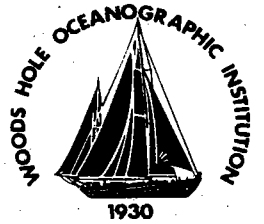
WHOI-90-44

Copy 2

Woods Hole Oceanographic Institution



The Seadata Program

by

Thomas W. Danforth

October, 1990

Funding was provided by the Office of Naval Research through Contract No. N00014-84-C-0134 and the U.S. Geological Survey under Contract No. 14-08-0001-A0245.

WHOI-90-44

The Seadata Program

by

Thomas W. Danforth

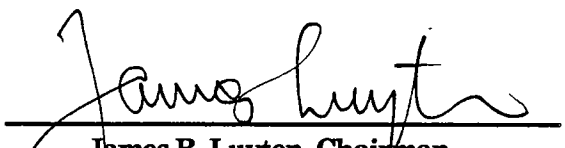
Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

October, 1990

Technical Report

Funding was provided by the Office of Naval Research through Contract No. N00014-84-C-0134 and the U.S. Geological Survey under Contract No. 14-08-0001-A0245.

Approved for Distribution:



James R. Luyten, Chairman
Department of Physical Oceanography

Table of Contents

Abstract

Current meter and meteorological instrument data are typically stored in the instrument on cassette tapes. Seadata, described in this report, is a PC version of the original CARP program (CAssette Reading Program) which transferred the data and prepared it for further processing. Also described are two programs which provide byte swapping which is necessary to use the PC data on a VAX/VMS computer. Some changes to the CARP format have been made and are documented here.

The Seadata Program

Introduction

Seadata is a personal computer (PC) version of the CARP program (CAsette Reading Program) which was written by Mary Hunt in 1972 for the Hewlett Packard (HP) 2100 series of computers. CARP was written to read current meter data which was stored on cassette tapes in the instruments. It was rewritten by Jerry Needell in 1983 as CRAP for use on LSI-11 computers. Both CARP and CRAP read data sent from a Model 12 cassette tape reader manufactured by Sea Data Corp., Newton, Mass. This version builds on the earlier versions by taking advantage of PC technology and the capabilities of the MS-DOS operating system.

In the process of writing seadata, the original CARP and CRAP data formats have been extended. As a result, two other programs (PCARPHP and PCARP) have been written to make the output from seadata look as though it came from an HP or an LSI computer. These programs run on a VAX running VMS and are also documented here.

While writing these programs, I have had the assistance of a number of people. Among them are Ken Prada, Melora Park Samelson, Robin Singer and Dave Aubrey.

Operating environments

Seadata is written in Turbo C and will run on 80286 or 80386 based PC's running MS-DOS Version 3.2 or greater. It also requires a parallel interface card based on an Intel 8255 processor which, in this case, was purchased from Optimal Technology, Inc. (part IB-24). This interface was modified so it would be compatible with the signals sent from the Model 12 tape reader. The modifications to the

interface and the cabling to connect it to the PC are documented in Appendices 1 and 2.

Since most of the programs which process cassette data are run on a VAX/VMS computer, the two programs (PCARP and PCARPHP) which convert the output to the CRAP or CARP format are written in VAX C and run under VMS.

Overview of the data flow

In order for the current meter data recorded on cassette tapes to be processed on a VAX running VMS, the data must go through several steps to get it ready. The first step is reading the tape using a Sea Data Model 12 reader. The tape reader is connected to a PC through a special parallel interface and the seadata program is run to collect the data and store it on the PC's hard disk. The output from the seadata program is two files with the extensions ".CMM" and ".DAT" for comment and data respectively. The comment file (".CMM") is a printable ascii file which contains some header information, the comments entered by the user, and information about the number of records processed and errors encountered. The data file (".DAT") is a modification of the CARP data files and contains the binary data read from the tape.

The ".CMM" and ".DAT" files can be used for processing on the PC if desired or they can be copied to a VAX for processing under the existing Buoy processing system. The files can be copied to a VAX in one of two ways. The first uses the file transfer program, ftp, and sends the files over ethernet. The ethernet transfer must be done in ascii mode for the ".CMM" file and binary mode for the ".DAT" file. An example of an ethernet file transfer is documented in Appendix 3, also documentation on ftp may be useful.

The second method of file transfer is via 9 track, ANSI labeled tape. ANSI tapes which are not created on VAX/VMS do not have some of the file handling information in the file header which is used by VMS. If tape copies are used, the

number of bytes in each record on the tape will be different than when the file is copied by ethernet. The files copied via ANSI labeled tape have an extra byte of carriage control which must be eliminated.

Since the tapes read on a PC are most likely to be processed on a VAX, two programs were written to convert the PC's data format to the VAX's data format. The program to produce a CRAP data file (looking like it came from an LSI) is called PCARP. This program requires the input of the two data files (".CMM" and ".DAT") and will produce a third file with an extension of ".LSI". The ".LSI" file is acceptable as input to the CARPBIT program used as part of the Buoy processing system. The program to produce a CARP format data file (looking like it came from an HP) is called PCARPHP. It also requires the input of the two data files and will produce a third file with an extension of ".HP". The ".HP" file can be used in processing by the USGS. PCARPHP also removes the extra carriage control characters introduced by copying the data to a VAX via ANSI labeled tape.

Name: Seadata

Version: 1.01 18-May-1990

Purpose: Seadata was written to replace the CARP and CRAP programs which were written to run on HP or LSI-11 computers.

Machine: 80286 or 80386 based IBM PC or compatible AT bus machine running MS-DOS version 3.2 or higher.

Language: Turbo C Version 2.0

Description: Seadata replaces the CARP and CRAP programs which were written to run on HP or LSI-11 computers respectively. The data is read from the Sea Data Model 12 cassette reader via a parallel interface (Intel 8255, 24-bit parallel I/O chip). The data is stored on input in one of three 2k byte buffers and reformatted into an output file which is a modification of the CARP format. The output file can be written to either virtual disk or hard disk. Once on disk, the files can be copied to network, tape, floppy, etc.

The function which this program provides was originally written for an HP computer and later rewritten for an LSI-11. The PC version provides some modifications:

1. Some of the Sea Data Model 12B readers have been modified to work only with an LSI-11. This modification was done to allow the LSI-11 to latch the data into its parallel port. The 8255 chip used in the parallel interface is fast enough to accept data from either version of the Sea Data reader.
2. The output buffers are much larger than before. The default output buffer size is 8k bytes. If smaller buffers are needed, the OUTB parameter in the include file seadata.h will need to be modified and the programs

recompiled and linked. Since the data is being written to disk, there may be some loss of performance and timing problems with smaller data buffers.

3. The LSI-11 version wrote characters indicating tape read errors to the terminal and/or printer. This version notes the type of errors on the monitor and prints a summary at the end of reading a tape.

4. There is no output to a printer while reading a cassette. A comment file, which summarizes the reading, is written to disk and can be printed after the reading is finished.

5. Some of the code needed for reading tapes with the model 0 or the 850 cartridge readers is in the seadata program; however, it has not been fully implemented or tested.

When seadata is executed, it first initializes its buffers and data areas and then prompts the user for the type of tape reader being used. The user must respond with a valid reader type before continuing. Next seadata paints four color bands on the screen. These bands are used to distinguish the usage of different parts of the screen. The top band is blue and is for permanent data about the program. The second band is black and is for prompting for information about the cassette being read. The third band is green and is used while reading the tape for a count of errors. The bottom band is brown and is used for error messages.

After this setup is completed, seadata prompts the user for the output file to use and opens the two files, comment and data. Next, seadata prompts for comments about the tape being read. Following the comments, seadata initializes the parallel input device and puts itself into a mode for collecting the data from the tape reader.

After the tape reading is finished, seadata writes the total number of tape records read and the number of errors both to the screen and to the comment file. Next it prompts the user for any further comments for the end of the tape reading.

At the end of reading a tape, seadata cycles to its beginning and asks the user if another tape is to be read. This cycle can continue until the tapes are finished or the disk is filled with data files.

Input: The seadata program is run by typing **"seadata"** at the DOS prompt. (assuming that seadata can be found in your path). While the program is running, it prompts the user for information. The requested prompts are discussed in the following paragraphs.

1. Seadata prompts for the reader type by printing:

   Please enter the type of reader you are using. Valid types are:

Model	*0 - 0*
Model	*12 - 12*
Cartridge	*850 - 850*

   Model #:

   The user should respond with the correct reader type. Valid entries are 0, 12, and 850. All other responses will be rejected.

   After the model number has been entered correctly, seadata will display this information as follows where **"xxx"** is the model number.

   Model # is **xxx**

2. Seadata prompts for the output file name by printing:

   Enter the root name of the file which you wish to use to store the data. The extensions '.DAT' and '.CMM' will be added for the data and comment files respectively.

   File name:

   At this point, the user needs to enter a name to use for storing the output data. The file can use any partition on the hard disk or on virtual disk. Space is a consideration as the output files may consume 2 to 3 megabytes for some of the longer tapes (between each tape, the program calculates the amount of free space on the partition just used). If a partition other than the default partition is to be used, it must be specified. The file name entered will have all characters to the right of any period (".") removed so the file name extensions can be added.

   If the file exists, seadata will ask if it is ok to write over the file. If not, then the user must specify a new file name or quit.

3. The third prompt requests the number of characters per tape record and is requested by printing:

   Enter the number of characters/cassette record (1-255):

   The user must enter a number greater than or equal to 1 and less than or equal to 255. If the entry is not in this range, the prompt will be reissued.

4. Comments are requested by printing:

   Enter comments – up to fifteen lines of information with a max. of 70 characters/line. End entry with an empty line (return only).

   1>

   The user can enter any comment information following the ">". The number of lines of comment is limited to 15 total. To end the comments, press the "enter" or "return" key at the ">" prompt.

5. To start the tape reading, seadata prompts the user as follows:

   Tape reading initialized.

   You may start the reader at any time.

   To stop the reading: Stop the reader, then

   press the ESC key.

   When this message is printed, the user may start the tape reader and the cassette records will be read. When the tape reading is finished, the user should stop the reader and then press the "ESC" or escape key. This will cause seadata to break out of its reading cycle, close the data file, and get additional comments for the end of the tape reading.

6. After reading a tape, seadata calculates the amount of free space on the disk partition just used and informs the user of the remaining space. Then it asks if another tape is to be read. The prompt is as follows where *"xxx"* is the byte count and *"Y"* is the partition designation:

   xxx bytes free on drive Y:

   Would you like to read another tape <yes or no>?

Output: The program produces two forms of output, information displayed on the monitor and the data files on disk.

The information on the monitor is written during the tape reading and tells the user how many cassette records have been read and how many errors have been found during the tape reading. The display is printed after 1000 records have been read, after every 3000th record has been read, and after the last record has been read. The errors reported are of four different types:

1. Parity errors are reported when a problem is found with the longitudinal parity in the record. All of the data in the cassette record is written to the output buffer.

2. Long records have more characters in them than the user indicated. The data up to the user-specified record length is written to the output buffer and all extra data is discarded. A tape will usually have at least one long record at the beginning of the tape. A long record will frequently generate a parity error.

3. Short records have fewer characters in them than the user indicated. All of the data in the short record is kept and the record is padded to the end with binary 0's and written to the output buffer. Short records are excluded from parity checking.

4. Tape errors are unknown errors which were found by the tape reader and indicated to the seadata program. All of the data in these records are written to the output buffer.

The display of this information is in the green color band on the monitor and appears as follows with the x's indicating the numeric fields.

Cassette records: xxxxx

Processing errors:

Parity	*long records*	*short records*	*tape errors*
xxxxx	xxxxx	xxxxx	xxxxx

The disk files are a modification of the CARP format as documented in the program document by Mary Hunt in 1972. The modifications are of two types, a separate printable file for comments and larger data buffers.

The comment file (filename extension ".CMM") is a printable ascii file with the first record an ascii form of the CARP comment header. The format used is as follows where x's indicate numeric fields.

0FFA 4A4E xxx xxxxx xxxx xxx xxx xxx xxxxx

The fields, from left to right, are

Comment header indicator; = 0x0FFA,
Data record indicator; = 0x4A4E,
Number of 4 bit characters/cassette record,
Number of cassette records/block of output data,
Number of 16 bit words in each cassette record,
An archaic flag indicating 9 track tape = −1,
Flag indicating reader type; = 0 for model 0 reader,
= 1 for 850 cartridge reader, = −1 for Model 12 reader,
Flag indicating PC used for reading tape; = −1,
Length of the output buffers.

All records in the comment file following the header record are the text of the comments entered by the user and a summary of the number of records read and errors encountered.

The data file (filename extension ".DAT") contains the data and headers in the same format as documented in the original CARP documentation with the exception that the buffers are 8192 bytes in length. The format includes a header of 20 bytes at the beginning of each output buffer followed by the cassette records and their headers. The buffer header contains the following information:

bytes 0–1 Data record indicator; = 0x4A4E,
bytes 2–3 Expected number of 4 bit characters per cassette record,
bytes 4–5 The usual number of cassette records in an output buffer.

bytes 6–7 The number of 16 bit words per cassette record,

bytes 8–9 Sequential number of this output buffer,

bytes 10–11 Number of cassette records actually in this output buffer. This will usually equal the value in bytes 4–5,

bytes 12–13 Error indicator; nonzero if an error occurred while writing the previous output buffer to disk,

bytes 14–19 Not used.

Following the header, the cassette records are packed into the output buffer. Each cassette record has a 6 byte header followed by the data. The header for the cassette records has the following format:

bytes 0–1 Number of 4 bit characters actually found in this cassette record. This will usually equal the value in bytes 2–3 in the buffer header.

bytes 2–3 Error indicator word. The bits have the following meaning:

- bits 0–3 parity error if any bit = 1,
- bit 7 short record if = 1,
- bit 8 long record if = 1,
- bit 12 record error if = 1,

bytes 4–5 Sequential number of this cassette record. If greater than 65535, this number will wrap to zero.

Following the cassette record header, there are $n - 3$ words of data where n is the number in bytes 6–7 of the buffer header.

Space not used in an output buffer is filled with binary 0's.

Errors and Diagnostics: As with most programs, seadata can produce errors. Most of the messages which are displayed are in the user interaction portions of the program; however, some errors with significant impact on the tape reading can occur during the reading. The most significant messages and procedures for dealing with the problems are listed below.

A. Data entry errors:

1. *Input error – the model type must be specified as either 0, 12, or 850. Please reenter.*

2. *Input error – the model type must be 1-3 characters and specified as either 0, 12, or 850. Please reenter.*

   These two errors can occur if the user enters the wrong information for the type of tape reader being used. The user should re-enter the information. The only responses which are valid are "0", "12", and "850". Four incorrect entries will cause the program to terminate.

3. *Unable to read comment information - continue <yes or no>?*

   This message may appear if there is some problem with the program reading the information being entered for comments about the tape reading. Try the entry again; however, if the problem persists, break out of the program by typing ^C (control-C) and run it again.

4. *The number of characters in the cassette record must be* $>= 1$ *and* < 256.

   The number of characters in a cassette record must be greater than 0 and less than 256. If this message appears, there is a problem with the number entered. The number should be entered with no decimal point, just numeric characters.

B. File and I/O errors:

1. *Error opening output file*

2. *Open failure*

3. *Unable to open file*

   Would you like to try again <yes or no>

4. *Fatal file open failure - exiting!*

   These four error messages may occur as a part of the process of opening the output files for the comments and data. Some of the messages may also have DOS error messages. If one of these messages occurs, it would be best to check the amount of free space on the disk, check the name of the files you wish to use, or check for a hardware error. Then, run the program again.

5. **"file"** *already exists*

   Would you like to write over the file? <yes or no>

   If this message occurs, the program has found that the file already exists. The user is given the opportunity to write over the file with new data or enter a different file name.

6. *Error on write to disk file.*

7. *Unable to write comment information - continue <yes or no>?*

   These error messages refer to difficulties while writing information to the comment file. If possible, a DOS error message is also printed. There may be a hardware problem and it may be best to restart the program or the computer.

C. Data collection errors:

1. *Data overrun - input buffers overflowed.*

   Cassette record # nnn

   Buffer count = ccc

   Program will terminate.

This message occurs if the PC is not able to keep up with the tape reader and process the data as fast as it is sent to the PC. This might be caused by a number of factors; however, the most likely cause is a fragmented or full hard disk which requires extra movement of the disk heads. This causes the disk write to take longer than it should and the program cannot keep up with the reader. If this happens, seadata will terminate and will need to be restarted. The value "*nnn*" is the current cassette record number and the value "*ccc*" is the count of overflowed input buffers.

2. *Unable to write data to output file - status = xxx*

   This error will occur if there is a hardware problem with the writing of data to the hard disk. The program will attempt to continue; however, it may be best to restart the program. The value "*xxx*" is the status returned by the write routine.

3. *Error in record character count [>3 or <1].*

   This is one of those "should never happen" errors. If it does happen, there is probably an error within the tape reader since it is sending an incorrect count of characters in the cassette record.

4. *Unable to find end of long record (> 300 char.).*
   The tape reading will abort.

   This error will occur if the PC encounters a cassette record which is longer than expected. The program will scan up to 300 additional characters searching for the end of the record flag from the tape reader. If the end of record flag is not found, seadata assumes that the tape records are messed up beyond hope and will abort the tape reading. The limit for this loop can be changed in the module handle.c, function l_rec.

5. *Fatal tape read error - status = xxx*

   Exiting -

   This error message is written after all attempts at recovery have failed and seadata is about to terminate. The status value "xxx" can be traced back to the portion of the code where the failure occurred.

Programmer: Thomas W. Danforth

Date: 18-May-1990

Name: PCARP and PCARPHP

Version: 1.0 18-May-1990

Purpose: These two programs were written to convert files produced by reading Sea Data cassettes on a PC to a form usable by processing programs on a VAX.

Machine: VAX/VMS

Language: VAX C

Description: PCARP and PCARPHP were written to convert files produced by the seadata program on a PC to a form which can be used by processing programs on a VAX. PCARP will make the data appear to have been produced by the CRAP program (run on an LSI-11), while PCARPHP will make the data appear to have been produced by the original CARP program (run on an HP).

Both of these programs read the data files produced by the seadata program. The output is a file with either an ".LSI" or ".HP" extension (".LSI" for the CRAP look-alike and ".HP" for the CARP look-alike).

Each of the programs reads parameters entered on the command line. PCARP uses only one parameter which is the root for the file name while PCARPHP uses two, the first being the root for the file name and the second being the mode by which the data was transferred to the VAX.

After processing the command information, the programs open the input files (one data and one comment) and create the output file. Next they build the CARP or CRAP output files from the information in the comment file followed by the information in the data file. The CRAP format requires PCARP to copy the buffer header and the cassette record headers; however, the data in the cassette records must have the bytes swapped on output. The CARP format, on the other hand, requires PCARPHP to swap the bytes in the

buffer header and the cassette record headers while copying the data in the cassette records. PCARPHP also shortens the 8192 byte input buffer to 1600 bytes for output.

Input: Both of these programs require two forms of input. One is via the command line and the other is the data files from the seadata program.

PCARP and PCARPHP are located in the directory BUOY:[SOFT.RUN] and should be executed by defining them as foreign commands so that the parameters required for their execution can be passed on the command line. The foreign command definitions are:

PCARP == "$BUOY:[SOFT.RUN]PCARP.EXE"
PCARPHP == "$BUOY:[SOFT.RUN]PCARPHP.EXE"

When executing the programs, the user can simply enter the name of the program followed by any necessary parameters as described below.

To execute PCARP, the following command should be entered

PCARP file

where "**file**" is the parameter which specifies the root name of the files to be converted. The program searches this string for a period (".") and deletes anything beyond the first period found, assuming that this is the file name extension. Any directory or logical name must, therefore, not contain a period. The file extensions .CMM and .DAT will be added by the program for the input file names. The output file will have the same root name with the extension of .LSI. The input and output files will be in the same directory. If the output file already exists, a new version will be created.

To execute PCARPHP, the following command should be entered.

PCARPHP file access

where **"file"** and **"access"** are the two positional parameters passed to the program. **"file"** is the root name of the input data files as described for PCARP above. **"access"** is the route via which the data reached the VAX (access method). This will be either a 9-track tape written on the PC or ethernet. The user must enter either **"tape"** or **"enet"** on the command line. If not specified and the file name is specified, **"access"** will default to **"tape"**.

If no parameters are specified on the command line, the programs will go into interactive mode and prompt for the parameters. The prompt for the file name is

File name:

to which the user should provide the root for the input file names as described above. The prompt for the access method is

Access method:

to which the user should respond with either **"tape"** or **"enet"**. If the response to the prompt is incorrect, the program will terminate.

The input data files, one of comments from the tape reading and one of data, have the format described in the seadata program.

Output: PCARP and PCARPHP produce output files which can be used by processing programs on the VAX. PCARP's output file (extension ".LSI") is very similar to the file produced by the CRAP program on an LSI-11 with two differences. The first difference is the addition of the buffer length field in the comment header. This uses two bytes (# 16-17) of previously unused space. The second is the size of the data buffers. PCARP's output data buffers are 8192 bytes in length.

The output file for PCARPHP (extension ".HP") is almost identical to files produced by the CARP program on an HP computer. The only extension to the original definition is that the data records are fixed at 1600 bytes in length.

These programs produce no output to the terminal if they complete their operation correctly.

Errors and Diagnostics: PCARP and PCARPHP produce three classes of error messages. All of them are considered fatal.

1. *Error opening comment file fff*

   Error: nnn

   Program exiting

2. *Error opening data file fff*

   Error: nnn

   Program exiting

3. *Error opening output file fff*

   Error: nnn

   Program exiting

   These three error messages are written to SYS$OUTPUT if the program has a problem opening one of the files. The file name is substituted for "*fff*" and the error number is substituted for "*nnn*". The user should fix the problem and run the program again.

4. *Comment data scan error*

   Program exiting

   This error message is written if the program cannot decode the header line in the comment file (".CMM"). Since the remainder of

the processing is dependent upon this information, the program terminates. The user should check to be sure that the file was transferred to the VAX correctly and that, if using PCARPHP, that the correct access method is specified.

5. *Error during read – nnn bytes read*

6. *Error during write – nnn bytes written*

   These two error messages are written if there is a problem either reading or writing one of the data files. The number of bytes read or written is reported as "*nnn*" as a possible indication of the problem or where the problem occurred. Both of these messages will be followed by the system error message which may be of assistance in fixing the problem.

Programmer: Thomas W. Danforth

Date: 18-May-1990

Appendix 1: 24 bit parallel interface from Optimal Technology, Inc.

The parallel interface used by seadata for reading tapes was purchased from Optimal Technology, Inc., Rt. 1 – Box 138, Earlysville, Virginia 22936. The price was $89.00 in 1989. The interface uses an Intel 8255 programmable, parallel processor chip.

In order for the Sea Data Model 12 reader to work with the PC, the strobe signal from the Model 12 had to be inverted so it would be recognized by the 8255 processor. For the data reading, pin #1 on J1 on the IB-24 is used to get the strobe signal from the reader (see schematic in Figure 1). The trace which connects this pin to bit PC7 on the 8255 was cut and the signal passed to a transistor (2N3904) to invert it. The output from the transistor was then wired to pins #3 & #14 on J1 on the IB-24. These are connected to bits PC2 and PC4 which are the strobe lines for units B & A, respectively, in the 8255.

The IB-24 was also modified to pass an interrupt signal from the 8255 processor to the AT bus. The interrupt used was IRQ 3, the COM 2 interrupt. The interrupt signal was wired to the AT bus by connecting a wire from pin #10 on J2 to bus line B25.

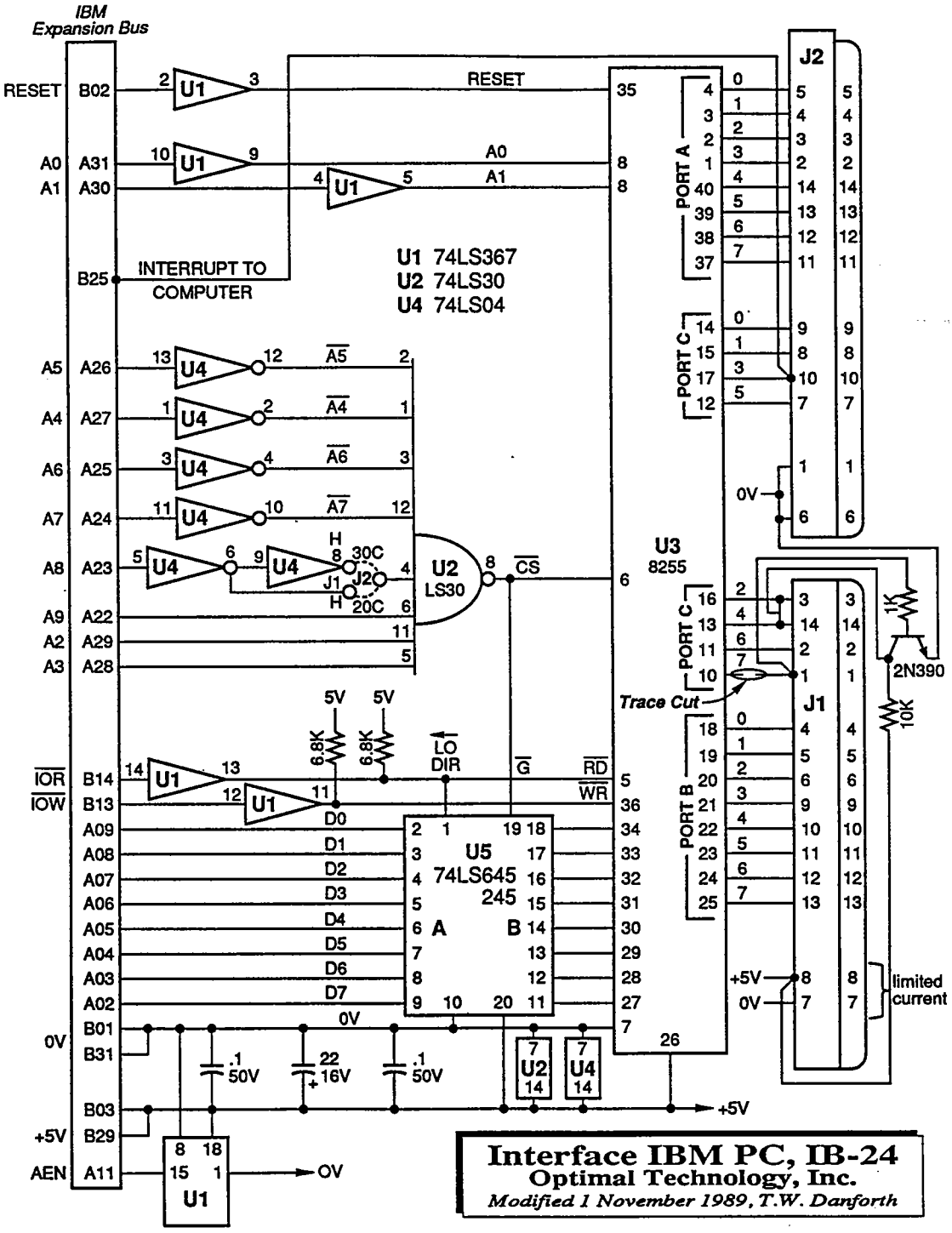

IBM
Expansion Bus
RESET
B02
A0
A31
A1
A30
B25
INTERRUPT TO
COMPUTER
U1 74LS367
U2 74LS30
U4 74LS04
A5
A26
A4
A27
A6
A25
A7
A24
A8
A23
A9
A22
A2
A29
A3
A28
IOR
B14
IOW
B13
A09
A08
A07
A06
A05
A04
A03
A02
0V
B01
B31
B03
+5V
B29
AEN
A11
U5
74LS645
245
D0
D1
D2
D3
D4
D5
D6
D7
U2
LS30
CS
DIR
G
RD
WR
U3
8255
PORT A
PORT B
PORT C
J1
J2
Trace Cut
1K
10K
2N390
limited
current
.1
50V
22
16V
.1
50V
Interface IBM PC, IB-24
Optimal Technology, Inc.
Modified 1 November 1989, T.W. Danforth

Figure 1

Appendix 2: Cabling documentation

A cable was designed for the PC to Model 12 reader which would put the data signals in the best order for the PC to use them. This is not the same as documented in the Model 12 reader documentation. The order of the data bits in the PC is as follows.

bit 15 – bit 12	Highest data "character"
bit 11 – bit 8	Middle data "character"
bit 7 – bit 4	Lowest data "character"
bit 3	end of record bit
bit 2	character count – high bit
bit 1	character count – low bit
bit 0	error flag

The IB-24 interface has two 14 pin connectors. Each is configured to use one of the data ports on the 8255 parallel chip (either A or B) and half of port C. Seadata uses ports A and B for data and uses the signals in port C for strobing the data into the 8255 chip and then interrupting the computer.

The wiring necessary to connect the IB-24 interface with J1 on the Model 12 reader is shown in Table 1.

Table 1: Wiring to connect the IB-24 interface with J1 on the Model 12 reader.

IB-24 function	Pin	25 pin connector	wire color	Model 12 – J1 Pin #	function
J2					
A3	2	13	blue – blk	11	DL 7
A2	3	12	orng – blk	10	DL 6
A1	4	11	blk – wht	9	DL 5
A0	5	10	orng	8	DL 4
C5					
C1					
C0					
A7	11	22	wht	15	DL 11
A6	12	23	green	14	DL 10
A5	13	24	red	13	DL 9
A4	14	25	red – grn	12	DL 8
	6	9	wht – blk,red	33	ground
J1					
C7	1	4	orng – grn	17	strobe
C6					
B0	4	3	blk	1	message
B1	5	2	blk – wht,red	2	WL0
B2	6	1	wht – blk	3	WL1
B3	9	14	red – wht	16	Last
B4	10	15	grn – wht,blk	4	DL0
B5	11	16	blue	5	DL1
B6	12	17	red – wht,blk	6	DL2
B7	13	18	blue – red	7	DL3

Appendix 3: Ethernet file copies

One of the most efficient methods of copying the seadata output files from a PC to a VAX is via ethernet using ftp (file transfer program). When using ftp, you must remember that the .CMM files are printable ascii and the .DAT files are binary. Therefore, you must use different methods of transfer within the ftp program, ascii for the .CMM files and binary for the .DAT files.

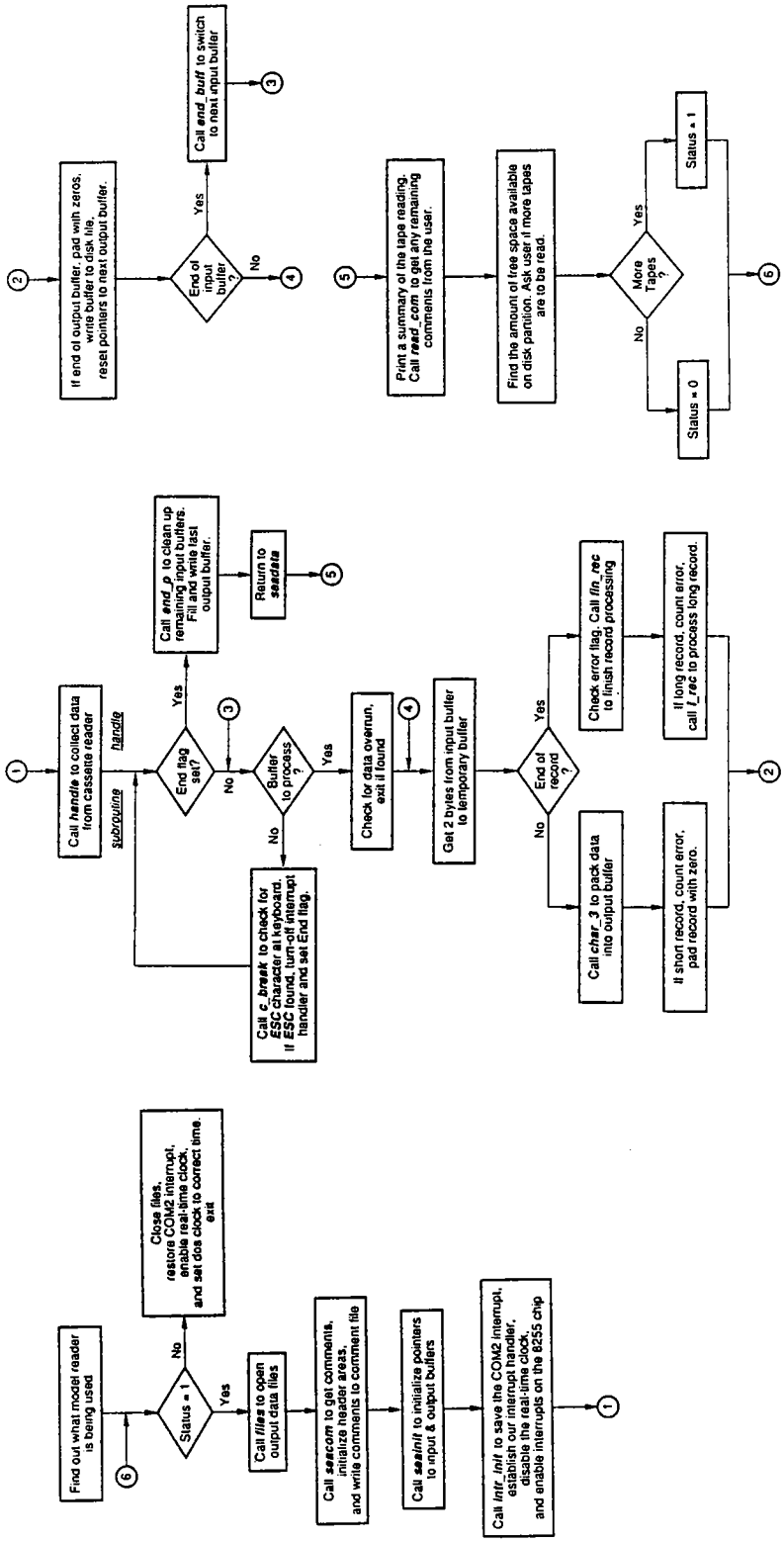
The file transfers can be accomplished as shown below. For a complete discussion of ftp, you should read a TCP/IP or ftp manual. To start ftp, you should simply enter ftp (assuming that ftp is available on your PC and can be found in your PATH).

ftp system	# system is the destination of the files.
Remote User Name:	# you enter your user name and password
Remote Password:	# for the destination system.
ftp> **cd direct**	# ftp responds with its prompt. The user can
	# enter commands to change directory and copy
	# files. A selection of useful commands is
	# given below.
ftp> **binary**	# Change to binary file copy.
ftp> **put file.DAT**	# Copy (put) the data file to the destination.
ftp> **ascii**	# Change to ascii file copy.
ftp> **put file.CMM**	# Copy the comment file to the destination.
ftp> **bye**	# Logoff destination system and exit ftp.

Useful commands include the following.

cd remote_directory	Change working directory to remote_directory.
pwd	Print name of remote working directory.
prompt	Toggle between prompting and no prompting for copying multiple files.
mput local_file	Copy one or more local_files to the remote working directory.
binary	Change file transfer mode to binary.
put local_file	Copy local file to the remote working directory.
ascii	Change file transfer mode to ascii.
bye	Logoff remote system and exit ftp.

Appendix 4: Flow diagram for seadata program



Appendix 5: Program listings

Programs are listed alphabetically.

```
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include "seadata.h"

/*      cbreak.c
/*      T.W. Danforth           15-Feb-1990
/*         TWD  18-May-1990
/*              Reset the three data ports (A, B, C) in the 8255 following
/*              the collection of data.  The reset puts the three ports
/*              to mode 0 with all set for input (high impedance on the
/*              lines).
/*
/*      This file contains two routines used for handling interrupts or
/*      breaks in the normal stream of data in the Seadata collection.
/*      The first routine checks for an ESCape character being entered on
/*      the keyboard.  The second is Sea_int which is the interrupt routine
/*      to get data from the 8255 parallel input processor.
  */

/*      c_break
/*      Routine to capture a ESC, write an ending message, and turn off
/*      the collection of data from the cassette reader.  The end of
/*      data collection is marked by setting  end_proc = 1.
/*
/*      Parameters:
/*              none.
/*
/*      Return values:
/*              none.
  */

void    c_break (void)
{
        extern  BUFF    D_buff;         /* data buffers */
        void    getrtime (int *), setdtime (int*);

        unsigned char i_mask;
        int     dtm[6];

        if (kbhit ()  &&  getch() == 0x1b)      /* if ESC char., then end */

           {  fflush (stdin);           /* attempt to clean out chars. */

              setvect (COM2, D_buff.oldfunc);   /* Restore COM2 interrupt. */
              outportb (CONTROL, 0x9B);         /* Reset the 8255 processor */

/*              Enable the real-time clock interrupt and disable the
/*              COM2 interrupt.
  */
              i_mask = inportb (0x21) | 0x08;   /* Get mask and reset COM2 */
              outportb (0x21, (i_mask & 0xFE)); /* Set mask and enable IRQ 0 */
```



```
/*              Get the value in the real-time clock and reset
/*              the DOS clock.
  */
              getrtime (dtm);           /* Get real_time clock value */
              setdtime (dtm);           /* set DOS clock */

              D_buff.L_count = D_buff.Rawpoint - D_buff.r_point[D_buff.which_raw];
              D_buff.inb_count ++;
              D_buff.end_proc = 1;

           }
        return ;
}


/*      Sea_int
/*      This routine removes the bytes of data from the 8255 parallel
/*      input ports.  The high order byte is comming in Port A and the
/*      low order byte is comming in Port B.
  */

void    interrupt Sea_int ()
{
        extern  BUFF    D_buff;         /* the data pointers and storage */

        *(D_buff.Rawpoint) = inportb (PORT_A);          /* get Port A */
        D_buff.Rawpoint ++;

        *(D_buff.Rawpoint) = inportb (PORT_B);          /* get Port B */
        D_buff.Rawpoint ++;

/*              Check for the end of the input buffer.  If found
/*              we need to switch to the next buffer.
  */
        if (D_buff.Rawpoint >= D_buff.Raw_end)
           {  D_buff.inb_count ++;              /* another buffer is ready */
              if (D_buff.which_raw == 2)
                 {  D_buff.which_raw = 0;  }
              else
                 {  D_buff.which_raw ++;  }

              D_buff.Rawpoint = D_buff.r_point[D_buff.which_raw];
              D_buff.Raw_end = D_buff.Rawpoint + D_buff.R_count;
           }

        outportb (CONTROL, 0x09);       /* Set Interrupt enable again, just in case */
        outportb (0x20, 0x20);          /* Set the End-of-Interrupt bit  */
                                        /* so normal processing can continue */

        return;
}
```

```
#include <dos.h>
/*      diskfr.c
/*      T.W. Danforth           14-Mar-90
/*      Routine to find the number of free bytes on the storage
/*      device being used for the Seadata cassettes.
/*
/*      Parameters:
/*              *disk   - Pointer to a character which contains the
/*                        upper case character for the designating the drive.
/*
/*      Returns:
/*              bytes   - a long integer which is the number of free bytes.
  */

long    disk_free (char *disk)
{
        typedef struct  {
                unsigned int    avail;
                unsigned int    total;
                unsigned int    bsec;
                unsigned int    sclus;
                }  dfree;
        dfree   df;

        unsigned char   d;
        unsigned long   bytes;

        d = *disk;
        d -= 64;                /* first char starts at 1 */
        getdfree (d, (struct dfree *) &df);

        bytes = (long) df.bsec * (long) df.sclus * (long) df.avail;
        return (bytes);
}
```