WHOI-92-43

C.2

Woods Hole Oceanographic Institution

IMET Shipboard Systems, Operations and Software User Manual

by

Kenneth E. Prada

November 1992

Technical Report

Funding was provided by the National Science Foundation under Grant Nos. OCE-92-04034 and OCE-87-09614.

WHOI-92-43

IMET Shipboard Systems, Operations and Software User Manual

by

Kenneth E. Prada

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

November 1992

Technical Report

Funding was provided by the National Science Foundation under Grant Nos. OCE-92-04034 and OCE-87-09614.

Approved for Distribution:

George V. Frisk

George V. Frisk, Chairman
Department of Applied Ocean Physics and Engineering

Contents

List of Figures

List of Tables

ABSTRACT

This report has two parts. The first is a copy of the *Operations and Software User Manual* prepared for use with the IMET shipboard software distribution. It describes the programs used to acquire and record data from IMET systems installed on *RV Knorr* and *RV Oceanus*. The second part adds appendix material that contains the documentation pages for programs and subroutines used in the IMET shipboard software system. These items are available through network or diskette access. This report has been prepared to give this information broader visibility and circulation.

1 INTRODUCTION

To meet World Ocean Circulation Experiment (WOCE) specified measurement needs for the 1990's, a development project was undertaken to provide new methods for meteorological measurements from ships and buoys. The results are the IMET system. Following successful prototype testing and evaluation, the technology for IMET was transferred to industry for small quantity commercial production.

KNORR and *OCEANUS* have the first permanent shipboard installations of IMET systems. This manual provides operational and software documentation for these installations.

2 HARDWARE INFORMATION

This section contains a brief description of the IMET system hardware. It is provided for general information background and is not a complete hardware reference for any portion of the IMET system. The user is directed to the IMET system technical manuals available from Alden Electronics in support of their IMET products.

2.1 IMET SYSTEM

IMET is a system of autonomous intelligent sensor modules interconnected using the EIA-485 standard. Figure 1 shows a block diagram of a shipboard IMET system. Each module contains a microcontroller programmed to handle the unique needs of the individual sensor. Typically a module samples its sensor as often as practical and, when interrogated, returns data in the form of one-minute averages. These data in are in calibrated scientific units. There are currently seven modules in the shipboard installation:

- wind speed and direction
- barometric pressure
- relative humidity
- air temperature
- .water temperature
- short wave radiation
- precipitation

In later configurations, modules for long wave radiation and ARGOS telemetry may be added.

On both *KNORR* and *OCEANUS*, these modules (except water temperature) are mounted on a mast at the bow. They are wired through a local junction box and 4-wire cable to the logger system. The cable provides power and communications between the data logging system and the modules. The water temperature module, installed in an engine room or bow chamber inlet, uses a separate 4-wire cable.

An additional cable is routed to the bow mast from a power supply if a forward compartment, typically the bosun's locker. This cable supplies low voltage to the precipitation sensor heating coils. The heaters prevent freezing of collected rain during cold weather. They are controlled by a thermostat internal to the sensor module and may be powered at all times.

2.2 DATA LOGGING SYSTEM

The data logging system performs several functions. These include:

- provide controlled power to IMET modules

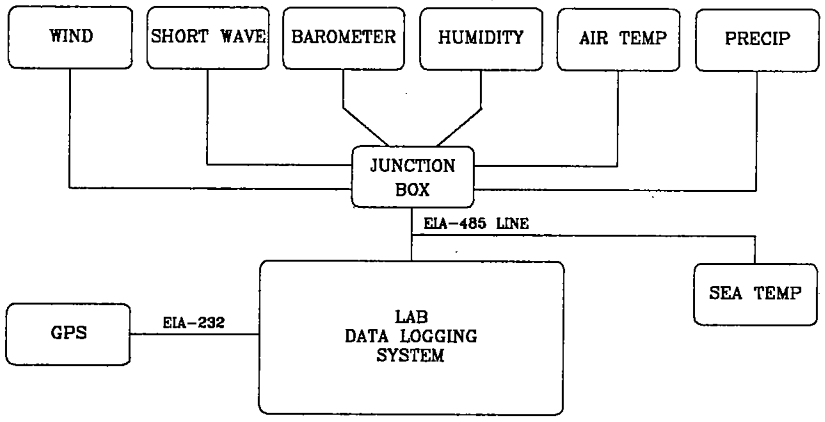


Figure 1: Block Diagram of Shipboard IMET System

- interrogate IMET modules
- format and record IMET data values
- archive data files to optical disk
- provide data to external users via EIA-232
- when available, acquire and record navigation from a local GPS receiver

2.2.1 Hardware Configuration

The data logging system uses a small footprint PC with a 25MHz 386 and numeric coprocessor. Figure 2 shows a block diagram of the logging system. Four megabytes of RAM are installed to support later operating system changes or enhancements.

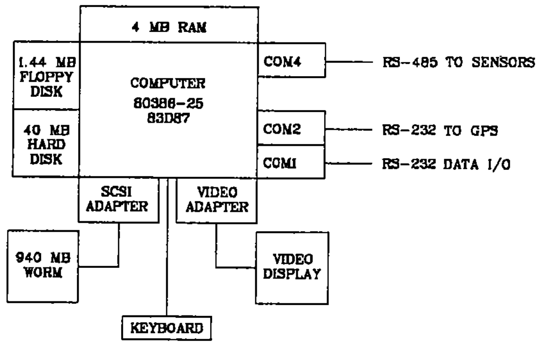


Figure 2: Block Diagram of IMET Data Logger

Standard peripherals include a 1.44 Mb floppy drive, 40 Mb hard drive, monochrome video display, and keyboard. The archive media is a 940 Mb (470 per side) WORM optical disk drive. There are three serial ports:

- COM1 - (EIA-232) is used for data communications to external users.
- COM2 - (EIA-232) provides communication with a GPS receiver (when provided).
- COM4 - (EIA-485) communications with the IMET modules.

Printer ports LPT1 and LPT2 are available but are not used in the current system configuration.

2.2.2 Module Power Supply

Power for IMET module operation is derived from a small laboratory power supply (Lambda LA-200). This supply is located at or near the data logging computer. It is adjusted to provide 12-13 volts DC. Its output is enabled or disabled by the computer. An auxiliary output signal from the COM485 card provides a logic level (low = on) to the rear connector strip on the power supply.

2.2.3 Module - Logger Interconnect

Communications between the data logging system and IMET modules uses the EIA-485 standard. EIA-485 is a two-wire balanced differential method with multipoint capability. It has excellent noise immunity and can be used over long distances at high baud rates (1 km at 100k BAUD).

Four wires interconnect the logger system with IMET modules. Two of the wires are used for the EIA-485 signals. The remaining wires carry power and common.

Aboard *KNORR* the ship's permanent wiring system connects the logger in the science chart room to the bow mast through several junction boxes. The same lines also extend to the sea surface temperature module located at the bow chamber raw water inlet.

Aboard *OCEANUS* two special cables have been installed. The first connects the logger to the bow mast. The second cable connects to the engine room where the sea surface temperature module is located in a raw water inlet line.

3 SOFTWARE

This section describes acquiring, installing, and using the operational software. Programs used for module test, maintenance, and calibration are discussed in the IMET systems Technical Documentation, available from Alden Electronics.

3.1 Acquiring Operational Software and Documentation

The IMET/UNOLS operational software distribution is available in two ways. It can be acquired using anonymous ftp methods over Internet and also is available on diskette to those without network access.

The distribution contains several groups of items:

- program binary executables (.EXE files)
- sources and objects for programs and libraries
- miscellaneous documents in LaTeX or PostScript format, which includes this manual in PostScript printable form.
- UNIX style manual pages for reference to both programs and library functions. Also included are Postscript versions of these manual pages for printer output.

3.1.1 Diskette Distribution

To order the diskette distribution, send a request, with a check for $50 to cover media, duplication, handling and postage, to:

IMET/UNOLS Shipboard Software Distribution
Attention: M. Tavares
Woods Hole Oceanographic Institution
Woods Hole, MA 02543

The request should specify diskette type. Three diskette formats are supported, 5-1/4" high density (1.2Mb) and 3-1/2" high and low density (1.44Mb and 720 kb). These formats are for use in the DOS environment.

Comments, feedback, and requests for other information also should be directed to the above address.

3.1.2 Network Distribution

The network distribution of the IMET/UNOLS operational software can be accessed using anonymous ftp to Internet address **isdl.whoi.edu**. A *tar* file can be accessed (in ftp binary mode) from the file **unols/imetship.tar** in the anonymous ftp directory of **isdl.whoi.edu**.

Three additional files should be transferred to your local directory. These are **read.me**, **tar.exe** and **hd_setup.bat**. The **read.me** file contains preliminary instructions for handling the distribution. **tar.exe** is a program to un-archive the **imetship.tar** file into appropriate directories and files. The file **hd_setup.bat** is a batch file containing procedures for automatically installing the distribution onto a hard disk system.

The command steps used to acquire the software distribution are shown in Table 1. The user responses are shown in *italics*, and the .... sequences indicate ftp activity and status responses.

Comments, feedback, and requests for other information can be sent via e-mail to *kegp@isdl.whoi.edu.*

```
ftp isdl.whoi.edu
connected to isdl.whoi.edu
isdl FTP server ready
Name: anonymous
Password: (enter your network id here)
Guest login OK, access restrictions apply
ftp> cd unols
CWD command successful
ftp> get hd_setup.bat
............................
ftp> binary
Type set to I
ftp> get imetship.tar
............................
ftp> get tar.exe
............................
ftp> bye
```

Table 1: Instructions for FTP Access to IMET/UNOLS Software Distribution

3.2 Installing Operational Software

The operational programs and support files can be automatically installed in the proper directories of the hard disk on the logger system. Installation procedures vary depending on how the software was acquired. These procedures include answering questions that produce a site configuration file. All installation procedures are interactive.

3.2.1 Installing From Floppy Diskette Distribution

1. Insert the diskette labeled *Installation* in the **A:** drive.
2. Type **A: <ENTER>**.
3. At the DOS prompt (A:>) type **FL_SETUP <ENTER>**.
4. A directory will be created (*UNOLS*) on the hard disk and files will be copied to it and its subdirectories.
5. Answer the procedure questions.
6. When procedure is complete, remove diskette.
7. Proceed to the following section on configuration

3.2.2 Installing From Network Distribution

This procedure assumes that installation is being performed on a DOS machine. The distribution can be unpacked on a UNIX machine if necessary but those instructions are not covered here.

There are three files obtained from the network distribution that must be in the root directory of the hard disk prior to starting the installation procedure.

- **imetship.tar**
- **tar.exe**
- **hd_setup.bat**

The following steps will install the software distribution on the hard disk.

1. Transfer to the root directory (**CD **)
2. At the DOS prompt (C:>) type **HD_SETUP <ENTER>**.
3. A directory will be created (*UNOLS*) on the hard disk and the **imetship.tar** file will be unpacked into it and its subdirectories.
4. Answer the procedure questions.
5. Proceed to the following section on configuration

3.3 System Configuration

Several steps must be taken to configure the system for operation. These include:

- installation of the drivers for the optical disk archive media.
- creation or editing of the **CONFIG.SYS** file.
- creation or editing of the **AUTOEXEC.BAT** file.
- creation or editing of the ship name and module configuration file.

3.3.1 Optical Disk Drivers

There should be instructions available with the purchased optical disk subsystem drivers. Follow these instructions to install any drivers or related programs for use of the optical disk drive. These installation procedures might involve formatting and partitioning the optical disk cartridge. This document does not deal with these device-specific issues. They are left to the judgement of the individual user based on local environment and needs. However, the operational software assumes that the archive drive will be available through the drive **E:** designation.

3.3.2 The CONFIG.SYS File

The system needs a **CONFIG.SYS** file. This file is used during the boot procedure. It contains certain commands which configure the DOS system. Refer to the DOS manual for details on the creation and contents of the **CONFIG.SYS** file. The **CONFIG.SYS** file must contain commands to load and configure the drivers needed to operate the optical disk drive. Instructions for these commands should be supplied with the optical disk software drivers. Follow these instructions and those supplied in the DOS documentation to generate a **CONFIG.SYS** file. An example file is shown in Appendix A.

The **CONFIG.SYS** file must contain a line that loads the **ANSI.SYS** file. This is a driver that makes the PC keyboard and video respond to ANSI compatible terminal commands. This driver is installed by inserting the following line in the **CONFIG.SYS** file:

```
DEVICE=C:\ANSI.SYS
```

The driver **ANSI.SYS** is a part of the DOS system file set and should be installed in the root directory.

3.3.3 The AUTOEXEC.BAT File

An **AUTOEXEC.BAT** file is a special batch file that is automatically executed whenever DOS is started. It is useful for executing certain commands before using the system.

If the UNOLS/IMET system is to be configured for turn-key operation, the **AUTOEXEC.BAT** file must have as its last line:

```
C:\SH_IMET
```

This runs the main operational program whenever the system is started or restarted. See the example in Appendix A.

3.3.4 The SHIPNAME File

The main operational program **SH_IMET** needs a file that contains ship name and module configuration information. This file is called **SHIPNAME** and should be located in the root directory. It is generally created during the software installation process.

The file contains two lines. The first line is the ship name. The second line contains the names, in lower case, of all modules present in the current system configuration. Table 2 contains a list of the module names and functions. The following lines are an example of a typical **SHIPNAME** file.

```
R/V Knorr
wnd bpr hrh tmp sst swr prc gps
```

The file contains only ASCII text and can be created and edited with any text editor.

NAME	FUNCTION
wnd	wind speed and direction
bpr	barometric pressure
hrh	relative humidity
tmp	air temperature
sst	sea temperature
swr	short wave radiation
lwr	long wave radiation
prc	precipitation
gps	Global Positioning

Table 2: IMET Module Names and Functions

3.4 Operational Program SH_IMET

SH_IMET is the primary program for data collection and recording. Its basic functions are:

- interrogate modules once each minute
- format and record the returned data values
- display data values, time, date, and messages on the CRT screen
- monitor keyboard input for user commands
- output data values to serial port, periodically or on request
- copy data file to archive optical cartridge once daily
- maintain a log file of program activity and errors
- control module power, cycling power when necessary to reset modules
- initialize, control and interrogate GPS unit, when provided

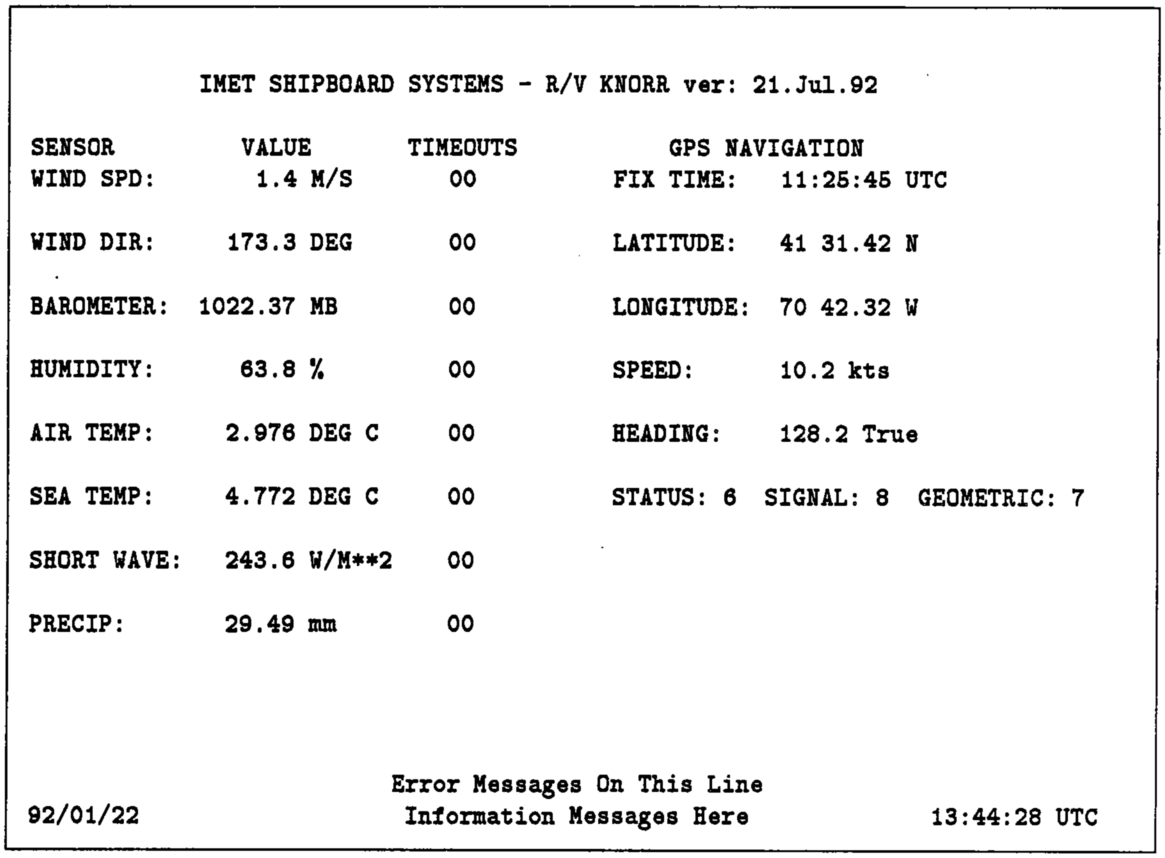
3.4.1 Module Interrogation

The **SH_IMET** program uses the data logging computer's hardware real-time clock for time-keeping. The hardware alarm interrupt feature of the real-time clock is used to provide an interrupt each second. At the start of each minute the program begins an interrogation sequence that includes all modules. If a module does not respond to interrogation within an alloted time, a time-out flag is set for that module (and the system). If more than ten timeouts are encountered for any module, it is removed from the interrogation list. The program monitors these time-out flags. When several have been encountered the module power supply is cycled in an effort to reset the modules and clear any hangups. The entire system is reset at the start of each day.

3.4.2 Data Format and Record

Data values received from the sensor modules are formatted into the netCDF interface specifications (see Appendix C). A record is written to a hard disk file each minute. Each file contains values for one full day. At the start of each day, the file from the prior day is copied to the archive media (optical disk) and the original hard disk file deleted.

File name conventions in the IMET system use chronology to generate the name. A file name consists of the year, month, and day. The name *911224.MET* is for a file containing data from 24 December 1991.



```
          IMET SHIPBOARD SYSTEMS - R/V KNORR ver: 21.Jul.92

SENSOR         VALUE       TIMEOUTS           GPS NAVIGATION
WIND SPD:        1.4 M/S      00         FIX TIME:    11:25:45 UTC

WIND DIR:      173.3 DEG      00         LATITUDE:    41 31.42 N

BAROMETER:   1022.37 MB       00         LONGITUDE:   70 42.32 W

HUMIDITY:       63.8 %        00         SPEED:       10.2 kts

AIR TEMP:      2.976 DEG C    00         HEADING:     128.2 True

SEA TEMP:      4.772 DEG C    00         STATUS: 6  SIGNAL: 8  GEOMETRIC: 7

SHORT WAVE:    243.6 W/M**2   00

PRECIP:        29.49 mm       00



                            Error Messages On This Line
92/01/22                    Information Messages Here             13:44:28 UTC
```

Figure 3: Example of SH_IMET Screen Display

3.4.3 Local Screen Display

The local video screen is used to display recent measurement values, time, date, information and error messages from the program. An example screen display is show in Figure 3.

Time, date and information messages are displayed on the bottom line of the screen. Time is in the lower right corner and updates each second. Date is in the lower left corner. The center of the bottom line shows information messages from the **SH_IMET** program. Error messages are displayed on the next-to-last line.

The top line of the display contains the program title and version. The remainder of the display contains sensor variable titles, values, and units. Values from IMET modules are in the left columns. GPS information is in the right columns (blank if GPS is not provided).

3.4.4 Keyboard Commands

There are several keyboard commands that **SH_IMET** recognizes. Table 3 lists these commands. All commands are formed by depressing two keys. The *ALT* key is always depressed as part of the command combination.

3.4.5 Serial Data Output

The COM1 port is used for data communications. It operates in two modes, periodic output or output on command receipt. The default mode at program initiation is periodic output at one-minute intervals. The

KEYS	ACTION
ALT-B	begin periodic serial data output activity (once each minute)
ALT-C	cycle power to sensor modules and re-enable all modules
ALT-E	end periodic serial data output activity
ALT-P	cycle power to sensor modules
ALT-S	execute a module sample cycle w/o recording
ALT-X	end program activity and return to interactive DOS prompt

Table 3: SH_IMET Command Key Sequences

NAME	VALUE	DESCRIPTION
DAY:	91/12/12	*date*
TIM:	12:23:00	*time*
WNS:	1.2	*wind speed in meters/second*
WND:	210.0	*wind direction in degrees*
BPR:	1009.0	*barometric pressure in millibars*
HUM:	43.8	*relative humidty in percent*
AIR:	22.4	*air temperature in degrees C*
SEA:	15.2	*sea surface temperature in degrees C*
SWR:	756.9	*short wave radiation in watts per square meter*
PRC:	10.34	*precipitation in millimeters*
FXT:	15:45:32	*fix time*
LAT:	41 31.47 N	*latitude in degrees and minutes*
LON:	70 40.30 W	*longitude in degrees and minutes*
HDG:	126.8	*heading in degrees true*
SPD:	8.44	*speed in knots*
FXQ:	6 9 8	*fix quality indicators*

Table 4: Serial Data Output Format

mode may be changed using keyboard commands (see Table 3) or by sending a command to this port. The recognized commands are:

- #IMETB - transmit periodically at one minute intervals
- #IMETE - stop periodic one minute transmissions
- #IMETS - transmit single data group and disable periodic output

The output from this port, in either mode, consists of a series of ASCII lines containing the measurement name and its value. Each line is terminated with a carraige-return line-feed pair and the total transmission is terminated with an ETX (03) character. Table 4 shows an example of a typical output group (with GPS). The variable name and value information are part of the output stream. The description column, added here (in *Italics*) for information only, does not appear in the actual output. A value may be replaced with "n/a" to indicate that the data in question was not available.

3.4.6 Global Positioning System

The *KNORR* IMET system includes a Global Positioning System (GPS) capability. This is implemented using a Magellan OEM GPS unit. Interface to the GPS unit uses the COM2 serial port.

Once initialized and collecting fixes, the program collects several fixes at 2-second intervals each minute at mid-minute. These fixes are averaged to reduce spurious noise and are recorded with the IMET measurements collected during the subsequent sampling period.

The information collected, displayed and recorded includes fix time, latitude, longitude, true heading and speed in knots. These values are shown in the right portion of the display screen.

3.5 Program OFFLOAD

A program is provided for data offload onto floppy diskette. When necessary, preferrably during port stays, data files from a prior cruise leg can be transferred to floppy diskette.

The **SH_IMET** program must be stopped (use ALT-X). Run program **OFFLOAD**.

The **OFFLOAD** program is interactive. It asks the user if diskettes are to be formatted prior to file transfer. It also requests the start and end dates for transfer. Following this interaction, the user need only feed the system diskettes upon request.

OFFLOAD calls the DOS format program to format the diskettes. It is called once for each diskette in the offloading sequence. However, the format program will ask the user if another diskette is to be formatted. The user should respond *no* to this request.

If the user supplies pre-formatted diskettes, they should be empty of files. The program determines the amount of space on the target floppy diskette and copies files until the diskette is full. Files are copied that satisfiy the start and end dates supplied by the user.

A typical file is approximately 120 kilobytes long without GPS and 150 kilobytes with GPS.

3.6 Program IMETLINK

The **IMETLINK** program provides th ability to communicate with a single sensor module using a variety of module commands. The main data logger program (**SH_IMET**) must be stopped in order to run **IMETLINK.**

This program is interactive and requests the address of the module to be used. The user should respond with a 5-character module address, for example, *SWR01* for the short wave module (Table 5 shows primary module addresses). The program then requests a command letter. Use the command *H* (help) to get the command set for the desired module.

When a command letter is entered, the program combines this with the module address and transmits the result. The lower portion of the screen is cleared and the program waits for a response. If a response is received, it is printed on the screen. If no response is received within 5 seconds, the program prints a timeout message at the bottom of the screen and requests a command letter again.

The program stays in a loop requesting a command letter until *ESC* is entered. This returns the program to the startup request for a module address. At this point a different address can be entered or the program terminated by typing *Q ENTER.*

4 DATA AND FORMATS

Both raw and calibrated values are acquired from IMET modules. Some raw values are recorded to allow later corrections if errors are indicated. Only calibrated values are displayed for local use. Table 5 lists the pertinent measurement values acquired from IMET modules whose address is shown in the first column.

4.1 netCDF

All data are stored in netCDF format. The netCDF data access interface [1] is a standard developed by the Unidata Program Center. It is supported at UCAR by the NSF Division of Atmospheric Sciences. The Unidata Program Center distributes the software and encourages its use for many scientific applications.

MODULE	MEASUREMENT	UNITS	RESOLUTION	RANGE
WND01	east wind velocity	meters/second	0.1	-50.0 to +50.0
WND01	north wind velocity	meters/second	0.1	-50.0 to +50.0
WND01	wind speed	meters/second	0.1	0.0 to +50.0
WND01	wind direction	degrees	0.1	0.0 to 359.0
BPR01	barometric pressure	millibars	0.1	980.0 to 1180.0
HRH01	relative humidity	percent	0.1	0.0 to 100.0
TMP01	air temperature	degrees C	0.001	-30 to + 40
SST01	sea temperature	degrees C	0.001	-5 to + 40
SWR01	short wave radiation	watts per square meter	0.1	0 to 1400
PRC01	precipitation	millimeters	0.1	0 to 50

Table 5: IMET Measurement Parameters

The netCDF format was chosen to support IMET systems because it is a national standard, fully supported, platform and language portable, versatile, and well documented.

This manual does not present a tutorial on the netCDF format. The reader is directed to reference [1] for detailed information concerning the netCDF data access interface.

Appendix C contains a definition of the netCDF files used for shipboard data storage. It is presented here as a convenience to users who need details of the data file content.

References

[1] Rew, Russell K., *netCDF User's Guide, An Interface for Data Access*, Version 1.11, March 1991, Unidata Program Center, 150 pp.

A DOS Directories and Files

A.1 Directory Layout

An example directory structure for the hard disk is:

```
C:\-----DOS
     |
     ---BIN
     |
     ---DEV
     |
     ---ETC
     |
     ---PCPLUS
     |
     ---TC----------INCLUDE------SYS
                    |
                    --LIB
                    |
                    --BGI
                    |
                    --BIN
```

Directory contents are defined in the following table:

DIRECTORY	CONTENTS
DOS	Files provided with Operating System (DOS)
BIN	General purpose executable files (EXE and COM)
DEV	Device drivers, e.g., optical disk handlers
ETC	Miscellaneous batch files
PCPLUS	PROCOMM communications software
TC	Borland Turbo C support software

A.2 Supplied Files

There are several files provided for use with the shipboard system. These typically reside in the C:\BIN directory with the exception of *SHIPNAME*, which must reside in the root directory.

485ON.EXE	Enable LA-200 Power Supply
485OFF.EXE	Disable LA-200 Power Supply
OFFLOAD.EXE	Offload IMET Files from Optical to Floppy
SH_IMET.EXE	Shipboard IMET Data Logger Program
CP.EXE	Copy program for Hard Disk to Optical Transfer
IMETLINK.EXE	A single module communications program
SHIPNAME	Ship Name and module configuration list

A.3 CONFIG.SYS and AUTOEXEC.BAT Files

Program execution occurs in the root directory (C:\) and is set in the startup AUTOEXEC.BAT batch file. The following is an example of the *CONFIG.SYS*. This may vary dependent on the drivers for the individual implemetation.

```
SHELL=C:\COMMAND.COM /E:512 /P
FILES=20
BUFFERS=20
BREAK=ON
DEVICE = C:\DEV\ANSI.SYS
DEVICE = C:\DEV\EXPRESS.SYS /HAHA:I330:D5
DEVICE = C:\DEV\XWORM3.SYS /TOLOV1 /F512 /B4 /I3
```

The *AUTOEXEC.BAT* file might contain:

```
ECHO OFF
PATH=C:\DOS;C:\BIN;C:\TC\BIN;C:\ETC;C:\
PROMPT $p$g
485ON
C:\SH_IMET
```

B Notes and Bugs

This appendix contains notes and information collected during test and initial use of the shipboard IMET system.

B.1 System Re-Start

The simplest and most reliable way to restart the system is by depressing the reset button on the computer front panel. This will re-boot the computer, starting a new execution of the **SH_IMET** program and cycling power to the IMET modules. The program re-opens the previously created data file for use.

If the system appears hung

- time on the screen display is not advancing
- values shown on the screen seem stuck
- values shown on the screen are unreal or bizarre
- screen is blank or does not contain proper display

Press the **RESET** button.

B.2 Checking and Setting Time

Time accuracy depends on the computer hardware real-time clock. This may operate to within several minutes of accuracy per year. It is necessary to check the accuracy periodically (once per month) and reset the clock if necessary.

The clock can be reset using the standard DOS **DATE** and **TIME** commands. In this version of DOS (3.30A) these commands also set the real-time clock values. **Remember to use UTC time (also known as GMT or ZULU).**

B.3 GPS Power Cycling

There is a problem with GPS use when AC power is cycled for whatever reason. This problem involves the use of early single channel Magellan OEM GPS units. When power is applied or cycled to the GPS unit and the enable toggle switch is in the **ON** position, the GPS unit will not function properly. If the program indicates improper operation of the GPS unit, cycle the switch on the GPS unit's chassis.

B.4 OFFLOAD Use

The **OFFLOAD** program is not intended for frequent use. Rather, it should be used between cruise legs, particlarly during port stops. This eliminates gaps in the data files caused when the data logging function is stopped to run the offload function. Although gaps are permissible and file continuity is maintained by the **SH_IMET** program, the user may forget to restart the program after offloading data files and valuable measurements could be lost.

B.5 Saving The Monitor

The video monitor will eventually have the phosphor permanently burned with the text image of displayed information. This will effect proper viewing to some degree. To slow the degradation process the brightness and/or contrast should be kept at low levels whenever possible.

C netCDF Data Variables and Attributes

The following is a definition for the shipboard IMET data set in netCDF Description Language (CDL).

```
netcdf ship {  // example netCDF specification for shipboard IMET

dimensions:
    record = unlimited;

variables:
    double  time(record);
        time:units      = "seconds of year";
        time:long_name  = "time";
    float   wnde(record);
        wnde:units      = ";m/s";
        wnde:long_name  = "wind east";
    float   wndn(record);
        wndn:units      = ";m/s";
        wndn:long_name  = "wind north";
    float   wnds(record);
        wnds:units      = ";m/s";
        wnds:long_name  = "wind speed";
    float   wndd(record);
        wndd:units      = "degrees";
        wndd:long_name  = "wind direction";
    float   wndc(record);
        wndc:units      = "degrees";
        wndc:long_name  = "wind compass";
    float   wndv(record);
        wndv:units      = "degrees";
        wndv:long_name  = "wind vane";
    float   bpr(record);
        bpr:units       = "mb";
        bpr:long_name   = "barometric pressure";
    float   hrh(record;
        hrh:units       = "percent";
        hrh:long_name   = "relative humidity";
    float   htmp(record);
        htmp:units      = "degC";
        htmp:long_name  = "air temperature";
        htmp:type       = "humidity";
    float   atmp(record);
        atmp:units      = "degC";
        atmp:long_name  = "air temperature";
    float   tapr(record);
        tapr:units      = "ohms";
        tapr:long_name  = "air prt resistance";
    float   tar2(record);
        tar2:units      = "ohms";
        tar2:long_name  = "air ref resistance 2";
    float   tar3(record);
        tar3:units      = "ohms";
        tar3:long_name  = "air ref resistance 3";
    float   stmp(record);
        stmp:units      = "degC";
        stmp:long_name  = "sea temperature";
    float   tspr(record);
        tspr:units      = "ohms";
        tspr:long_name  = "sea prt resistance";
    float   tsr2(record);
        tsr2:units      = "ohms";
        tsr2:long_name  = "sea ref resistance 2";
    float   tsr3(record);
        tsr3:units      = "ohms";
        tsr3:long_name  = "sea ref resistance 3";
    float   srad(record);
        srad:units      = "watts per square meter";
        srad:long_name  = "short wave radiation";
    float   prc(record);
        prc:units       = "mm";
        prc:long_name   = "precipitation";

// if GPS available

    double fixt(record);
        fixt:units      = "seconds of year";
        fixt:long_name  = "fix time";
    float  lat(record);
        lat:units       = "degrees";
        lat:long_name   = "latitude";
    float  lon(record);
        lon:units       = "degrees";
        lon:long_name   = "longitude";
    float  head(record);
        head:units      = "degrees true";
        head:long_name  = "heading";
    float  speed(record);
        speed:units     = "km/hr";
        speed:long_name = "speed";
    char   fixst(record);
        fixst:long_name = "fix status";
    char   fixsq(record);
        fixsq:long_name = "signal quality";
    char   fixgq(record);
        fixgq:long_name = "geometric quality";
    }

// global attributes

    :title           = "IMET Data from KNORR";
    :platform        = " R/V KNORR";
    :instrument type = "IMET system";
    :history         = " ";
```

D Command and Subroutine Documentation Pages

This appendix contains UNIX style documentation (manual) pages for IMET shipboard programs and subroutines. Programs are intended to be invoked directly by the user. Subroutines are functions intended to be called by a user's program.

Each of the two sections begins with an introduction page **INTRO()**. This page contains general information concerning the section and an ordered list of the section contents. The sections consist of a number of independent entries of a page or so each. The name of the entry is in the upper corners of its pages together with the section type. Entries within each section are alphabetized. The revision date of the entry is generally at the bottom of each page.

All entries are based on a common format of subsections, not all of which will always appear. These subsections are:

- **Name** lists the exact names of the commands or subroutines covered under the entry and gives a very short description of their purpose.
- **Syntax** summarizes the use of the program or subroutine being described. A few conventions are used:
  1. Boldface words are considered literals and are typed just as they appear.
  2. Square brackets [ ] around an argument indicate that the argument is optional
  3. When an argument is given as 'name', it usually refers to a file name.
- **Description** discusses in detail the subject item and its use.
- **Return Value** defines values returned from subroutines, where applicable.
- **Files** gives the names of files which are built into or specifically used by the function.
- **See Also** give pointers to related information.
- **Diagnostics** discusses diagnostic indications which may be produced by the command or subroutine. Self-explanatory messages are not listed.
- **Notes** provides general information and may include known bugs and deficiencies. Occasionally a suggested fix may be described.

NAME

intro – introduction to UNOLS/IMET commands

DESCRIPTION

This section describes user commands for the UNOLS/IMET shipboard software. These commands, listed below in alphabetical order, are described in the following pages.

LIST OF COMMANDS

Name	**Appears on Page**	**Description**
485_off	**485_off**(1)	disable IMET module power supply
485_on	**485_on**(1)	enable IMET module power supply
imetlink	**imetlink**(1)	communication link with IMET module
offload	**offload**(1)	offload IMET files from archive to floppy
sh_imet	**sh_imet**(1)	shipboard IMET acquisition program

NAME

485_off – disable IMET module power supply

SYNOPSIS

485_off

DESCRIPTION

485_off disables the power unit supplying the IMET module network. This is typically a Lambda Model LA-200 power supply controlled through a back-panel input. The control line is derived from the EIA-485 control card in the PC. The control line is set low to enable power supply output or high (+5v) to disable output.

SEE ALSO

485_on(1)

COPYRIGHT

AUTHORS

K. Prada, Department of Applied Ocean Physics and Engineering

NAME

485_on – enable IMET module power supply

SYNOPSIS

485_on

DESCRIPTION

485_on enables the power unit supplying the IMET module network. This is typically a Lambda Model LA-200 power supply controlled through a back-panel input. The control line is derived from the EIA-485 control card in the PC. The control line is set low to enable power supply output or high (+5v) to disable output.

SEE ALSO

485_off(1)

COPYRIGHT

AUTHORS

K Prada, Department of Applied Ocean Physics and Engineering

NAME

imetlink – communicate with individual IMET modules

SYNOPSIS

imetlink

DESCRIPTION

imetlink() is used to communicate with a single IMET module. The program is interactive and requests the address of the module to be used. The user should respond with a 5-character module address, for example, *SWR01* for the short wave module. The program then requests a command letter. Use the module command *H* to get the command set for the desired module.

When a command letter is entered, the program combines this with the module address and transmits the result. The lower portion of the screen is cleared and the program waits for a response. If a response is received, it is printed on the screen. If no response is received within 5 seconds, the program prints a timeout message at the bottom of the screen and requests a command letter again.

The program stays in a loop requesting a command letter until **ESC** is entered. This returns the program to the startup request for a module address. At this point a different address can be entered or the program terminated by typing **Q ENTER**.

SEE ALSO

Operations and Software User Manual.

COPYRIGHT

AUTHORS

K. Prada, Department of Applied Ocean Physics and Engineering

NAME

offload – transfer files from archive media to floppy diskette

SYNOPSIS

offload

DESCRIPTION

offload transfers data files from the archive media (optical disk) to floppy diskettes. The program is interactive. It asks the user if diskettes are to be formatted prior to file transfer. It also requests the numeric values for start and end transfer dates. Following this interaction, the user need only feed the system diskettes upon request.

If requested, **offload**

calls the DOS format program to format each diskette in the offloading sequence. However, as part of each call the format program will ask the user if another diskette is to be formatted. The user should respond *no* to this request.

If the user supplies pre-formatted diskettes, they should be empty of files. The program determines the amount of space on the target floppy diskette and copies files until the diskette is full. Files are copied that satisfy the start and end dates supplied by the user.

SEE ALSO

sh_imet(1), *Operations and Software User Manual.*

COPYRIGHT

Copyright (c) 1992, Woods Hole Oceanographic Institution

AUTHORS

K. Prada, Department of Applied Ocean Physics and Engineering

NAME

sh_imet – primary IMET shipboard data logging program

SYNOPSIS

sh_imet

DESCRIPTION

sh_imet is the primary program for data collection and recording in the IMET shipboard system. Its basic functions are:

- interrogate modules once each minute
- format and record the returned data values
- display data values, time, date, and messages on the CRT screen
- monitor keyboard input for user commands
- output data values to serial port, periodically or on request
- copy data file to archive optical cartridge once daily
- maintain a log file of program activity and errors
- control module power, cycling power when necessary to reset modules
- initialize, control and interrogate GPS unit, when provided

Data values received from IMET sensor modules are formatted into netCDF. A record is written to a hard disk file each minute. Each file contains values for one full day. At the start of each day, the file from the prior day is copied to archive media (optical disk) and the original hard disk file deleted.

Data file name conventions use chronology to generate the name. A file name consists of the year, month, and day. The name *911224.MET* is for a file containing data from 24 December 1991.

Data values from the one minute interrogations are displayed on the screen in the format shown below.

```
                 IMET SHIPBOARD SYSTEMS - R/V OCEANUS ver: 21.Jul.92

   SENSOR         VALUE          TIMEOUTS   GPS NAVIGATION
   WIND SPD:      1.4 M/S           00      FIX TIME:        11:25:45 UTC
   WIND DIR:      173.3 DEG                 LATITUDE:        41 31.42 N
   BAROMETER:     1022.37 MB        00      LONGITUDE:       70 42.32 W
   HUMIDITY:      63.8 %            00      SPEED:           10.2 kts
   AIR TEMP:      2.976 DEG C       00      HEADING:         128.2 True
   SEA TEMP:      4.772 DEG C       00      STATUS:           6 8 7
   SHORT WAVE:    243.6 W/M**2      00
   PRECIP:        29.49 mm          00


                              Error Messages On This Line
          92/01/22            Information Messages Here        13:44:28 UTC
```

The program currently recognizes several keyboard sequences during operation. These are:

KEYS	ACTION
ALT-B	begin periodic serial data output activity (once each minute)
ALT-C	cycle power to sensor modules and re-enable all modules
ALT-E	end periodic serial data output activity
ALT-P	cycle power to sensor modules
ALT-S	execute a module sample cycle w/o recording
ALT-X	end program activity and return to interactive DOS prompt

Data values are transmitted using the EIA-232 port COM1 once each minute. This feature can be controlled from the keyboard (see table above) or from the EIA-232 line (see *Operations and Software User Manual*). The table below shows an example of a typical output group (with GPS). The variable name and value information is part of the output stream. The description column, added here for information only, does not appear in the actual output.

NAME	VALUE	DESCRIPTION
DAY:	91/12/12	*date*
TIM:	12:23:00	*time*
WNS:	1.2	*wind speed in meters/second*
WND:	210.0	*wind direction in degrees*
BPR:	1009.0	*barometric pressure in millibars*
HUM:	43.8	*relative humidty in percent*
AIR:	22.4	*air temperature in degrees C*
SEA:	15.2	*sea surface temperature in degrees C*
SWR:	756.9	*short wave radiation in watts per square meter*
PRC:	10.34	*precipitation in millimeters*
FXT:	15:45:32	*fix time*
LAT:	41 31.47 N	*latitude in degrees and minutes*
LON:	70 40.30 W	*longitude in degrees and minutes*
HDG:	126.8	*heading in degrees true*
SPD:	8.44	*speed in knots*
FXQ:	6 9 8	*fix quality indicators*

SEE ALSO

offload(1), senddata(3), showdata(3), hotkey(3), *Operations and Software User Manual.*

COPYRIGHT

AUTHORS

K. Prada, Department of Applied Ocean Physics and Engineering

NAME

intro – introduction to user-level library functions for UNOLS/IMET

DESCRIPTION

This section describes user-level library routines for the UNOLS/IMET shipboard software. These functions are listed below in alphabetical order. This list does not contain all functions used in the system. Instead it deals with those functions of most importance to system operations.

LIST OF LIBRARY FUNCTIONS

Name	Appears on Page	Description
cdftime	**cdftime**(3)	convert standard time to seconds of year
com_1	**com_1**(3)	COM1 serial input/output functions
com_2	**com_2**(3)	COM2 serial input/output functions
com_85	**com_85**(3)	COM85 serial input/output functions
dayear	**dayear**(3)	calculate sequential day of the year
diskdata	**diskdata**(3)	control disk writes and archive transfers
getdtime	**getdtime**(3)	get date and time from DOS system clock
getrtime	**getrtime**(3)	get date and time from hardware real-time clock
get_imet	**get_imet**(3)	interrogate IMET modules
getship	**getship**(3)	read shipname file for name and module list
hotkey	**hotkey**(3)	test for and return 2 character hot key sequence
imet_err	**imet_err**(3)	create error message into buffer
imtime	**longtime**(3)	convert long integer date/time to individual values
info_mess	**info_mess**(3)	display information message on line 24
pemess	**pemess**(3)	display error message on line 23
senddata	**senddata**(3)	transmit data values via EIA-232 port
setdtime	**setdtime**(3)	set date and time in DOS system clock
setrtime	**setrtime**(3)	set date and time in hardware real-time clock
sh_cdf	**sh_cdf**(3)	netCDF interface functions
sh_con	**sh_con**(3)	console I/O functions
sh_log	**sh_log**(3)	logbook file entry functions
showdata	**showdata**(3)	data display functions
timeclr	**dostime**(3)	disable real-time alarm interrupt
timeinit	**dostime**(3)	initialize real-time alarm interrupt
totime	**longtime**(3)	convert time to UNIX long integer format

NAME

cdftime – convert standard time to time in seconds of the year for netCDF use.

SYNOPSIS

```
double cdftime(day, hour, minute, second)
int day, hour, minute;
float second;
```

DESCRIPTION

cdftime() converts standard time, as given in the calling arguments, to seconds of the year for use in netCDF storage. Argument *day* is the sequential day of the year (1-366).

RETURN VALUES

cdftime() returns a double value that represents the second of the year (31622400.0 for 366 days). This result can be passed to the netCDF functions for value storage.

SEE ALSO

dayear(3)

BUGS

This function does not currently detect or report any errors in passed arguments, e.g., *day* greater than 366.

NAME

rts1off, rts1on, com1brk, com1get, com1init, com1int, com1off, com1put, com1rdy, com1str

SYNOPSIS

```
void rts1off(void)

void rts1on(void)

void com1brk(void)

int com1get(void)

void com1init(baud)
  int baud;

void com1int(str,cnt,end)
  char *str;
  int cnt,end;

void com1off(void)

void com1put(ch)
  int ch;

int com1rdy(void)

void com1str(str,pace)
  char str;
  int pace;
```

DESCRIPTION

These are control and input/output functions for the COM1 interface.

rts1off() disables the RTS bit output from the UART.

rts1on() enables the RTS bit output from the UART.

com1brk() generates a 350 millisecond break on the serial output.

com1get() gets a character from the UART. If no character is ready, the function waits until a character has been received.

com1init() initializes the COM1 EIA-232 serial port. The *baud* argument may be any of the standard baud rates between 300 and 19200. Other UART parameters are fixed at no parity, 8 data bits, 1 stop bit.

com1int() initializes the serial functions for input of a string under interrupt control. The argument *str* is a pointer to a character array. The argument *cnt* specifies the maximum number of characters to be input and *end* specifies a specific character that will end the input sequence (e.g. carraige return). The function uses global variable *com1cnt* as a character counter. It is set to *cnt* when the function is initialized and decrements to zero as characters are received. When *com1cnt* reaches zero, the desired count or end character has been reached.

com1off() disables the serial input interrupt functions.

com1put() outputs a single character to the UART.

com1rdy() tests for an input character ready from the UART. A zero is returned if no character is ready. Otherwise a 1 is returned.

com1str() outputs a string to the UART. Argument *str* is a pointer to a null terminated string. The *pace* argument specifies a delay in milliseconds.

SEE ALSO

com_2(3), com_85(3)

NAME

rts2off, rts2on, com2brk, com2get, com2init, com2int, com2off, com2put, com2rdy, com2str

SYNOPSIS

```
void rts2off(void)

void rts2on(void)

void com2brk(void)

int com2get(void)

void com2init(baud)
  int baud;

void com2int(str,cnt,end)
  char *str;
  int cnt,end;

void com2off(void)

void com2put(ch)
  int ch;

int com2rdy(void)

void com2str(str,pace)
  char str;
  int pace;
```

DESCRIPTION

These are control and input/output functions for the COM2 interface.

rts2off() disables the RTS bit output from the UART.

rts2on() enables the RTS bit output from the UART.

com2brk() generates a 350 millisecond break on the serial output.

com2get() gets a character from the UART. If no character is ready, the function waits until a character has been received.

com2init() initializes the COM2 EIA-232 serial port. The *baud* argument may be any of the standard baud rates between 300 and 19200. Other UART parameters are fixed at no parity, 8 data bits, 1 stop bit.

com2int() initializes the serial functions for input of a string under interrupt control. The argument *str* is a pointer to a character array. The argument *cnt* specifies the maximum number of characters to be input and *end* specifies a specific character that will end the input sequence (e.g. carraige return). The function uses global variable *com2cnt* as a character counter. It is set to *cnt* when the function is initialized and decrements to zero as characters are received. When *com2cnt* reaches zero, the desired count or end character has been reached.

com2off() disables the serial input interrupt functions.

com2put() outputs a single character to the UART.

com2rdy() tests for an input character ready from the UART. A zero is returned if no character is ready. Otherwise a 1 is returned.

com2str() outputs a string to the UART. Argument *str* is a pointer to a null terminated string. The *pace* argument specifies a delay in milliseconds.

SEE ALSO

com_1(3), com_2(3)

NAME

rts85off, rts85on, com85brk, com85get, com85init, com85int, com85off, com85put, com85rdy, com85str

SYNOPSIS

void rts85off(void)

void rts85on(void)

void com85brk(void)

int com85get(void)

```
void com85init(baud)
  int baud;

void com85int(str,cnt,end)
  char *str;
  int cnt,end;
```

void com85off(void)

```
void com85put(ch)
  int ch;
```

int com85rdy(void)

```
void com85str(str,pace)
  char str;
  int pace;

void power485(mode)
int mode;
```

DESCRIPTION

These are control and input/output functions for the EIA-485 interface.

rts85off() disables the RTS bit output from the UART.

rts85on() enables the RTS bit output from the UART.

com85brk() generates a 350 millisecond break on the serial output.

com85get() gets a character from the UART. If no character is ready, the function waits until a character has been received.

com85init() initializes the EIA-485 serial port. The *baud* argument may be any of the standard baud rates between 300 and 19200. Other UART parameters are fixed at no parity, 8 data bits, 1 stop bit.

com85int() initializes the serial functions for input of a string under interrupt control. The argument *str* is a pointer to a character array. The argument *cnt* specifies the maximum number of characters to be input and *end* specifies a specific character that will end the input sequence (e.g. carraige return). The function uses global variable *com85cnt* as a character counter. It is set to *cnt* when the function is initialized and decrements to zero as characters are received. When *com85cnt* reaches zero, the desired count or end character has been reached.

com85off() disables the serial input interrupt functions.

com85put() outputs a single character to the UART.

com85rdy() tests for an input character ready from the UART. A zero is returned if no character is ready. Otherwise a 1 is returned.

com85str() outputs a string to the UART. Argument *str* is a pointer to a null terminated string. The *pace* argument specifies a delay in milliseconds.

power485() enables or disables the power supply for the IMET net. Argument *mode* determines which function, 0 for disable, 1 for enable. The function sets or clears the *OUT1* pin on the UART.

SEE ALSO

com_1(3), com_2(3)

NAME

dayear – calculate the sequential day of the year.

SYNOPSIS

int dayear(year, month, day)
int year, month, day;

DESCRIPTION

dayear() calculates the sequential day of the year (1-366) from the supplied arguments *year, month,* and *day*. Argument *year* may be specified as either two or four digits. If only two digits are supplied, the function determines the century based on whether the year is greater than or less than 50.

RETURN VALUES

dayear() returns an integer value between 1 and 366 or -1 if given arguments are not valid.

SEE ALSO

cdf_time(3)

BUGS

This function only works correctly for years between 1950 and 2049 when the year is specified with two digits.

NAME

diskdata – control disk write operations and archive transfers

SYNOPSIS

int diskdata(void)

DESCRIPTION

diskdata() is called periodically (once per minute) to control transfer of data records to disk. The function calls **putcdf()** to perform the actual disk write operation.

Once per day at 00:00 this function closes the daily file and and starts a new one. The prior file is then copied to the archive media, typically optical disk at drive E:. If the transfer to archive media is successful, the prior file is deleted from hard disk.

Once per month, the collected log file for the prior month is transferred to archive media.

RETURN VALUES

diskdata() returns zero if successful or -1 if any errors are encountered.

SEE ALSO

setcdf(3), endcdf(3), putcdf(3)

NAME

getwind, getbpr, gethrh, gettmp, getsst, getswr, getprc

SYNOPSIS

```
#include "externs.h"

int getwind(module)
int module;

int getbpr(module)
int module;

int gethrh(module)
int module;

int gettmp(module)
int module;

int getsst(module)
int module;

int getswr(module)
int module;

int getprc(module)
int module;
```

DESCRIPTION

These functions perform the interrogation of IMET modules. The module address is specified with argument *module*, a number that is used to generate an address string for module interrogation (e.g., WND01). The table below relates function to module type.

FUNCTION	Module Type
getwind()	wind
getbpr()	barometric pressure
gethrh()	relative humidity
gettmp()	air temperature
getsst()	sea temperature
getswr()	short wave radiation
getprc()	precipitation

Each function generates an address string and transmits this string to the EIA-485 module network. The function then waits for a response for approximately two seconds. If there is no response, a timeout variable for each function is incrementd. If more than ten timeouts are encountered, the module is removed from the active list. A proper response is scanned for expected values.

RETURNS

These functions return zero if successful or 1 if no response from the module.

NOTES

Power cycling, either user requested, automatic based on timeouts, or the daily power cycle event, restores all modules to the active list.

NAME

getdtime – get date and time from DOS system software clock

SYNOPSIS

void getdtime(dtm)
int *dtm;

DESCRIPTION

getdtime() acquires the date and time from the DOS system software clock and returns the values to a user integer array. The array is of length 6 and contains:

dtm[0] = year(00-99)
dtm[1] = month(1-12)
dtm[2] = day(1-31)
dtm[3] = hour(0-23)
dtm[4] = minute(0-59)
dtm[5] = second(0-59)

RETURN VALUES

none

NOTES

This function should not be confused with function **getrtime.** This function derives the time from a DOS system software clock while **getrtime** derives the time from a hardware clock chip.

SEE ALSO

setdtime(3), getrtime(3), setrtime(3)

NAME

getrtime – get date and time from real-time clock chip

SYNOPSIS

void getrtime(dtm)
int *dtm;

DESCRIPTION

getrtime() acquires the date and time from the AT system hardware real-time clock chip and returns the values to a user integer array. The array is of length 6 and contains:

```
dtm[0] = year(00-99)
dtm[1] = month(1-12)
dtm[2] = day(1-31)
dtm[3] = hour(0-23)
dtm[4] = minute(0-59)
dtm[5] = second(0-59)
```

RETURN VALUES

none

SEE ALSO

setrtime(3), getdtime(3), setdtime(3)

NAME

getship – get ship name and sensor module list

SYNOPSIS

int getship(void)

DESCRIPTION

getship() opens the *SHIPNAME* file and reads two lines. The first line contains the name of the vessel. The second line contains a list of active module names, in lower case, for use by the *SH_IMET* program. The following table relates lower case names to module.

NAME	FUNCTION
wnd	wind speed and direction
bpr	barometric pressure
hrh	relative humidity
tmp	air temperature
sst	sea temperature
swr	short wave radiation
lwr	long wave radiation
prc	precipitation
gps	Global Positioning

A typical file might contain:

```
R/V KNORR
wnd bpr hrh tmp sst swr prc gps
```

RETURNS

The function does not currently return any meaningful value.

BUGS

There is no error indicator returned. This should be added in the next revision.

NAME

hotkey – scan keyboard input for hotkey commands

SYNOPSIS

int hotkey(void)

DESCRIPTION

hotkey() tests for a keyboard hit and evaluates the input. If the result is a two character sequence beginning with zero (ALT and Function keys), the value of the secondary code is returned.

The *SH_IMET* program currently recognizes several hotkey sequences. These are listed in the following table with their functions.

KEYS	ACTION
ALT-B	begin periodic serial data output activity (once each minute)
ALT-C	cycle power to sensor modules and re-enable all modules
ALT-E	end periodic serial data output activity
ALT-P	cycle power to sensor modules
ALT-S	execute a module sample cycle w/o recording
ALT-X	end program activity and return to interactive DOS prompt

RETURNS

The function returns zero if no hotkey sequence is detected. Otherwise it returns the secondary key code.

NAME

imet_err – register and print an error message

SYNOPSIS

```
void imet_err(message)
  char *message;
```

DESCRIPTION

imet_err() writes an error string as determined by argument *message* as the last error message received. The date and time are prepended to the message. This new string is displayed on the screen as the last error message. Each time the screen is re-written, this message is displayed, until a new error message is registered.

SEE ALSO

pemess(3), info_mess(3), showdata(3)

NAME

imtime – convert long integer date/time to individual values

SYNOPSIS

void imtime(dtm)
long *dtm;

DESCRIPTION

imtime() function converts a long integer (typical UNIX style storage) date and time into individual variables. The long integer value is typically derived from the function **time.**

The resulting variables are stored as global integers and are: *gyr, gmo, gda, ghr, gmn,* and *gsc.*

RETURN VALUES

none

SEE ALSO

totime(3)

NAME

info_mess – display an information message on the screen

SYNOPSIS

```
void info_mess(message)
  char *message;
```

DESCRIPTION

info_mess() displays a message string as determined by argument *message* on line 24 of the display console.

SEE ALSO

pemess(3), imet_err(3), showdata(3)

NAME

pemess – display currently registered error message

SYNOPSIS

void pemess(void)
  char *message;

DESCRIPTION

pemess() displays the currently registered error message string to the console, centered on line 23. If the error message string is NULL, there is no display output.

SEE ALSO

info_mess(3), imet_err(3), showdata(3)

NAME

senddata – transmit data values via COM1 port

SYNOPSIS

void senddata(void)

DESCRIPTION

senddata() transmits data values using the COM1 serial port. The function is typically called once per second from the main program. It operates in two modes. The first mode involves transmitting data values once per minute. The alternate mode is to transmit values upon receiving a command from the COM1 port.

The output from this port, in either mode, consists of a series of ASCII lines containing the measurement name and its value. Each line is terminated with a carraige-return line-feed pair and the total transmission is terminated with an ETX (03) character. The table below shows an example of a typical output group (with GPS). The variable name and value information is part of the output stream. The description column, added here for information only, does not appear in the actual output.

NAME	VALUE	DESCRIPTION
DAY:	91/12/12	*date*
TIM:	12:23:00	*time*
WNS:	1.2	*wind speed in meters/second*
WND:	210.0	*wind direction in degrees*
BPR:	1009.0	*barometric pressure in millibars*
HUM:	43.8	*relative humidty in percent*
AIR:	22.4	*air temperature in degrees C*
SEA:	15.2	*sea surface temperature in degrees C*
SWR:	756.9	*short wave radiation in watts per square meter*
PRC:	10.34	*precipitation in millimeters*
FXT:	15:45:32	*fix time*
LAT:	41 31.47 N	*latitude in degrees and minutes*
LON:	70 40.30 W	*longitude in degrees and minutes*
HDG:	126.8	*heading in degrees true*
SPD:	8.44	*speed in knots*
FXQ:	6 9 8	*fix quality indicators*

If the system configuration does not support a GPS receiver, the data lines containing navigation information do not appear as part of the transmitted output.

The first time the **sendata()** function is called, an interrupt input handler is invoked to capture any commands sent to the COM1 port. Each time the function is called thereafter, the interrupt handler is checked to determine if any commands have been received. These commands can control both periodic and single transmissions. The commands are:

#IMETE - end periodic transmissions
#IMETB - begin periodic transmissions
#IMETS - perform single transmission

The global variable *sendflag* affects operation of the **sendata()** function. It can be set by the main program to control transmit activities. *sendflag* can have three values:

0 - disable periodic transmission
1 - enable periodic transmission
2 - transmit data once

RETURN VALUES

none

BUGS

The condition of the interrupt input function is only tested on entry to this function. When the main program is busy with data collection or other activities, this function is not entered. Hence, commands received by the COM1 port may not get immediate attention.

SEE ALSO

showdata(3)

NAME

setdtime – set date and time to DOS system software clock

SYNOPSIS

void setdtime(dtm)
int *dtm;

DESCRIPTION

setdtime() sets the date and time into the DOS software system clock. The values to be set are derived from a user supplied integer array. The array is of length 6 and contains:

dtm[0] = year(00-99)
dtm[1] = month(1-12)
dtm[2] = day(1-31)
dtm[3] = hour(0-23)
dtm[4] = minute(0-59)
dtm[5] = second(0-59)

RETURN VALUES

none

NOTES

This function should not be confused with function **setrtime.** This function sets the time of a DOS system software clock while **setrtime** sets the time of a hardware clock chip.

SEE ALSO

getdtime(3), getrtime(3), setrtime(3)

NAME

setrtime – set date and time to real-time clock chip

SYNOPSIS

void setrtime(dtm)
int *dtm;

DESCRIPTION

setrtime() sets the date and time into the AT system hardware real-time clock chip. The values to be set are derived from a user supplied integer array. The array is of length 6 and contains:

dtm[0] = year(00-99)
dtm[1] = month(1-12)
dtm[2] = day(1-31)
dtm[3] = hour(0-23)
dtm[4] = minute(0-59)
dtm[5] = second(0-59)

RETURN VALUES

none

SEE ALSO

getrtime(3), getdtime(3), setdtime(3)

NAME

chekcdf, setcdf, setid, putcdf, endcdf

SYNOPSIS

#include <netcdf.h>

void chekcdf(void)

void setcdf(void)

void setid(void)

void putcdf(void)

void endcdf(void)

DESCRIPTION

These are the netCDF interface functions for the shipboard IMET systems.

chekcdf() determines the presence of the daily output file. If the file exists, it is opened and setup with calls to the netCDF library and **setid().** If the file does not exist, it is created will a call to **setcdf().** If the file exists, but has zero length, it is deleted.

setcdf() creates and configures a netCDF file appropriate for data storage of shipboard IMET values. This is accomplished with calls to the netCDF library.

setid() sets the proper variable id information for a re-opened data file.

putcdf() writes the appropriate values to a data record in the open netCDF file.

endcdf() closes the netCDF data file currently in use.

NAME

center, clreol, clrinv, clrscr, gotoxy, setinv

SYNOPSIS

```
void center(line,string)
  int line;
  char * string;

void clreol(void)

void clrinv(void)

void clrscr(void)

void gotoxy(column,line)
  int column,line;

void setinv(void)
```

DESCRIPTION

These are screen control functions for the video device using the ANSI.SYS driver.

center() displays the selected string *string* in the center of the specified line *line.*

clreol() clears the current line from the cursor to the end of line.

clrinv() clears the inverse video mode.

clrscr() clears the entire screen and positions the cursor at home.

gotoxy() positions the cursor at the selected *line* and *column.*

setinv() starts the inverse video mode for subsequent characters.

NAME

logstart, logstop, logenter

SYNOPSIS

int logstart(void)

int logstop(void)

int logenter(void)

DESCRIPTION

These functions control access and entries to a monthly log file that is used to keep track of errors and other *SH_IMET* program activities. Each message entered into the log file is prepended with the current date and time.

logstart() creates a file name based on the current year and month, in the form **yymm.log.** The file is opened and a program start message is written into the file.

logstop() writes a program stop message to the log file.

logenter() writes a line to the log file containing the timeout list for the modules.

BUGS

These functions should return an indicator if the file cannot be opened. An additional function should be added to allow a user defined message to be entered into the log file.

NAME

showdata – display IMET and GPS data values to screen

SYNOPSIS

void showdata(void)

DESCRIPTION

showdata() formats and displays values from IMET and GPS measurements on the video screen. These values include the cumulative number of timeouts experienced with each IMET module. The following shows an example screen layout and contents. This function is generally called after each one-minute sample sequence.

```
                 IMET SHIPBOARD SYSTEMS - R/V OCEANUS ver: 21.Jul.92

SENSOR          VALUE          TIMEOUTS  GPS NAVIGATION
WIND SPD:       1.4 M/S           00     FIX TIME:         11:25:45 UTC
WIND DIR:       173.3 DEG                LATITUDE:         41 31.42 N
BAROMETER:      1022.37 MB        00     LONGITUDE:        70 42.32 W
HUMIDITY:       63.8 %            00     SPEED:            10.2 kts
AIR TEMP:       2.976 DEG C       00     HEADING:          128.2 True
SEA TEMP:       4.772 DEG C       00     STATUS:            6 8 7
SHORT WAVE:     243.6 W/M**2      00
PRECIP:         29.49 mm          00


                               Error Messages On This Line
         92/01/22              Information Messages Here          13:44:28 UTC
```

RETURN VALUES

none

SEE ALSO

senddata(3)

NAME

timeclr – disable real-time alarm interrupt

SYNOPSIS

void timeclr(void)

DESCRIPTION

timeclr() clears the alarm interrupt from the PC-AT system real-time clock. This interrupt was initialized using the **timeinit** function. This function clears the interrupt mask in the interrupt controller and disables interrupt output from the real-time clock chip.

RETURN VALUES

none

SEE ALSO

timeinit(3)

NAME

timeinit – initialize real-time alarm interrupt

SYNOPSIS

timeinit(mode)
int mode;

DESCRIPTION

timeinit() initializes an alarm interrupt from the PC-AT system real-time clock. An interrupt vector is installed that points a simple interrupt handler. The real-time clock is set to generate an interrupt at an interval dependent upon argument *mode,* which has three valid values. The following list shows the mode values and corresponding interrupt periods.

mode 0 = 500 milliseconds
mode 1 = 1 second
mode 2 = 1 minute

Each time an interrupt occurs, the global variable *timflg* is set to 1. The user is responsible for monitoring this variable and resetting its value to 0 after use.

RETURN VALUES

timeinit() returns a zero if the initialization is successful, or non-zero if the specified mode argument is not valid.

SEE ALSO

timeclr(3)

NAME

totime – convert time values to UNIX long integer form

SYNOPSIS

long totime(yr,mo,da,hr,mn,sc)
int yr,mo,da,hr,mn,sc;

DESCRIPTION

totime() converts the individual values of date and time passed as arguments into a single long value.

RETURN VALUES

The long integer time in UNIX format is returned.

SEE ALSO

imtime(3)

DOCUMENT LIBRARY

March 11, 1991

Distribution List for Technical Report Exchange

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology &
Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological
Observatory
Columbia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Stennis Space Center
NSTL, MS 39522-5001

Marine Sciences Collection
Mayaguez Campus Library
University of Puerto Rico
Mayaguez, Puerto Rico 00708

Library
Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB
UNITED KINGDOM

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

50272-101

REPORT DOCUMENTATION PAGE	**1. REPORT NO.** WHOI-92-43	**2.**	**3. Recipient's Accession No.**
4. Title and Subtitle IMET Shipboard Systems, Operations and Software User Manual			**5. Report Date** November 1992
			6.
7. Author(s) Kenneth E. Prada			**8. Performing Organization Rept. No.** WHOI-92-43
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			**10. Project/Task/Work Unit No.**
			11. Contract(C) or Grant(G) No. (C) OCE-92-04034 (G) OCE-87-09614
12. Sponsoring Organization Name and Address National Science Foundation			**13. Type of Report & Period Covered** Technical Report
			14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-92-43.

16. Abstract (Limit: 200 words)

This report has two parts. The first is a copy of the Operations and Software User Manual prepared for use with the IMET shipboard software distribution. It describes the programs used to acquire and record data from IMET systems installed on R/V Knorr and R/V Oceanus. The second part adds appendix material that contains the documentation pages for programs and subroutines used in the IMET shipboard software system. These items are available through network or diskette access. This report has been prepared to give this information broader visibility and circulation.

17. Document Analysis a. Descriptors

IMET
meteorological
shipboard systems

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement Approved for public release; distribution unlimited.	**19. Security Class (This Report)** UNCLASSIFIED	**21. No. of Pages** 61
	20. Security Class (This Page)	**22. Price**

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce