

DBSUBH - Reads the Subheader Records from the GDAS header file.

DBTINC - Adds a specified number of basic time intervals to a specified date.

DBWRIT - Writes data values to a GDAS data file.

NPTSDB - Returns the row number in the data matrix given station identifier, parameter descriptor, and height.

NRECDB - Returns the record number of the record in the data file corresponding to a particular date.

5. DOCUMENTATION OF SUBROUTINES

5.1 DBDAT

DBDAT (NAME, LUN, IHEAD, RMISS, DBERR)

Purpose: This subroutine creates a GDAS data file with the VAX filename NAME and file extension '.DAT' and initializes all data values in the data file to the missing data flag for each PTS.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS data file.

LUN - FORTRAN I/O unit to be associated with GDAS data file.

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (1-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

RMISS - Array of dimension IHEAD(3) containing missing data flag value for all PTS's in this data set.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O Units used:

LUN - FORTRAN I/O unit associated with GDAS data file for this data set.
6 - Used for error reports.

5.2 DBDATE

DBDATE (IYEARS, IMON, IDAY, IDINC, IYEARSM, IMONM, IDAYM, DBERR)

Purpose: This subroutine adjusts time by incrementing the time arguments (IYEARS,IMON,IDAY) by the specified number of DAYS specified by IDINC.

Arguments on input:

IYEARS - Starting year (no limit restriction).

IMON - Starting month (1 - 12).

IDAY - Starting day (1 - 31).

IDINC - Number of days to increment time arguments. No limit on the number of days to advance time.

Arguments on output:

IYEARSM - The adjusted year (no limit restriction).

IMONM - The adjusted month (1 - 12).

IDAYM - The adjusted day (1 - 31).

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O units used:

6 - Used for error reports.

5.3 DBHED

DBHED (NAME, IHEAD, DSDES, DESCR, NVERT, RLAT, RLON, BIAS, RMISS, REDIT, UNITS, DBERR)

Purpose: This subroutine creates a GDAS header file with the VAX filename NAME and file extension '.HED'.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS header file.

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (1-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

DSDES - CHARACTER*80 array of dimension IHEAD(12) containing data system descriptor records for this GDAS data set.

DESCR - CHARACTER*8 array of dimension IHEAD(3) containing station identifier (DESCR(1:5)), parameter descriptor (DESCR(6:7)), and type (DESCR(8:8)) for all PTS's in this data set.

NVERT - Array of dimension IHEAD(3) containing height or depth relative to water level in centimeters for all PTS's in this data set.

RLAT - Array of dimension IHEAD(3) containing latitude in degrees north for all PTS's in this data set.

RLON - Array of dimension IHEAD(3) containing longitude in degrees west for all PTS's in this data set.

BIAS - Array of dimension IHEAD(3) containing biases for data values for all PTS's in this data set.

RMISS - Array of dimension IHEAD(3) containing missing data flag value for all PTS's in this data set.

REDIT - Array of dimension IHEAD(3) containing edited data flag value for all PTS's in this data set.

UNITS - CHARACTER*48 array of dimension IHEAD(3) containing description of units for all PTS's in this data set.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O Units used:

99 - Temporarily associated with GDAS header file for this data set.
6 - Used for error reports.

5.4 DBINDX

DBINDX (IHEAD, IY2, IJ2, IT2, JINDEX, DBERR)

Purpose: This function calculates the elapsed time, in basic time units, between the beginning date of the data base stored in IHEAD as year, Julian day, and time in seconds (IHEAD(7)-IHEAD(9)) and a given date given as year (IY2), Julian day (IJ2), and time (IT2). The subroutine then returns the column index number of the data matrix corresponding to date IY2, IJ2, and IT2. The subroutine also makes sure that a recording was made at the time specified.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file. (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical UNIT number associated with the GDAS data file.

IY2 - Year for which column index number will be returned.

IJ2 - Julian day (or month if INTERV = 5) for which column index number will be returned. Value not used if INTERV = 6.

IT2 - Time in basic time units for which column index number will be returned. Value not used if INTERV > 3.

Arguments on output:

JINDEX - Column index number of the data matrix corresponding to date IY2, IJ2, and IT2.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O units used:

6 - Used for error reports.

5.5 DBINFO

DBINFO (NAME, IPARl, IPAR2, IPAR3, IPAR4, IPAR5, IPAR6, DBERR)

Purpose: Prints six different types of summaries of the information in the header file for data set NAME. This subroutine assumes that a file with the VAX filename NAME and the file extension '.HED' is a GDAS header file. If any one of the values of IPARl through IPAR5 is equal to 1 then the summary report corresponding to that parameter is printed. If any one of the values of IPARl through IPAR5 is not equal to 1 then the summary report corresponding to that parameter is not printed. See the input arguments for a description of the summary report that is associated with a particular parameter.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS data set.

IPARl - Prints the descriptor records and the Header Record.

IPAR2 - Prints all the parameter descriptor and height pairs for each station identifier.

IPAR3 - Prints the station location for each station identifier.

IPAR4 - Prints all the station identifiers for each parameter descriptor and height pair.

IPAR5 - Prints a subheader summary including all subheader parameters except the 48-character descriptor for units.

IPAR6 - Prints a subheader summary including station identifier, parameter descriptor, height, and 48-character descriptor for units.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

I/O units used:

6 - Formatted summary and error reports.
99 - Temporarily associated with header file.

Subroutines called:

DBMMDD - Accepts the year and Julian day and returns the month and day of month.

DBDATE - Adjusts time by incrementing the time arguments by the specified number of seconds.

DBTINC - Adds a specified number of basic time units to a specified date.

5.6 DBJDAT

DBJDAT (IYEARS, IMON, IDAY, IJULS, DBERR)

Purpose: This subroutine converts a Gregorian calendar date to the corresponding Julian day number IJULS. The Julian day number IJULS is computed from the given day IDAY, month IMON, and year IYEARS. without using tables. The procedure is valid for any valid Gregorian calendar date. Leap year is defined to be any year divisible by 4 except centenary years not divisible by 400.

Arguments on input:

IYEARS - Year for which Julian day is to be calculated.

IMON - Month for which Julian day is to be calculated.

IDAY - Day for which Julian day is to be calculated.

Arguments on Output:

IJULS - Julian day calculated from IYEARS, IMON and IDAY.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O Units used:

6 - Used for error report.

5.7 DBLOOK

DBLOOK (IHEAD, DESCR, NVERT, RMISS, DBERR)

Purpose: This subroutine allows the user to scan the data file to determine if data are available, partially available, or missing. The subroutine is useful because of the speed with which the user can cursor through the data matrix to determine the availability of data, Each entry in the table created by this subroutine for data

bases in time-series format represents a whole record as defined by the value of NCOL in the data base header file. Each entry in the table created by this subroutine for data bases in time-synchronous format represents an individual data value. Input parameters IHEAD, DESCR, and NVERT are initialized by calling GDAS subroutine DBOPEN. Input parameter RMISS is initialized by calling GDAS subroutine DBSUBH.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

DESCR - CHARACTER*8 array of dimension IHEAD(3) containing station identifier (DESCR(1:5)), parameter descriptor (DESCR(6:7)), and type (DESCR(8:8)) for all PTS's in this data set.

NVERT - Array of dimension IHEAD(3) containing height or depth relative to water level in centimeters for all PTS's in this data set.

RMISS - Array of dimension IHEAD(3) containing missing data flag value for all PTS's in this data set.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

DBMMDD - Accepts the year and Julian day and returns the month and day of month.

DBTINC - Adds 'INC' basic time units to the date.

NPTSDB - Returns the row number in the data matrix, given a station identifier, parameter descriptor, and height.

NRECDB - Returns the record number, given a PTS and a date.

I/O units used:

5 - Used for information prompting.
6 - Used for reporting results.

5.8 DBMMDD

DBMMDD (IYEARS, IJULS, IMON, IDAY, DBERR)

Purpose: This routine accepts the year and Julian day (IYEARS,IJULS) as input, and returns the month and day of month (IMON,IDAY).

Arguments on input:

IYEARS - Year for which month and day of month are calculated.

IJULS - Julian day for which month and day of month are calculated.

Arguments on output:

IMON - Month calculated from date IY1, and IJUL1.

IDAY - Day of month calculated from date IY1, and IJUL1.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL ‘and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O units used:

6 - Used for error reports.

5.9 DBNEW

DBNEW (NAME, LUN, IHEAD, DSDES, DESCR, NVERT, RLAT, RLON, BIAS, RMISS, REDIT, UNITS, DBERR)

Purpose: This subroutine creates a GDAS header file and a GDAS data file with the VAX filename NAME and file extensions of ’.HED’ and ’.DAT’ respectively. The GDAS data file remains associated with FORTRAN I/O unit LUN.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS header and data files.

LUN - FORTRAN I/O unit to be associated with GDAS data file.

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track

of the FORTRAN logical unit number associated with the GDAS data file.

DSDES - CHARACTER*80 array of dimension IHEAD(12) containing data system descriptor records for this GDAS data set.

DESCR - CHARACTER*8 array of dimension IHEAD(3) containing station identifier (DESCR(1:5)), parameter descriptor (DESCR(6:7)), and type (DESCR(8:8)) for all PTS's in this data set.

NVERT - Array of dimension IHEAD(3) containing height or depth relative to water level in centimeters for all PTS's in this data set.

RLAT - Array of dimension IHEAD(3) containing latitude in degrees north for all PTS's in this data set.

RLON - Array of dimension IHEAD(3) containing longitude in degrees west for all PTS's in this data set.

BIAS - Array of dimension IHEAD(3) containing biases for data value for all PTS's in this data set.

RMISS - Array of dimension IHEAD(3) containing missing data flag value for all PTS's in this data set.

REDIT - Array of dimension IHEAD(3) containing edited data flag value for all PTS's in this data set.

UNITS - CHARACTER*48 array of dimension IHEAD(3) containing description of units for all PTS's in this data set.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

DBHED - Creates a GDAS header file.
DBDAT - Creates a GDAS data file.

I/O units used:

LUN - FORTRAN I/O unit associated with GDAS data file for this data set.
99 - temporarily associated with GDAS header file for this data set.
6 - Used for error reports.

5.10 DBOPEN

DBOPEN (NAME, LUN, IHEAD, DESCR, NVERT, NDIM, DBERR)

Purpose: This subroutine opens an existing GDAS data file with the VAX filename NAME and associates it with FORTRAN I/O unit LUN. It also reads the GDAS header file for this data file and initializes the arrays IHEAD, DESCR, and NVERT. The GDAS data file NAME'.DAT' will be opened as READONLY if LUN is greater than 0. It will be opened for WRITE ACCESS if LUN is less than 0.

Note: If multiple GDAS data bases are used by a program at the same time, each data base will require its own IHEAD, DESCR, and NVERT arrays.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS header and data files.

LUN - FORTRAN I/O unit to be associated with GDAS data file. The actual FORTRAN I/O unit number used is the absolute value of LUN.

NDIM - Dimension of arrays DESCR and NVERT as dimensioned in calling program.

Arguments on output:

IHEAD - Array of dimension 21. IHEAD is returned as the 20 integer values that describe the organization of the data file. IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

IHEAD(1) = NCOL The number of data values per record for each parameter for a data file in "time-series" format, or 1 for a data file in "time-synchronous,, format.

IHEAD(2) = NROW The number of data values per record for each time for a data file in "time-synchronous,, format, or 1 for a data file in 'time-series" format. NROW must be a submultiple of NUMPTS.

IHEAD(3) = NUMPTS The number of parameter time-series (PTS's) in the data set. A PTS is a time series of values of a particular parameter at a particular location (latitude, longitude, and height or depth from surface).

IHEAD(4) = NMAX Total number of data records in the data file for each PTS.

IHEAD(5) = INTERV Basic time interval indicator.

6 = Years
5 = Months
4 = Days
3 = Hours
2 = Minutes
1 = Seconds
-1 = Tenths of Seconds
-2 = Hundredths of Seconds

Figure 3 illustrates the effect a particular choice of INTERV has on the meaning of IYEARS, IJULS, and ITIMES.

IHEAD(6) = NINTRV Recording interval, i.e., the time interval between successive data values in units of the basic time interval (INTERV).

IHEAD(7) = IYEARS The year of the first record in the data file.

IHEAD(8) = IJULS The Julian day (or month if INTERV = 5) of the first record in the data file. Not used if INTERV = 6. See Figure 3 for definitions of IJULS for different values of INTERV.

IHEAD(9) = ITIMES The number of basic time intervals (INTERV) after 0000 local time (see IHEAD(11)) at which the first record in the data file starts. Not used if INTERV > 3. See Figure 3 for definitions of ITIMES for different values of INTERV.

IHEAD(10) = NDSDES The number of data system descriptor records.

IHEAD(11) = ITZONE Number of hours between local time zone for this data base and GMT (i.e., 4 = EDT, 5 = EST, 6 = CST, 0 = GMT).

IHEAD(12-20) Unused (reserved for future use).

DESCR - CHARACTER*8 array of dimension IHEAD(3) returned as the station identifier (DESCR(1:5)), parameter descriptor (DESCR(6:7)), and type (DESCR(8:8)) for all PTS's in this data set.

NVERT - Array of dimension IHEAD(3) returned as the height or depth relative to water level in centimeters for all PTS's in this data set.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O units used:

LUN - Associated with GDAS data file for this data set.
99 - Temporarily associated with GDAS header file for this data set.
6 - Used for error reports.

5.11 DBREAD

DBREAD (IHEAD, NPTS, IYR, JDAY, ITIM, NDATA, DATA, DBERR)

Purpose: This subroutine retrieves data values from a GDAS data file. For a data file in "time-series" format, NDATA data values for PTS index number NPTS starting at IYR, JDAY, ITIM are returned in the array DATA. For a data file in "time-synchronous" format, data for PTS index NPTS through NPTS+NDATA-1 at IYR, JDAY, ITIM are returned in the array DATA. Function NPTSDB can be used to determine NPTS for a given station identifier, parameter descriptor, and height. For a "time-series" data file, an error results if the index number of the last data value requested is greater than the maximum index. For a "time-synchronous" data file, an error results if NPTS+NDATA+1 is greater than the total number of PTS's in the data base.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (1-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

NPTS - The index number of the PTS for which data are to be read.

IYR - Desired year.

JDAY - Desired Julian day (month if INTERV = 5). Not used if INTERV = 6.

ITIM - Desired time in basic time units. Not used if INTERV = 4, 5 or 6.

NDATA - The number of data values to read.

Arguments on output:

DATA - Array in which data values are returned by DBREAD.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

NRECDB - Returns the record number, given a PTS and a date.
DBINDX - Returns the column index number of the data matrix, given a date.

I/O units used:

IHEAD(21) - FORTRAN I/O unit associated with GDAS data file for this data set.

6 - Used for error reports.

5.12 DBSUBH

DBSUBH (NAME, IHEAD, RLAT, RLON, BIAS, RMISS, REDIT, UNITS, DBERR)

Purpose: This subroutine reads the GDAS header file NAME and returns values for RLAT, RLON, BIAS, RMISS, REDIT, and UNITS for all PTS's in this data set.

Arguments on input:

NAME - CHARACTER*80 VAX filename for GDAS header file.

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

Arguments on output:

RLAT - Array of dimension IHEAD(3) returned as latitude in degrees north for all PTS's in this data set.

RLON - Array of dimension IHEAD(3) returned as longitude in degrees west for all PTS's in this data set.

BIAS - Array of dimension IHEAD(3) returned as biases for data values for all PTS's in this data set.

RMISS - Array of dimension IHEAD(3) returned as missing data flag value for all PTS's in this data set.

REDIT - Array of dimension IHEAD(3) returned as edited data flag value for all PTS's in this data set.

UNITS - CHARACTER*48 array of dimension IHEAD(3) returned as description of units for all PTS's in this data set.

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

None

I/O units used:

99 - Temporarily associated with GDAS header file for this data set.
6 - Used for error reports.

5.13 DBTINC

DBTINC (IHEAD, IYR, IMON, IDAY, IJUL, ITIME, INC,
IYEARSM, IMONM, IDAYM, IJULM, IHM, IMINM,
ISECM, ITENM, IHUNM, DBERR)

Purpose: This subroutine adds the number of basic time units indicated by input parameter 'INC' to the date and converts the time calculated in basic time units to hours, minutes, seconds, tenths of seconds, and hundredth of seconds.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

IYR - Year for which INC basic time units will be added.

IMON - Month for which INC basic time units will be added. Not used if INTERV > 4.

IDAY - Day of month for which INC basic time units will be added. Not used if INTERV > 4.

IJUL - Julian day if $-2 \leq$ INTERV ≤ 4 for which INC basic time units will be added. Month if INTERV = 5 for which INC basic time units will be added. Not used if INTERV = 6.

ITIME - Time in basic time units for which INC basic time units will be added. Not used if INTERV > 3.

INC - Number of basic time units to increment date and time arguments.

Arguments on output:

IYEARSM - Adjusted year after incrementing day and time arguments.

IMONM - Adjusted month after incrementing day and time arguments. Not adjusted if INTERV = 6.

IDAYM - Adjusted day of month after incrementing day and time arguments. Not adjusted if INTERV > 4.

IJULM - Adjusted Julian day if 4 ≤ INTERV ≤ -2 after incrementing day and time arguments. Month adjusted if INTERV = 5 and not adjusted if INTERV = 6.

IHM - Calculated hour after incrementing day and time arguments. Not calculated if INTERV > 3.

IMINM - Calculated minute after incrementing day and time arguments. Not calculated if INTERV > 2.

ISECM - Calculated second after incrementing day and time arguments. Not calculated if INTERV > 1.

ITENM - Calculated tenth of second after incrementing day and time arguments. Not calculated if INTERV > -1.

IHUNM - Calculated hundredth of second after incrementing day and time arguments. Not calculated if INTERV > -2.

DBERR - LOGICAL variable returned with the value of **.FALSE.** if no errors are detected and the value of **.TRUE.** if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

DBDATE - Adjusts time by incrementing the time arguments by the specified number of days.

I/O units used:

None

5.14 DBWRIT

DBWRIT (**IHEAD**, NPTS, IYR, JDAY, **ITIM**, NDATA, DATA, DBERR)

Purpose: This subroutine writes data values to a GDAS data file. For a data file in "time-series" format, NDATA data values for PTS index number NPTS starting at IYR, JDAY, **ITIM** are written to the data file from the array DATA. For a data file in "time-synchronous" format, data for PTS index NPTS through NPTS+NDATA-1 at IYR, JDAY, **ITIM** are written to the data file from the array DATA. Function NPTSDB can be used to determine NPTS for a given station identifier, parameter descriptor, and height. For a "time-series" data file, an error results if the index number of the last data value requested is greater than the maximum index. For a "time-synchronous" data file, an error results if **NPTS+NDATA+1** is greater than the total number of **PTS's** in the data base.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (1-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

NPTS - The index number of the PTS for which data are to be written.

IYR - Desired year.

JDAY - Desired Julian day (month if INTERV = 5). Not used if INTERV = 6.

ITIM - Desired time in basic time units. Not used if INTERV = 4, 5, or 6.

NDATA - The number of data values to write.

DATA - Array from which data values are written by DBWRIT.

Arguments on output:

DBERR - LOGICAL variable returned with the value of .FALSE. if no errors are detected and the value of .TRUE. if an error is detected. Calling program must declare this variable to be of type LOGICAL and is responsible for checking this variable to see if an error has occurred.

Subroutines called:

NRECDB - Returns the record number, given a PTS and a date.
DBINDX - Returns the column index number of the data matrix, given a date.

I/O units used:

IHEAD(21) - FORTRAN I/O unit associated with GDAS data file for the data set.
6 - Used for error reports.

5.15 NPTSDB

FUNCTION NPTSDB (IHEAD, DESCR, NVERT, STA, DES, IHIGH)

Purpose: This function returns the row number in the data matrix for a given station identifier (STA), parameter descriptor (DES), and height (IHIGH). The arrays IHEAD, DESCR, and NVERT must have previously been initialized by DBOPEN. The value of the function is negative if an error is detected.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD

(l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

DESCR - CHARACTER*8 array of dimension IHEAD(3) containing station identifier (DESCR(1:5)), parameter descriptor (DESCR(6:7)), and type (DESCR(8:8)) for all PTS's in this data set.

NVERT - Array of dimension IHEAD(3) containing height or depth relative to water level in centimeters for all PTS's in this data set.

STA - CHARACTER*5 station identifier.

DES - CHARACTER*2 parameter descriptor.

IHIGH - Height or depth relative to water surface, in centimeters.

Subroutines called:

None

I/O units used:

6 - Used for error reports.

5.16 NRECDB

FUNCTION NRECDB (IHEAD, NPTS, IYR, JDAY, ITIM)

Purpose: This function returns the record number of the record in the data file corresponding to data for IYR, JDAY, ITIM for parameter time series number NPTS. The value of the function is negative if an error is detected.

Arguments on input:

IHEAD - Array of dimension 21. IHEAD is expected to contain the 20 integer values that describe the organization of the data file (see IHEAD (l-20) in Sec. 2.1). IHEAD(21) is used by the GDAS to keep track of the FORTRAN logical unit number associated with the GDAS data file.

NPTS - The index number of the PTS.

IYR - Desired year.

JDAY - Desired Julian day (month if INTERV = 5). Not used if INTERV = 6.

ITIM - Desired time in basic time units. Not used if INTERV = 4, 5 or 6.

Subroutines called:

DBINDX - Returns the column index number of the data matrix given a date,

I/O units used:

6 - Used for error reports.

6. GDAS SAMPLE PROGRAM

```
      PROGRAM GDASM
C
C     This is a sample program that demonstrates the use of the GDAS
C  data base subroutines.
C     This program can be used to create a new user-defined header
C  file by creating a formatted ASCII file with the name of the data
C  base and extension '.ASC' that contains the header file information
C  in the format described below.
C     LINE 1 of the formatted ASCII header file is the Header Record, which
C  consists of 20 integer values describing the organization of the data
C  file.
C     LINES 2 through (NDSDES + 1) of the formatted ASCII header file
C  consist of 80-character descriptor records.
C     LINES (NDSDES + 2) through line (NDSDES + NUMPTS + 1) of the formatted
C  ASCII header file contain Subheader Records, one for each PTS in the
C  data system.
C     This program reads the formatted ASCII file and sends the data to
C  subroutine 'DBNEW' where an unformatted binary file with an extension
C  '.HED' is created containing the header file information. In
C  addition an initialized unformatted binary data file is created
C  with the extension '.DAT'.  Once the data base has been defined in
C  this way the user can extract summaries of the header data, add data
C  to the data file, and examine data in the data file.
C     This program runs interactively (i.e., all parameter prompting
C  is output to the terminal and all parameter input is read from the
C  keyboard) regardless of an assign statement.
C     This program is set up to run interactively.  However, data that are
C  to be written to the data base can optionally be read from a file by using
C  an ASSIGN statement (e.g., ASSIGN INPUTDATA.DAT FOR005).  On program
C  termination the DEASSIGN statement should be given (e.g., DEASSIGN
C  FOR005).  Data that are to be read from the data base can optionally be
C  written to a file by using an ASSIGN statement (e.g., ASSIGN OUTPUTDATA.DAT
C  FOR006).  On program termination the DEASSIGN statement should be
C  given (e.g., DEASSIGN FOR006).
C     This program will support all basic time intervals (Header file
C  parameter number 5 = INTERV) except tenths and hundredths of seconds
C  (INTERV = -1,-2).
C
C  HISTORY: Written by Edward W. Lynn, December 1984.
C
C  Array containing Header Record of the data base header file
C
      DIMENSION IHEAD(21)
C
C  Array containing the Descriptor Records of the data base header file
```

```
C
      CHARACTER DSDES(5000)*80

C Array containing Subheader Records of the data base header file
C
      DIMENSION NVERT(5000),RLAT(5000),RLON(5000)
      DIMENSION BIAS(5000),RMISS(5000),REDIT(5000)
      CHARACTER DESCR(5000)*8, UNITS(5000)*48
      LOGICAL DBERR
      DIMENSION  DATA(5000)
      CHARACTER  NAME*80, NAME1*80, STA*5, DES*2, ANS*1
C
C Set dimension of arrays
C
      DATA IDIM/5000/
C
C Get data base file name,  do not include file extension,
C default file extensions will apply
C
    5 WRITE(*,'(A)') ' ENTER DATA BASE FILE NAME:'
      READ(*,'(A)', ERR=5,END=5) NAME
C
C User will either create or open an existing data base
C
   15 WRITE(*,'(A)') ' ENTER N IF YOU WANT TO CREATE A NEW DATA BASE'
      WRITE(*,'(A)')
     1' ENTER 0 IF YOU WANT'TO OPEN AN EXISTING DATA BASE'
      READ(*,'(A)',ERR=15,END=15) ANS
      CALL  STR$UPCASE(ANS,ANS)
      IF(ANS .NE. 'N' .AND. ANS .NE. '0') GOT0 15
C ***************************************************************************
C
C Open data base to prepare for access
C
      IF(ANS .EQ. '0') THEN
C
C Determine access type for data base and assign a LUN to the
C data base data file.  LUN must be < 0 to have data base opened
C  for write access
C
   17  WRITE(*,'(A)')
     1 ' ENTER R IF YOU WANT DATA BASE OPENED FOR READ ACCESS ONLY'
       WRITE(*,'(A)')
     1 ' ENTER W IF YOU WANT DATA BASE OPENED FOR READ/WRITE ACCESS'
       READ(*,'(A)',ERR=17,END=17) ANS
       CALL  STR$UPCASE(ANS,ANS)
       IF(ANS .NE. 'R' .AND. ANS .NE. 'W') GOT0 17
       IF(ANS .EQ. 'R') THEN
        LUN = 1
       ELSE
        LUN = -1
       END IF
       CALL DBOPEN(NAME,LUN,IHEAD,DESCR,NVERT,IDIM,DBERR)
```

```
        IF(DBERR) GOT0 5
        INTERV = IHEAD(5)
C
C ***************************************************************************
C
C Create new data base header file and initialize data base data file
C
      ELSE
C
C Confirm ASC, HED, and DAT filenames
C
        INDX = INDEX(NAME,' ') -1
        NAME1(1:INDX+4) = NAME(1:INDX) // '.ASC'
   18   WRITE(*,'(3A)')
     1 '  THE HEADER FILE IN FILE: ',NAME(1:INDX),'.ASC WILL BE USED TO'
        WRITE(*,'(3A)')
     1 '  CREATE GDAS HEADER FILE: ',NAME(1:INDX),'.HED AND GDAS'
        WRITE(*,'(3A)')
     1 '  DATA FILE: ',NAME(1:INDX),'.DAT IS THIS OK? (INPUT Y..OR..N)'
        READ(*,'(A)',ERR=18,END=18) ANS
        CALL STR$UPCASE(ANS,ANS)
        IF(ANS .NE. 'Y' .AND. ANS .NE. 'N') GOT0 18
        IF(ANS .EQ. 'N') GOT0 5
C
C Open formatted ASCII header file, file name is assumed to be the same
C as the data base with the extension '.ASC'
C
        OPEN(1,FILE=NAME1,STATUS='OLD')
C
C Fill parameters to send to 'DBNEW' subroutine, read the Header Record
C
        READ(1,*,ERR=99,END=99) (IHEAD(I),I=1,11)
        NUMPTS = IHEAD(3)
        INTERV = IHEAD(5)
        NDSDES = IHEAD(10)
C
C Make sure IDIM is greater than or equal to NUMPTS
C
        IF(IDIM .LT. NUMPTS) THEN
         WRITE(*,*)
     1   ' DIMENSION IDIM IS TO SMALL IN GDASM'
         STOP
        END IF
C
C Read the data system Descriptor Records
C
        DO 10 I = 1, NDSDES
         READ(1,'(A)',ERR=99,END=99) DSDES(I)
   10 CONTINUE
C
C Read the Subheader Records
C
        DO 11 I = 1, NUMPTS
```

```
        READ(1,*,ERR=99,END=99) DESCR(I),NVERT(I),RLAT(I),RLON(I),
     1  BIAS(I),RMISS(I),REDIT(I),UNITS(I)
   11 CONTINUE
       GOT0 27
   99  WRITE(*,'(2A)')
     1 ' ERROR OR END OF FILE FOUND ON FILE: ',NAME(1:INDX)
       CLOSE(l)
       GOT0 5
C
C Now send initialized arrays to DBNEW to create a new
C header file and initialized data file
C
   27 CALL DBNEW(NAME,LUN,IHEAD,DSDES,DESCR,NVERT,RLAT,RLON,BIAS,
     1  RMISS,REDIT,UNITS,DBERR)
       IF(DBERR) GOT0 5
      END IF
C
C Type the list of options
C
   20  WRITE(*,*) ' '
      WRITE(*,'(A)')  ' ENTER S IF YOU WANT SUMMARY INFORMATION'
      WRITE(*,'(A)')  ' ENTER R IF YOU WANT TO READ DATA'
      WRITE(*,'(A)')  ' ENTER W IF YOU WANT TO WRITE DATA'
      WRITE(*,'(A)')  ' ENTER Q IF YOU WANT TO QUIT'
      READ(*,'(A)',ERR=20,END=20) ANS
      CALL  STR$UPCASE(ANS,ANS)
      IF(ANS .NE. 'S' .AND. ANS .NE. 'R' .AND. ANS .NE. 'W' .AND.
     1   ANS .NE.  'Q') GOT0 20
C
C ***************************************************************************
C
C List options available for data base summaries
C
      IF(ANS .EQ. 'S') THEN
   22   WRITE(*,'(A)') ' '
       WRITE(*,'(A)')
     1 ' OPTION: 1 PRINTS THE DESCRIPTOR RECORDS AND THE HEADER RECORD'
       WRITE(*,'(A)')
     1 ' OPTION: 2 PRINTS ALL THE PARAMETER DESCRIPTOR AND HEIGHT',
     2 '           PAIRS FOR EACH STATION IDENTIFIER'
       WRITE(*,'(A)')
     1 ' OPTION: 3 PRINTS THE STATION LOCATION FOR EACH STATION',
     2 '           IDENTIFIER'
       WRITE(*,'(A)')
     1 ' OPTION: 4 PRINTS ALL THE STATION IDENTIFIERS FOR EACH',
     2 '           PARAMETER DESCRIPTOR AND HEIGHT PAIR'
       WRITE(*,'(A)')
     1 ' OPTION: 5 PRINTS A SUBHEADER SUMMARY INCLUDING ALL SUBHEADER',
     2 '           PARAMETERS EXCEPT 48 CHARACTER DESCRIPTOR FOR UNITS'
       WRITE(*,'(A)')
     1 ' OPTION: 6 PRINTS A SUBHEADER SUMMARY INCLUDING STATION',
     2 '           IDENTIFIER, PARAMETER DESCRIPTOR, HEIGHT, AND',
     3 '           48 CHARACTER DESCRIPTOR FOR UNITS'
```

```
      WRITE(*,'(A)') ' OPTION: 7 RETURN TO MAIN MENU'
      WRITE(*,'(A)') ' '
      WRITE(*,'(A)') ' ENTER OPTION NUMBER:'
      READ(*,*,ERR=22,END=22) IOP
      IF(IOP .LT. 1 .OR. IOP .GT. 7) GOT0 22
      IF(IOP .EQ. 1) THEN
       CALL DBINFO(NAME,1,0,0,0,0,0,DBERR)
      ELSE IF(IOP .EQ. 2) THEN
       CALL DBINFO(NAME,0,1,0,0,0,0,DBERR)
      ELSE IF(IOP .EQ. 3) THEN
       CALL DBINFO(NAME,0,0,1,0,0,0,DBERR)
      ELSE IF(IOP .EQ. 4) THEN
       CALL DBINFO(NAME,0,0,0,1,0,0,DBERR)
      ELSE IF(IOP .EQ. 5) THEN
       CALL DBINFO(NAME,0,0,0,0,1,0,DBERR)
      ELSE IF(IOP .EQ. 6) THEN
       CALL DBINFO(NAME,0,0,0,0,0,1,DBERR)
      ELSE IF(IOP .EQ. 7) THEN
       GOT0 20
      END IF
      GOT0 22
C
C ***************************************************************************
C
C Example of reading data from data base, given Start Date, Station
C Identifier, Parameter Descriptor, Height, and number of data
C values required
C
      ELSE IF(ANS .EQ. 'R') THEN
   25 IF(INTERV .EQ. 6) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR TO READ DATA BASE    (E.G. 1984)'
        READ(*,*,ERR=25,END=25) IYR
       ELSE IF(INTERV .EQ. 5) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR AND MONTH TO READ DATA BASE    (E.G. 1984,2)'
        READ(*,*,ERR=25,END=25) IYR, IJUL
       ELSE IF(INTERV .EQ. 4) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR AND JULIAN DAY TO READ DATA BASE    (E.G. 1984,345)'
        READ(*,*,ERR=25,END=25) IYR, IJUL
       ELSE IF(INTERV .EQ. 3) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR, JULIAN DAY AND TIME IN HOURS TO READ DATA BASE',
     2  ' (E.G. 1984,345,12)'
        READ(*,*,ERR=25,END=25) IYR, IJUL, ITIME
       ELSE IF(INTERV .EQ. 2) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR, JULIAN DAY AND TIME IN MINUTES TO READ DATA BASE',
     2  ' (E.G. 1984,345,720)'
        READ(*,*,ERR=25,END=25) IYR, IJUL, ITIME
       ELSE IF(INTERV .EQ. 1) THEN
        WRITE(*,'(A)')
```

```
     1 '  ENTER YEAR, JULIAN DAY AND TIME IN SECONDS TO READ DATA BASE',
     2  ' (E.G. 1984,345,43200)'
        READ(*,*,ERR=25,END=25) IYR, IJUL, ITIME
       ELSE
        WRITE(*,'(A)')
     1  ' SORRY, SAMPLE MAIN PROGRAM ''G. CM" DOES NOT SUPPORT BASIC',
     2  ' TIME INTERVALS LESS THEN A SECOND.'
        STOP
       END IF
C
   30 WRITE(*,'(A)')
     1' ENTER THE STATION IDENTIFIER, PARAMETER DESCRIPTOR, AND',
     2' HEIGHT OR DEPTH RELATIVE TO WATER LEVEL IN CENTIMETERS',
     3' (E.G.  ''TWR'',''AT'',500)'
       READ(*,*,ERR=30,END=30) STA, DES, IHIGH
C
C Get the column index corresponding to the particular Station Identifier,
C Parameter Descriptor, and Height
C
       NPTS = NPTSDB(IHEAD,DESCR,NVERT,STA,DES,IHIGH)
       IF(NPTS .LT. 0) GOT0 20
C
   35  WRITE(*,'(A)')
     1 ' ENTER THE NUMBER OF DATA VALUES THAT YOU WOULD LIKE READ'
       READ(*,*,ERR=35,END=35) NDATA
C
C On return from 'DBREAD', NDATA values will be in array DATA for your use
C
       CALL DBREAD(IHEAD,NPTS,IYR,IJUL,ITIME,NDATA,DATA,DBERR)
       IF(DBERR) GOT0 20
       WRITE(6,*) (DATA(I),I=1,NDATA)
       GOT0 20
C
C ***************************************************************************
C
C Example of writing data to data base given Start Date, Station
C Identifier, Parameter Descriptor, Height, and number of data
C values required
C
      ELSE IF(ANS .EQ. 'W') THEN
   40 IF(INTERV .EQ. 6) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR TO WRITE DATA BASE    (E.G. 1984)'
        READ(*,*,ERR=40,END=40) IYR
       ELSE IF(INTERV .EQ. 5) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR AND MONTH TO WRITE DATA BASE    (E.G. 1984,2)'
        READ(*,*,ERR=40,END=40) IYR, IJUL
       ELSE IF(INTERV .EQ. 4) THEN
        WRITE(*,'(A)')
     1  ' ENTER YEAR AND JULIAN DAY TO WRITE DATA BASE  (E.G. 1984,345)'
        READ(*,*,ERR=40,END=40) IYR, IJUL
       ELSE IF(INTERV .EQ. 3) THEN
```

```
         WRITE(*,'(A)')
     1   ' ENTER YEAR, JULIAN DAY AND TIME IN HOURS TO WRITE DATA BASE',
     2   ' (E.G. 1984,345,12)'
         READ(*,*,ERR=40,END=40) IYR, IJUL, ITIME
        ELSE IF(INTERV .EQ. 2) THEN
         WRITE(*,'(A)')
     1   ' ENTER YEAR, JULIAN DAY AND TIME IN MINUTES TO WRITE DATA BASE'
     2   ,' (E.G. 1984,345,720)'
         READ(*,*,ERR=40,END=40) IYR, IJUL, ITIME
        ELSE IF(INTERV .EQ. 1) THEN
         WRITE(*,'(A)')
     1   ' ENTER YEAR, JULIAN DAY AND TIME IN SECONDS TO WRITE DATA BASE'
     2   ,' (E.G. 1984,345,43200)'
         READ(*,*,ERR=40,END=40) IYR, IJUL, ITIME
        ELSE
         WRITE(*,'(A)')
     1   ' SORRY, SAMPLE MAIN PROGRAM "GDASM" DOES NOT SUPPORT BASIC',
     2   ' TIME INTERVALS LESS THEN A SECOND.'
         STOP
        END IF
C
   45    WRITE(*,'(A)')
     1' ENTER THE STATION IDENTIFIER, PARAMETER DESCRIPTOR, AND',
     2' HEIGHT OR DEPTH RELATIVE TO WATER LEVEL IN CENTIMETERS',
     3' (E.G.  ''TWR'',''AT'',500)'
        READ(*,*,ERR=45,END=45) STA, DES, IHIGH
C
C  Get the column index corresponding to the Particular Station Identifier,
C  Parameter Descriptor, and Height
C
        NPTS =  NPTSDB(IHEAD,DESCR,NVERT,STA,DES,IHIGH)
        IF(NPTS .LT. 0) GOT0 20
C
   50  WRITE(*,'(A)')
     1' ENTER THE NUMBER OF DATA VALUES THAT YOU WOULD LIKE TO WRITE'
        READ(*,*,ERR=50,END=50) NDATA
C
   55  WRITE(*,'(A,I4,A)')
     1' ENTER ',NDATA,' VALUES THAT YOU WOULD LIKE TO WRITE'
        READ(5,*,ERR=56,END=56) (DATA(I),I=1,NDATA)
        GOT0 57
   56  WRITE(*,*) ' ERROR IN INPUTING DATA'
        GOT0 20
C
C  On return from 'DBWRIT', NDATA values from array DATA will have
C  been written to the data base
C
   57 CALL DBWRIT(IHEAD,NPTS,IYR,IJUL,ITIME,NDATA,DATA,DBERR)
       IF(DBERR) GOT0 20
       GOT0 20
      END IF
      END
```

7. SUPPLEMENTARY DATA FOR GDAS

7.1 GDAS Files

GDAS SOURCE CODE:

[PLM.GDAS]GDAS.FOR	FILE CONTAINING GDAS SUBROUTINE SOURCE CODE.
[PLM.GDAS]GDASM.FOR	EXAMPLE MAIN PROGRAM USING GDAS SUBROUTINES.

GDAS OBJECT LIBRARY:

[PLM.GDAS]GDAS.OLB	OBJECT LIBRARY CONTAINING GDAS SUBROUTINES

7.2 Example Compilation, Link, and Run of Users Main Program with GDAS Subroutine Object Library

```
FOR prog
LINK  prog,[PLM.GDAS]GDAS/L
RUN prog
```

7.3 How To Run the Example Interactive Main Program

```
RUN [PLM.GDAS]GDASM
```

7.4 Examples of Data Base Data Files and Header Files in GDAS Format

NDBC DATA

```
DISK$NDBC:[NDBC]NDBC1981.DAT, DISK$NDBC:[NDBC]NDBC1981.HED
DISK$NDBC:[NDBC]NDBC1982.DAT, DISK$NDBC:[NDBC]NDBC1982.HED
DISK$NDBC:[NDBC]NDBC1983.DAT, DISK$NDBC:[NDBC]NDBC1983.HED
DISK$NDBC:[NDBC]NDBC1984.DAT, DISK$NDBC:[NDBC]NDBC1984.HED
```

NOTE THAT THE NDBC DATA ARE ON A MOUNTABLE DISK PACK.
USE THE MOUNTPACK COMMAND TO MOUNT THE DISK (I.E., MOUNTPACK NDBC).

☆U.S. Government Printing Office: 1987—756-657/40196