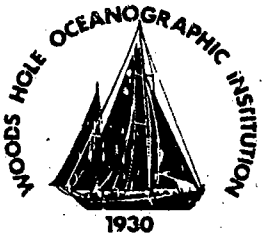
WHOI-91-38

Copy 1

Woods Hole Oceanographic Institution



Multiple Convergence Zone Acoustic Telemetry Feasibility Test Report

by

Josko A. Catipovic
Keith von Der Heydt
John Stevens Merriam
Woods Hole Oceanographic Institution

and

Geir Helge Sandsmark
University of Trondheim (Norway)

November 1991

Technical Report

Funding was provided by the Office of Naval Technology under Grant No. N00014-90-C-0098.

WHOI-91-38

Multiple Convergence Zone Acoustic Telemetry Feasibility Test Report

by

Josko A. Catipovic
Keith von Der Heydt
John Stevens Merriam
Woods Hole Oceanographic Institution

and

Geir Helge Sandsmark
University of Trondheim (Norway)

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

November 1991

Technical Report

Funding was provided by the Office of Naval Technology under Grant No. N00014-90-C-0098.

Approved for Distribution:

Albert J. Williams 3rd

Albert J. Williams 3rd, Chairman
Department of Applied Ocean Physics and Engineering

Multiple Convergence Zone Acoustic Telemetry Feasibility Test Report

Josko A. Catipovic
Keith von Der Heydt
John Stevens Merriam
Woods Hole Oceanographic Institution

Geir Helge Sandsmark
University of Trondheim (Norway)

1.0 Summary

This report describes a multiple CZ acoustic telemetry experiment conducted off the coast of California 1/28/90 - 2/2/90. The goal was to design a maximally robust high speed underwater modem suitable for data telemetry for submerged platforms and moorings.

Six modulation methods were used to transmit data at rates from 1 to 1000 baud, corresponding to bit rates up to 3kbit/sec. The modulation formats were:

1. Multiple Frequency Shift Keying (MFSK) and Binary Expurgated Modulation (BEXPERM)
2. Duobinary Frequency Shift Keying
3. Quadrature Phase -Shift Keying (QPSK)
4. 8 Quadrature Amplitude Modulation (8QAM)
5. Continuous Phase Modulation (CPM) 2DPM4 and 2CPFSK4
6. Trellis coded 8PSK

In addition, a large number of channel probe sequences was transmitted in order to estimate channel multipath, fluctuation dynamics and spatial diversity characteristics relevant to acoustic data telemetry.

The data was transmitted from a 1 kHz source suspended from the R/V McGaw, and received on a multichannel vertical array tended by the R/V Point Sur The multichannel data was digitally recorded using floating-point digitizers and stored on optical disk for further processing. Approximate transmission ranges were 70, 140, 200 and 250 km. Approximately 8 hrs of transmission were recorded at each data range.

2.0 Experiment hardware

The transmitter used digital waveform generation to cycle through the modulation formats and data rates using a minimum of dedicated hardware. The transmitter was based on a

personal computer utilizing an AT&T DSP32C digital signal processor plug-in board with analog to digital (A/D) converters for real-time waveform generation. The receiver was 1 32-element 1 km long vertical array deployed between 500m and 1500 m. It was tethered through a 3 km tether to the R/V Point SUR, which housed the multichannel digital recorders. The R/V Point SUR also deployed a profiling CTD to measure the acoustic channel profile.

2.1 Transmitter

The transmitter block diagram is shown in Figure 1 Blocks of waveforms to be transmitted were precomputed and stored on disk. The transmitter program placed the blocks in memory and sequenced the A/D through a predetermined block sequence to generate the desired analog waveform. The A/D output was a $\mp 3V$ analog waveform supplied to an ELGAR 1751SL 1.7 kW power amplifier.[1] The power amp provided an analog output at 0 to 200 Vrms, which was fed to a 4:1 step-up transformer for more efficient power matching at the transducer.

The transducer was a Lockheed/Sanders Model 30 class IV flextensional transducer. Its specifications are shown in Figure 3 through 5. The voltage response is 145 - 147 db re 1 volt per μPa, yielding a maximum transmit level of 203-205 dB, well within the transducer power rating.[2] During the experiment, the transmitter was generally operated 5-8 dB below this peak level. Output RMS voltage levels are logged among the transmission parameters for each transmission block.

The transducer was deployed 100m below the surface. It was attached to the stern frame of the vessel and generally followed the pitching motion of the vessel. Vertical displacements of up to 2 m from the mean transmitter depth can be expected due to ship motion, particularly during the early transmitted sequences, when the logged sea state was relatively high (6'- 8'). Horizontal drift velocities as high as 0.9 knots were also observed during the experiment, and resultant Doppler shift of up to 2 knots is expected in the data set.

The transducer was deployed on a 250 m "CTD wire" cable. The 0.322" diameter cable contained three #19 awg conductors within a steel armor. The overall measured cable re-

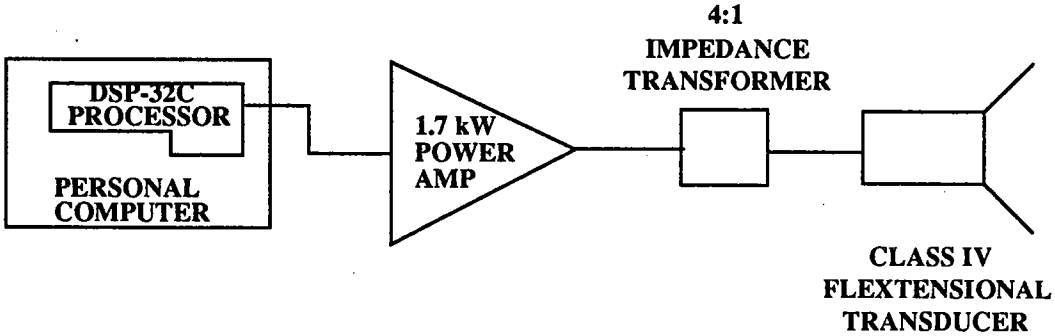


Figure 1: Waveform Transmitter Hardware

sistance is represented as a 11.56Ω resistance in series with the load. The cable is rated at a maximum voltage of 600V. However, our cable samples repeatedly withstood 1000V loads for extended periods of time, and no degradation or change in cable characteristics was noted after the cruise.

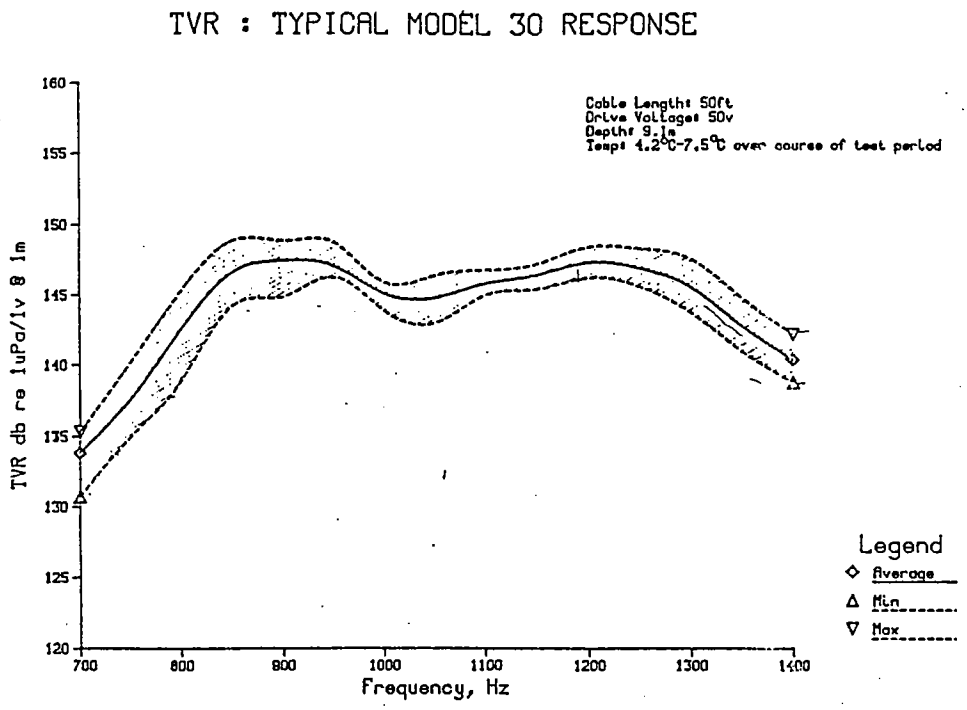


Figure 2: Model 30 Transducer transmit levels (courtesy Lockheed/Sanders)

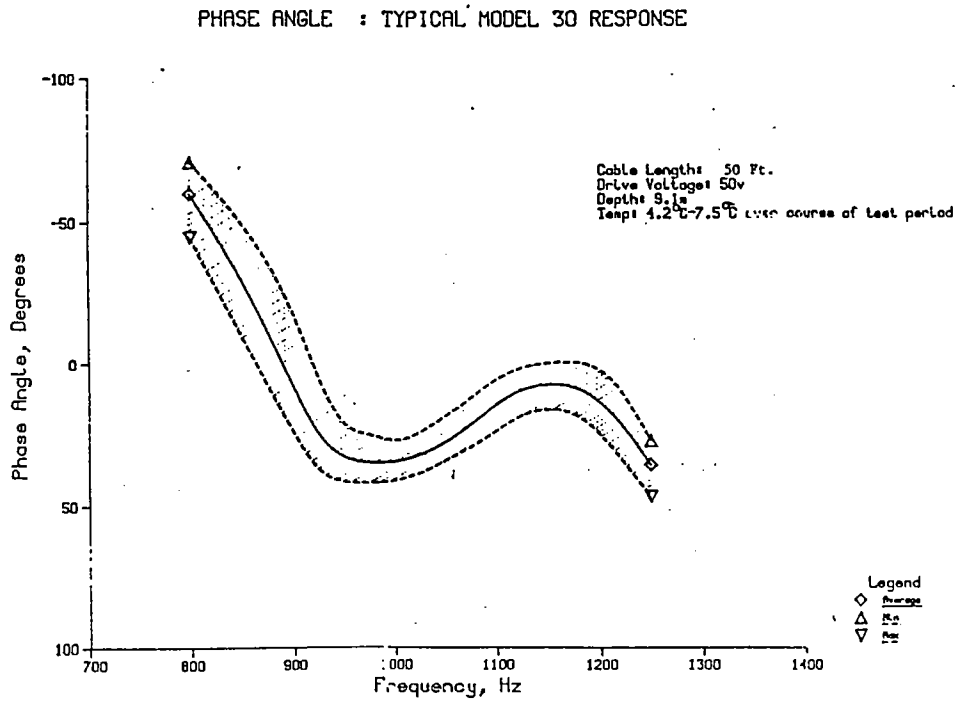


Figure 3: Model 30 transducer transmit phase angle (courtesy Lockheed/Sanders)

3.0 Waveform Generation

Experimental waveforms were stored on an NEC Powermate SX portable computer and transmitted via an AT Bus add-in board, AC5-C0-J, manufactured by Communications Automation and Control (CAC). The CAC board contains a 50 Mhz AT&T DSP32C Digital Signal Processor chip, 256 Kbyte 0 wait state ram and dual 14 bit D/A and A/D converters for general signal processing use. The board is I/O mapped into the AT Bus which can transfer data from the NEC PC's memory to the DSP32C's memory space at rates up to 1.5 Mbytes per second. To accomplish the data transfer, a PC program read raw data from the PC hard disk and passed the data to a program on the DSP32C which then clocked the data to one of the CAC board's D/A converters. The output sample rate was generated by a Syntest SI-101 Frequency Synthesizer box and was set to 4 Khz for this experiment.

Test waveforms (described in later sections) were generated on MATLAB, normalized to +/- 1.0, and stored as double precision floating point UNIX files. These were then converted to single precision binary floating point files using the MATLAB utility TRANSLATE386. A DOS program "MATCONV.EXE" was then used to generate a 16 bit integer DOS binary file from the binary floating point files. MATCONV scales the single precision floating point data to 14 bits to match the D/A converter on the CAC board but the data is saved as 16 bit integers on disk.

A data transmission program written in Microsoft C, '123_CNTL.EXE', was used to read the prestored binary waveforms and transfer the data to the CAC board. At start up, 123_CNTL.EXE downloads another program 'CACDAC' to the DSP32C. The two routines then interact through the use of two 'flip-flop' buffers in the DSP32C memory. This arrangement enables 123_CNTL to read a block of waveform data from disk and pass the data to one of the two buffers in the DSP memory while CACDAC clocks data out of the other buffer. Then, when CACDAC has clocked out that buffer's data, the two programs switch buffers. As long as 123_CNTL is able to read a block of disk data and send it to the CAC board before CACDAC has clocked out its buffer, then real time contraints have been met and the transmitted waveforms are continuous. Given the relatively slow 4K sample per second output rate, 123_CNTL only uses about 25% of its available time. Most of the time used by 123_CNTL is dependent on the hard disk access time. In other experiments the PC RAMDRIVE has been used to speed up reading the disk files and output rates of 60Khz have been accomplished.

CACDAC is written in C and compiled with the AT&T DSP32C C compiler. As mentioned, it handshakes with 123_CNTL to receive data to be transmitted. This data is sent to the D/A converter through the software programmable DSP32C serial I/O port using Direct Memory Access (DMA). The output sample frequency clocks the DMA address pointer through one of the two flip-flop buffers. The DMA controller is not capable of sending the address pointer back in a circular buffer fashion, so the software must periodically check the status of the DMA pointer and move it to another data buffer when the current buffer is exhausted. CACDAC handles this function and ensures that no discontinuities occur in the

data.

Transmitted waveforms (e.g. 2DPM4 or 2CPFSK4 of section 4) were actually sequences of smaller binary waveform files which were designated "id1" files. A given waveform was then formed by placing the names of the .id1 files in an appropriate order in a so-called ".nam" (for name) file which is an ASCII file read by 123_CNTL. 123_CNTL accepts a .nam file as input and opens up as many as 255 of the .id1 files listed. 123_CNTL then cycles through the list of files reading in their data and transferring it to the DSP. Many of the waveforms transmitted contained the some of the same patterns, so it made sense to form the waveforms from collections of smaller ones. Furthermore, to prestore all of the waveforms transmitted as complete separate entities would have used too much disk space. The .id1 files themselves required a total of about 20 Mbytes of PC disk space.

Transmission of data for the experiment was done during 90 minute windows when the Heard Island experiment was not active. To automate the transmission process, DOS batch files were used to form sequences 1 through 12 (section 4.0) by calling 123_CNTL a number of times, each time with an appropriate .nam file as input. A given sequence would last approximately 65 minutes.

4.0 Receiver

4.0.1 Array description

Signals were recorded from a 32 channel array of hydrophones suspended vertically in the water column. The primary purpose of this array was the reception of the 57 Hz signal during the Heard Island Feasibility Experiment. Telemetry experiments were conducted during the periods that the Heard Island source was not being received. The array was electro-mechanically connected to the R/V Pt. Sur from Moss Landing Marine Labs, Moss Landing CA. Recording systems were housed and operated on board the Pt. Sur. The receiver configuration is shown in Figure 4.

The 32 hydrophone sensors in the array were identical, each consisting of a small cylindrical ceramic element with an internal preamp resulting in a sensitivity at its output of -170 dB re. 1 $\mu Pa/\sqrt{Hz}$The frequency response is nominally flat up to at least 2 kHz with an omnidirectional characteristic. The low end -3dB point was 10 Hz due to the sensor capacitance and the preamp input impedance. The sensors were supplied with bipolar power via the cable and each had an independent, unshielded twisted pair that transmitted the signal the length of the cable (from a minimum distance of 1.6km to a maximum of 3km from sensor #31) to the recording systems aboard ship. At the shipboard end, signals were tapped off via isolated BNC's on a patch box to a bank of differential input amplifiers.The input to each of these was shunted by a 1% 10k resistance. Each side of the differential amplifiers was connected to local common through a 1 megohm bias resistor. The inboard 100 feet of the cable was shielded and care was taken with grounding to minimize 60 Hz pickup. The driving concern with 60 Hz contamination was the proximity of the Heard Island signals and was clearly not an issue for telemetry purposes. The net effect however of cable impedances and the 10k shunt is that the signals provided to the acquisition system

were attenuated by 4 to 7 dB at 1 kHz depending on transmission distance on the cable. This of course has no effect on signal-to-noise ratio but must be included in absolute level calculations.

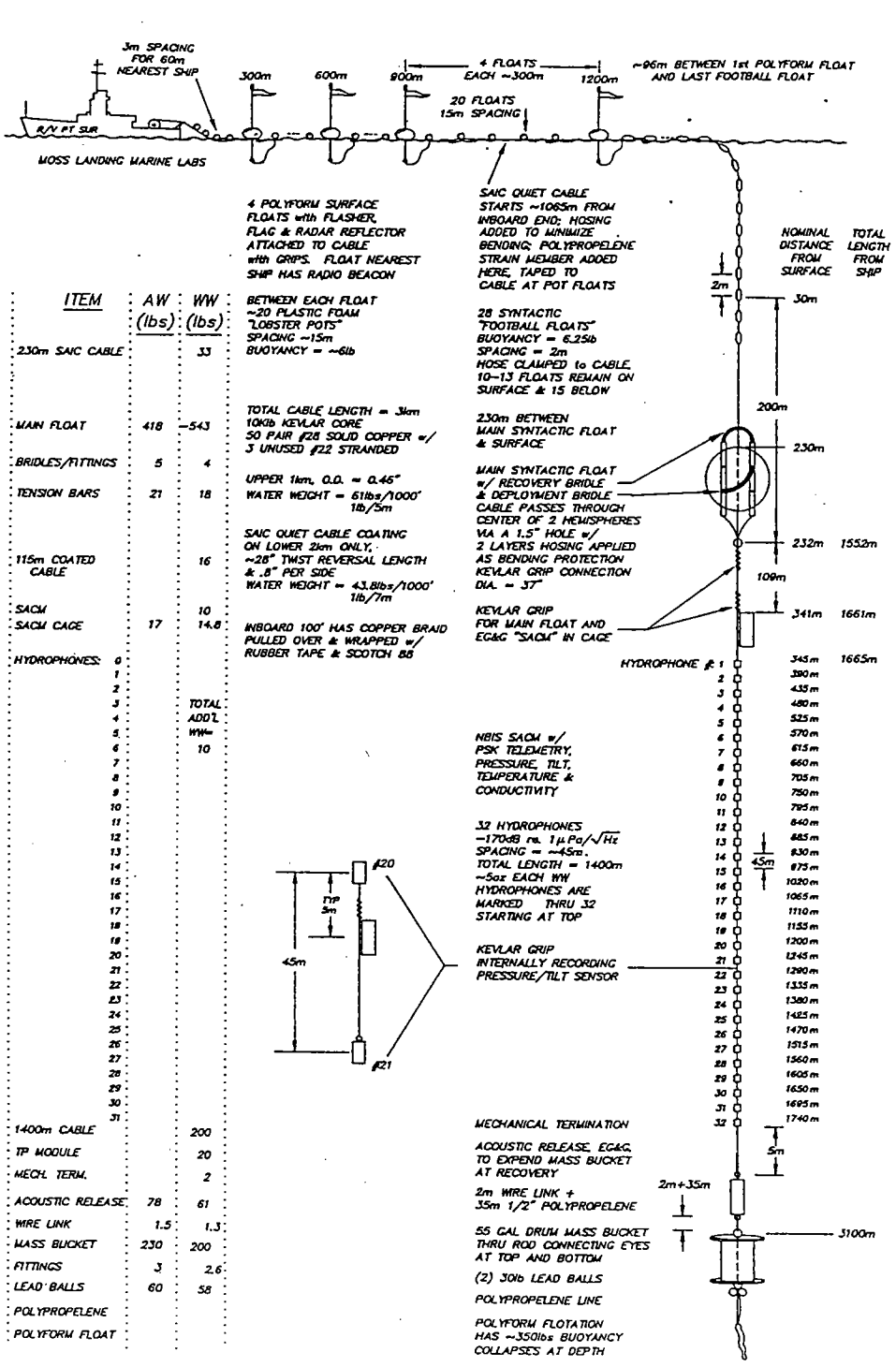


Figure 4: Receiver array configuration

A fixed gain of 20dB was applied to all channels at the differential amplfier, each of which drove 4 single ended unity gain buffers. This allowed some isolation among of each of the 3 independent recording systems used for the Heard Island work. Telemetry signals were recorded only on theWHOI multichannel system.

4.0.2 Array Deployment

The array was suspended in the water column to bracket the sound channel axis at about 800 meters with the top sensor (\#0) at 350 meters depth and the bottom sensor,(#31) at 1750 meters. The sensors were linearly spaced every 45 meters along the 1400 meter length of the array. The suspension system was designed to hold the array at this depth and minimize the effect of swell motion observed to be about a 12-13 second period. The array was ballasted to maintain this attitude with a slack surface tether section of approximately 1.1km. Refer to the drawing of Figure xx for a details of the configuration. The ship continuously attempted to minimize strain on the cable by maneuvering very slightly ahead or astern to maintain an S-shaped surface tether configuration. If the ship maintained a strain of 100 pounds or more on the surface tether, the array would slowly rise and assume an increasing angle with the vertical at the top.

Two depth and inclination recording instruments were attached to the array as shown on the drawing. The upper unit telemetered data at a 2.5 second period in realtime to a logging system on the ship. The lower unit recorded data at 1 minute period internally. Unfortunately, data from neither of these units is available for most of the telemetry transmissions. Judging from the benign sea state during telemetry experiments and depth/ tilt data acquired previously during Heard Island transmissions, a tilt estimate of less than 5 degrees and depth variation of less than 50 meters can be made.

4.0.3 Data Recording System

The WHOI acquisition system is auto-gain-ranged, thereby offering a very large measurement range (>120 dB). The system consists of a a suite of cards, 1 per channel, on a common bus and interfaced to a generic PC/AT computer. Data were stored on optical disks but can be stored on any SCSI device. Each card is independent and includes the following functionality in order from input:

- programmable fixed gain block, 0 to 42 dB, 6 dB per step
- selectable 4 pole Butterworth high pass filter
- programmable 8 pole Butterworth low pass filter, 20 to 5120 Hz, 20 Hz steps
- 5 gain range amplifier (0,18,36,54,72 dB), with 13 bit ADC at each output
- microcontroller

When configuring an acquisition session, the PC communicates with all connected amplifier cards to program fixed gain, select the high pass filter, select the lowpass cutoff, initiate offset nulling operations and set other parameters related to the sampling process.

A common sample period pulse is provided to all channels for simultaneous sampling, i.e. there is no skew in sampling time across channels. For each sample the microcontroller selects the correct gain based on amplitude and slew rate, makes an offset correction and

outputs a 16 bit data word upon being addressed by a high speed interface in the PC. The 16 bit sample consists of a 13 bit mantissa which is the output from one of the ADC's and a 3 bit gain word representing the dynamic gain applied to the signal for that sample.

The amplifier interface is essentially a programmable sequencer with FIFO buffering and the ability to transparently store data via DMA into the full 16 megabyte memory space of the PC/AT (ISA bus) machine. This ability touse **extended** memory space makes it possible to use multiple buffers and to devote significant amounts of memory space to achieve continuous acquisition rates at speeds in excess of 1 Mbyte/sec with an inexpensive computer.

As buffers fill, SCSI transfers from memory to the optical disk are initiated by the acquisition program. Using the Optimem disks, continuous acquisition can occur as fast as 200k samples per second. For the telemetry receptions, up to 32 channels were sampled at 4 kHz.

The acquisition parameters of concern for processing the telemetry data are the following:

- 4 kHz sampling rate

10 Hz low cut filter on GRA and 10 Hz low cut at the sensor for -6dB point at 10 Hz and -30 dB/oct rolloff below 10 Hz.

- 1400 Hz low pass -3dB point, 48 dB/oct rolloff
- 20 dB fixed gain for all data
- 4 to 7 dB attenuation at 1 kHz as a function of sensor position on the cable, i.e. -4 dB @ sensor #0 and -7 dB @ sensor \#31

4.0.4 Format of data on Optical Disk

All data for this experiment series was recorded on an Optimem 1000M optical disk drive using 12 inch PDO media with 1.2 gigabytes per side. The data from 16 files is contained on 4 disks. It amounts to a total of 8.8 GB. Refer to Appendix 4 for information on individual files. The file structure of these disks is **not compatible** with any operating system. A suite of DOS based programs has been written to read these disks. To date we have not read data directly with a UNIX based machine though theoretically that should be possible.

Each optical disk starts and ends with a disk header that is one sector long. Thus there is a disk header sector at the first logical block (LBA 0) and the last logical block, (LBA 1172499) for 1.2 GB per side disks. The file directory for a disk exists on the disk in 2 forms:

1. The "compact" directory which is stored backwards sequentially starting at the last LBA, i.e. at the location of the 2nd disk header. It is a series of directory entries, each one sector long, with one entry per data file. The compact directory is normally used since a listing of all files on the disk will be found in one place, i.e. the end of the disk. Typically then, LBA 1172499 would contain a disk header sector, LBA 1172498 would contain the directory entry of the first file written, LBA 1172497 for the second file, and so on.

2. Separate directory entries, each 1 sector long and identical to the entries in the compact area, are found preceding each file on the disk.

Thus, the procedure used during acquisition is to write data starting at LBA 0, while writing directory entries backward from the end. Two types of files are on the disks: *.DAT files which hold the data, and *.HDR files, which are ASCII files that contain info about the corresponding *.DAT files, eg phone sensitivity, sample rate, etc. The *.HDR files are independent files that have two sets of directory entries as do data files.

Data files from the TELEMETRY experiments are multiplexed and groundtrue (an "active" bit is a "0"). A data file is divided into records which have NO STANDARD LENGTH! The record length will be constant however, throughout any given data file. A data file is a contiguous sequence of records, each record beginning with a sector containing a record header,in which the record size, number of channels, etc.\/ are identified. The time to the microsecond of the first sample in any record and its number is recorded in the record header. Most of the information in record headers will be the same throughout a file. Following each record header sector will be a number of sectors containing multiplexed data strung out in this manner: chan 1 value, chan 2 value, \ldots, chan N value, chan 1 value, chan 2 value, \ldots etc. Each data value is two bytes in length and is written as an integer which means that if one examines a sequence of raw values, the least significant byte of a 2-byte value occurs first. There will be an equal number of values for all channels in one record. After the last sector in a record, the following sector will contain the next record header, followed by more data. In this way, all the data for one file occupies a contiguous area on the disk.

Information about the size of an entire data file is not contained in the record headers. This, along with information about the file's address, is found in the corresponding directory entry of the file. Appendix 2 contains the "C" language structures that describe the information held in directory entries and data record headers for data files. Also, a "C" routine is given that has been used to convert the stored 2-byte raw data floating point format to native single precision floating point.

4.1 Accessing Telemetry data from the optical disks

Presently, the only way to access data on optical disks is via the Powermate PC, with Internet hostname **necco.whoi.edu** in Bigelow 313. It is connected to WHOInet and can be accessed by all machines on the net. The network software on this machine is public domain NCSA, which does not allow a remote user to get an MS-DOS shell on the PC from a remote UNIX machine. This means somebody has to physically be sitting with the PC in order to get data. This isn't so bad, since someone has to to be there anyway to mount the optical disk required. The PC is equipped with a SCSI interface to the Optimem disk drives. Before powering up the PC, the Optimem disk drive must be on, so that the PC's initialisation routines will see the drive. A message, " WORM device found" will be displayed upon successful startup. The main programs for interacting with the optical disk are in the OPT directory, and are named: **odir** and **odrd**. A third program that is useful for browsing through record headers in an optical datafile is **od2gld.** These programs are all written in C, and come with many options. We will discuss only a few here, but a complete listing of options can be found at the start of **odrd.c** and **odir.c** .

ODIR

Just typing **odir** , in most cases, will provide the user with a list of all data files on the currently mounted optical disk. If a new disk has just been loaded into the drive, you may get an error message the first time \tt odir \rm is executed. Just execute it again. In rare cases, the directory area of the disk may have been clobbered, and some of the options to \tt odir \rm may have to be invoked. SEEK HELP!!.

OD2GLD:

Before reading a file, a user might like to browse through an optical disk file to find out useful parameters like the record length in bytes and time, the GMT time a record was acquired, etc. A program was written on the PC to convert optical disk files directly to GLD file format. (For a colorful discussion of "GLD" format refer to the document **Practical Guide to POLAR.WHOI.edu and ARCTIC.MIT.edu** found in Bigelow 313.) It probably will never be necessary to run this program on the PC, since a lot of floating point operations are done in converting 16 bit aquisition format to 32 bit floats, and doing this on a PC takes forever. However, **od2gld** , has a nice user interface, which displays record headers, file length info, interactively. Invoke the program this way:

od2gld filename

A menu appears with certain default settings which probably needn't be altered. You are then asked if you want to skip through the file. If you do, you can see record headers anywhere in the file. When you are done, kill the program (^ C) and delete the zerolength **filename.gld** file it created.

ODRD:

This program copies data from the optical disk to a disk file on the PC. The format of data on the disk, which is multiplexed, is preserved. The simplest invocation of odrd :

odrd -bi fname1 -bo fname2

would copy an entire data file, **fname1** to the PC as file **fname2** . This normally would result in an error because most files on the optical disk are huge, and the total storage for this purpose on the PC is about 30 Mbs. The following command:

odrd -bi fname1 -bo fname2 -st recno -rs howmany

where **recno** is the record number to start from, and **howmany** specifies the number of records to use, will be of more use. A typical TELEMETRY data file has 12 channels of data. If the user does not require to read all channels, there will be more storage space left on the PC to read in more records. Two options **-siftk** and -siftd are available for "sifting" through multiplexed data, and only extracting certain channels. For instance:

odrd -bi fname1 -bo fname2 -st recno -rs howmany -siftk

will result in the program requesting the user the channel numbers he wishes to keep. (-siftd asks which to delete). The resulting PC file for the sift options are in the original, multiplexed format, with headers adjusted to reflect the decreased number of channels.

OPTICAL (ODAS) FORMAT TO GLD FORMAT:

When the desired image of the optical diskfile is available on the PC, the next thing to do is to move it to a minicomputer or workstation. This can be accomplished over the network using ftp. On the PC, type **ftp hostname** , supplying the desired destination host. After logging in to **ftp**, type binary , since a binary file tranfer is necessary. Then **send PCfilename** .

OD2GLDV:

If the destination machine is a microVAX, after the transfer is made, the program **od2gldv** , a VAX version of **od2gld** , can be run. Like **od2gld** , od2gldv , is user-friendly. Its menu contains the defaults for converting TELEMETRY date. Options for selecting channels at this point are also available. To do so, create a file with a text editor that includes pairs of channelnumber status , where status = 1 if the channel is to be included. In **od2gldv** 's main menu, enter the name of the created channel file. A GLD file will be created with the selected channels only, i.e. the number of rows will be equal to the number of selected channels.

4.1.1 MATLAB Format

A defacto data file format that is being used is the format forMATLAB *.mat files. This format is necessary to load external data intoMATLAB with MATLAB's **load** command. The format is very similar to GLD format: data is arranged in matrix form, and a small header PRECEDES the matrix. The differences are:

1. One *.mat file can hold multiple matrices (called "variables" in MATLAB). Each matrix, with its header, is just concatenated to the previous variable's in the *.mat file.

2. For each variable in a *.mat file, the header preceding the data is 20 bytes (5 four-byte long integers) long. This header contains information about the type of host machine, size of the matrix, whether it is real or complex data, and the length of the variable name. Following these five bytes is a string containing the variable name (terminated by a NUL character). This means that the header does not have a constant length. Its length depends on the number of characters in the variable name.

Native MATLAB matrices ("variables") are stored in column order in memory. GLD files are stored in row order. ALSO, all variables in MATLAB are converted to double precision (eight bytes per value). If you **save filename variablename** a variable in MATLAB , the variable is put on disk with the name filename.mat . The data stored will be in column order, with 8 bytes per value.

4.1.2 GLD To MATLAB Conversion

We have written a program, **gld2mat** , that converts a GLD file into *.mat format, for porting to MATLAB . The program requests input file name, and then asks for an output file name which should end in .mat . It then requests a variable name which will reference the entire GLD file once you are in MATLAB . After this the user is prompted for a destination machine type, supplying codes for PC, SUN, MAC, and VAX computers. You pick the desired code. A one variable *.mat file is then produced. The user can then start MATLAB , and load *.mat to obtain access to the GLDfiles's data as a variable. Two things will have happened. The header information supplied by **gld2mat** will have taken care of the row-wise versus column-wise orientation problem, and the size of the variable in MAT-

LAB will have doubled. Type whos to see the current MATLAB storage. If you later save this variable, the output *.mat file will be twice as long as the input file, and it will be column-oriented. As of 9/1990, there is no way to save a file in single precision, although "*they are working at it*" . The double precision "feature" of MATLAB makes it difficult to work with a typical GLDfile, which may be on the order of 1 MByte or more. Once loaded into MATLAB , it consumes twice as much memory. Using the MATLAB clear command only frees the memory to the MATLAB process, the cleared memory is not made available to the Operating System. Typically, UNIX will slow down if MATLAB memory approaches the hardware limit, even if the user has cleared a huge variable. It is best to partition a huge variable, save the partitions, and exit MATLAB . Then re-invoke MATLAB , and load the saved partitions of data.

Another quick hint on **gld2mat** . If you wish to see your data values in a *.mat file from UNIX with an **od -f** command, you must use a variable name that is 3, or 7 characters long. The length of a *.mat header is not an even multiple of four unless you do this, and the subsequent float values will appear to be garbage.

4.1.3 Optical Disc To MATLAB Conversion

For those who wish to use optical disk data in MATLAB , without having to go through the steps of generating a GLD file, there are 2 routines, **od2matsp** and **od2matv** , for the SUN Sparcstation, and the microVAX, respectively. The friendly user interface to these programs is almost identical to that of **od2gld(v)**(see above).

Again, the sequence of events to move data from the optical disk to MATLAB on the SPARC station is:

1. In Bigelow 313, get help to install the right optical disk in the drive and use the PC program ODIR to look at the optical directory for your file

2. Use the PC program OD2GLD to check record times if desired tomake sure you will get the data you want.

3. Use the PC program ODRD to read data from the optical disk onto the PC hard disk....-probably no more than 30 MB at a time.

4. Use ftp {\em binary} at the PC to transfer the data to workstation of choice

5. Use OD2MATSP or OD2MATV to convert the data to native float in MATLAB format

5.0 Transmitted Data

The objective of the experiment was to evaluate a number of promising data modulation techniques at a wide range of data rates.The transmitted data is arranged into sequences approximately 70 min in duration. The 70 min duration derives from the recording capabilities of the receiver and the transmission interleave schedule with the Heard Island experiment, which was on-line for approximately 1 hour at 3 hour intervals, i.e from 01:00 to 02:00 PST, 04:00 to 05:00 PST, 07:00 to 08:00 PST etc. An additional 15 minutes of listen time was reserved to receive time-dispersed arrivals from Heard Island.

The data to be transmitted was organized into sequences. Each sequence is approximately 20 min in length. The sequences were in turn combined into 60-70 min. blocks for transmission. The two-step procedure was used to allow recombining the sequences in the field without having to manipulate the individual short segments. The structure of individual sequences is detailed below.

5.0.1 Sequence 1

This sequence contains 4PSK and 8QAM sequences. It is approximately 19 minutes in duration and organized as follows:

Modulation	Baud rate	Duration
4PSK	1 Hz	3 min 5 sec
4PSK	3.33 Hz	1 min 7 sec
4PSK	10 Hz	40.3 sec
4PSK	33.3 Hz	43.4 sec
4PSK	100 Hz	29.3 sec
4PSK	333 Hz	33.9 sec
4PSK	1 kHz	23.5 sec
8QAM	333 Hz	1 min 4.6 sec
8QAM	1 Hz	2 min 16 sec
8QAM	3.33 Hz	1 min 54.9 sec
8QAM	10 Hz	2 min 16.3 sec
8QAM	33.3 Hz	2 min 38.6 sec
8QAM	100 Hz	54.9 sec
8QAM	1 kHz	23.5 sec

Individual sequences are separated by1 sec quiet periods and are acessible by time-indexing from the block beginning. The individual waveforms consist of several signal files put together contiguously as shown in figure 6. Each waveform is started with a barker code transmitted with chip duration equal to the symbol duration for the actual waveform. Then follows a 1s silent interval before the data sequence is started with a preamble. Then follows one or more periods of a periodic data sequence derived from a maximum length shift register sequence. After the data sequence follows a 1 s. pause followed by the Barker followed by a new 1s pause.

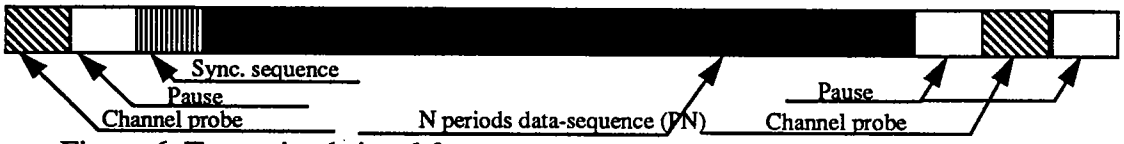


Figure 6. Transmitted signal format.

5.0.2 Sequence 2

This sequence contains channel probe sequences as well as MFSK and duobinary FSK transmissions. It is organized as follows:

Waveform	type/data rate	duration
500 Hz FM sweep	channel probe / 1 sec	90 sec
500 Hz FM sweep	channel probe / 5 sec	140 sec
1023 chip PN code	channel probe / 4 msec chip	368 sec
4 CW tones	channel probe / 50 Hz spacing	180 sec
16 CW tones	channel probe / 50 Hz spacing	180 sec
64 CW tones	channel probe / 10 Hz spacing	180 sec
500 Hz FM sweep	channel probe / 5 sec	140 sec
1023 chip PN code	channel probe / 4 msec chip	368 sec
FSK - 1 Hz	1 bit/sec	180 sec
4FSK- 1 Hz	4 bits/sec	180 sec
16FSK - 1 Hz	16 bits/sec	162 sec
32FSK - 1 Hz	32 bits/sec	162 sec
500 Hz FM sweep	channel probe / 5 sec	140 sec
1023 chip PN code	channel probe / 4 msec chip	368 sec
FSK - 10 Hz	10 bit/sec	180 sec
4FSK- 10 Hz	40 bits/sec	180 sec
16FSK - 10 Hz	160 bits/sec	162 sec
32FSK - 10 Hz	320 bits/sec	162 sec
Duobin. FSK - 10 Hz	20 bits/sec	210 sec
Duobin FSK - 100 Hz	200 bits/sec	210 sec

Individual sequences are separated by 5 sec quiet periods and are acessible by time-indexing from the block beginning.

5.0.3 Sequence 3

This sequence contains Digital Phase Modulation (DPM) transmissions at data rates from 3 1/3 to 1000 baud. Modulation index = 0.8.

Modulation	Baud rate	Duration
2DPM4	3.33 hz	6 min. 39.9 s
2DPM4	10 hz	2 min 15.3 s
2DPM4	33.3 hz	43.3 s
2DPM4	100 hz	29.1 s
2DPM4	333 hz	20.3 s
2DPM4	1 khz	13.3 s

5.0.4 Sequence 4

(sequence 15)

This sequence contains a 22 minute transmission of a 1 baud digital phase modulation with modulation index = 0.8. It is treated separately becuase of its duration.

Modulation	Baud rate	Duration
2DPM4	1 hz	21 min. 56 s

5.0.5 Sequence 5

This sequence contains Continuous Phase Frequency Shift Keying modulation (CPFSK) transmissions at baud rates from 3 1/3 to 1000. Modulation index = 0.8.

Modulation	Baud rate	Duration
2CPFSK4	3.33 Hz	6 min. 39.9 s
2CPFSK4	10 Hz	2 min 40.5 s
2CPFSK4	33.3 Hz	1 min 6.2 s
2CPFSK4	100 Hz	44.4 s
2CPFSK4	333 Hz	52.3 s
2CPFSK4	1 kHz	19.4 s

5.0.6 Sequence 6

This sequence contains a 22 min. transmission of a 1 baud CPFSK waveform with modulation index = 0.8.

Modulation	Baud rate	Duration
2CPFSK4	1 hz	21 min. 56 s

5.0.7 Sequence 7

This sequence is identical to sequence 3 except the modulation index = 1.0

(sequence 30 through 35)

Modulation	Baud rate	Duration
2DPM4	3.33 hz	6 min. 39.9 s
2DPM4	10 hz	2 min 15.3 s
2DPM4	33.3 hz	43.3 s
2DPM4	100 hz	29.1 s
2DPM4	333 hz	33.9 s
2DPM4	1 khz	13.3 s

5.0.8 Sequence 8

This sequence is identical to sequence 4 except the modulation index = 1.0.

(sequence 29)

Modulation	Baud rate	Duration
2DPM4	1 hz	21 min. 56 s

5.0.9 Sequence 9

This sequence is identical to sequence 5 except the modulation index = 1.0

(sequence 37 through 42)

Modulation	Baud rate	Duration
2CPFSK4	3.33 Hz	5 min. 15.3 s
2CPFSK4	10 Hz	2 min 40.5 s

2CPFSK4	33.3 Hz	1 min 6.2 s
2CPFSK4	100 Hz	44.4 s
2CPFSK4	333 Hz	52.3 s
2CPFSK4	1 kHz	19.4 s

5.0.10 Sequence 10

(sequence 36)

This sequence is identical to sequence 5 except the modulation index = 1.0

Modulation	Baud rate	Duration
2CPFSK4	1 hz	21 min. 56 s

5.0.11 Sequence 11

(sequence 44 through 49)

This sequence contains trellis coded 8PSK waveforms transmitted at baud rates from 3 1/3 to 1000.

Modulation	Baud rate	Duration
r=2/3 coded 8PSK	3.33 hz	6 min. 43.8 s
r=2/3 coded 8PSK	10 hz	2 min 16.6 s
r=2/3 coded 8PSK	33.3 hz	1 min 21.8 s
r=2/3 coded 8PSK	100 hz	29.2 s
r=2/3 coded 8PSK	333 hz	33.9 s
r=2/3 coded 8PSK	1 khz	13.3 s

5.0.12 (sequence 43)

This sequence contains a 1 baud trellis coded 8 PSK waveform

Modulation	Baud rate	Duration
r=2/3 coded 8PSK	1 hz	21 min. 19 s