
Applied Research Laboratory

The Pennsylvania State University

Post Office Box 30

State College, PA 16804

REVERBERATION, NOISE, AND SIGNAL GENERATION ALGORITHM (RENSGEN)

BY: J. E. SENTZ AND J. WAKELEY

Technical Note

File No. 86-64

18 April 1986

Copy No. 8

20091130166

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

Department of the Navy

Space and Naval Warfare Systems Command

Contract No. N00024-85-C-6041

Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804



REVERBERATION, NOISE, AND SIGNAL GENERATION ALGORITHM (RENSGEN)

BY: J. E. SENTZ AND J. WAKELEY

Technical Note
File No. 86-64
18 April 1986

Copy No. 8

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

Department of the Navy
Space and Naval Warfare Systems Command
Contract No. N00024-85-C-6041

Abstract:

The purpose of this study was to develop an Artificial Intelligence Expert System to detect signals in a range-Doppler map of noise and reverberation. This memorandum documents the initial phase of the study, i.e., development of a mathematical representation for a gain-controlled range-Doppler map capable of simulating various in-water acoustic background conditions.

The range-Doppler map representation was generated by an algorithm written in COMMON LISP identified as RENSGEN. RENSGEN was designed to generate a range-Doppler map with options for the presence of signals, noise, and a representation of reverberation assuming a pulsed pure tone transmitted signal. A simulated range-Doppler map of noise and reverberation generated by RENSGEN is shown in Figure 1.

This work was the result of a joint effort by The Applied Research Laboratory and the Mathematics Department of The Pennsylvania State University. Through a Mathematics Work-Study program, they provide a one-year assistantship to University seniors majoring in Mathematics.

Figure 1. A simulated range-Doppler map of noise and reverberation.

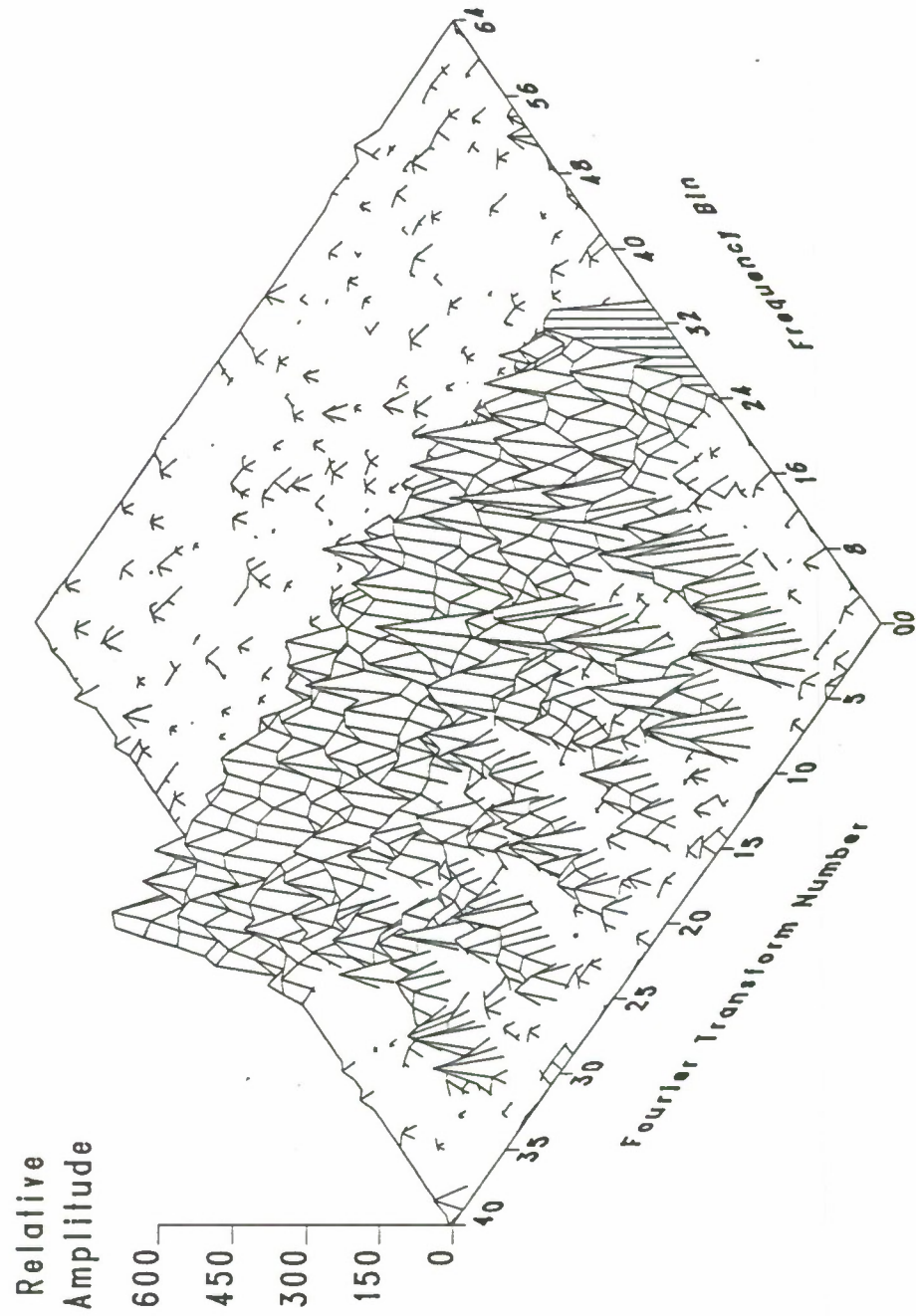


TABLE OF CONTENTS

	Page No.
Abstract	1
Table of Contents	3
List of Figures	4
Introduction	5
REverberation, Noise, and Signal GENERation	8
Appendix A - RENSGEN LISP code	20
Appendix B - Sample Interactive Terminal Sessions ...	40
Appendix C - Sample Output Files	42
Appendix D - Three Dimensional Graph Program	46
Appendix E - Instructions for Using the VAX/LISP	51
Appendix F - Interesting LISP Functions	54
Appendix G - Data for Generation of Figures	57
Appendix H - References	60

LIST OF FIGURES

Figure No.		Page No.
1	A Simulated Range-Doppler Map of Noise and Reverberation	2
2	Measured Reverberation	6
3	Simulated Reverberation	6
4	Constant-Level Signals	10
5	Constant-Level Signals with Bar	10
6	Variable-Level Signals	11
7	Signals with Noise Background, High Signal/Noise Ratio	13
8	Signals with Noise Background, Low Signal/Noise Ratio	13
9	Signals with Reverberation	14
10	Signals with Reverberation and Boundary Returns	15
11	Signals, Noise, Reverberation, and Boundary Returns	16
12	A 90-degree Rotation of Figure 11	16
13	Repositioned Signals	18

INTRODUCTION

This memorandum reports on the first part of a study designed to apply Artificial Intelligence to the detection of acoustic signals in the presence of underwater noise and reverberation environments.

The algorithm this memorandum addresses is called RENSGEN. RENSGEN is an acronym for REverberation, Noise and Signal GENERation. As the name suggests, RENSGEN is concerned with simulating a frequency domain representation of noise and reverberation to be used in the study of signal detection. RENSGEN was designed to simulate the range-Doppler map of the type of measured data shown in Figure 2.

The subsequent portion of the study deals with a program entitled RENSID (REverberation, Noise and Signal IDentification), which attempts to characterize, by amplitude and location, the signals generated in RENSGEN. RENSID will be discussed at a later date.

A range-Doppler map is a three dimensional data presentation of successive Fourier transformed, or fast-Fourier transformed, time domain information to the frequency domain. Frequency is generally expressed as Doppler, relative to the measuring platform, or segmented into frequency bins. Successive Fourier transforms represent increasing time which may be converted to range separation between measuring platform and acoustic reflector. Elapsed time and range separation between source and reflector are related by the speed of sound. Hence the name range-Doppler map.

Figure 2 displays reverberation measured from a moving platform with clearly recognizable surface and bottom returns. Figure 2 is comparable to the approximation generated by RENSGEN in Figure 3. The main difference between Figures 2 and 3 is that frequency (or frequency bin) and the time (or

Figure 2. Measured reverberation

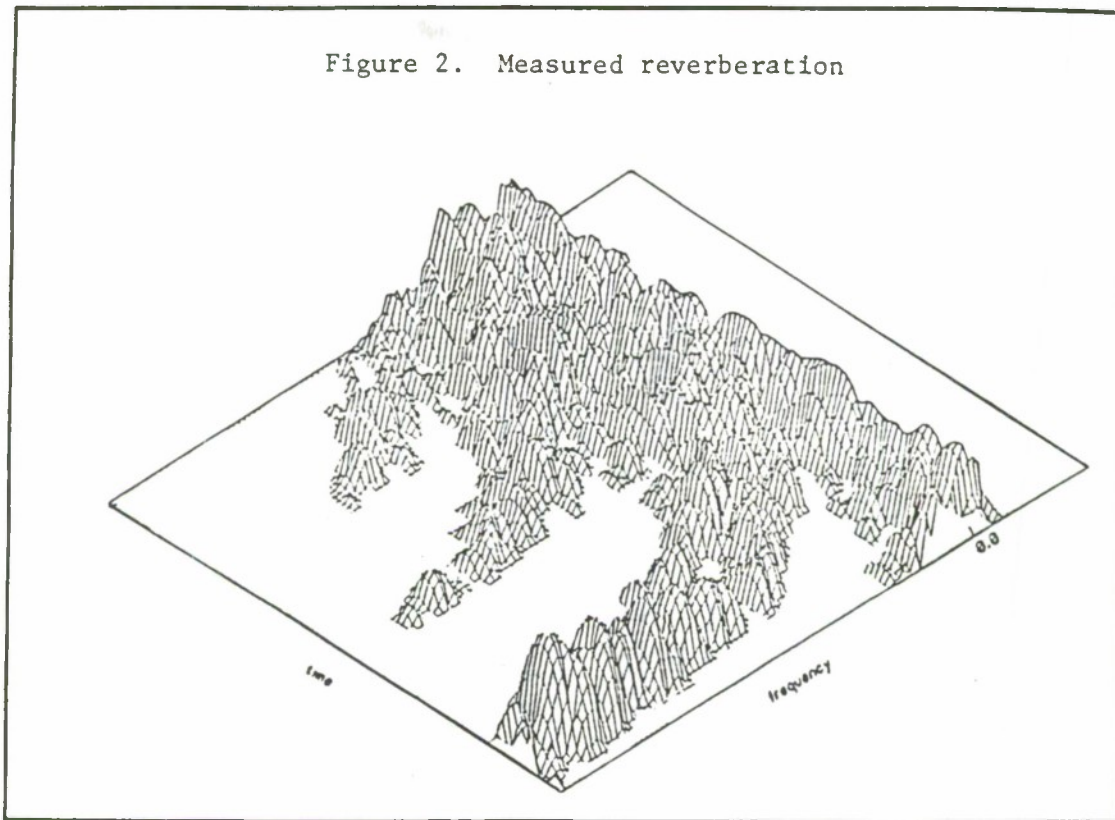
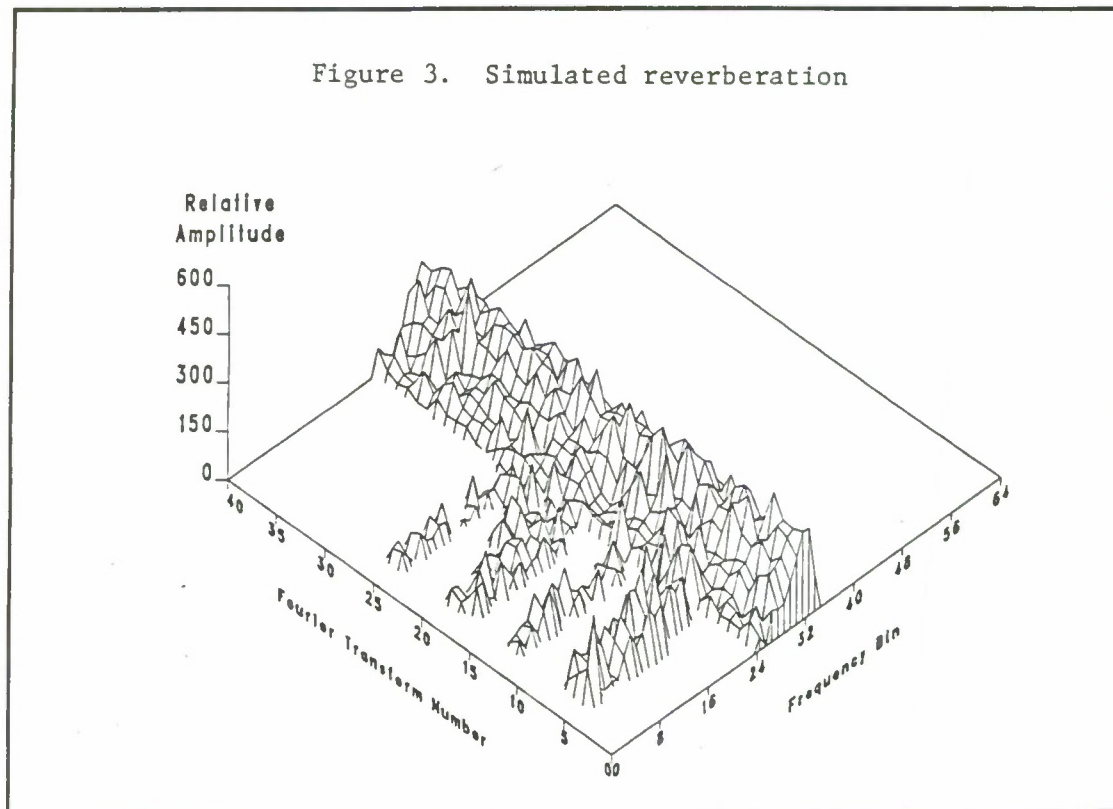


Figure 3. Simulated reverberation



Fourier transform number) scales are greater in Figure 3 by a factor of approximately two.

RENSGEN generates range-Doppler maps by one of two methods. The first method involves random placement of selected signals and optional addition of noise and reverberation environments. The second method uses a regeneration of maps, retaining original map components, but changing the positions of selected signals, i.e., the simulation of reflector motion.

The development of this program was used to study LISP (a language of Artificial Intelligence), and to gain knowledge about the field of Artificial Intelligence (AI). Familiarization with LISP and AI gained during the implementation of this algorithm was applied to the formulation of the subsequent RENSID algorithm.

REVERBERATION, NOISE, AND SIGNAL GENERATION

The REverberation, Noise and Signal GENeration algorithm (hereafter referred to as RENSGEN) generates simulated range-Doppler maps to provide data for a signal detection algorithm entitled RENSID. Each execution of RENSGEN results in one of two basic methods for generating a range-Doppler map. The first method available generates a new range-Doppler map. The second method regenerates a map by retaining previous map background components, i.e., noise and reverberation, and repositioning the simulated signals on the map. The LISP code for the RENSGEN algorithm is presented in Appendix A.

The first method of map generation produces a map from original data. Prior to the production of the map, data specifying the desired map conditions must be supplied. Samples of this input data and the method by which it is attained can be found in Appendix B. The range-Doppler map may include (or exclude) signals, noise, and reverberation. For the purpose of this memorandum reverberation has been divided into two components: (1) boundary returns, i.e., easily distinguishable returns, both single and multiple, from the air-water surface and the bottom capable of being measured with relatively wide angle acoustic beams, and (2) reverberation, i.e., a composite of volume and boundary reverberation not easily separable. The map may contain constant-level or variable-level signals and boundary returns. The number of each type of signal and the mean amplitude levels for all components are input.

The alternative method of map generation is to produce a second-generation map, that is, a map which is regenerated from a previous map. The new map will retain all of the components of the first map (i.e., number and type of signals, presence and level of noise, reverberation, and boundary returns). The difference between the two maps will be in the positioning of the signals.

Whereas the first map contained signals which were all placed on the map randomly by the algorithm, this second-generation repositions selected signals according to a time-velocity relation. Any signals not repositioned in this manner will again be randomly placed on the map. (Note: Random components of the original map, e.g. noise levels, are again random in the second-generation map. However, the distribution of these random levels are centered around the same previously selected mean.)

The RENSGEN algorithm has the capability of placing seven different signal types on a range-Doppler map. These presently available signal types are demonstrated in Figures 4 and 5. The signals selected spell the word LOT, for the person in Genesis who was directed not to look back on the "plains" where he lived, but look forward for "new sources of data." The types are referred to as L, 'fat' L, O, 'fat' O, T, 'fat' T, and bar. Note the portions of the 'bar' and the 'O' that are common in Figure 5 combine. This combination is formed by using the square root of the sum of squares of the amplitudes of the signals involved. Signals can appear on a map in one of two forms; constant-level signals or variable-level signals. For constant-level signals, the amplitude of each cell crossing is equal to the mean crossing level (supplied by the user). Randomly varying signals have cell crossing amplitudes which are Rayleigh distributed about the mean crossing level. Figure 6 shows the constant-level signal configuration of Figure 4 with variable-level signals and no background included.

RENSGEN is capable of including up to three types of background or noise-related components on a range-Doppler map. These components are called 'flat' noise, reverberation (or weighted noise), and boundary returns.

Figure 4. Constant-level signals

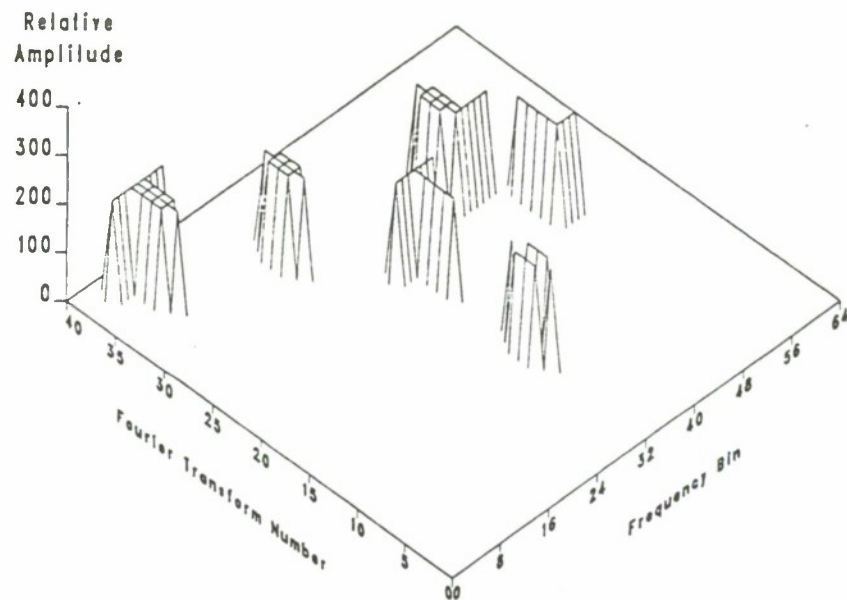


Figure 5. Constant-level signals with bar

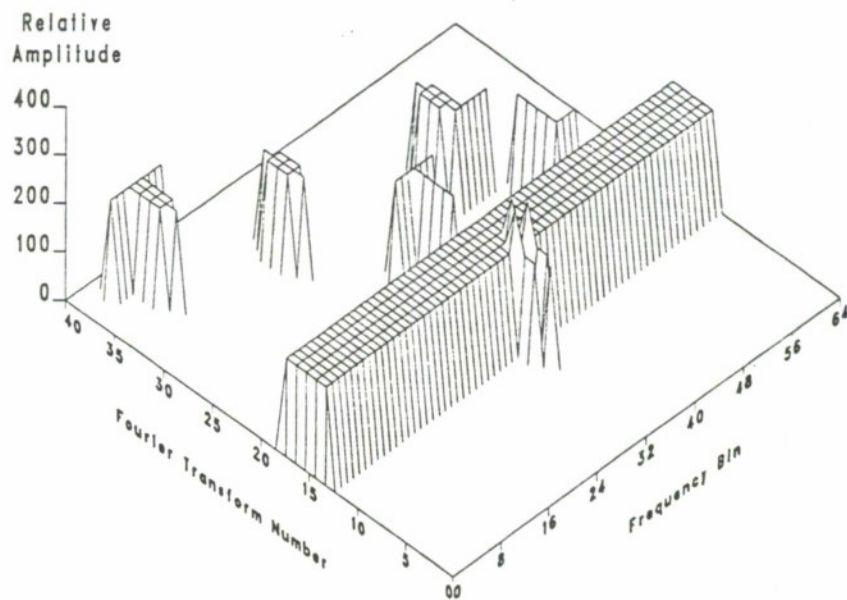


Figure 4. Constant-level signals

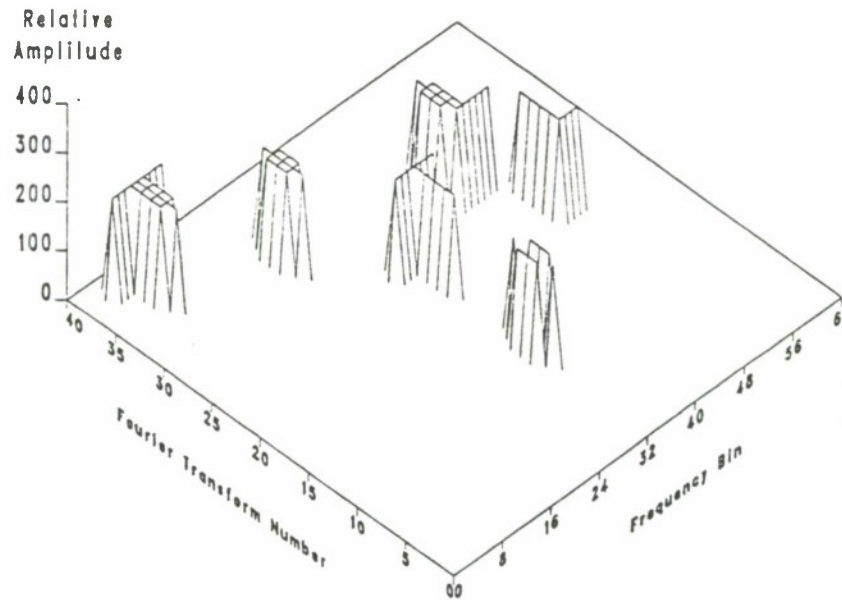
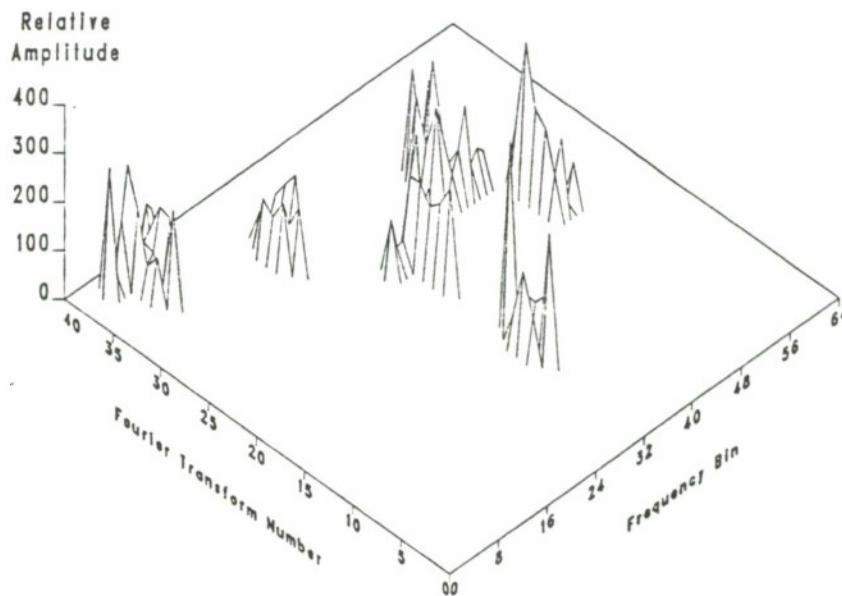


Figure 6. Variable-level signals



'Flat' noise to be added to the map is generated by a Raleigh density function about a user-supplied mean value. An example of a range-Doppler map generated with a high (10:1) mean signal to noise level ratio is shown in Figure 7. Figure 8 shows a map with a low (4:1) mean signal to noise ratio.

Reverberation, or weighted noise, is also generated by a form of the Raleigh distribution. However, the Raleigh density function in this case is altered so that a weighting factor can be introduced. The reverberation is designed to peak at the frequency bin value of 32 which is defined as zero Doppler. Figure 9 shows a map with signals and reverberation. The mean amplitude of the reverberation is supplied by the user. The user also supplies a level for the random component of the reverberation (analogous to 'flat' noise), and a value which determines the shape (width) of the reverberation peak.

Boundary returns can be included on the range-Doppler map in many forms. Boundary return components include returns from the surface, the bottom, and various combinations of surface and bottom (e.g., returns reflected from the surface once and the bottom once (S1B1), the bottom twice and the surface once (B2S1), etc.). An example of a map with all possible boundary returns implemented in RENSGEN is given in Figure 10. Figure 11 demonstrates a map which contains all signals (excluding the bar) and noise-related components that RENSGEN is capable of producing. Figure 12 shows the same data shown in Figure 11 with a 90-degree rotation of the horizontal plane about a vertical axis.

Figure 7. Signals with noise background,
high signal to noise ratio

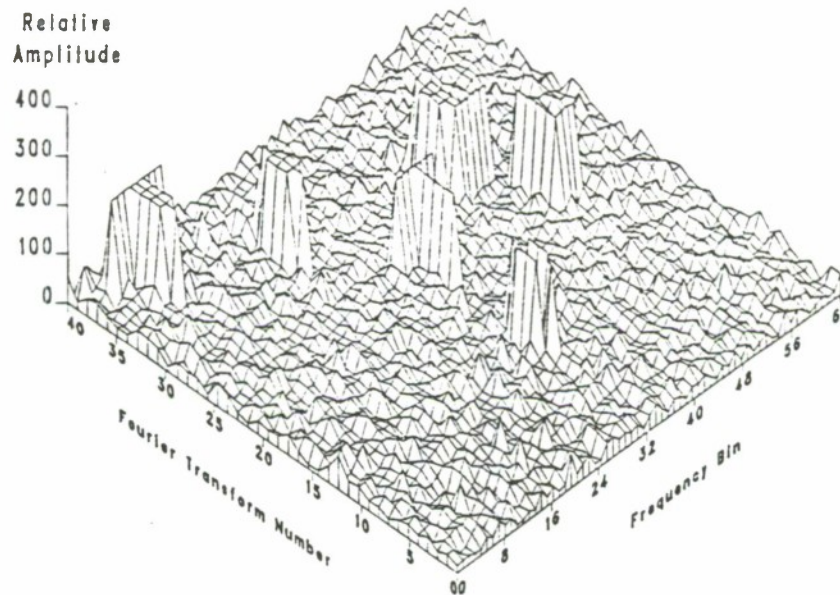


Figure 8. Signals with noise background,
low signal to noise ratio

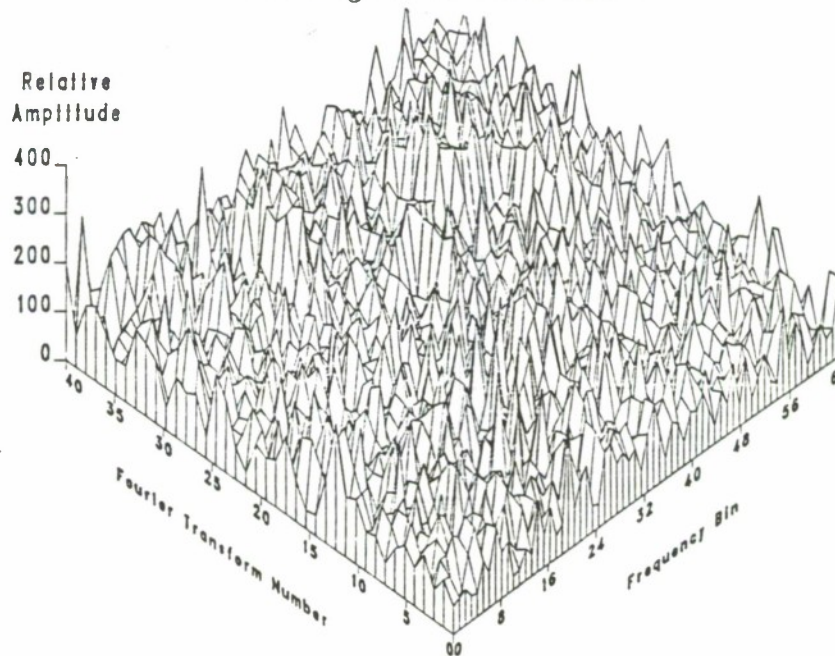


Figure 4. Constant-level signals

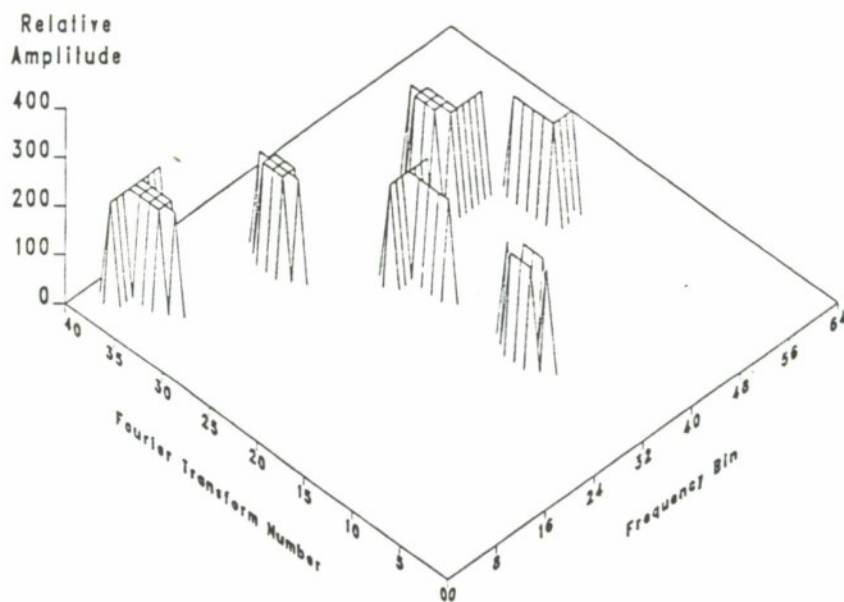


Figure 9. Signals with reverberation

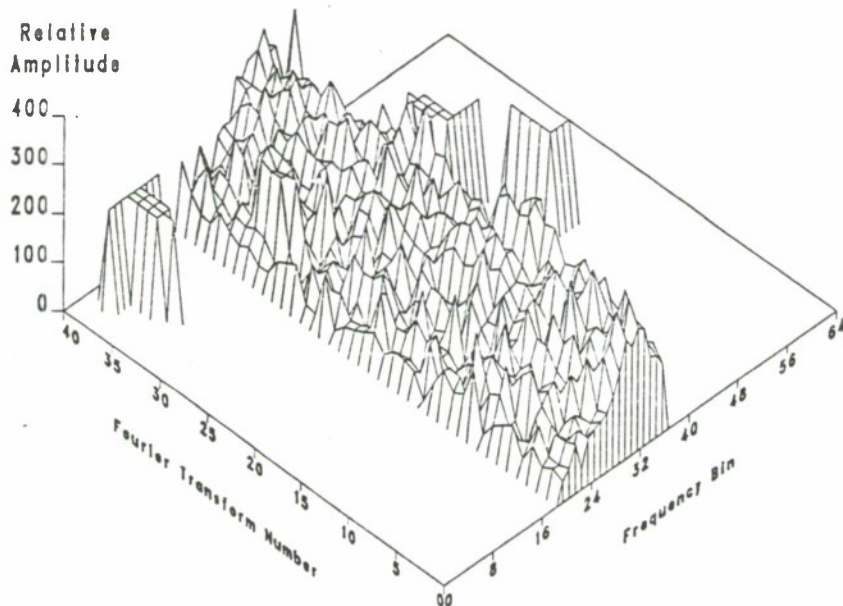


Figure 4. Constant-level signals

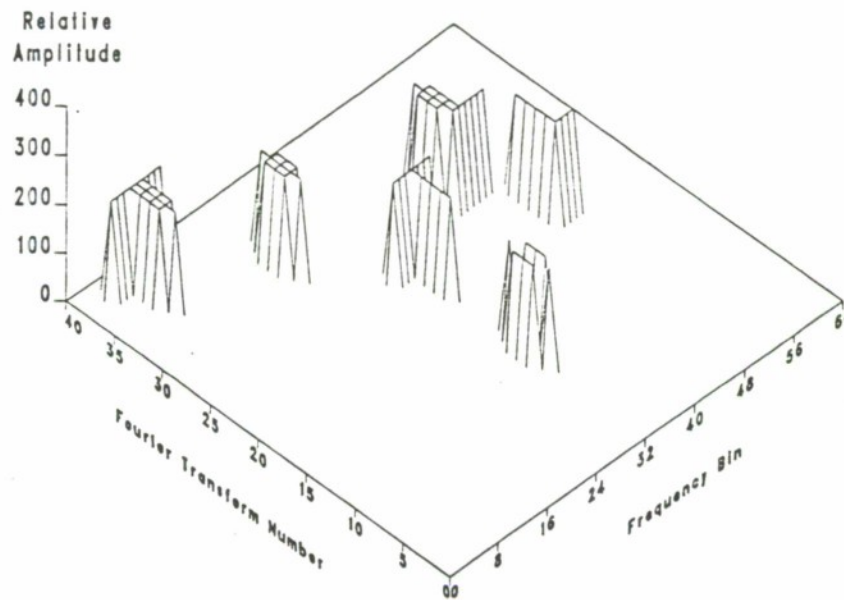


Figure 10. Signals with reverberation and boundary returns

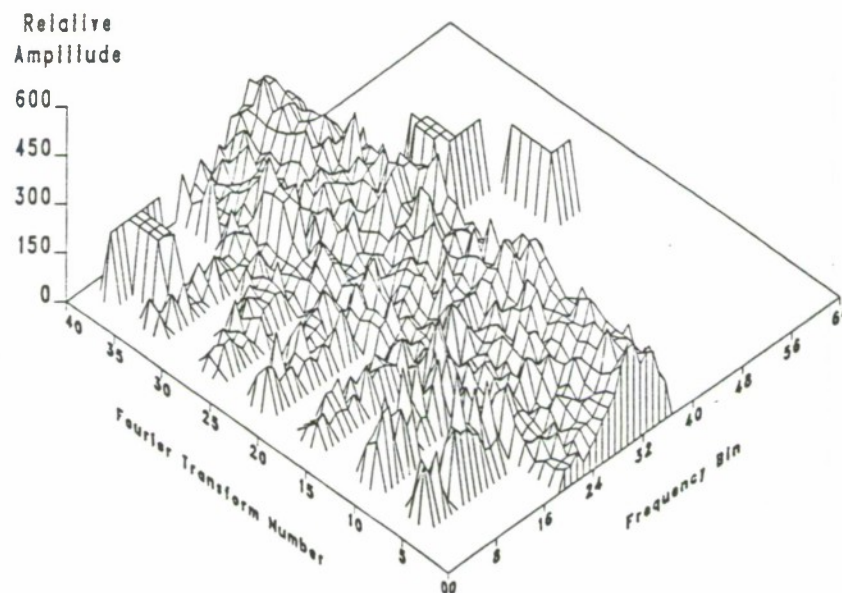


Figure 11. Signals, noise, reverberation
and boundary returns

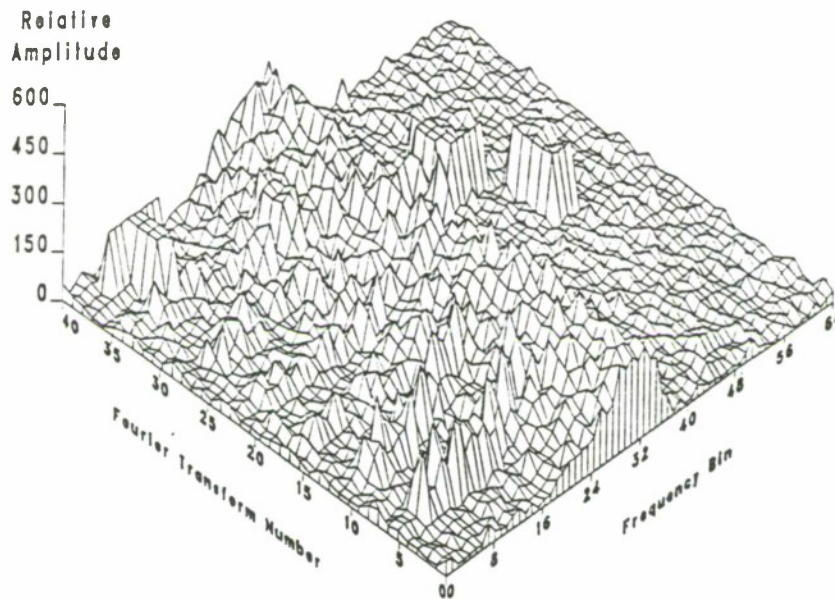
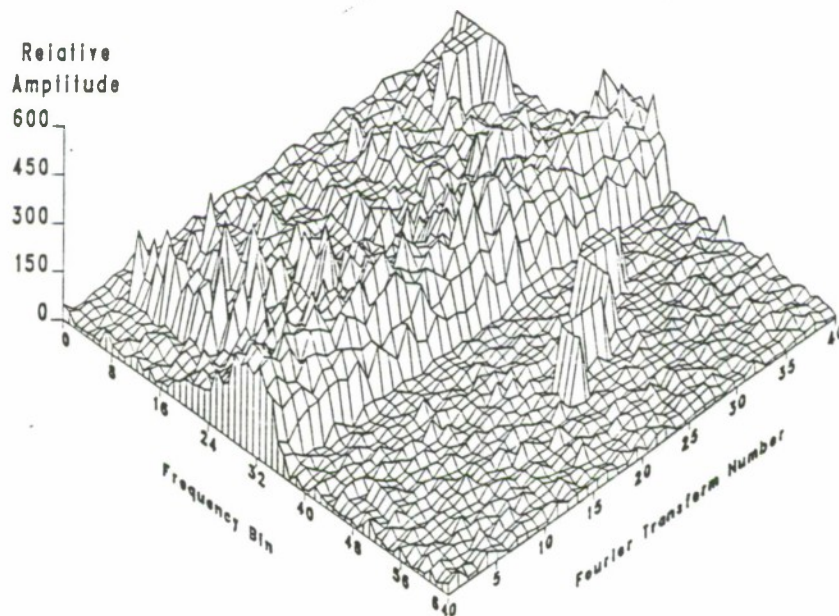


Figure 12. A 90-degree rotation of Figure 11



A second-generation map is produced by regenerating a previous map with the same basic components. The main objective in the second-generation process is to estimate the change in position of the signals after a fixed time interval. The positional change is calculated assuming straight line motion by the acoustic reflector located directly in front of the measuring platform. Signals may also be repositioned in order to eliminate overlapping and to separate from excessive interference. The second map retains the number and type of signals in the original map as well as levels of signals, noise, reverberation and boundary returns whenever appropriate. The regeneration process does, however, generate unique values for individual map cells while still obeying the same random distribution functions of the original method. The second-generation map shown in Figure 13 was regenerated using Figure 4 as the previous map. In this example, the 'fat' L, 'fat' T, 'fat' O and L signals were repositioned. (Note that the 'fat' T is no longer visible on the map, having moved beyond the upper limit of the scale.)

RENSGEN is run in an interactive LISP environment. Sample interactive terminal sessions are presented in Appendix B. The samples are records of actual data used in generation of Figure 11 and Figure 13 (Method 1 and 2, respectively). RENSGEN creates four output files during each execution, and uses one input file when the second-generation method is selected.

An input file, CENTERS.LSP, is used when a specific map is to be regenerated. This file contains all of the data input to the program during the original generation as well as identification and location of each signal present on the map.

Figure 4. Constant-level signals

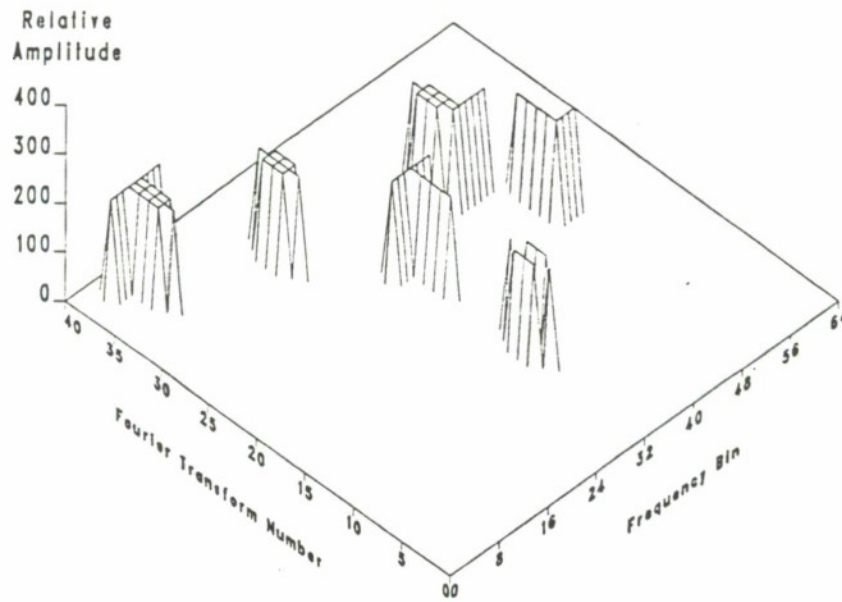
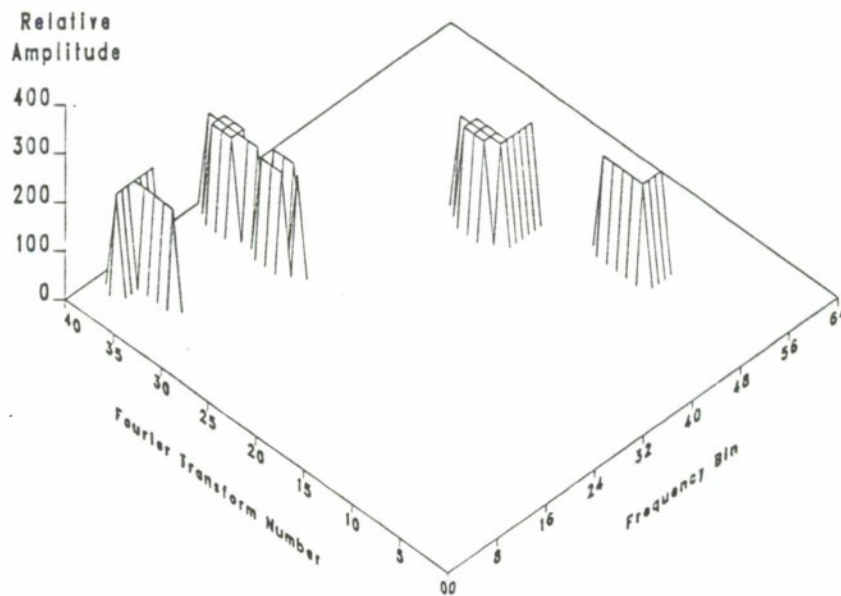


Figure 13. Repositioned signals



RENSGEN creates three files for a quick look at the resulting output, SIGNALS.LSP, READIN.LSP, and CENTERS.LSP (See Appendix C). A fourth file, PLOT3D.LSP, is also created by RENSGEN for the purpose of computer aided plotting. The file stored in SIGNALS.LSP presents either a quantized representation of the signal and/or noise level values or a representation that shows only the presence of a signal. The files stored in READIN.LSP and PLOT3D.LSP present the actual signal and/or noise level values. The READIN.LSP file is to be read and used in the RENSID.LSP program, designed to identify the signals. The PLOT3D.LSP file is to be read and used in the RDLOT.FOR program to produce three-dimensional graphs of the range-Doppler map, Appendix D. The file stored in CENTERS.LSP keeps a record of the information input into the program to be used by the program if a certain map is to be regenerated with specified signals repositioned.

Additional information concerning the implementation and use of RENSGEN is included in Appendices E, F, G, and H. Appendix E provides instructions for using the VAX/LISP. Appendix F outlines some interesting LISP functions used in the implementation of RENSGEN. Appendix G lists the input data used in generation of Figures 1 through 13. Appendix H contains references used in writing RENSGEN.

APPENDIX A

RENSGEN LISP CODE

APPENDIX A

LISP Program: RENSGEN.LSP

AUTHORS: Joe Wakeley, ARL/PSU
and
Jodi E. Sentz, MATH/PSU

REVISED: 27 February 1986

- REFERENCES: (1) Wilensky, Robert, "LISPCRAFT," University of California, Berkely, W.W. Norton & Company, 1984.
(2) Winston, Patrick H. and Born, Berthold K. P., "LISP," Massachusetts Institute of Technology, Addison-Wesley Publishing Company, 1981.
(3) Steele, Guy L., "Common LISP - The Language," Digital Press, 1984.

```

////////////////////////////////////
(DEFUN REVERBERATION ()
  (SETQ LDOK (DPEN "SIGNALS.LSP,1" :DIRECTION :OUTPUT :IF-EXISTS :NEW-VERSION))
  (SETQ REVIDEN (DPEN "READIN.LSP,1" :DIRECTION :DUTPUT :IF-EXISTS :NEW-VERSION))
  (SETQ PLDT3D (DPEN "PLDT3D.LSP,1" :DIRECTION :DUTPUT :IF-EXISTS :NEW-VERSION))

  (WRITE-CHAR #\NEWLINE LDDK) (WRITE-CHAR #\NEWLINE LDDK)
  (WRITE-CHAR #\NEWLINE LDDK) (WRITE-CHAR #\NEWLINE LOOK)
  (WRITE-CHAR #\NEWLINE LOOK) (WRITE-CHAR #\NEWLINE LDDK)
  (WRITE-CHAR #\NEWLINE LOOK) (WRITE-CHAR #\NEWLINE LOOK)

  (RESET)
  (INITIAL)

  (TERPRI)
  (PRINC "Would you like a new (N) range-Doppler map or")
  (PRINC " will this be a second (S) generation map? [N/S] ")

  (COND ((EQUAL (READ) 'S) (REGEN))
    (T (PRDMPT)))

  (WRITE-VALUE "SIGNAL MEAN : " SIGS_PLACE SIG_MU)
  (WRITE-VALUE "REVERBERATION MEAN LEVEL : " REV_WEIGHT_NDISE REV_WN_MU)
  (WRITE-VALUE "FLAT NDISE MEAN : " FLAT_NDISE FN_MU)

  (COND ((EQUAL QUANT 1)
    (WRITE_Q_VALUE "QUANTIZED SIGNAL MEAN : " SIGS_PLACE SIG_MU)
    (WRITE_Q_VALUE "QUANTIZED REVERBERATION MEAN LEVEL : "
      REV_WEIGHT_NDISE REV_WN_MU)
    (WRITE_Q_VALUE "QUANTIZED FLAT NDISE MEAN : " FLAT_NDISE FN_MU)))

  (SET_SIGS SET_NUM_FL SET_NUM_FD SET_NUM_FT
    SET_NUM_L SET_NUM_O SET_NUM_T SET_NUM_B)

  (BDUND_WRITE MIDDLE BDUND_GEN)
  (WRITE CEN_SIG :STREAM MIDDLE)

  (CLDSE LDDK)
  (CLDSE REVIDEN)
  (CLDSE MIDDLE)
  (CLOSE PLDT3D))

  FUNCTION TO READ IN DATA, INITIALIZE,
  RUN RENSGEN, AND SET UP DUTPUT FILE

  OPEN OUTPUT FILE LOOK FOR
  QUANTIZED SIGNAL VALUES
  OPEN OUTPUT FILE REVIDEN FOR
  SIGNAL VALUES TO BE USED IN REVID
  OPEN DUTPUT FILE PLDT3D FOR PLDTTNG
  SIGNAL VALUES WITH REVPLT

  PUT BLANK LINES IN DUTPUT FILE LDDK
  SD DUTPUT WILL BE CENTERED
  DN PAGE

  CALL FUNCTION TO INITIAITE VARIABLES
  CALL FUNCTION TO INITIALIZE IN_LINE_FT

  PRDMPT TO SET TYPE OF MAP

  IF REPEAT, REGENERATE MAP
  OR
  IF NEW, PRDVIDE PROMPTS

  WRITE SIGNAL MEAN TO FILE REVIDEN
  WRITE REVERB MEAN LEVEL TO FILE REVIDEN
  WRITE FLAT NDISE MEAN TO FILE REVIDEN

  WRITE QUANTIZED SIGNAL MEAN
  TO FILE LDDK

  WRITE QUANTIZED REVERBERATION
  MEAN LEVEL TO FILE LOOK

  WRITE QUANTIZED WEIGRTED NDISE
  MEAN TO FILE LDDK

  CALL SET_SIGS FUNCTION TO
  EXECUTE PROGRAM

  CALL BOUND WRITE FUNCTION TO
  WRITE TO FILE MIDDLE
  WRITE CEN_SIG TO FILE MIDDLE

  CLDSE DUTPUT FILE LOOK
  CLDSE DUTPUT FILE REVIDEN
  CLOSE DUTPUT FILE MIDDLE (DPENED IN REGEN)
  CLDSE OUTPUT FILE PLDT3D
  
```

```

=====
INITIAL CONDITIONS I

(DEFUN INITIAL ()
  (PROG (GENER KOUNT COUNT)
    (SETQ GENER ())
    (SETQ KOUNT 0)
    (SETQ COUNT 0)
    (SETQ IN_LINE_FT ())

    LOOP1
      (SETQ COUNT 0)
      (SETQ KOUNT (+ KOUNT 1))
      (SETQ ROS (* KOUNT 100))
      (GENSYM ROS)

      (COND ((> KOUNT 40)
        (SETQ IN_LINE_FT (REVERSE IN_LINE_FT))
        (RETURN IN_LINE_FT))
        (T (GO LOOP1)))

    LOOP2
      (SETQ COUNT (+ COUNT 1))
      (SETQ GENER (GENSYM "C"))
      (SETQ IN_LINE_FT (CONS GENER IN_LINE_FT))

      (COND ((> COUNT 63) (GO LOOP1))
        (T (GO LOOP2))))

    ; FUNCTION TO ADDRESS 40 CONSECUTIVE 64 POINT
    ; FOURIER TRANSFORMS CALLED IN_LINE_FT

    ; LOOP TO SET UP ROW NUMBERS
    ; RESET COLUMN COUNTER
    ; INCREMENT ROW COUNTER
    ; SET ROS TO 100 * NUMBER OF ROWS
    ; START GENERATOR AT VALUE OF ROS
    ; IF ALL 40 ROWS GENERATED,
    ; REVERSE IN_LINE_FT
    ; RETURN
    ; OTHERWISE, CONTINUE AT LOOP2

    ; INCREMENT COLUMN COUNTER
    ; GENERATE A VALUE WITH THE PREFIX "C"
    ; ADD THE VALUE TO IN_LINE_FT
    ; IF ALL 64 ARE DONE, GO TO LOOP1
    ; IF NOT, REPEAT
=====

```

```

=====
INITIAL CONDITIONS II

(DEFUN RESET ()
  (SETQ SIG_FL '())
  (SETQ SIG_FO '())
  (SETQ SIG_FT '())
  (SETQ SIG_L '())
  (SETQ SIG_O '())
  (SETQ SIG_T '())
  (SETQ SIG_B '())

  (SETQ LOT_SIGS '())
  (SETQ ADD_SIGS '())

  (SETQ CEN_SIG '())

  (SETQ MID_SIG '())
  (SETQ NUMBER 1)

  (SETQ YES_STARS 9)
  (SETQ NO_STARS 1)

  (SETQ OUTPUT_LIST ())
  (SETQ OUTPUT_FILE ())
  (SETQ OUTPUT_FILE1 ())

  (SETQ COUNT 0)
  (SETQ KOUNT 0)

  (SETQ SET_NUM_FL 0)
  (SETQ SET_NUM_FO 0)
  (SETQ SET_NUM_FT 0)
  (SETQ SET_NUM_L 0)
  (SETQ SET_NUM_O 0)
  (SETQ SET_NUM_T 0)
  (SETQ SET_NUM_B 0)

  (SETQ SIGS_PLACE 0)
  (SETQ REV_WEIGHT_NOISE 0)
  (SETQ FLAT_NOISE 0)
  (SETQ BOUND_FLAG 0)

  (SETQ SIG_MU 0)
  (SETQ REV_WN_MU 0)
  (SETQ REV_ALPBA 0)
  (SETQ REV_ALPBA2 0)
  (SETQ FN_RU 0)
  (SETQ BD_VAL_SQ 0)

  (SETQ QUANT 0)

  ; FUNCTION TO RESET VARIABLES TO RUN
  ; THE PROGRAM

  ; 'FAT' L SIGNALS
  ; 'FAT' O SIGNALS
  ; 'FAT' T SIGNALS
  ; L SIGNALS
  ; O SIGNALS
  ; T SIGNALS
  ; B SIGNALS

  ; ALL SIGNALS
  ; ADDITIONAL SIGNALS REPOSITIONED

  ; LIST OF SIGNAL SHAPES AND CENTERS

  ; NEWLY GENERATED SIGNAL AND CENTER
  ; NUMBER OF SIGNALS TO REPOSITION

  ; ELEMENT FOR OUTPUT FILE
  ; ELEMENT FOR OUTPUT FILE

  ; OUTPUT LIST FOR FILE LOOK
  ; OUTPUT LIST FOR FILE REVIDEN
  ; OUTPUT LIST FIR FILE PLOT3D

  ; COUNTER FOR OUTPUT FILE LOOK
  ; COUNTER FOR OUTPUT FILE REVIDEN

  ; VARIABLES TO READ IN
  ; DESIRED NUMBER OF SIGNALS
  ; OF EACH TYPE

  ; FLAG TO PLACE SIGNALS
  ; FLAG TO PLACE WEIGHTED NOISE
  ; FLAG TO PLACE NOISE
  ; FLAG TO PLACE BOUNDARY CONDITIONS

  ; VALUE FOR SIGNAL MEAN
  ; VALUE FOR REVERB WEIGHTED NOISE MEAN
  ; VALUE FOR SHAPE OF REVERB WEIGHT NOISE
  ; VALUE FOR REVERB RANDOM COMPONENT
  ; VALUE FOR FLAT NOISE MEAN
  ; VALUE FOR BOUNDARY RETURN

  ; FLAG TO QUANTIZE
=====

```

```

.....
(DEFUN WRITE_VALUE (TO_WRITE ID_FLAG IVALUE)                                ; FUNCTION TO WRITE HEADING AND VALUES
                                          ; TO FILE REVIDEN

( WRITE TD_WRITE :STREAM REVIDEN)                                           ; WRITE HEADING TO REVIDEN

(COMD ((EQUAL ID_FLAG 1)                                                     ; WRITE VALUE TO REVIDEN
      (WRITE XVVALUE :STREAM REVIDEN))

      (T (WRITE '0 :STREAM REVIDEN)))

( WRITE-CHAR #\NEWLINE REVIDEN))                                           ; SKIP A LINE
.....

.....
(DEFUN WRITE_Q_VALUE (TD_WRITE ID_FLAG VALUE_MU)                          ; FUNCTION TO WRITE HEADING AND VALUES
                                          ; TO FILE LDDK

(CMDND ((EQUAL ID_FLAG 1) (WRITE TD_WRITE :STREAM LOOK)
      (SETQ QXVALUE (+ (TRUNCATE (/ VALUE_MU 32)) 1))
      (WRITE QXVALUE :STREAM LDDK)
      (WRITE-CHAR #\NEWLINE LDDK))))

```

```

(DEFUN REGEN ( )
  (MAKE-PATHNAME :VERSIOND :NEWEST) (SETQ INPUT (DPEN "CENTERS.LSP"))
  (SETQ SIGS_PLACE (READ INPUT))
  (CDMD ((EQUAL SIGS_PLACE 1)
    (SETQ SET_NUM_FL (READ INPUT))
    (SETQ SET_NUM_FD (READ INPUT))
    (SETQ SET_NUM_FT (READ INPUT))
    (SETQ SET_NUM_L (READ INPUT))
    (SETQ SET_NUM_O (READ INPUT))
    (SETQ SET_NUM_T (READ INPUT))
    (SETQ SET_NUM_H (READ INPUT))
    (SETQ SIG_MU (READ INPUT))
    (SETQ RAM_SIG_LEV (READ INPUT)))))
  (SETQ FLAT_NOISE (READ INPUT))
  (SETQ FN_MU (READ INPUT))
  (SETQ REV_WEIGHT_NOISE (READ INPUT))
  (SETQ REV_WN_MU (READ INPUT))
  (SETQ REV_ALPHA (READ INPUT))
  (SETQ REV_ALPHA2 (READ INPUT))
  (SETQ QUANT (READ INPUT))
  (BOUND READ INPUT)
  (SETQ CEN_SIG (READ INPUT))
  (CLDSE INPUT)

  (SETQ MIDDLE (OPEN "CENTERS.LSP;1" :DIRECTIDW :DUTPUT :IF-EXISTS :NEW-VERSIOND))
  (WRITE SIGS_PLACE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (CDMD ((EQUAL SIGS_PLACE 1)
    (WRITE SET_NUM_FL :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_FD :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_FT :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_L :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_O :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_T :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SET_NUM_H :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE SIG_MU :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE RAM_SIG_LEV :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)))
  (WRITE FLAT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE FN_MU :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE REV_WEIGHT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE REV_WN_MU :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE REV_ALPHA :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE REV_ALPHA2 :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (WRITE QUANT :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)

  (REPOSITIOND SIGNAL_PRDMPT))
  FUNCTIOND TO REGENERATE SIGNALS FROM
  ANDTHER MAP
  READ DATA FROM FILE CONSTRUCTED IN PRDMPT
  FUNCTIOND DURING PREVIOUS RUN OF PRDGRAM
  CALL BOUND_READ TO PROMPT FOR
  BOUNDARY RETURNS
  CLDSE FILE FOR INPUT
  DPEN DUTPUT FILE MIDDLE TO STDRE
  REGENERATIDW INFORMATION
  WRITE DATA BACK INTO FILE
  CALL FUNCTIOND TO PRDMPT FOR REPOSITIOND

```

```

=====
(DEFUN REPOSITION_SIGNAL_PROMPT ()
  (CONO ((EQUAL SIGS_PLACE 1)
    (PRINC "The last range Ooppler map had these signals and center values: ")
    (TERPRI)
    (PRINC CEN_SIG)
    (TERPRI)
    (PRINC "How many of these signals are to be repositioned? ")
    (SETQ NUMBER (READ)))
  (GENERATE NUMBER))
  (SET 'CEN_SIG MIO_SIG))
=====

```

FUNCTION TO PROMPT FOR
SIGNALS TO REPOSITION

PROMPT TO READ NUMBER OF SIGNALS TO
BE REPOSITIONED

CALL TO FUNCTION GENERATE
ADD REGENERATED SIGNALS AND CENTER VALUES
TO CEN_SIG

```

=====
(DEFUN GENERATE (NUMB)
  (PROG (ANS)
    LOOP
      (COND ((EQUAL NUMB 0) (RETURN NUMB)))
      (T
        (SETQ PARAM (ASK_REGEN))
        (SETQ ANS (CAR PARAM))
        (SETQ TIME_CHANGE (CDR PARAM))
        (CONO ((EQUAL (CAR ANS) 'FL)
          (SETQ SET_NUM_FL (- SET_NUM_FL 1))
          (UP_DATE SET_NUM_FL 'PLACE_SIG_FL 'FL))
          ((EQUAL (CAR ANS) 'FO)
          (SETQ SET_NUM_FO (- SET_NUM_FO 1))
          (UP_DATE SET_NUM_FO 'PLACE_SIG_FO 'FO))
          ((EQUAL (CAR ANS) 'FT)
          (SETQ SET_NUM_FT (- SET_NUM_FT 1))
          (UP_DATE SET_NUM_FT 'PLACE_SIG_FT 'FT))
          ((EQUAL (CAR ANS) 'L)
          (SETQ SET_NUM_L (- SET_NUM_L 1))
          (UP_DATE SET_NUM_L 'PLACE_SIG_L 'L))
          ((EQUAL (CAR ANS) 'O)
          (SETQ SET_NUM_O (- SET_NUM_O 1))
          (UP_DATE SET_NUM_O 'PLACE_SIG_O 'O))
          ((EQUAL (CAR ANS) 'T)
          (SETQ SET_NUM_T (- SET_NUM_T 1))
          (UP_DATE SET_NUM_T 'PLACE_SIG_T 'T))
          ((EQUAL (CAR ANS) 'B)
          (SETQ SET_NUM_B (- SET_NUM_B 1))
          (UP_DATE SET_NUM_B 'PLACE_SIG_B 'B))
          (T
            (SETQ NEW ())
            (SETQ NUMB (+ NUMB 1)))
          (SETQ MIO_SIG (CONS NEW MID_SIG))
          (SETQ NUMB (- NUMB 1))
          (GO LOOP))))
  )
=====

```

FUNCTION TO GENERATE NEW POSITIONS
FOR SELECTED SIGNALS

WHEN ALL SIGNALS REPOSITIONED, QUIT

SET UP GEOMETRY RELATIVE TO PREVIOUS
RANGE OOPPLER MAP AND RETURN DESIRED
SIGNAL TO REPOSITION AND TIME CHANGE
GET ANSWERS FROM REGEN
QUERY TO REPOSITION SIGNALS

IF SIGNAL TO BE REPOSITIONED IS A 'FAT' L,
TAKE 1 FROM FL COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS A 'FAT' O,
TAKE 1 FROM FO COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS A 'FAT' T,
TAKE 1 FROM FT COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS AN L,
TAKE 1 FROM L COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS AN O,
TAKE 1 FROM O COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS A T,
TAKE 1 FROM T COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS A BAR,
TAKE 1 FROM B COUNTER
CALL UPDATE FUNCTION TO ADD NEW SIGNAL

IF SIGNAL TO BE REPOSITIONED IS INCORRECT,
SET NEW TO NIL
ADJUST COUNTER TO ITERATE AGAIN

ADD NEW SIGNAL AND CENTER VALUE
DECREASE NUMBER OF SIGNALS TO REPOSITION

```

=====
(DEFUN ASK_REGEN ()
  (TERPRI)
  (PRINC " Enter a signal to reposition,")
  (PRINC " as it appears above. ")
  (TERPRI)
  (SETQ ANS (READ))
  (PRINC " Enter the desired time change (sec).")
  (SETQ TIME_CHANGE (READ))
  (SETQ CENTER (CADR ANS))
  (SETQ Y (TRUNCATE (/ CENTER 64)))
  (SETQ X (- CENTER (* Y 64)))
  (SETQ Y (- Y (TRUNCATE (/ (* TIME_CHANGE (- X 32)) 18.75))))
  (SETQ NEW_CENTER (+ (* 64 Y) X))
  (SETQ NEW_CENTER_B (TRUNCATE (/ NEW_CENTER 64)))
  (CONS ANS TIME_CHANGE))
  ; FUNCTION TO PROMPT FOR SIGNAL AND TIME
  ; CHANGE, AND FIND NEW COORDINATES
  ; PROMPT FOR SIGNAL TO BE REPOSITIONED
  ; READ ANSWER
  ; ENTER ELAPSED TIME SINCE PRECEDING
  ; RANGE DOPPLER MAP
  ; SET CENTER VALUE
  ; SET Y COORDINATE
  ; SET X COORDINATE
  ; COMPUTE NEW Y COORDINATE
  ; COMPUTE NEW CENTER FOR SIGNALS
  ; COMPUTE NEW CENTER FOR BAR
  ; RETURN VALUES OF ANS AND TIME_CHANGE
=====

(DEFUN UP_OATE (SET_NUM_ID PLACE_SIG_ID ID)
  (COND ((EQUAL ID 'H)
    (SETQ ID_CENTER NEW_CENTER_H))
    (T
    (SETQ IO_CENTER NEW_CENTER)))
  (SETQ AOO_SIGS (APPEND (FUNCALL PLACE_SIG_ID IO_CENTER) AOO_SIGS))
  (SETQ NEW (CONS ID (CONS IO_CENTER '()))))
  ; FUNCTION TO ADD REPOSITIONED SIGNALS
  ; ADJUSTMENT FOR BAR SIGNAL
  ; ADD NEW SIGNAL TO AOO_SIGS
  ; ADD LISTING OF NEW SIGNAL TO NEW
=====

```

```

=====
(DEFUN PROMPT ()
  (SETQ MIDDLE (OPEN "CENTERS.LSP" :DIRECTION :OUTPUT :IF-EXISTS :NEW-VERSION))
  (TERPRI)
  (PRINC "Are signals desired on the range Doppler map? [Y/N] ")
  (COND ((EQUAL 'Y (READ)) (ASK_SIG))
    (T
    (WRITE SIGS_PLACE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)))
  (TERPRI)
  (PRINC "Is noise desired on the range Ooppler map? [Y/N] ")
  (COND ((EQUAL 'Y (READ)) (SETQ FLAT_NOISE 1) (ASK_N))
    (T
    (WRITE FLAT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE FN_MU :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)))
  (TERPRI)
  (PRINC "Is reverberation desired on the range Ooppler map? [Y/N] ")
  (COND ((EQUAL 'Y (READ)) (SETQ REV_WEIGHT_NOISE 1) (ASK_REV))
    (T
    (WRITE REV_WEIGHT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE REV_WN_MU :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE REV_ALPHA :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
    (WRITE REV_ALPHA2 :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)))
  (BOUND_PROMPT_MOD)
  (TERPRI)
  (PRINC "Is a quantized representation of values (Q) or a representation")
  (PRINC " that shows only the presence of a signal (P) desired? [D/P] ")
  (COND ((EQUAL 'Q (READ)) (SETQ DUANT 1))
    (T
    (WRITE QUANT :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)))
  ; FUNCTION TO PROMPT FOR DESIRED TYPE OF MAP
  ; OPEN OUTPUT FILE MIDDLE
  ; PROMPT TO INCLUDE SIGNALS
  ; IF SIGNALS DESIRED, CALL ASK
  ; WRITE SIGS_PLACE TO FILE MIDDLE
  ; PROMPT TO INCLUDE NOISE
  ; WRITE FLAT NOISE FLAG TO FILE MIDDLE
  ; WRITE FLAT NOISE MEAN TO FILE MIDDLE
  ; PROMPT TO INCLUDE REVERBERATION
  ; WRITE REVERB WEIGHTED NOISE FLAG TO
  ; FILE MIDDLE
  ; WRITE REVERB WEIGHTED NOISE MEAN TO
  ; FILE MIDDLE
  ; WRITE REVERB ALPHA TO FILE MIDDLE
  ; WRITE REVERB ALPHA2 TO FILE MIDDLE
  ; CALL BOUND_PROMPT_MOD TO PROMPT
  ; FOR BOUNDARY RETURNS
  ; PROMPT FOR TYPE OF GRAPH DESIRED
  ; WRITE DUANT TO FILE MIDDLE
=====

```

```

////////////////////////////////////
(OEFUN ASK_SIG ()
; FUNCTION TO ASK FOR DESIRED NUMBER
; OF EACH TYPE OF SIGNAL
;
; SET FLAG FOR SIGNALS
; WRITE SIGS_PLACE TO FILE MIOOLE
; PROMPT FOR DESIRED NUMBER OF EACH
; TYPE OF SIGNAL
;
(SETQ SIGS_PLACE 1)
(WRITE SIGS_PLACE :STREAM MIOOLE) (WRITE-CHAR #\NEWLINE MIDDLE)
(TERPRI)
(PRINC "Enter the number of 'fat' L signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_FL (READ))
(MIN_MAX_WRITE SET_NUM_FL 0 12 1)
(TERPRI)
(PRINC "Enter the number of 'fat' O signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_FO (READ))
(MIN_MAX_WRITE SET_NUM_FO 0 12 1)
(TERPRI)
(PRINC "Enter the number of 'fat' T signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_FT (READ))
(MIN_MAX_WRITE SET_NUM_FT 0 12 1)
(TERPRI)
(PRINC "Enter the number of L signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_L (READ))
(MIN_MAX_WRITE SET_NUM_L 0 12 1)
(TERPRI)
(PRINC "Enter the number of O signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_O (READ))
(MIN_MAX_WRITE SET_NUM_O 0 12 1)
(TERPRI)
(PRINC "Enter the number of T signals desired (MAX 12) ... ")
;
(SETQ SET_NUM_T (READ))
(MIN_MAX_WRITE SET_NUM_T 0 12 1)
(TERPRI)
(PRINC "Enter the number of bsrs desired (MAX 12) ... ")
;
(SETQ SET_NUM_B (READ))
(MIN_MAX_WRITE SET_NUM_B 0 12 1)
(TERPRI)
(PRINC "Enter the desired mean for ")
(PRINC "the level of these signals (MAX 255) ... ")
;
(SETQ SIG_MU (READ))
(MIN_MAX_WRITE SIG_MU 1 255 100)
;
(TERPRI)
(PRINC "Are constant level signals (C) or ")
(PRINC "signal level that vary randomly (R) desired? [C/R] ")
;
(SETQ RAN_SIG_LEV (READ))
(WRITE RAN_SIG_LEV :STREAM MIOOLE) (WRITE-CHAR #\NEWLINE MIOOLE))
////////////////////////////////////

```

```

////////////////////////////////////
(OEFUN MIN_MAX_WRITE (SET_NUM_ID MIN_VAL MAX_VAL DEF_VAL)
; FUNCTION TO CHECK BOUNDS OF VALUES
; AND WRITE THE VALUES TO FILE MIDDLE
;
; SET MINIMUM NUMBER OF SIGNALS MIN_VAL
; SET MAXIMUM NUMBER OF SIGNALS MAX_VAL
; SET DEFAULT VALUE
;
; WRITE NUMBER OF SIGNALS TO FILE MIDDLE
;
(COND ((NUMBERP SET_NUM_ID)
(COND ((< SET_NUM_ID MIN_VAL) (SETQ SET_NUM_ID MIN_VAL)))
(COND ((> SET_NUM_ID MAX_VAL) (SETQ SET_NUM_ID MAX_VAL))))
(T (SETQ SET_NUM_ID DEF_VAL)))
(WRITE SET_NUM_ID :STREAM MIOOLE) (WRITE-CHAR #\NEWLINE MIOOLE))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN ASK_N ()
  (WRITE FLAT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (PRINC "Enter the desired mean for ")
  (PRINC "the level of the noise (MAX 255) ... ")
  (SETQ FN_MU (READ))
  (MIN_MAX_WRITE FN_MU 1 255 20))
  ; FUNCTION TO ASK FOR MEAN
  ; OF NOISE LEVEL
  ; WRITE FLAT_NOISE TO FILE MIDDLE
  ; PROMPT FOR MEAN OF NOISE
  ; CALL MIN_MAX_WRITE TO CHECK RANGES
////////////////////////////////////

(DEFUN ASK_REV ()
  (WRITE REV_WEIGHT_NOISE :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE)
  (PRINC "Enter the desired mean for ")
  (PRINC "the level of the reverberation (MAX 255) ... ")
  (SETQ REV_WN_MU (READ))
  (MIN_MAX_WRITE REV_WN_MU 1 255 200)
  (TERPRI)
  (PRINC "Enter the desired value to determine ")
  (PRINC "the shape of the reverberation (MAX 64) ... ")
  (SETQ REV_ALPHA (READ))
  (MIN_MAX_WRITE REV_ALPHA 1 64 5)
  (TERPRI)
  (PRINC "Enter the desired mean for ")
  (PRINC "the random component of the reverberation (MAX 255) ... ")
  (SETQ REV_ALPHA2 (READ))
  (MIN_MAX_WRITE REV_ALPHA2 1 255 50))
  ; FUNCTION TO ASK FOR REVERBERATION
  ; PARAMETERS
  ; WRITE REV_WEIGHT_NOISE TO FILE MIDDLE
  ; PROMPT FOR REVERBERATION MEAN
  ; CALL MIN_MAX_WRITE TO CHECK RANGES
  ; PROMPT FOR REVERBERATION SHAPE
  ; CALL MIN_MAX_WRITE TO CHECK RANGES
  ; PROMPT FOR NOISE MEAN
  ; CALL MIN_MAX_WRITE TO CHECK RANGES
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN SET_SIGS (SET_SIG_FL SET_SIG_FO SET_SIG_FT SET_SIG_L SET_SIG_O SET_SIG_T SET_SIG_H)
  (PICK_SIG_FL SET_SIG_FL)
  (PICK_SIG_FO SET_SIG_FO)
  (PICK_SIG_FT SET_SIG_FT)
  (PICK_SIG_L SET_SIG_L)
  (PICK_SIG_O SET_SIG_O)
  (PICK_SIG_T SET_SIG_T)
  (PICK_SIG_H SET_SIG_H)
  (SETQ LOT_SIGS (APPEND SIG_FL SIG_FO SIG_FT SIG_L SIG_O SIG_T SIG_H ADO_SIGS))
  (DELETE NIL LOT_SIGS)
  (CHECKEO LOT_SIGS)
  (TEST_RANGE_DOPPLER_MAP))
  ; FUNCTION TO EXECUTE SIGNAL GENERATION
  ; CALL FUNCTION TO PICK THE RANDOM
  ; SIGNAL POSITIONS
  ; PUT THE SIGNALS ALL IN ONE LIST
  ; DELETE ANY NILS FROM SIGNAL LIST
  ; CALL FUNCTION TO CHECK FOR
  ; DUPLICATE POINTS IN SIGNAL LIST
  ; CALL FUNCTION TO DISPLAY SIGNALS
  ; ON MAP
////////////////////////////////////

(DEFUN CHECKED (CHECK_LIST)
  (PROG (TEST)
    (SETQ TEST ())
    (SETQ APPEAR ())
    LOOP
    (COND ((EQUAL CHECK_LIST ()) (RETURN CHECK_LIST))
      (T
        (SETQ PRE_LENGTH (LENGTH CHECK_LIST))
        (SETQ TEST (CAR CHECK_LIST))
        (SETQ CHECK_LIST (DELETE TEST CHECK_LIST))
        (SETQ POST_LENGTH (LENGTH CHECK_LIST))
        (SETQ DIFF (- PRE_LENGTH POST_LENGTH))
        (SETQ APPEAR (CONS TEST (CONS DIFF APPEAR)))
        (GO LOOP))))))
  ; FUNCTION TO CHECK FOR DUPLICATE POINTS
  ; IN SIGNAL LIST
  ; WHEN ENTIRE LIST IS CHECKED, QUIT
  ; FIND LENGTH OF CHECK LIST
  ; LOOK AT FIRST ELEMENT OF CHECK LIST
  ; DELETE ALL OCCURENCES OF FIRST ELEMENT
  ; FIND LENGTH OF SHORTENED CHECK_LIST
  ; DIFFERENCE IN LENGTHS OF TWO LISTS
  ; SAVE NUMBER OF OCCURENCES OF EACH ELEMENT
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_FL (NUM_SIG_FL)                                ; FUNCTION TO RANDOMLY PLACE 'FAT' L SIGNALS
  (SETQ SIG_FL (PICK_HELPER NUM_SIG_FL 'FL 'PLACE_SIG_FL)))    ; CALL PICK_HELPER FUNCTION TO PLACE SIGNALS
////////////////////////////////////

(DEFUN PLACE_SIG_FL (CENTER)                                   ; FUNCTION TO PLACE DESIGN OF 'FAT' L
  (CONS (NTH (- CENTER 128) IN_LINE_FT)                       ; .....
    (CONS (NTH (- CENTER 127) IN_LINE_FT)                     ; .....
      (CONS (NTH (- CENTER 126) IN_LINE_FT)                   ; .....
        (CONS (NTH (- CENTER 125) IN_LINE_FT)                 ; .....
          (CONS (NTH (- CENTER 124) IN_LINE_FT)               ; .....
            (CONS (NTH (- CENTER 123) IN_LINE_FT)              ; .....
              (CONS (NTH (- CENTER 65) IN_LINE_FT)             ; .....
                (CONS (NTH (- CENTER 64) IN_LINE_FT)           ; .....
                  (CONS (NTH (- CENTER 63) IN_LINE_FT)         ; .....
                    (CONS (NTH (- CENTER 1) IN_LINE_FT)         ; .....
                      (CONS (NTH CENTER IN_LINE_FT)             ; .....
                        (CONS (NTH (+ CENTER 1) IN_LINE_FT)     ; .....
                          (CONS (NTH (+ CENTER 63) IN_LINE_FT) ; .....
                            (CONS (NTH (+ CENTER 64) IN_LINE_FT) ; .....
                              (CONS (NTH (+ CENTER 65) IN_LINE_FT) ; .....
                                (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))))))))))
  )
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_HELPER (NUM_SIG_ID ID PLACE_SIG_ID)              ; FUNCTION TO RANDOMLY PLACE
  (PROG (COUNTR)                                              ; SIGNALS ON A RANGE DOPPLER MAP
    (SETQ COUNTR NUM_SIG_ID)
    (SETQ SIG_CEN ())
    (SETQ SIG_ID ())
    LOOP
      (COND ((EQUAL COUNTR 0) (RETURN SIG_ID))                ; WHEN ALL SIGNALS PLACED, QUIT
        (T
          (SETQ SIG_CEN ())                                     ; RESET SIG_CEN
          (COND ((EQUAL ID 'H)
            (SETQ SIG_CENTER (RANDOM 39)))                       ; PICK RANDOM CENTER VALUE
              (T
                (SETQ SIG_CENTER (WITHIN_HOUNDS (RANDOM 2560)))) ; WITHIN HOUNDS OF MAP
          (SETQ SIG_CEN (CONS ID (CONS SIG_CENTER SIG_CEN)))    ; LABEL TYPE OF SIGNAL WITH CENTER VALUE
          (SETQ CEN_SIG (CONS SIG_CEN CEN_SIG))                 ; ADD LABEL TO CEN_SIG LIST
          (SETQ SIG_ID (APPEND SIG_ID (FUNCALL PLACE_SIG_ID SIG_CENTER))) ; CALL PLACE_SIG * FUNCTION TO
                                                                    ; GENERATE SIGNAL AND ADD TO SIG_ID
          (SETQ COUNTR (- COUNTR 1)))                             ; DECREASE COUNTER
        )
      (GO LOOP)))
////////////////////////////////////

(DEFUN WITHIN_HOUNDS (RAND)                                   ; FUNCTION TO CHECK COLUMN POSITION OF
  (PROG ()                                                    ; SIGNAL CENTERS TO ENSURE NO SIGNAL
    LOOP                                                       ; OVERLAPS OVER 'EDGE' OF MAP
      (COND ((MEMBER (- RAND (* 64 (TRUNCATE (/ RAND 64))))
        (0 1 2 3 4 5 6 7 58 59 60 61 62 63))                  ; IF COLUMN POSITION = -6 TO 7 (MOD 64)
        (SETQ RAND (RANDOM 2560))                                ; PLACE SIGNAL CENTER IN A NEW COLUMN
        (GO LOOP))
        (T (RETURN RAND))))
  )
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_FO (NUM_SIG_FO)                                ; FUNCTION TO RANDOMLY PLACE 'FAT' O SIGNALS
  (SETO SIG_FO (PICK_HELPER NUM_SIG_FO 'FO 'PLACE_SIG_FO)))
////////////////////////////////////

(DEFUN PLACE_SIG_FO (CENTER)                                   ; FUNCTION TO PLACE DESIGN OF 'FAT' O
  (CONS (NTH (- CENTER 128) IN_LINE_FT)
    (CONS (NTH (- CENTER 65) IN_LINE_FT)
      (CONS (NTH (- CENTER 64) IN_LINE_FT)
        (CONS (NTH (- CENTER 63) IN_LINE_FT)
          (CONS (NTH (- CENTER 1) IN_LINE_FT)
            (CONS (NTH CENTER IN_LINE_FT)
              (CONS (NTH (+ CENTER 1) IN_LINE_FT)
                (CONS (NTH (+ CENTER 63) IN_LINE_FT)
                  (CONS (NTH (+ CENTER 64) IN_LINE_FT)
                    (CONS (NTH (+ CENTER 65) IN_LINE_FT)
                      (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))))))
  (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_FT (NUM_SIG_FT)                                ; FUNCTION TO RANDOMLY PLACE 'FAT' T SIGNALS
  (SETO SIG_FT (PICK_HELPER NUM_SIG_FT 'FT 'PLACE_SIG_FT)))
////////////////////////////////////

(DEFUN PLACE_SIG_FT (CENTER)                                   ; FUNCTION TO PLACE DESIGN OF 'FAT' T
  (CONS (NTH (- CENTER 128) IN_LINE_FT)
    (CONS (NTH (- CENTER 65) IN_LINE_FT)
      (CONS (NTH (- CENTER 64) IN_LINE_FT)
        (CONS (NTH (- CENTER 63) IN_LINE_FT)
          (CONS (NTH (- CENTER 1) IN_LINE_FT)
            (CONS (NTH CENTER IN_LINE_FT)
              (CONS (NTH (+ CENTER 1) IN_LINE_FT)
                (CONS (NTH (+ CENTER 63) IN_LINE_FT)
                  (CONS (NTH (+ CENTER 64) IN_LINE_FT)
                    (CONS (NTH (+ CENTER 65) IN_LINE_FT)
                      (CONS (NTH (+ CENTER 124) IN_LINE_FT)
                        (CONS (NTH (+ CENTER 125) IN_LINE_FT)
                          (CONS (NTH (+ CENTER 126) IN_LINE_FT)
                            (CONS (NTH (+ CENTER 127) IN_LINE_FT)
                              (CONS (NTH (+ CENTER 128) IN_LINE_FT)
                                (CONS (NTH (+ CENTER 129) IN_LINE_FT)
                                  (CONS (NTH (+ CENTER 130) IN_LINE_FT)
                                    (CONS (NTH (+ CENTER 131) IN_LINE_FT)
                                      (CONS (NTH (+ CENTER 132) IN_LINE_FT) ())))))))))))))))))
  (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_L (NUM_SIG_L)                                , FUNCTION TO RANDOMLY PLACE L SIGNALS
  (SETQ SIG_L (PICK_HELPER NUM_SIG_L 'L 'PLACE_SIG_L)))
////////////////////////////////////

////////////////////////////////////
(DEFUN PLACE_SIG_L (CENTER)                                  , FUNCTION TO PLACE DESIGN OF L
  (CONS (NTH (- CENTER 128) IN_LINE_FT)                      , **
    (CONS (NTH (- CENTER 127) IN_LINE_FT)                    , **
      (CONS (NTH (- CENTER 126) IN_LINE_FT)                  , **
        (CONS (NTH (- CENTER 125) IN_LINE_FT)                , *****
          (CONS (NTH (- CENTER 64) IN_LINE_FT)                ,
            (CONS (NTH CENTER IN_LINE_FT)                      , ** = CENTER
              (CONS (NTH (+ CENTER 64) IN_LINE_FT)
                (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_O (NUM_SIG_O)                                , FUNCTION TO RANDOMLY PLACE O SIGNALS
  (SETQ SIG_O (PICK_HELPER NUM_SIG_O 'O 'PLACE_SIG_O)))
////////////////////////////////////

////////////////////////////////////
(DEFUN PLACE_SIG_O (CENTER)                                  , FUNCTION TO PLACE DESIGN OF O
  (CONS (NTH (- CENTER 128) IN_LINE_FT)                      ,
    (CONS (NTH (- CENTER 65) IN_LINE_FT)                      ,
      (CONS (NTH (- CENTER 63) IN_LINE_FT)                    ,
        (CONS (NTH (- CENTER 1) IN_LINE_FT)                   ,
          (CONS (NTH (+ CENTER 1) IN_LINE_FT)                  ,
            (CONS (NTH (+ CENTER 63) IN_LINE_FT)               ,
              (CONS (NTH (+ CENTER 65) IN_LINE_FT)             ,
                (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_T (NUM_SIG_T)                                ; FUNCTION TO RANDOMLY PLACE T SIGNALS
  (SETQ SIG_T (PICK_HELPER NUM_SIG_T 'T 'PLACE_SIG_T)))
////////////////////////////////////

(DEFUN PLACE_SIG_T (CENTER)                                  ; FUNCTION TO PLACE OESIGN OF T
  (CONS (NTH (- CENTER 128) IN_LINE_FT)
    (CONS (NTH (- CENTER 64) IN_LINE_FT)
      (CONS (NTH CENTER IN_LINE_FT)
        (CONS (NTH (+ CENTER 64) IN_LINE_FT)
          (CONS (NTH (+ CENTER 128) IN_LINE_FT)
            (CONS (NTH (+ CENTER 126) IN_LINE_FT)
              (CONS (NTH (+ CENTER 127) IN_LINE_FT)
                (CONS (NTH (+ CENTER 128) IN_LINE_FT)
                  (CONS (NTH (+ CENTER 129) IN_LINE_FT)
                    (CONS (NTH (+ CENTER 130) IN_LINE_FT)
                      (CONS (NTH (+ CENTER 131) IN_LINE_FT) ())))))))))))))
  ())))))))))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN PICK_SIG_B (NUM_SIG_B)                                ; FUNCTION TO RANOOMLY PLACE BARS
  (SETQ SIG_B (PICK_HELPER NUM_SIG_B 'B 'PLACE_SIG_B)))
////////////////////////////////////

(DEFUN PLACE_SIG_B (CENTER)                                  ; FUNCTION TO PLACE OESIGN OF BAR
  (PROG (RESULT TIM)
    (SETQ TIM 0)
    (SETQ RESULT ())
    (SETQ CENTER (* CENTER 64))
    LOOP
      (CONO (> TIM 63) (RETURN RESULT))
      (T
        (SETQ RESULT (CONS (NTH (+ (- CENTER 128) TIM) IN_LINE_FT)
          (CONS (NTH (+ (- CENTER 64) TIM) IN_LINE_FT)
            (CONS (NTH (+ CENTER TIM) IN_LINE_FT)
              (CONS (NTH (+ CENTER (+ 64 TIM)) IN_LINE_FT)
                (CONS (NTR (+ CENTER (+ 128 TIM)) IN_LINE_FT) RESULT)))))))
        (SETQ TIM (+ TIM 1)))
      (GO LOOP)))
  ())))))
////////////////////////////////////

```

```

=====
(OEFUN TEST_RANGE_DOPPLER_MAP ()
  (PROG (COUNTER)
    (SETQ COUNTER 0)
    (SETQ NUMB 2497)
    (SETQ COL_NUMB 1)
    LOOP
      (SETQ COUNTER (+ COUNTER 1))
      (CONDO ((< COUNTER 65)
        (CELL_LEVEL NUMB)
        (SETQ NUMB (+ NUMB 1))
        (SETQ COL_NUMB (+ COL_NUMB 1))
        (GO LOOP))
        (T
          (SETQ NUMB (- NUMB 128))
          (SETQ COL_NUMB 1)
          (CONDO ((< NUMB 1) (RETURN NUMB))
            (T
              (SETQ COUNTER 0)
              (GO LOOP))))))
  )
)
=====
; FUNCTION TO CALL CELL_LEVEL FUNCTION
;
; INITIALIZE COUNTER
; START AT FIRST ELEMENT IN ROW 40
; SET COLUMN NUMBER TO 1
;
; LOOP TO CALL CELL_LEVEL FUNCTION
; FROM ROW 40 TO ROW 1
;
; INCREMENT COUNTER FOR COLUMN POSITION
;
; FOR ALL 64 COLUMN POSITIONS IN ONE ROW,
; CALL CELL_LEVEL
; INCREMENT NUMBER
; INCREMENT COLUMN NUMBER
; REPEAT LOOP
;
; GO TO BEGINNING OF PREVIOUS ROW
; RESET COLUMN NUMBER
; IF ALL ROWS CALLED, QUIT
;
; IF MORE ROWS TO CALL,
; RESET COUNTER FOR COLUMN POSITION
; REPEAT LOOP
=====

```

```

=====
(OEFUN CELL_LEVEL (POSIT)
  (SETQ POSIT (- POSIT 1))
  (SETQ SUM_VALUE 0)
  (SETQ TOT_SIG_VAL 0)
  (SETQ REV_W_NOISE_VALUE 0)
  (SETQ F_NOISE_VALUE 0)
  (SETQ BD_VAL_SQ 0)
  (IF_OESIREO POSIT)
  (CONDO ((EQUAL REV_WEIGHT_NOISE 1)
    (OISTRIBUTE_REV_WN REV_WN_MU)
    (SETQ REV_W_NOISE_VALUE (* REV_W_NOISE_VALUE REV_W_NOISE_VALUE)))
    ((EQUAL FLAT_NOISE 1)
    (OISTRIBUTE_FN MU)
    (SETQ F_NOISE_VALUE (* VALUE VALUE)))
    ((EQUAL BOUND_FLAG 1)
    (CALC_BO_VAL POSIT BOUNO_RET
      BOUNO_GEN BOUNO_CONS_LEV BOUNO_AVG_LEV)))
  (SETQ SUM_VALUE (+ (+ (+ TOT_SIG_VAL REV_W_NOISE_VALUE) F_NOISE_VALUE)
    BO_VAL_SQ))
  (SETQ FINAL_VALUE (TRUNCATE (SQRT SUM_VALUE)))
  (SETQ OUTPUT_FILE (CONS FINAL_VALUE OUTPUT_FILE))
  (SETQ OUTPUT_FILE1 (CONS FINAL_VALUE OUTPUT_FILE1))
  (COND (> FINAL_VALUE 256) (SETQ FINAL_VALUE 256))
  (CONDO ((EQUAL QUANT 1)
    (SETQ STARS (+ (TRUNCATE (/ FINAL_VALUE 32)) 1))
    (SETQ OUTPUT_LIST (CONS STARS OUTPUT_LIST)))
    ((EQUAL COUNT 1)
    (SETQ COUNT (+ COUNT 1))
    (SETQ KOUNT (+ KOUNT 1))
    (COND ((EQUAL COUNT 64)
      (SETQ OUTPUT_LIST (REVERSE OUTPUT_LIST))
      (WRITE OUTPUT_LIST :STREAM LOOK) (WRITE-CHAR #\NEWLINE LOOK)
      (SETQ OUTPUT_FILE1 (REVERSE OUTPUT_FILE1))
      (WRITE OUTPUT_FILE1 :STREAM PLOT3D) (WRITE-CHAR #\NEWLINE PLOT3D)
      (SETQ OUTPUT_FILE1 ())
      (SETQ COUNT 0)
      (PRINC " ")
      (SETQ OUTPUT_LIST ()))
      ((EQUAL KOUNT 32)
      (SETQ OUTPUT_FILE (REVERSE OUTPUT_FILE))
      (WRITE OUTPUT_FILE :STREAM REVIOEN) (WRITE-CHAR #\NEWLINE REVIOEN)
      (SETQ OUTPUT_FILE1 ())
      (SETQ KOUNT 0))))
  )
)
=====
; FUNCTION TO ASSIGN VALUES OF SIGNAL, NOISE
; AND REVERBERATION TO A SINGLE CELL
;
; DECREMENT POSIT TO ACCOUNT FOR STARTING POSITION OF 0
; INITIALIZE SUM VALUE
; INITIALIZE TOTAL SIGNAL VALUE
; INITIALIZE WEIGHTED NOISE VALUE
; INITIALIZE FLAT NOISE VALUE
; INITIALIZE BOUNDARY VALUES
;
; PLACE SIGNALS IF DESIRED
;
; IF WEIGHTED NOISE IS TO APPEAR,
; CALL FOR WEIGHTED OISTRIBUTION
; SQUARE THE VALUE OF REV_W_NOISE_VALUE
;
; IF FLAT NOISE IS TO APPEAR,
; CALL FOR RANDOM OISTRIBUTION
; SQUARE THE VALUE OF F_NOISE_VALUE
;
; IF BOUNDARY VALUES DESIRED,
; CALCULATE BOUNDARY VALUES
;
; ADD VALUE OF SIGNALS, REVERBERATION, NOISE,
; AND BOUNDARY RETURN
; TAKE SQUARE ROOT OF SUM OF SQUARES
; ADD ELEMENT TO OUTPUT_FILE
; ADD ELEMENT TO OUTPUT_FILE1
; MAXIMUM FINAL VALUE IS 256 FOR COMPUTATION OF
; OF QUANTIZED VALUES
;
; QUANTIZE THE VALUE = (VALUE / 32) + 1
; ADD QUANTIZED VALUE TO OUTPUT_LIST
;
; INCREMENT COUNTERS FOR OUTPUT FILES
;
; WHEN COUNTER IS AT 64,
; REVERSE OUTPUT_LIST
; WRITE LINE TO FILE LOOK AND SKIP TO NEXT LINE
; REVERSE OUTPUT_FILE1
; WRITE LINE TO FILE PLOT3D AND SKIP TO NEXT LINE
; RESET OUTPUT_FILE1
; RESET COUNTER TO 0
; SKIP TO NEXT LINE ON SCREEN
; RESET OUTPUT_LIST
;
; WHEN COUNTER IS AT 32,
; REVERSE OUTPUT_FILE
; WRITE LINE TO FILE REVIOEN AND SKIP TO NEXT LINE
; RESET OUTPUT_FILE1
; RESET COUNTER TO 0
=====

```

```

(OEFUN IF_OESIREO (RO_POS)
/ FUNCTION TO PLACE SIGNALS ON MAP
(CONO ((EQUAL SIGS_PLACE 1)
/ PLACE SIGNALS IF OESIREO
(CONO ((EQUAL (NTH RO_POS IN LINE_FT)
(CAR (MEMBER (NTH RO_POS IN LINE_FT) LOT_SIGS)))
/ CHECK IF ELEMENT IS IN SIGNAL 'POINT' LIST
(SETQ TIME_APP (CAOR (MEMBER (NTH RO_POS IN LINE_FT) APPEAR)))
/ READ HOW MANY TIMES IT APPEARS
(ITERATE)
/ CALL ITERATE FUNCTION
(CONO ((EQUAL QUANT 0)
(SETQ OUTPUT_LIST (CONS YES_STARS OUTPUT_LIST))))
/ ADD ELEMENT TO OUTPUT LIST
(PRINC *****))
/ PRINTS ** TO SCREEN IN POSITION OF SIGNAL
(T
(PRINC " ")
/ PRINTS BLANK TO SCREEN WHERE NO SIGNAL EXISTS
(CONO ((EQUAL QUANT 0)
(SETQ OUTPUT_LIST (CONS NO_STARS OUTPUT_LIST))))))
/ ADD ELEMENT TO OUTPUT_LIST
.....
(OEFUN ITERATE ()
/ FUNCTION TO RECALL OISTRIBUTE FUNCTION
/ FOR OVERLAPPING SIGNALS
(PROG (TIMES)
/ SET TIMES TO TIME_APP
(SETQ TIMES TIME_APP)
LOOP
(CONO ((EQUAL TIMES 0) (RETURN TIMES))
(T
/ IF POINT APPEARS 0 ADDITIONAL TIMES, QUIT
/ OTHERWISE,
(CONO ((EQUAL RAN_SIG_LEV 'C)
(SETQ VALUE_SIG_MU))
/ IF SIGNAL IS TO BE CONSTANT LEVEL,
/ SET SIGNAL TO SIGNAL MEAN
(T
(DISTRIBUTE SIG_MU)))
/ CALL FOR RANDOM OISTRIBUTION OF SIGNAL LEVELS
(SETQ TOT_SIG_VAL (+ (* VALUE VALUE) TOT_SIG_VAL))
/ ADD VALUE OF SIGNAL SQUAREO
(SETQ TIMES (- TIMES 1))
/ DECREASE TIMES
(GO LOOP))))
/ REPEAT LOOP
.....

```

```

(DEFUN OISTRIBUTE (MU)
  (SETQ X (* 2 (* MU MU)))
  (SETQ Y (LOG (+ (RANDOM 1.0) 0.00001)))
  (SETQ Z (* -1 (* X Y)))
  (SETQ VALUE (TRUNCATE (SQRT Z))))

  FUNCTION TO FINO A RAYLEIGH DISTRIBUTION
  X = 2 * MU * MU
  Y = LN ((RANDOM 1) + 0.00001)
  Z = - (X * Y)
  VALUE = Z ** 1/2

  =====

(DEFUN DISTRIBUTE_REV_WN (REV_WN_MU)
  (SETQ FBIN 32)
  (SETQ A (- (+ FBIN REV_ALPHA) COL_NUMB))
  (SETQ H (* (* REV_WN_MU REV_ALPHA) (SORT (EXP 1)))))
  (SETQ C (/ A (* REV_ALPHA REV_ALPHA)))
  (SETQ O (* (/ (* A A) (* 2 (* REV_ALPHA REV_ALPHA))) -1))
  (SETQ E (EXP O))
  (SETQ F (* H (* C E)))

  FUNCTION TO OISTRIBUTE REVERB WEIGHTEO NOISE
  INITIALIZE FBIN
  A = FBIN + REV_ALPHA - COL_NUMB
  H = REV_WN_MU * REV_ALPHA * (SORT E)
  C = A / (REV_ALPHA * REV_ALPHA)
  O = -1 * [(A * A) / (2 * REV_ALPHA * REV_ALPHA)]
  E = E ** O
  F = H * C * E

  (CONO ((OR (< A 0) (< F (/ REV_WN_MU 100))) (SETQ REV_W_NOISE_VALUE 0)), NO WEIGHTEO NOISE TYPE REVERBERATION VALUE FOR
  VALUES OF COL_NUMB GREATER THAN (FBIN + REV_ALPHA)

  (T
    (OISTRIBUTE REV_ALPHA2)
    (SETQ RAN_NOISE_VALUE)
    (SETQ REV_W_NOISE_VALUE (+ RAN_NOISE_VALUE F))))

  CALL OISTRIBUTE FOR RANOOM BACKGROUND NOISE
  SET RAN_NOISE_VALUE TO VALUE GENERATEO
  AOO RAN_NOISE_VALUE TO REVERB WEIGHTEO NOISE VALUE
  =====

```

```

=====
(DEFUN BOUND_PROMPT_MOD ()                                ; FUNCTION TO PROMPT FOR BOUNDARY CONDITIONS
  (TERPRI)
  (PRINC "Are boundary returns desired? [Y/N] ..... ")
  (SETQ BOUND_GEN (READ))
  (COND ((EQUAL BOUND_GEN 'Y) (BOUNDARY)                  ; IF DESIRED, GENERATE BOUNDARY RETURNS
        (SETQ BOUND_FLAG 1))
        (T (SETQ BOUND_GEN 'NO) (SETQ BOUND_FLAG 0) (SETQ BOUND_RET ()))) ; OTHERWISE, QUIT
  )
=====

(DEFUN BOUNDARY ()                                         ; FUNCTION TO PROMPT FOR BOUNDARY CONDITION VALUES
  (TERPRI) (PRINC "Platform depth (m) .....")
  (PRINC "..... ") (SETQ PLATD (READ))

  (TERPRI) (PRINC "Platform velocity (m/sec) .....")
  (PRINC "..... ") (SETQ PLATV (READ))

  (TERPRI) (PRINC "Water depth (m) .....")
  (PRINC "..... ") (SETQ WATERD (READ))

  (TERPRI) (PRINC "Average level of all boundary returns .....")
  (PRINC "..... ") (SETQ BOUND_AVG_LEV (READ))

  (TERPRI) (PRINC "Are constant boundary returns desired? [Y/N] ..")
  (PRINC "..... ") (SETQ BOUND_CONS_LEV (READ))

  (SURF_GEN PLATD PLATV)                                     ; GENERATE BOUNDARY RETURNS
  (BOTT_GEN PLATD PLATV WATERD)
  (S1B1_GEN PLATD PLATV WATERD)
  (B1S1_GEN PLATD PLATV WATERD)
  (S2B1_GEN PLATD PLATV WATERD)
  (B2S1_GEN PLATD PLATV WATERD)
  (S2B2_GEN PLATD PLATV WATERD)
  (B2S2_GEN PLATD PLATV WATERD)

  (SETQ BOUND_RET (APPEND SURF_RET BOTT_RET S1B1_RET B1S1_RET
                           S2B1_RET B2S1_RET S2B2_RET B2S2_RET)) ; COMBINE ALL COMPONENTS OF BOUNDARY RETURNS
                                                                    ; TO GET BOUND_RET VALUE
  )
=====

```

```

=====
(DEFUN SURF_GEN (DEPTH1 VELOC)                             ; FUNCTION TO PROMPT FOR AND COMPUTE SURFACE RETURNS
  (SETQ SURF_RET ())
  (TERPRI)
  (PRINC "Are surface boundary returns desired? [Y/N] ..... ")
  (SETQ SURF_RIDGE (READ))
  (COND ((EQUAL SURF_RIDGE 'Y)                             ; IF DESIRED, GENERATE SURFACE BOUNDARY RETURNS
        (PROG (BCOUNT)
              (SETQ BCOUNT 31)
              LOOP
              (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                    (RETURN SURF_RET))
                    (T
                     (SETQ DELF (* (- 32 BCOUNT) 20))
                     (SETQ ANG (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                     (SETQ STIME (/ DEPTH1 (* 750 (SIN ANG))))
                     (SETQ SURF_RET (APPEND (TRANS_SIG STIME BCOUNT)
                                              SURF_RET))
                     (SETQ BCOUNT (- BCOUNT 1))
                     (GO LOOP))))
              )
        (T (SETQ SURF_RIDGE 'NO)))
  )
=====

(DEFUN TRANS_SIG (SIG_CENT BCNT)                           ; FUNCTION TO REPRESENT TRANSMITTED SIGNAL
                                                                    ; IN BOUNDARY RETURN
  (SETQ TRANS_CELLS ())
  (SETQ STINC (* 64 (TRUNCATE (/ SIG_CENT 0.05))))
  (SETQ BCELL (+ (+ BCNT 1) STINC))
  (SETQ BCELLP (+ BCELL 64))
  (SETQ BCELLM 0)
  (SETQ TRANS_CELLS (CONS BCELLP (CONS BCELLM
                                         (CONS BCELL TRANS_CELLS))))
  (DELETE 0 TRANS_CELLS))
=====

```

```

=====
(OEFUN BOTT_GEN (OEPB1 VELOC DEPTB2)                                , FUNCTION TO PROMPT FOR AND COMPUTE BOTTOM RETURNS
(
  (SETQ BOTT_RET ())
  (TERPRI)
  (PRINC "Are bottom boundary returns desired? [Y/N] ..... ")
  (SETQ BOTT_RIDGE (READ))
  (COND ((EQUAL BOTT_RIDGE 'Y)                                     , IF DESIRED, GENERATE BOTTOM BOUNOARY RETURNS
    (PROG (BCOUNT)
      (SETQ BCOUNT 31)
      LOOP
      (CONO ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
        (RETURN BOTT_RET))
      (T
        (SETQ OELF (* (- 32 BCOUNT) 20))
        (SETQ ANG (ACOS (- 1 (/ OELF (* 10 VELOC)))))
        (SETQ STIME (/ DEPTB1 (* 750 (SIN ANG))))
        (SETQ BTIME (- (/ DEPTB2 (* 750 (SIN ANG))) STIME))
        (SETQ BOTT_RET (APPEND (TRANS SIG BTIME BCOUNT)
          (BOTT_RET)))
        (SETQ BCOUNT (- BCOUNT 1))
        (GO LOOP))))))
  (T
    (SETQ BOTT_RIDGE 'NO)))
=====

```

```

=====
(OEFUN S1B1_GEN (OEPB1 VELOC DEPTB2)                                , FUNCTION TO PROMPT FOR AND COMPUTE
, ONE SURFACE AND ONE BOTTOM RETURN
(
  (SETQ S1B1_RET ())
  (TERPRI)
  (PRINC "Are surf(1),bott(1) boundary returns desired? [Y/N] ..... ")
  (SETQ S1B1_RIDGE (READ))
  (COND ((EQUAL S1B1_RIDGE 'Y)                                     , IF DESIRED, GENERATE S1B1 BOUNOARY RETURNS
    (PROG (BCOUNT)
      (SETQ BCOUNT 31)
      LOOP
      (CONO ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
        (RETURN S1B1_RET))
      (T
        (SETQ OELF (* (- 32 BCOUNT) 20))
        (SETQ ANG1 (ACOS (- 1 (/ OELF (* 10 VELOC)))))
        (SETQ SUMO (+ DEPTB1 DEPTB2))
        (SETQ OIFO (- DEPTB2 DEPTB1))
        (SETQ ANG2 (ATAN (/ (* SUMO (TAN ANG1)) OIFO)))
        (SETQ TTN (+ (* DIFO (SIN ANG2)) (* SUMO (SIN ANG1))))
        (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
        (SETQ SB TIME (/ TTN TTD))
        (SETQ S1B1_RET (APPEND (TRANS SIG SB_TIME BCOUNT)
          (S1B1_RET)))
        (SETQ BCOUNT (- BCOUNT 1))
        (GO LOOP))))))
  (T
    (SETQ S1B1_RIDGE 'NO)))
=====

```

```

////////////////////////////////////
(DEFUN B1S1_GEN (DEPTB1 VELOC DEPTB2)
; FUNCTION TO PROMPT FOR AND COMPUTE
; ONE BOTTOM AND ONE SURFACE RETURN

(SETQ B1S1_RET ())
(TERPRI)
(PRINC "Are bott(1)surf(1) boundary returns desired? [Y/N] ..... ")
(SETQ B1S1_RIDGE (READ))
(COND ((EQUAL B1S1_RIDGE 'Y)
; IF DESIRED, GENERATE B1S1 BOUNDARY RETURNS

(PROG (BCOUNT)
(SETQ BCOUNT 31)

LOOP

(COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
(RETRN B1S1_RET))

(T

(SETQ DELF (* (- 32 BCOUNT) 20))
(SETQ ANGL (ACOS (- 1 (/ DELF (* 10 VELOC))))))
(SETQ DIF2 (- (* 2 DEPTB2) DEPTB1))
(SETQ ANG2 (ATAN (/ (* DIF2 (TAN ANGL)) DEPTB1)))
(SETQ TTN (+ (* DEPTB1 (SIN ANG2)) (* DIF2 (SIN ANGL))))
(SETQ TTD (* 1500 (* (SIN ANGL) (SIN ANG2))))
(SETQ SB_TIME (/ TTN TTD))
(SETQ B1S1_RET (APPEND (TRANS_SIG SB_TIME BCOUNT)
B1S1_RET))
(SETQ BCOUNT (- BCOUNT 1))
(GO LOOP))))))

(T
(SETQ B1S1_RIDGE 'NO)))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN S2B1_GEN (DEPTB1 VELOC DEPTB2)
; FUNCTION TO PROMPT FOR AND COMPUTE
; TWO SURFACE AND ONE BOTTOM RETURN

(SETQ S2B1_RET ())
(SETQ S2B1_RET1 ())
(SETQ S2B1_RET2 ())
(TERPRI)
(PRINC "Are surf(2)bott(1) boundary returns desired? [Y/N] ..... ")
(SETQ S2B1_RIDGE (READ))
(COND ((EQUAL S2B1_RIDGE 'Y)
; IF DESIRED, GENERATE S2B1 BOUNDARY RETURNS

(PROG (BCOUNT)
(SETQ BCOUNT 31)

LOOP

(COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
(SETQ S2B1_RET (APPEND S2B1_RET1 S2B1_RET2))
(RETRN S2B1_RET))

(T

(SETQ DELF (* (- 32 BCOUNT) 20))
(SETQ ANGL (ACOS (- 1 (/ DELF (* 10 VELOC))))))
(SETQ SUM2 (+ (* 2 DEPTB2) DEPTB1))
(SETQ ANG2 (ATAN (/ (* SUM2 (TAN ANGL)) DEPTB1)))
(SETQ TTN (+ (* DEPTB1 (SIN ANG2)) (* SUM2 (SIN ANGL))))
(SETQ TTD (* 1500 (* (SIN ANGL) (SIN ANG2))))
(SETQ SB_TIME (/ TTN TTD))
(SETQ S2B1_RET1 (APPEND (TRANS_SIG SB_TIME BCOUNT)
S2B1_RET1))

(SETQ TTN2 (* 2 (+ DEPTB1 DEPTB2)))
(SETQ TTD2 (* 1500 (SIN ANGL)))
(SETQ SB2_TIME (/ TTN2 TTD2))
(SETQ S2B1_RET2 (APPEND (TRANS_SIG SB2_TIME BCOUNT)
S2B1_RET2))

(SETQ BCOUNT (- BCOUNT 1))
(GO LOOP))))))

(T
(SETQ S2B1_RIDGE 'NO)))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN B2S1_GEN (DEPTB1 VELOC DEPTB2)
, FUNCTION TO PRDMPRT FDR AND COMPUTE
, TWO BOTTDIM AND DNE SURFACE RETURN

  (SETQ B2S1_RET ())
  (SETQ B2S1_RET1 ())
  (SETQ B2S1_RET2 ())
  (TERPRI)
  (PRINC "Are bott(2) surf(1) boundary returns desired? [Y/N] ..... ")
  (SETQ B2S1_RIDGE (READ))
  (CDND ((EQUAL B2S1_RIDGE 'Y)
, IF DESIRED, GENERATE B2S1 BOUNDARY RETURNS

    (PROG (BCDUNT)
      (SETQ BCDUNT 31)

      LOOP

      (CDND ((DR (EQUAL BCDUNT 0) (<= BCDUNT (- 32 (/ VELOC 2))))
        (SETQ B2S1_RET (APPEND B2S1_RET1 B2S1_RET2))
        (RETURN B2S1_RET))

      (T

        (SETQ DELF (* (- 32 BCDUNT) 20))
        (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
        (SETQ DIF1 (- (* 3 DEPTB2) DEPTB1))
        (SETQ DIFD (- DEPTB2 DEPTB1))
        (SETQ ANG2 (ATAN (/ (* DIF1 (TAN ANG1)) DIFD)))
        (SETQ TTN (+ (* DIFD (SIN ANG2)) (* DIF1 (SIN ANG1))))
        (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
        (SETQ SB_TIME (/ TTN TTD))
        (SETQ B2S1_RET1 (APPEND (TRANS SIG SB_TIME BCDUNT)
          B2S1_RET1))

        (SETQ TTN2 (* 2 (- (* 2 DEPTB2) DEPTB1)))
        (SETQ TTD2 (* 1500 (SIN ANG1)))
        (SETQ SB2_TIME (/ TTN2 TTD2))
        (SETQ B2S1_RET2 (APPEND (TRANS SIG SB2_TIME BCDUNT)
          B2S1_RET2))

        (SETQ BCDUNT (- BCDUNT 1))
        (GD LDDP))))))

    (T
      (SETQ B2S1_RIDGE 'ND)))
////////////////////////////////////

```

```

////////////////////////////////////
(DEFUN S2B2_GEN (DEPTB1 VELOC DEPTB2)
, FUNCTION TO PRDMPRT FDR AND COMPUTE
, TWO SURFACE AND TWO BOTTOM RETURNS

  (SETQ S2B2_RET ())
  (SETQ S2B2_RET1 ())
  (SETQ S2B2_RET2 ())
  (TERPRI)
  (PRINC "Are surf(2) bott(2) boundary returns desired? [Y/N] ..... ")
  (SETQ S2B2_RIDGE (READ))
  (CDND ((EQUAL S2B2_RIDGE 'Y)
, IF DESIRED, GENERATE S2B2 BOUNDARY RETURNS

    (PRDG (BCDUNT)
      (SETQ BCDUNT 31)

      LOOP

      (CDND ((DR (EQUAL BCDUNT 0) (<= BCDUNT (- 32 (/ VELOC 2))))
        (SETQ S2B2_RET (APPEND S2B2_RET1 S2B2_RET2))
        (RETURN S2B2_RET))

      (T

        (SETQ DELF (* (- 32 BCDUNT) 20))
        (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
        (SETQ SUM3 (+ (* 3 DEPTB2) DEPTB1))
        (SETQ DIFD (- DEPTB2 DEPTB1))
        (SETQ ANG2 (ATAN (/ (* SUM3 (TAN ANG1)) DIFD)))
        (SETQ TTN (+ (* DIFD (SIN ANG2)) (* SUM3 (SIN ANG1))))
        (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
        (SETQ SB_TIME (/ TTN TTD))
        (SETQ S2B2_RET1 (APPEND (TRANS SIG SB_TIME BCDUNT)
          S2B2_RET1))

        (SETQ SUMD2 (+ DEPTB1 DEPTB2))
        (SETQ ANG22 (ATAN (/ (* SUMD2 (TAN ANG1)) DIFD)))
        (SETQ TTN2 (+ (* DIFD (SIN ANG22)) (* SUM3 (SIN ANG1))))
        (SETQ TTD2 (* 1500 (* (SIN ANG1) (SIN ANG22))))
        (SETQ SB2_TIME (/ TTN2 TTD2))
        (SETQ S2B2_RET2 (APPEND (TRANS SIG SB2_TIME BCDUNT)
          S2B2_RET2))

        (SETQ BCDUNT (- BCDUNT 1))
        (GD LDDP))))))

    (T
      (SETQ S2B2_RIDGE 'ND)))
////////////////////////////////////

```

```

=====
DEFUN B2S2_GEN (DEPTB1 VELOC DEPTH2)
; FUNCTION TO PROMPT FOR AND COMPUTE
; TWO BOTTOM AND TWO SURFACE RETURNS

(SETQ B2S2_RET ())
(SETQ B2S2_RET1 ())
(SETQ B2S2_RET2 ())
(ITERPRI)
(PRINC "Are bott(2)surf(2) boundary returns desired? [Y/N] .... ")
(SETQ B2S2_RIDGE (READ))
(COND ((EQUAL B2S2_RIDGE 'Y)
; IF DESIRED, GENERATE B2S2 BOUNDARY RETURNS

(PROG (BCOUNT)
(SETQ BCOUNT 11)

LOOP

(COND ((OR (EQUAL BCOUNT 0) (< BCOUNT (- 32 (/ VELOC 2))))
(SETQ B2S2_RET (APPEND B2S2_RET1 B2S2_RET2))
(RETURN B2S2_RET))

(T

(SETQ DELF (* (- 32 BCOUNT) 20))
(SETQ ANGL (ACOS (- 1 (/ DELF (* 10 VELOC))))))
(SETQ DIF4 (- (* 4 DEPTB2) DEPTB1))
(SETQ DIF2 (- DEPTB2 DEPTB1))
(SETQ ANG2 (ATAN (/ (* DIF4 (TAN ANGL)) DEPTB1)))
(SETQ TTN (* (* DEPTB1 (SIN ANG2)) (* DIF4 (SIN ANGL))))
(SETQ TTD (* 1500 (* (SIN ANGL) (SIN ANG2))))
(SETQ SB_TIME (/ TTN TTD))
(SETQ B2S2_RET1 (APPEND (TRANS_SIG SB_TIME BCOUNT)
B2S2_RET1))

(SETQ DIF22 (- (* 2 DEPTB2) DEPTB1))
(SETQ ANG22 (ATAN (/ (* DIF22 (TAN ANGL)) DEPTB1)))
(SETQ TTN2 (* (* DEPTB1 (SIN ANG22)) (* DIF4 (SIN ANGL))))
(SETQ TTD2 (* 1500 (* (SIN ANGL) (SIN ANG22))))
(SETQ SB2_TIME (/ TTN2 TTD2))
(SETQ B2S2_RET2 (APPEND (TRANS_SIG SB2_TIME BCOUNT)
B2S2_RET2))

(SETQ BCOUNT (- BCOUNT 1))
(GO LOOP))))

(T
(SETQ B2S2_RIDGE 'NO)))
=====

```

```

=====
DEFUN BOUND_WRITE (BOUND_FILE IF_BOUND_GEN)
; FUNCTION TO WRITE BOUNDARY CONDITION VALUES
; TO FILE BOUND_FILE (OUTPDT FILE CENTERS.LSP)

(WRITE IF_BOUND_GEN :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)

(COND ((EQUAL IF_BOUND_GEN 'Y)

(WRITE PLATD :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE PLATV :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE WATERD :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE BOUND_AVG_LEV :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE BOUND_CONS_LEV :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE SURF_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE SURF_RET :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE BOTT_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE BOTT_RET :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE S1B1_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE B1S1_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE S1B1_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE B1S1_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE S2B2_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE B2S2_RIDGE :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)
(WRITE BOUND_RET :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE))

(T
(WRITE BOUND_RET :STREAM BOUND_FILE) (WRITE-CBAR #\NEWLINE BOUND_FILE)))
=====

```

```

1 (DEFUN BOUND_READ (INPUT_FILE)
2   (SETQ BOUND_GEN (READ INPUT_FILE))
3   (CONDO ((EQUAL BOUND_GEN 'Y)
4     (SETQ PLATD (READ INPUT_FILE))
5     (SETQ PLATV (READ INPUT_FILE))
6     (SETQ WATERD (READ INPUT_FILE))
7     (SETQ BOUND_AVG_LEV (READ INPUT_FILE))
8     (SETQ BOUND_CON5_LEV (READ INPUT_FILE))
9     (SETQ SURF_RIDGE (READ INPUT_FILE))
10    (SETQ SURF_RET (READ INPUT_FILE))
11    (SETQ BOTT_RIDGE (READ INPUT_FILE))
12    (SETQ BOTT_RET (READ INPUT_FILE))
13    (SETQ S1B1_RIDGE (READ INPUT_FILE))
14    (SETQ B1S1_RIDGE (READ INPUT_FILE))
15    (SETQ S2B1_RIDGE (READ INPUT_FILE))
16    (SETQ B2S1_RIDGE (READ INPUT_FILE))
17    (SETQ S2B2_RIDGE (READ INPUT_FILE))
18    (SETQ BOUND_RET (READ INPUT_FILE)))
19   (T
20     (SETQ BOUND_RET (READ INPUT_FILE)))))

```

```

/ FUNCTION TO CALCULATE BOUNDARY RETURNS
(DEFUN CALC_BD_VAL (RD_POS RET_LIST IF_GEN IF_CONS_LEV AVG_LEV)
  (CONO ((AND (EQUAL IF_GEN 'Y)
              (EQUAL IF_CONS_LEV 'Y)
              (EQUAL RD_POS (CAR (MEMBER RD_POS RET_LIST)))))
    (COND ((OR (EQUAL RD_POS (CAR (MEMBER RD_POS SURF_RET)))
              (EQUAL RD_POS (CAR (MEMBER RD_POS BOTT_RET)))))
      (SETQ BD_VAL AVG_LEV))
    (T (SETQ BD_VAL (/ AVG_LEV 2)))))
  ((AND (EQUAL IF_GEN 'Y)
        (EQUAL RD_POS (CAR (MEMBER RD_POS RET_LIST)))))
    (COND ((OR (EQUAL RD_POS (CAR (MEMBER RD_POS SURF_RET)))
              (EQUAL RD_POS (CAR (MEMBER RD_POS BOTT_RET)))))
      (SETQ BD_VAL (DISTRIBUTE AVG_LEV)))
    (T (SETQ BD_VAL (DISTRIBUTE (/ AVG_LEV 2))))))
  (T (SETQ BD_VAL 0)))
(SETQ BD_VAL_SQ (* BD_VAL BD_VAL))
/ SQUARE VALUE OF BD_VAL TO ADD TO AMPLITUDE LEVEL

```

APPENDIX B

SAMPLE INTERACTIVE TERMINAL SESSIONS

APPENDIX B

Sample Interactive Terminal Sessions

Method 1 - New range-Doppler map :

Would you like a new (N) range-Doppler map or will this be a second-generation (S) map? [N/S]

Are signals desired on the range Doppler map? [Y/N]
 Enter the number of 'fat' L signals desired (MAX 12) ...
 Enter the number of 'fat' O signals desired (MAX 12) ...
 Enter the number of 'fat' T signals desired (MAX 12) ...
 Enter the number of L signals desired (MAX 12) ...
 Enter the number of O signals desired (MAX 12) ...
 Enter the number of T signals desired (MAX 12) ...
 Enter the number of bars desired (MAX 12) ...
 Enter the desired mean for the level of these signals (MAX 255) ...
 Are constant level signals (C) or signal level that vary randomly (R) desired? [C/R]
 Is noise desired on the range Doppler map? [Y/N]
 Enter the desired mean for the level of the noise (MAX 255) ...
 Is reverberation desired on the range Doppler map? [Y/N]
 Enter the desired mean for the level of the reverberation (MAX 255) ...
 Enter the desired value to determine the shape of the reverberation (MAX 64) ...
 Enter the desired mean for the random component of the reverberation (MAX 255) ...
 Are boundary returns desired? [Y/N] ...
 Platform depth (m) ...
 Platform velocity (m/sec) ...
 Water depth (m) ...
 Average level of all boundary returns ...
 Are constant boundary returns desired? [Y/N] ...
 Are surface boundary returns desired? [Y/N] ...
 Are bottom boundary returns desired? [Y/N] ...
 Are surf(1) bott(1) boundary returns desired? [Y/N] ...
 Are bott(1) surf(1) boundary returns desired? [Y/N] ...
 Are surf(2) bott(1) boundary returns desired? [Y/N] ...
 Are bott(2) surf(1) boundary returns desired? [Y/N] ...
 Are surf(2) bott(2) boundary returns desired? [Y/N] ...
 Are bott(2) surf(2) boundary returns desired? [Y/N] ...

Is a quantized representation of values (Q) or a representation that shows only the presence of a signal (P) desired? [Q/P]

Method 2 - Second-generation range-Doppler map :

Would you like a new (N) range-Doppler map or will this be a second-generation (S) map? [N/S]

The last range Doppler map had these signals and center values: ((T 1440) (O 800) (L 1457) (FT 2248) (FO 2006) (FL 1834))

How many of these signals are to be repositioned?
 Enter a signal to reposition, as it appears above.
 Enter the desired time change (sec).
 Enter a signal to reposition, as it appears above.
 Enter the desired time change (sec).
 Enter a signal to reposition, as it appears above.
 Enter the desired time change (sec).
 Enter a signal to reposition, as it appears above.
 Enter the desired time change (sec).

APPENDIX C

SAMPLE OUTPUT FILES

APPENDIX C1

CENTERS.LSP

1	Flag for presence of signals
1	Number of 'fat' L signals
1	Number of 'fat' O signals
1	Number of 'fat' T signals
1	Number of L signals
1	Number of O signals
1	Number of T signals
0	Number of bars
200	Mean level of amplitude for signals
C	Type of signal (constant or variable)
1	Flag for presence of noise
20	Mean level of amplitude for noise
1	Flag for presence of reverberation
200	Mean level of amplitude for reverberation
5	Level for shape of reverberation
50	Mean level of amplitude for background noise
1	Flag for quantified values
1	Flag for presence of boundary returns
200	Platform depth
600	Platform velocity
60	Water depth
100	Mean level of amplitude for boundary returns
N	Flag for constant boundary returns
Y	Flag for presence of surface returns
(388 324 389 325 390 326 391 327 ...)	Surface ridge values (computed by RENSSEN)
Y	Flag for presence of bottom returns
(708 644 709 645 710 646 711 647 ...)	Bottom ridge values (computed by RENSSEN)
Y	Flag for presence of surface-bottom returns
Y	Flag for presence of bottom-surface returns
Y	Flag for presence of surface-bottom-surface returns
Y	Flag for presence of bottom-surface-bottom returns
Y	Flag for presence of surface-bottom-surface-bottom returns
Y	Flag for presence of bottom-surface-bottom-surface returns
(T 1440) (O 800) (L 1457) (ET 2248) (FO 2006) (EL 1834))	Signals and center values

This file contains quantized values of each cell generated in RENSCEN. Each row below corresponds to one row on the range-Doppler map.

QUANTIZED SIGNAL MEAN : "7"
QUANTIZED REVERBERATION MEAN LEVEL: "7"

This file contains the individual cell values computed in RENGSEN.
The rows below are referenced in RENGSEN as rows 0 through 39 (read from bottom to top).
Each row of the range-Doppler map corresponds to two lines of output in READIN.LSP. For example,
the values in row 39 of the map are contained in the top two rows below, starting with (52 9 27 ...
and ending with ... 21 19 8).

[illegible]

APPENDIX D

THREE-DIMENSIONAL GRAPH PROGRAM

[illegible][illegible]

```

C C C measure data line length, and read data
C C C
DO J=1,NROW
READ (99,120) LENGTH, STRING(1:LENGTH)
FORMAT (Q,A)
120
C C change right and left parentheses to blanks
C C C
STRING(1:1)=' '
STRING(LENGTH:LENGTH)=' '
READ (STRING,*) (NUMBIX(J,I), I=1,NCOL)
C IF (ROTATION.EQ.'Y') THEN
DO I=1,NCOL
NCOLR=NROW+1-J
NUMBIXR(I,NCOLR)=NUMBIX(J,I)
END DO
END IF
END DO
CLOSE (UNIT=99)
write output to files
C C C
WRITE (6,700)
FORMAT (2X,'Are output data files desired (Y/N)',$)
700 READ (5,710) FILD
FORMAT (A)
710 CALL STRUPCASE(FILD,FILD)
IF (FILD.EQ.'Y') THEN
IF (ROTATION.EQ.'Y') THEN
OPEN (UNIT=20,STATUS='NEW')
OPEN (UNIT=40,STATUS='NEW')
OPEN (UNIT=87,STATUS='NEW')
((NUMBIXR(J,I), I=1,NROW-20),J=1,NCOL)
WRITE (20,131)
((NUMBIXR(J,I), I=NROW-19,NROW),J=1,NCOL)
WRITE (40,131)
((NUMBIXR(J,I), I=1,NROW),J=1,NCOL)
WRITE (87,131)
CLOSE (UNIT=20)
CLOSE (UNIT=40)
CLOSE (UNIT=87)
ELSE
OPEN (UNIT=32,STATUS='NEW')
OPEN (UNIT=64,STATUS='NEW')
OPEN (UNIT=88,STATUS='NEW')
((NUMBIX(J,I), I=1,NCOL-32),J=1,NROW)
WRITE (32,130)
((NUMBIX(J,I), I=NCOL-31,NCOL),J=1,NROW)
WRITE (64,130)
((NUMBIX(J,I), I=1,NCOL),J=1,NROW)
WRITE (88,130)
CLOSE (UNIT=32)
CLOSE (UNIT=64)
CLOSE (UNIT=88)
END IF
END IF
FORMAT (32(1X,I3))
130 FORMAT (20(1X,I3))
131
C C C

```

```
C
C
C      initialize TEMPLATE and page lay out
C
CALL USLPDF
CALL UPSET('OUTPUTFILE', 7, )
CALL UASGN( 7., 'RDPLLOT.PDF\')
CALL USTART

C
CALL UDIMEN( 9., 6.5 )
CALL UPSET('WIDTH', 10. )
CALL UOUTLN
CALL ARL_HIDE_FONT

C
C      produce a two line plot title
C
CALL USET('DEVICE')
CALL UPSET('HORIZONTAL_SIZE', 0.28)
CALL UPSET('VERTICAL_SIZE', 0.40)
CALL UPSET('WIDTH', 5.0)
CALL USET('JUSTIFY')
CALL UPINT(4.5, 6.0, XREF(TITLE))
CALL UPSET('HORIZONTAL_SIZE', 0.21)
CALL UPSET('VERTICAL_SIZE', 0.30)
CALL UPSET('WIDTH', 4.0)
CALL UPINT(4.5, 5.5, XREF(TITLE1))
CALL USET('VIRTUAL')
CALL ARL_HIDE_FONT
C
C
```


produce the plot

APPENDIX E

INSTRUCTIONS FOR USING THE VAX/LISP

APPENDIX E
Instructions for using VAX/LISP

The VAX/LISP may be entered, from the dollar sign prompt, by typing LISP and then <RTN>. A prompt will appear LISP>.

NOTE: (1) Functions may be entered at this point by the operator to become familiar with the VAX/LISP dialect. However, it should be noted that functions entered at this time will be destroyed when the operator exits the VAX/LISP.

(2) To create a permanent record of desired functions the EDT editor may be used prior to entering LISP to create a file identified by FILENAME.LSP and loaded as mentioned below. This should provide a permanent record of the desired files. If the LISP editor is used (entered through the command (ED "RENSGEN.LSP")), a permanent record of the file will be provided by the following:

Enter GOLD COMMAND and type EXIT.

A prompt will appear, 'EXITING THE EDITOR.

ALL BUFFERS WILL BE LOST. ENTER [Y]

TO CONTINUE:'

Enter a Y.

The message 'EXITING...' appears.

If any changes have been made, a prompt appears 'BUFFER RENGGEN.LSP IS MODIFIED. DO YOU WANT ITS CONTENTS SAVED [Y]:'.

Enter Y to provide a record of changes.

Messages of 'WRITING FILE RENGGEN.LSP' and

'WROTE FILE ... RECORDS' will be followed

by the prompt LISP>.

RENSGEN.LSP may then be entered into the VAX/LISP by typing (LOAD "RENSGEN.LSP") and then pressing <RTN>. A message 'LOADING CONTENTS OF FILE RENGGEN.LSP' will appear, all the functions within the program will be sequentially listed, a message 'FINISHED LOADING' and a T will appear if the program loaded correctly, and then a LISP> prompt is given.

NOTE: At this point any function in RENGGEN.LSP may be exercised by typing the function name and appropriate argument(s) enclosed in parentheses, and then <RTN>.

The program RENGGEN.LSP may be exercised by typing (REVERBERATION) after the LISP> prompt following the loading of the program, and answering the questions that follow. To get out of VAX/LISP type (EXIT)

after the LISP> prompt. The dollar sign prompt will then appear.

If the program has been compiled, follow the same procedure to load and run, but replace RENSGEN.LSP with RENSGEN.FAS. To compile a LISP program, enter LISP/COMPILE RENSGEN.LSP after the dollar sign prompt.

APPENDIX F

INTERESTING LISP FUNCTIONS

APPENDIX F Interesting LISP Functions

FUNCALL - (FUNCALL fn a1 a2 ... an) applies the function (fn) to the arguments a1, a2, ... an as if the function was called directly (i.e., (fn a1 a2 ... an)). The FUNCALL function is useful when it is necessary to pass a function name as an argument to another function. The function may not be a special form or macro. An example of the use of FUNCALL can be found in the UP-DATE function of RENSGEN.

GENSYM - GENSYM invents a print name and creates a new symbol with that print name. The invented print name consists of a prefix and a decimal representation of a number. The number is incremented on each call to the GENSYM function.

Example:
(gensym 'num) --> num1
(gensym) --> num2
(gensym 5) --> num5
(gensym 'new) --> new6

GENSYM is used in the INITIAL function of RENSGEN.

PROG - A PROG construct establishes a local environment in a particular function. The variables x1, x2, ... xn below are all local variables.

```
(PROG (x1 x2 ...xn)
  (setq x1 1 x2 2 ... xn n)
  label1
    (cond (.... (return))
      ....
      (t (go label2)))
  label2
    (cond (.... (return))
      ....
      (t (go label1))))
```

Examples of the PROG are found in many functions in RENSGEN, including INITIAL and GENERATE.

LOOP - (LOOP form1 form2 ... formn)
Each form of a LOOP is evaluated in turn. When formn has been evaluated, then form1 is evaluated again, and so on, repeatedly. The LOOP construct never returns a value. Its execution must be terminated explicitly, using a RETURN or THROW, for example. LOOPS in RENSGEN occur in the INITIAL and GENERATE functions.

File reading - To read information from a file, the file must first be opened:

```
(make-pathname :version :newest)
(setq INPUTFILE (open "FILENAME.FILETYPE"))
```

Then to set a variable (var) to a value from the file:

```
(setq var (read INPUTFILE))
```

When all information has been read from the file, the file is closed:

```
(close INPUTFILE)
```

The REGEN function contains examples of file reading in RENSGEN.

File writing - To write to a file, again the file must be opened:

```
(setq OUTPUTFILE (open "FN.FT;l"
:direction :output :if-exists :new-version))
```

One of the functions in RENSGEN to write to files is the REGEN function.

Compiling - To compile a LISP program, the command

```
LISP/COMPILE FN.LSP
```

can be issued after the dollar sign prompt. If the LISP editor is being used, the command is

```
(compile "FILENAME.LSP").
```

(NOTE: A LISP file must have the filetype LSP.)

APPENDIX G

DATA FOR GENERATION OF FIGURES

APPENDIX G

Data for Generation of Figures

Figure 1 :

Signal Types	- None
Noise Mean	- 20
Reverberation Mean	- 200
Reverberation Shape	- 3
Reverberation Noise	- 50
Platform Depth	- 200 (m)
Platform Velocity	- 60 (m/sec)
Water Depth	- 600 (m)
Boundary Return Mean	- 100
Cutoff for Graph	- 35 Variable

Figure 2: Measured Data

Figure 3:

Signal Types	- None
Reverberation Mean	- 200
Reverberation Shape	- 3
Reverberation Noise	- 50
Platform Depth	- 200 (m)
Platform Velocity	- 60 (m/sec)
Water Depth	- 450 (m)
Boundary Return Mean	- 75 Variable
Cutoff for Graph	- 25

Figure 4:

Signal Types	- L,FL,O,FO,T,FT
Signal Mean	- 200 Constant

Figure 5:

Signal Types	- L,FL,O,FO,T,FT,Bar
Signal Mean	- 200 Constant

Figure 6:

Signal Types	- L,FL,O,FO,T,FT
Signal Mean	- 200 Variable

Figure 7:

Signal Types	- L,FL,O,FO,T,FT
Signal Mean	- 200 Constant
Noise Mean	- 20

Figure 8:

Signal Types	- L,FL,O,FO,T,FT
Signal Mean	- 200 Constant
Noise Mean	- 50

Figure 9:

Signal Types	- L,FL,O,FO,T,FT
Signal Mean	- 200 Constant
Reverberation	- 200
Reverberation Shape	- 5
Reverberation Noise	- 50

Figure 10: Signal Types - L,FL,O,FO,T,FT
 Signal Mean - 200 Constant
 Reverberation - 200
 Reverberation Shape - 5
 Reverberation Noise - 50
 Platform Depth - 200 (m)
 Platform Velocity - 60 (m/sec)
 Water Depth - 600 (m)
 Boundary Return Mean - 100 Variable

Figure 11: Signal Types - L,FL,O,FO,T,FT
 Signal Mean - 200 Constant
 Noise Mean - 20
 Reverberation - 200
 Reverberation Shape - 5
 Reverberation Noise - 50
 Platform Depth - 200 (m)
 Platform Velocity - 60 (m/sec)
 Water Depth - 600 (m)
 Boundary Return Mean - 100 Variable

Figure 12: Data same as Figure 11 with 90 Degree
 Rotation of Horizontal Plane about a
 Vertical Axis

Figure 13: Signal Types - L,FL,O,FO,T,FT
 Repositioned Signals - L,FL,FO,FT
 Change in Time - 10 (sec)

April 18, 1986
JES:JW:mdp

APPENDIX H

REFERENCES

APPENDIX H

References

- 1) Wilensky, Robert, "LISPCRAFT," University of California, Berkeley, W. W. Norton and Company, 1984.
- 2) Winston, Patrick, H. and Horn, Berthold K. P., "LISP," Massachusetts Institute of Technology, Addison-Wesley Publishing Company, 1981.
- 3) Steele, Guy L., "Common LISP - The Language," Digital Press, 1984.

DISTRIBUTION LIST FOR UNCLASSIFIED TN 86-64, by J. E. Sentz and
J. Wakeley, dated 18 April 1986

Office of Naval Technology (ONT)
800 N. Quincy Street
Arlington, VA 22217-5000
Attention: A. J. Faulstich, OCNR 23
Copy 1

Office of Naval Technology (ONT)
800 N. Quincy Street
Arlington, VA 22217-5000
Attention: D. C. Houser, OCNR 232
Copy 2

Commander
Naval Ocean Systems Center (NOSC)
San Diego, CA 92152-5000
Attention: William Tagney, Code 613
Copy 3

Commander
Naval Ocean Systems Center (NOSC)
San Diego, CA 92152-5000
Attention: Paul Reeves, Code 632
Copy 4

Commander
Naval Sea Systems Command (NAVSEA)
Department of the Navy
Washington, DC 20362-5101
Attention: Walter Rankin, Code 63D
Copy 5

Commanding Officer
Naval Coastal Systems Center (NCSC)
Panama City, FL 32407-5000
Attention: Paul Kurtz, Code 40
Copy 6

Naval Underwater Systems Center (NUSC)
Department of the Navy
Newport, RI 02841-5047
Attention: C. Albanese, Code 8212
Copy 7

Chris Eggen
Applied Physics Laboratory
University of Washington
1013 N.E. 40th St.
Seattle, WA 98195
Copy 8

DISTRIBUTION LIST FOR UNCLASSIFIED TN 86-64, by J. E. Sentz and
J. Wakeley, dated 18 April 1986

Dr. Anthony S. Maida
333 Whitmore Lab
The Pennsylvania State University
University Park, PA 16802
Copy 9

J. E. Sentz
5672-88 Stevens Forest Rd.
Columbia, MD 21045
Copy 10