BOEING

15 [illegible]

This document [illegible] by any holder only with specific prior approval [illegible]

REV LTR

THE BOEING COMPANY

[redacted] THE BOEING COMPANY WITHOUT THE PRIOR APPROVAL OF THE AIR ASW AND SURVEILLANCE PROGRAM ORGANIZATION.

Seattle, Wash 98124

CODE IDENT. NO. 81205

NUMBER (14) D189-90022-2

TITLE (6) ADVANCED ACOUSTIC SENSOR SYSTEM SOFTWARE

(9) Final report

ORIGINAL RELEASE DATE 4-3-74. FOR THE RELEASE DATE OF SUBSEQUENT REVISIONS, SEE THE REVISIONS SHEET. FOR LIMITATIONS IMPOSED ON THE DISTRIBUTION AND USE OF INFORMATION CONTAINED IN THIS DOCUMENT, SEE THE LIMITATIONS SHEET.

MODEL

ISSUE NO.

ISSUE TO

PREPARED UNDER:

[ ] CONTRACT NO.

[X] IR&D

[ ] OTHER

(11) 3 Mar 74

(12) 95 p.

(10) PREPARED BY GREG SCALLON, CHARLES CRETIN

SUPERVISED BY GLEN McMILLEN

APPROVED BY D. SCHULE

APPROVED BY D. MAYER

SHEET 1473

D1 4802 1430 ORIG 3/71

059 600 DN

TABLE OF CONTENTS

FOREWORD

The IR&D Acoustic Sensor Technology task for 1973 was to:

> Enlarge the technology base applicable to anti-submarine warfare in the detection, localization, classification and tracking of threat submarines. Perform analyses, trade studies and evaluations of operational and developmental acoustic sensor systems using computer models, laboratory testing and limited flight testing to determine the most effective system for future ASW patrol airplanes.

This series of documents has been prepared to describe the Independent Research and Development activities, including selection of the advanced Acoustic Sensor System for Airborne ASW, which was installed in the Boeing ASW test airplane during 1973 and the testing which was accomplished.

The family of documents includes:

- Advanced Acoustic Sensor System for Airborne ASW, D189-90022.
- Advanced Acoustic Sensor Subsystem Specifications, D189-90022-1.
- Advanced Acoustic Sensor System Software, D189-90022-2.
- Advanced Acoustic Sensor System Calibration and Testing, D189-90022-3.

This document describes the Advanced Acoustic Sensor System Software.

ABSTRACT

As a result of acoustic sensor analyses, trade studies and evaluations completed during 1973, an acoustic sensor system concept was developed and components were selected, procured and installed in the Boeing 720 ASW test airplane.

The applications software, designed and programmed for operation in the Interdata Model 85 computer, controls data acquisition, processing and display. The nucleus of this software is the real-time program which controls data acquisition from the analog-to-digital converter (ADC), performs the Fast Fourier Transform (FFT), line integration and line tracking algorithms, and communicates with the operator through the Hazeltine 2000 CRT/keyboard and the electrographic recorder displays.

In November 1973, an end-to-end test of the Acoustical Signal Processing and Analysis System (ASPAS) was run. The test employed an analog test tape, first with a sine wave with no acoustic noise, and then with a sine wave with noise added for approximately thirty minutes. The tests clearly demonstrated the capability of the ASPAS to detect and process signal-to-noise (S/N) levels lower than -20 decibels.

The acoustic sensor system was exercised during six flight-tests with varying degrees of success during 1973. This flight testing culminated in an operable system of hardware and software with the capability of recording acoustic data and subsequently playing it back on the ground for detail study and analysis.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

A to D	Analog-to-Digital
A/D	Analog-to-Digital
ADC	Analog-to-Digital Converter
ALI	Automatic Line Integration
ASCII	American Standard Code for Information Interchange
ASPAS	Acoustic Signal Processing and Analysis System
ASW	Anti-Submarine Warfare
BT	Bathythermal
CDF	Correlated Doppler Fixing
CRT	Cathode-Ray Tube
DIFAR	Directional Frequency Analysis and Recording
DOS	Disc Operating System
EGR	Electrographic Recorder
FFT	Fast Fourier Transform
FM VHF	Frequency Modulated Very-High Frequency
HEX	Hexadecimal
I/O	Input/Output
LOFAR	Low Frequency Analysis and Recording
MAD	Magnetic Anomaly Detection
NADC	Naval Air Development Center, Johnsville, Pennsylvania
NDDT	N-Buoy Doppler Differential Tracking Algorithm

LIST OF ABBREVIATIONS (Continued)

PROPLOSS	Propagation Loss Program
PSW	Program Status Word
S/N	Signal-to-Noise

KEY WORD LIST

Acoustic

Airborne ASW

Directional Frequency Analysis

Doppler Fixing

Fixing Algorithms

Low Frequency Analysis

Real-time

Software

Sonobuoy

Fast Fourier Transform

1.0 INTRODUCTION

1.0 INTRODUCTION

The work of developing major software components for the Acoustic Processing System fell into three groups:

(1) REAL-TIME ACOUSTIC SOFTWARE which is used to detect, classify and track submarines.

(2) ACOUSTIC SUPPORT SOFTWARE which augments those functions mentioned above but does not require real-time data from the sonobuoy sensors.

(3) ACOUSTIC DEVELOPMENT SOFTWARE which facilitates the generation of all the acoustic software.

Boeing has extensive experience with various software disciplines related to acoustic processing. This background was combined with the expertise of NADC to help formulate the specific objectives which were implemented in 1973.

Group 1 - Basic LOFAR processing has been implemented and flight tested. Meaningful LOFAR-grams for up to eight channels have been generated. The operator can dynamically command the FFT size (256, 512, 1024, 2048 and 4096 points), hanning options, gram formats, sampling rates, etc.

NDDT is a fixing algorithms suggested by NADC. N-Buoy Doppler Differential Tracking has been implemented by Boeing in FORTRAN and optimized for run-time and storage requirements.

CDF is another fixing program. Correlated Doppler Fixing is an improved NDDT-type algorithm producing greater accuracy using less a priori information and far fewer operator input variables.

DIFAR is a program designed to produce a bearing estimate from DIFAR sonobuoy data. Detection and bearing estimates from two or more DIFAR buoys allow a "fix" of the target's location. The program provides bearing estimates, signal-to-noise estimates, and a bearing confidence estimate. The bearing estimate is corrected for the estimated bias as a function of the signal-to-noise.

Group 2 - PROPLOSS and TASDA are two FORTRAN programs given to Boeing by NADC.

PROPLOSS is a program which calculates a ray path given the sonobuoy geometry and oceanographic conditions. These ray paths predict zones of convergence by computing the propagation loss at different geometrical points.

TASDA is a program which processes PROPLOSS outputs to guide the placement of sonobuoys for a given strategy, i.e., to maintain a given probability of detecting a submarine, to best cover an area, etc.

Group 3 - The mainframe manufacturer provided some general support software; A FORTRAN compiler, an assembler, a Disc Operating System (DOS), General Linking Loader, etc. Boeing has designed and implemented a real-time checkout monitor which provides the operator with interactive debug tools to help integrate, validate and verify the acoustic processing software being developed on the ASPAS test bed.

In general Boeing has developed its software as general purpose modules permitting swift reconfiguration as required by typical R&D programs. These goals have been partly unattainable in that much of the software in Group 1 and all in Group 3 was written in assembly language specifically tailored to exploit the peculiarities of our hardware configuration. This was done for processing efficiency - both timing and sizing constraints were recognized as severe limitations to the quantity and quality of data necessary. The other software was written in FORTRAN and made to work on Boeing's main computer system (large IBM machines), then converted to run on the ASPAS test bed. These programs are very flexible because of the degree of modularity.

Boeing's two main goals in its software development have been to:

(1) provide the necessary acoustic processing system to exercise the acoustic facility and acoustic signal processing analysis system software,

(2) establish test bed environment in which (ASPAS) functions can be investigated (i.e., trade studies, etc.).

In 1973, most of the emphasis was directed toward the first of these goals, of evolving a working system to gain ASW experience. Boeing has developed such a system and satisfactorily demonstrated it in flights against targets of opportunity.

2.0 GROUP 1 -- REAL-TIME PROCESSING

2.1 The ASPAS System

2.2 LOFAR Processing

2.3 Users Guide for Real-Time Acoustic System Software

2.3.1 Startup

- o Configuration & Turn-On
- o Loading
- o Initialization
- o Terminating
- o Restarting

2.3.2 Real-Time Controls

- o Translator Frequency Selections (Center Frequency & Bandwidth)
- o Input Data Quality
- o Processing Resolution
- o Hanning Option
- o Line Integration

2.3.3 Electrographic Recorder (EGR)

- o Turn-on
- o Software Enable/Disable
- o Calibration Options
- o Track Assignments
- o CRT Status

2.3.4 Sensor Metrics

- o Enable/Disable
- o RF Channel Assignments
- o Entering the Sensor's Position (Range/Bearing & X/Y Coordinates)

2.0 GROUP 1 -- REAL-TIME PROCESSING

2.1 The ASPAS System

Figure 2-1 shows the Boeing ASPAS ASW configuration operating in a real-time environment. After sonobuoy sensors are dropped into the ocean, they transmit a FM VHF signal containing frequency and broad audio data emitted by the target submarine. These data are continuously broadcast from each sensor to the ASPAS system located in the Boeing ASW test aircraft where they are recorded for future reference and submitted to dedicated processing hardware. The operator is provided with two electrographic recorders (EGR) as display devices and certain data is presented on a CRT display. The operator can dynamically direct this real-time processing from a control keyboard. Figure 2-2 shows the relationships between various ASPAS real-time functional components -- not all of which have been fully implemented and integrated. Those elements identified with an "*" are not currently available in the operational flight software. Figure 2-3 identifies the digital computer functions as well as the peripheral components of direct interest to the real-time operational software. That processing not included in the "Applications Software" box is collectively referred to as LOFAR processing and produces EGR recordings of spectral energy from continuous sonobuoy data. Note that this LOFAR processing can be augmented by various applications programs such as doppler differential fixing algorithms or DIFAR, etc.

The Real-Time Operational Software System is composed of three different subsystems: (1) an interactive control monitor, (2) an applications package and (3) basic LOFAR processing system. The modular construction is designed to allow for simple extensions to the interactive control

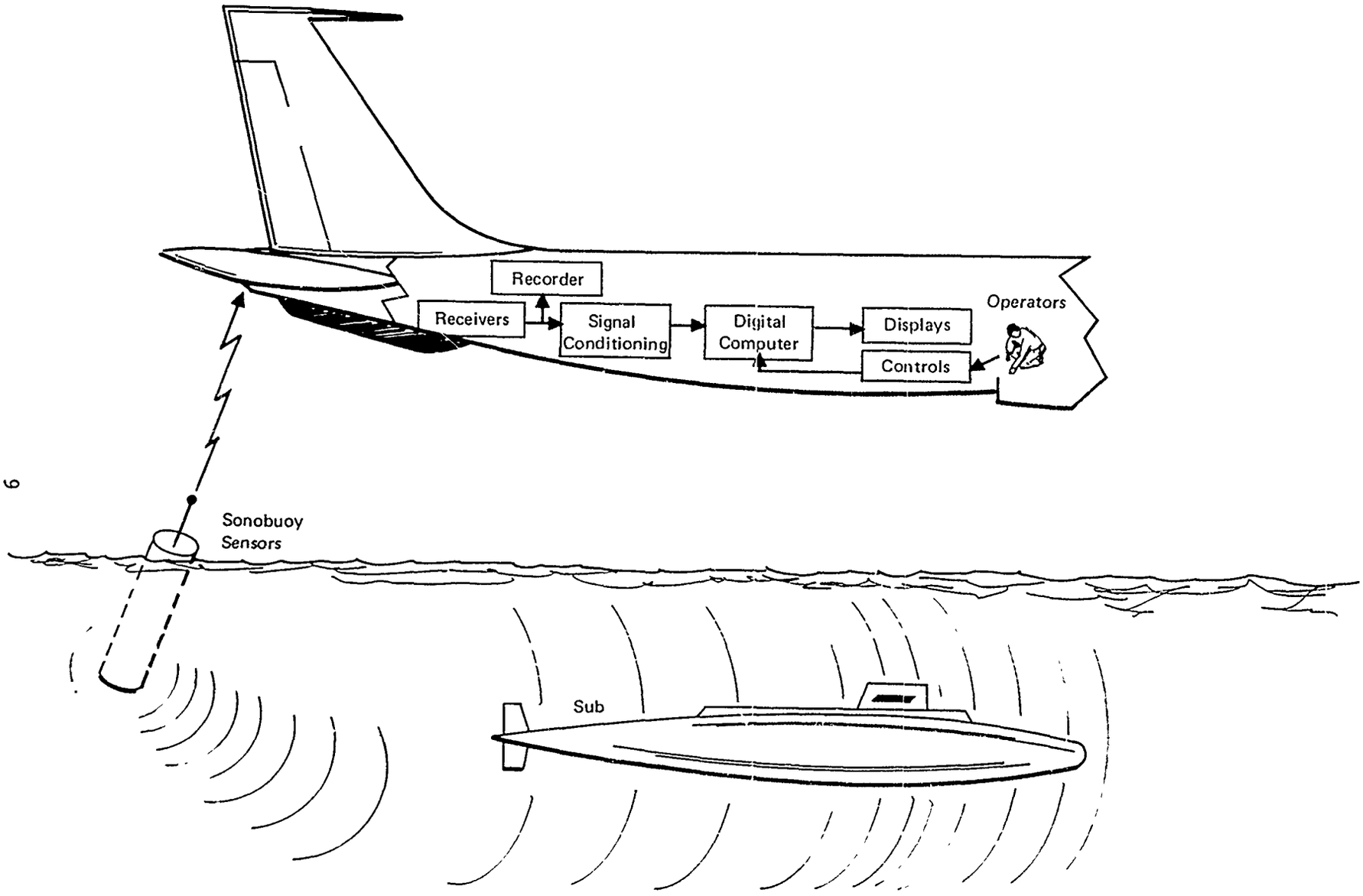


Figure 2-1 : ASPAS Real-Time Operation

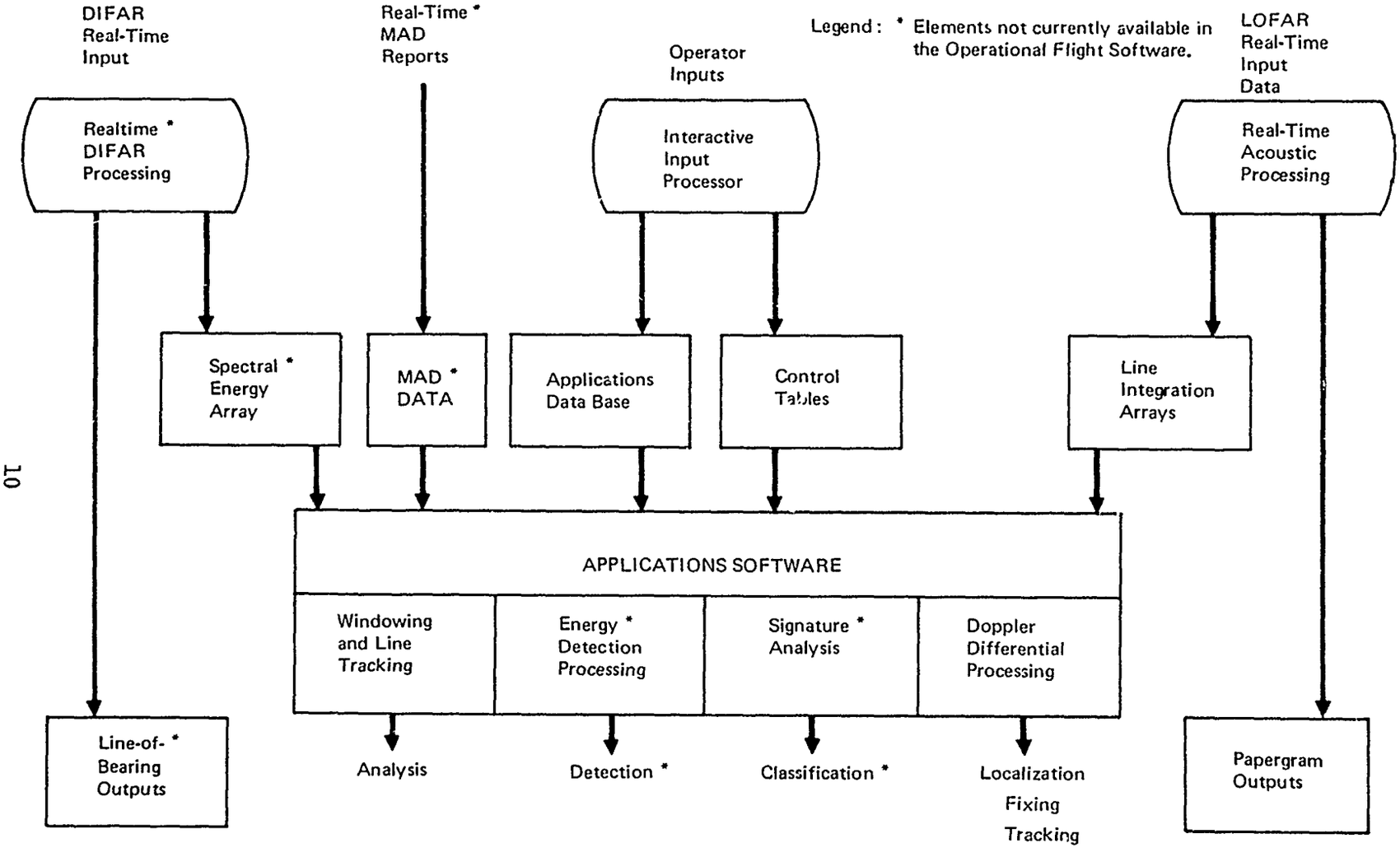


Figure 2-2 : Acoustic Processing

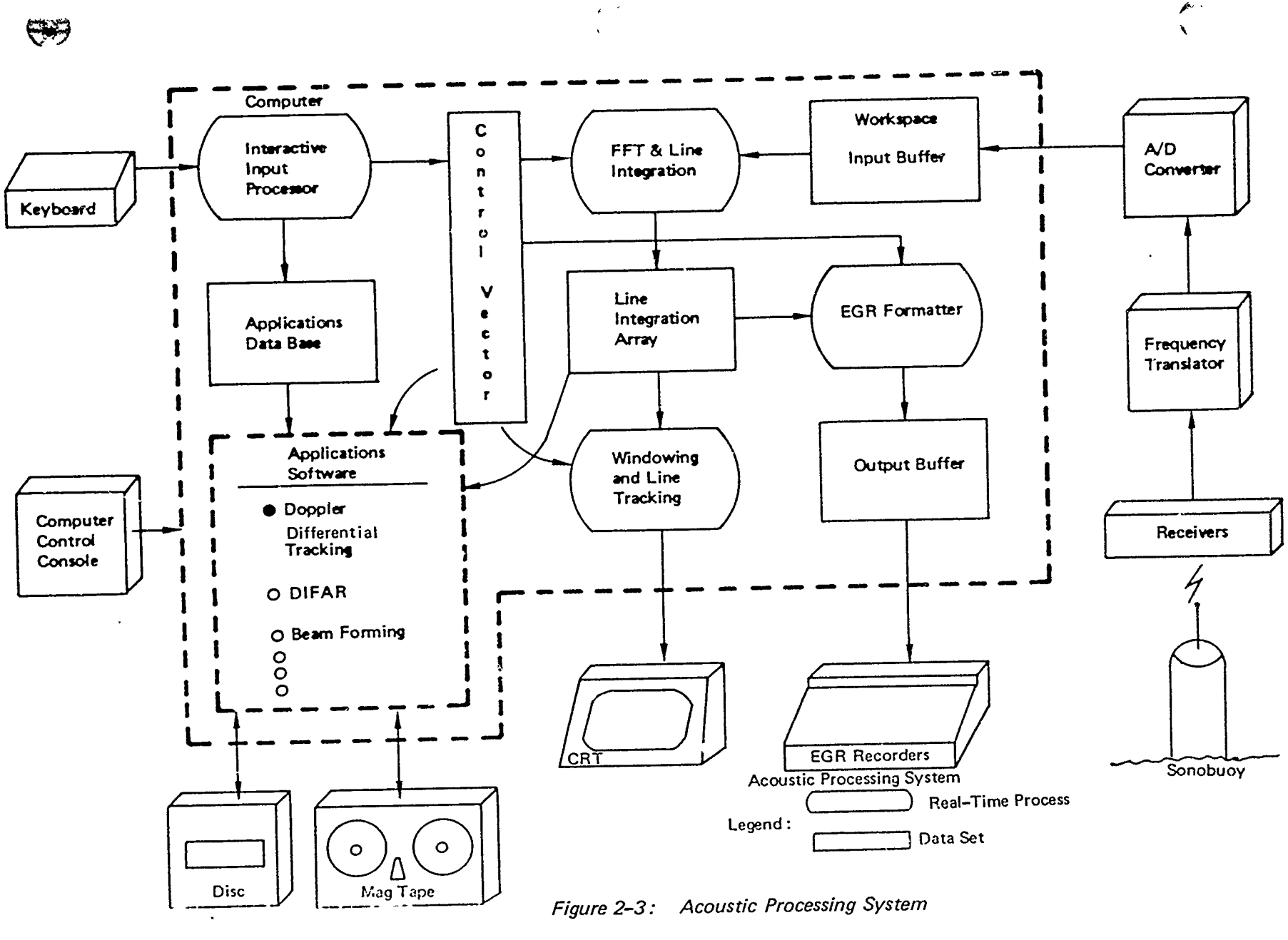


Figure 2-3: Acoustic Processing System

monitor to interface operator commands with any application program via control vectors as well as providing a good working structure into which new applications software modules may be readily inserted. The LOFAR processing is divided into numerous functional subroutines within a general framework such that new functions can be substituted directly without extensive redevelopment of the entire software system. The organization of all software around a general data-set concept facilitates this modular construction by separating the data design from the functional logic.

2.2 LOFAR Processing

LOFAR processing is organized into a real-time interrupt-driven scheme such that it will dynamically usurp control of the processor and its resources to accomplish the necessary calculations. This scheme of interrupting the applications program during execution to produce LOFAR-grams and update line integration arrays allows separation of LOFAR software from applications software while still maintaining the tight cyclic timing constraints inherent in real-time processing of continuous sensor data.

The LOFAR processing can be described as follows:

(1) A setup routine which initiates a cycle by establishing an input buffer for sensor data.

(2) A pre-Fast Fourier Transform (FFT) routine which organizes the input data for the FFT routine.

(3) A FFT routine which computes the spectral energy corresponding to the input time series data.

(4) An Automatic Line Integration (ALI) routine which updates line integration arrays retained in main memory.

(5) An EGR routine which formats and displays ALI data on the electrographic recorders.

Figure 2-4 shows this sequence.

Each of these routines is actually composed of numerous subroutines. Table 2-1 shows the FORTRAN equivalent of the FFT used.

The interactive control monitor is also an interrupt-driven process which dynamically analyzes all operator-typed commands.

This monitor interfaces with the other operational real-time software via control vectors which effectively isolate the timing of these programs from the specific instance and sequence of operator command entries. The checkout monitor is an optional part of the interactive control monitor. The real-time CRT supervisor is another part of the interactive control monitor (See Figure 2-5).

Table 2-2 shows the memory allocation scheme used for the real-time operational software. The disc operating system (DOS) occupies approx. 2000 hexadecimal bytes of storage and is used only to support any FORTRAN applications programs, none of the LOFAR software uses any of the DOS. This represents an area of future expansion, i.e., to substitute a small FORTRAN support kernel in place of DOS which is simplified by the fact that the interactive control monitor includes real-time I/O supervisors available from FORTRAN. Current versions provide slightly more than 2000 hexadecimal bytes for applications programs but the checkout monitor option can be excluded which will release more than 1000 hexadecimal

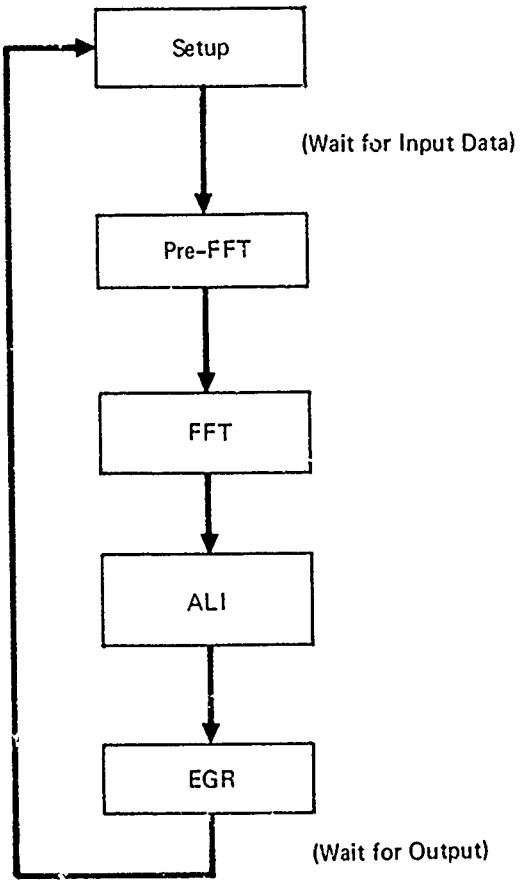


Figure 2-4 : LOFAR Processing

FAST FOURIER TRANSFORM ALGORITHM
-- FORTRAN EQUIVALENT

```
      DIMENSION X(N), Y(N)
      M = LOG2 (N)
      DO 10 LO = 1, M
      LMX = 2** (M-LO)
      LIX = 2* LMX
      SCL - 6.283185/LIX
      DO 10 LM = 1, LMX
      ARG = (LM-1)*SCL
      C = COS (ARG)
      S = SIN (ARG)
      DO 10 LI = LIX, N, LIX
      J1 - L1 - LIX + LM
      J2 = J1 + LMX
      T1 = X(J1) - X(J2)
      T2 = Y(J1) - Y(J2)
      X(J1) = X(J1) + X(J2)
      Y(J1) = Y(J1) + Y(J2)
      X(J2) = C*T1 + S*T2
10    Y(J2) = C*T2 - S*T1

      END
```

TABLE 2-1

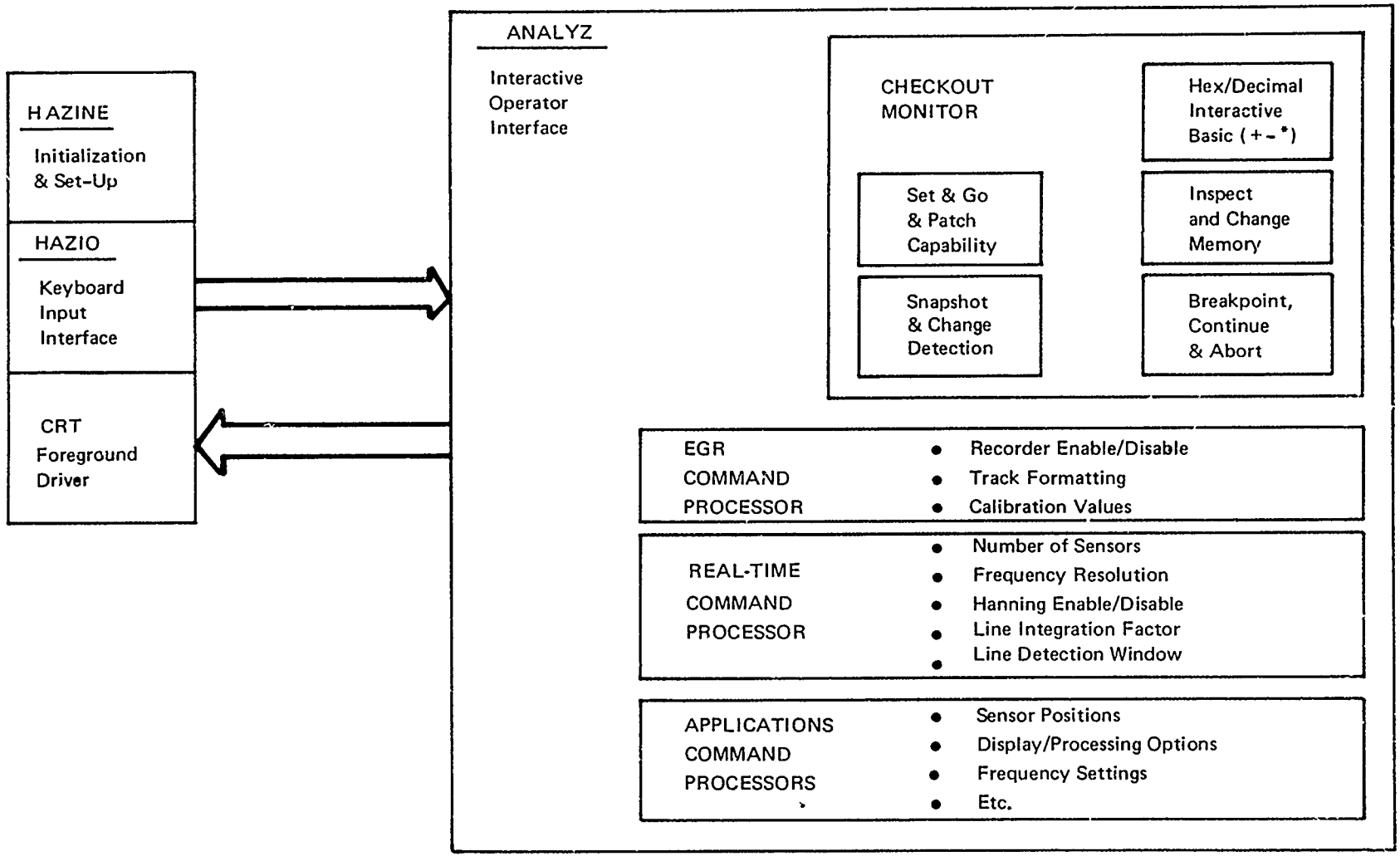


Figure 2-5 : Interactive Control Monitor

DATA DESIGN MEMORY MAP

LOCATION (IN HEX)	STORAGE FOR
0 thru 2D0	DEDICATED HARDWARE LOCATIONS
2D1 thru 2540	DOS (DISC OPERATING SYSTEM)
2550 thru 6550	BUFFERS & WORKSPACE AREA
6551 thru 6570	ALI SCALEFACTORS
6571 thru 8570	ALI ARRAYS
8571 thru 8D70	NOISE ARRAYS
9000 thru 9194	FORTRAN MAINPROGRAM
9195 thru 94D0	REALTIME LOFAR SUPERVISOR
94D1 thru A006	REALTIME LOFAR ROUTINES
A00D thru B9DE	INTERACTIVE CONTROL MONITOR
B9DF thru CBEO	CRT DRIVERS AND FORMATTERS
CBE1 thru CE5E	FORTRAN SUPPORT LIBRARY
CE5F thru ED94	CDF APPLICATIONS PROGRAMS
F500 thru FFFF	FORTRAN COMMONS

NOTE: Memory locations CE5E thru FFFF can be overlaid with other applications programs.

TABLE 2-2

bytes of additional storage for applications programs. Applications programs usually can be divided into small modules which can be sequentially loaded and executed, effectively overlaying themselves and minimizing the real-time memory requirements. Recall that the LOFAR and interactive control monitor are interrupt-driven, which requires that they be resident along with the buffers/workspaces and line arrays such that they occupy approx. A800 hexadecimal bytes.

2.3 User's Guide For Real-Time Acoustic System Software

2.3.1 Startup

Configuration and Turn-on

The real-time system requires a data source - either the RF receivers or the analog tape. The switching matrix on the acoustic operator's console selects the input source. The frequency translator passes only a portion of the entire spectrum - (10-, 50-, or 200-Hz windows) switch selectable on the translator's faceplate. The center of this bandpass must be dialed into the translator using the thumb-wheel switches also located on the translator's faceplace. The center frequency dialed in is numerically calculated as the desired center frequency plus 10% of the bandwidth. For example: a 50 Hz window centered about 100 Hz (i.e., 75 to 125 Hz processing) is specified by selecting the 50 Hz bandpass and dialing in 100 + 10% of 50 = 100 + 5 = 105 Hz. The translator's output (10, 50, or 200 Hz signals) is wired into the A to D converter which inputs digitized signals to the computer for processing. The CRT is tied into the computer such that it functions as the real-time control console. The electrographic recorders are output devices which graphically display the spectral power across the selected band. Other elements in the digital system such as the disc, magnetic tape unit and teletype machines are not used once the real-time process commences.

Obviously, the circuit breakers for the input device switch matrix, frequency translator, A to D converter, computer, CRT, and EGR's must be energized. The receivers, analog tape unit, frequency translator, A to D converter computer, computer memory, CRT, and EGR's have ON-OFF switches on their respective faceplates which must be turned on to operate.

The computer can be started up (cold started) only by employing both the disc and the teletype. Thus, these hardware units must be powered up. If the Real-time Operational Software System is not residing on disc, then the magnetic tape unit must also be powered up.

Loading

The ASW Acoustic Log is kept with the hardware and always contains the name and designations for the different system tapes. Tape #1273 is currently under construction and incorporates the second-generation real-time software. This system includes noise modeling, frequency normalization, real-time interactive keyboard control, improved status display on the CRT, individual line integration per channel, etc.

Using Interdata's Disc Operating System (DOS) the desired real-time operational software package is loaded. Usually the latest version is filed on disc "D6", file "OBJ2" under the name of "GREGRT" and can be easily loaded.

Initialization

Prior to starting the program, all essential hardware must be powered-up (see "Configuration & Turn-On). Consult the ASW Acoustic Log for the correct starting address and execute by "ST XXX" from the DOS or by setting the address into the computer's maintenance panel and executing. The real-time operation system does not require DOS to be resident nor does it assume any initialized computational state, only that the CRT be on-line.

Termination

To terminate the real-time operational program, the user can use the CRT's keyboard to enter "EX=D" if the checkout monitor has been "sys-gened" into the particular real-time operating system being exercised, or in any case the computer's maintenance switches can be set to "F" and a local maintenance panel interrupt will transfer control to the DOS via an illegal instruction. Termination can also be effected by transferring control to any desired location. Such transfers can be accomplished using the checkout monitor "EX=XXX" or by setting the target address into the maintenance switches and executing a maintenance panel interrupt. It must be understood that a simple branch to "2DO" will not restart the DOS and that some of the DOS I/O capabilities are severely compromised by the real-time I/O mechanisms. However, the "EX=D" and maintenance switch "F" with local interrupt restore the DOS to its full capability. (See Command Table 2-3.)

Restarting

Restarting can mean anything from re-initialization of one or more parameters, to various stages of "warm starts". The real-time operational software can re-sync. the CRT by setting "E" into the maintenance switches and executing a maintenance interrupt. The ASW Acoustic Log contains the restart address if it is different from the initial starting address. Selecting "EX=XXX" from the checkout monitor or setting the switches on the maintenance panel and executing a maintenance interrupt will execute a restart.

COMMAND TABLE

EX=D	Termination by recalling DOS.
EX=9000	Restart.
FW=XX	Frequency bandwidth in Hz (decimal integer).
R#=n	FFT resolution as a power of two.
HN=n	n = 1 enable, n = 0 disable hanning.
E#=i j k ...	Enable sensor i and j and k and ...
D#=i j k ...	Disable sensor i and j and k and ...
CF=XX	Center frequency in Hz (decimal integer).
IC=.XX	Set integration constant for all channels.
Ij=.XX	Set integration constant for channel j.
ER=n	Enable EGR, n=0 or 3 both, =1=right, =2=left.
ER=-n	Disable EGR n=3=both, =1=right, =2=left.
EC=XX	EGR calibration constant, in %, (0 to 100).
Ti=j	EGR track i will display channel j.
Ci=j	Channel i (1 to 8) will be RF#j (1-31).
Ri=XX.X	Range from origin to sensor i in yds.
Bi=XX.X	Bearing from origin to sensor i in degrees.
Xi=XX.X	X displacement from origin to sensor i in yds.
Yi=XX.X	Y displacement from origin to sensor i in yds.
LF=XX	Line tracking window-lower freq. (in Hz).
HF=XX	Line tracking window-higher freq. (in Hz).

TABLE 2-3

2.3.2 Real-Time Controls

Translator Frequency Selections (Center Frequency & Bandwidth)

The CRT keyboard is used to enter the center frequency and the bandwidth settings of the translator. "CF=nnnn" will establish the center frequency for all processing. Note that the thumb-wheel switches on the translator contain this same center frequency plus 10% of the bandwidth setting. The bandwidth command is "FW=nnnn". This command must be issued to establish the A to D conversion/sampling rate. The frequency spectrum displayed on the EGR's is identical to this "FW" value. (NOTE: To initialize the A to D conversion (e.g., after a power-down on the A to D) this "FW" command must be reissued.)

Example:

"FW=200" Translator band switch set to 200 will sample the A to D converter 400 times a second.

NOTE: The "FW" value does not have to match the translator's bandpass setting but it is recommended that FW $\geq$ bandpass for signal processing mathematics in the FFT.

Input Data Quality

The CRT displays a vector of raw data quality under the title "R". The following symbols may be found "-, 1, 2, 3, 4, 5, 6, 7, 8, +". The "-" (minus) symbol means that insufficient data was input to allow processing. If only "-" symbols are observed for all sensors, then either (1) the A to D converter is not reporting (turn ON and issue FW command) or no sensors are defined (issue an E# command). For resolution codes of 11 and 12, the number of sensors processed is degraded, R#=11 will limit the number of sensors to a maximum of four (first four only), R#=12 will process only the first or first and

second channels. These limitations are due to the fixed memory workspace constraint and can be responsible for "-" symbols in the raw data quality vector displayed on the CRT.

The "+" (plus) symbol means that the A to D converter was overdriven at least once on that channel. The integers 0 thru 8 mean:

CODE	PEAK VALUE BETWEEN (VOLTS)		
0	0	to	0.039
1	0.039	to	0.078
2	0.078	to	0.156
3	0.156	to	0.312
4	0.312	to	0.625
5	0.625	to	1.25
6	1.25	to	2.5
7	2.5	to	5.0
8	5.0	to	10.0

NOTE: By observing the rate at which the raw data quality vector is updated on the CRT, the operator knows the period associated with the input data collection process. If this period is 30 seconds then the A to D converter is OFF-LINE, the SELCH is HUNG, etc. ... Note: This condition will automatically display all "-" symbols.

The input time requirement can be observed by noting the interval between the start of the raw data quality update. The interval between the start of the raw data quality update to the line track frequency update is the signal processing time requirement.

Processing Resolution

The frequency transformation resolution is set by the number of input points to the FFT. The user specifies this parameter via the "R#=n" command. "n" may be one of the following; "8, 9, 10, 11, 12". Values less than eight will be promoted to 8, those greater than 12 will be degraded to 12.

CODE	NUMBER OF INPUT (TIME) POINTS	NUMBER OF FREQUENCY CELLS (BINS)
8	256	128
9	512	256
10	1024	512
11	2048	1024
12	4096	2048

Due to fixed memory workspace, limitations "R#=11" can only process the first four channels, and "R#=12" can process only the first two input channels. When in conflict, the number of channels are decreased until the memory constraints are satisfied.

Hanning Option

The input data for each channel can be submitted to a hanning process which multiplies the raw data by a triangular wave. This option applies to none or all channels only. The command "HN=1" will enable hanning while "HN=0" will disable the option. At startup hanning is disabled. The CRT will display "HAN." in the top line to signify that the option has been enabled and " " to inform the operator that hanning is not being performed.

Line Integration

Line integration between successive power spectrum calculations can be evoked individually for each channel. Line integration is an exponential scheme where the user specifies the decay constant, or the fraction of the previous integrated value to be retained, e.g., specifying zero means none of the previous integration value is to be retained or no line integration is to be performed while 0.9 means that the decay is only 10% per processing cycle and can be expressed in terms of time for any combination of resolution (R#) and bandwidth (FW). The command to set all the eight integration constants is "IC=.XXX". To change a single channel, line integration factor "Ij=.XXX" will set only channel "j"'s integration factor, note: j=1, 2, 3, 4, 5, 6, 7, 8.

Examples:

"IC=0.653" will set all channels to retain only 65.3% of their integrated values per cycle.

"I5=0" will delete line integration on channel 5 only.

"I3=.5" will establish a 50% integration factor for channel 3.

NOTE: These commands require operands less than unity, otherwise the command will be rejected. Negative operands are likewise rejected.

2.3.3 Electrographic Recorder

Turn-on

The EGR's have ON-OFF switches in addition to a circuit breaker. Knurled wheels on either side of the pinch roller must be positioned to apply the paper advance force. The calibration switch on the upper left faceplate must be set to "COMPUTER".

Software Enable/Disable

The command "ER=n" will enable/disable the computer generated output for the EGR's.

ER=0	Enables both papergrams.
ER=1	Enables the outboard EGR.
ER=2	Enables the inboard EGR.
ER=3	Enables both papergrams.
ER=-1	Disables the outboard EGR.
ER=-2	Disables the inboard EGR.
ER=-3	Disables both papergrams.

Disabled papergrams do not receive any output data from the computer while enabled papergrams are in constant communication with the computer. The real-time operational software updates the papergram display at the end of each cycle.

Calibration Options

The calibration switch on the upper left of the EGR faceplate permits local generation of a set of fixed patterns independent of the computer. In addition, the computer can be programed to produce any desired scale of gray. The command "EC=nn" established this gray level where nn is in percent of full scale. (0 to 100 only).

Example:

"EC=50" will set the calibration constant to one-half of full scale.

"EC=10" will set the calibration constant to ten percent of full scale.

Setting of this calibration constant does not, in itself, cause this selected calibration level to be displayed. The unit must be "TURNED-ON", "ENABLED" and the calibration value must be routed to various traces/tracks with "TRACK ASSIGNMENTS".

Track Assignments

Each electrostatic recorder is divided into four equal bands or tracks. These tracks are numbered 1 thru 8 where: Track 1 is the most outboard, 4 is the right-edge track on the outboard recorder, 5 is the left-edge track on the inboard recorder and track 8 is the furthest inboard. Double-width tracks are identified as Track A, B, C, and D where: Track A is the left-half of the outboard recorder, B is the right-half of the outboard recorder, C is the left-half of the inboard recorder and D is the right-half of the inboard recorder. Quad-width tracks are designated E and F where: Track E is the entire outboard recorder and Track F is the entire inboard recorder. The following chart shows these designations and the relationships between them.

TRACK ASSIGNMENT			LOCATION	
E	A	1	1st Quarter	OUTBOARD RECORDER
		2	2nd Quarter	
	B	3	3rd Quarter	
		4	4th Quarter	

TRACK ASSIGNMENT			LOCATION	
F	C	5	1st Quarter	INBOARD RECORDER
		6	2nd Quarter	
	D	7	3rd Quarter	
		8	4th Quarter	

The command "Ti = j" is the general TRACK ASSIGNMENT directive. "i" denotes the track to be assigned (i = 1, 2, 3, 4, 5, 6, 7, 8, A, B, C, D, E, F) and "j" is the data to be displayed. Operationally "j" is the channel designator -- for example:

"T3 = 5" means display channel 5 onto track 3 (third-quarter on outboard recorder).

"TA = 2" means channel 2 is displayed on track A (left-half of outboard recorder).

"TF = 4" means channel 4 is displayed across the entire inboard recorder.

When "j" is zero, a zero output value will be displayed on the selected track. When "j" is nine, the calibration value is displayed onto the designated track. Also, if "j" is greater than the highest channel being processed, then "j" is interpreted as though it were a nine and the calibration value is displayed.

The allowed "j" values are then "0, 1, 2, 3, 4, 5, 6, 7, 8, 9".

Example:

"TE = 9" will assign the calibration value across the outboard recorder.

"TF = 0" will assign a zero value across the inboard recorder.

"TC = 2" will display channel 2 on the left-half of the inboard recorder. If the current number of active sensors is less than 2, then the calibration value will be displayed in place of the channel 2 data.

NOTE: To write on the EGR's, they must also be "ON", and "ENABLED". Track Assignments can overlay, no combination or sequence is illegal.

CRT Status

The CRT displays a list of current track assignments (right-end of the top line). The eight codes correspond to tracks 1 thru 8. Display Codes "0 thru 9" correspond to the standard (quarter-width) trace. Display codes "LX" and "RX" mean that this quarter-track is displaying the left-half or right-half of channel "X". Display codes of "1X", "2X", "3X", "4X" mean that this quarter-track is displaying the first-quarter of channel "X", second-quarter of channel "X", third-quarter of channel "X", and fourth-quarter of channel "X" respectively.

2.3.4 Sensor Metrics

Enable/Disable

Sensors can be enabled by issuing the "E# = $n_1 n_2 n_3$ ..." command.

Example:

"E# = 1375" will enable sensor 1 and sensor 3 and sensor 7 and sensor 5.

"D# = 38" will disable sensor 3 and sensor 8.

Obviously multiple sensors can be enabled/disabled simultaneously, in fact repetition of the sensor identification is allowed, e.g., "E# = 111112" will enable sensors 1 and 2. The operand string can be composed only of the symbols 1 thru 8, otherwise the command will be rejected (ignored completely).

RF Channel Assignments

Each sensor has a unique RF channel which it uses to report acoustic data to the aircraft in flight. These channels are numbered 1 thru 31 by the Navy. To record this RF channel for specific sensors the user enters:

"Ci = j" where "i" is the sensor number ($1 \leq$ "i" ≤ 8) and "J" is the RF channel ($1 \leq$ "J" ≤ 31).

Example:

"C5 = 17" means that sensor 5 is using RF Channel 17.

This information is not used in the computer; it is retained on the CRT display for the purpose of providing fast/easy reference for the operator. This display occupies the right-end of the fourth line on the screen.

Entering the Sensor's Position (Range/Bearing and X/Y Coordinates

The sensor's position must be entered into the Real-time Operational System from the CRT keyboard. A local reference grid is assumed whose origin is known to the operator. All sensors are defined relative to this grid. Usually this grid is in polar coordinates (range and bearing) where range is expressed in yards and bearing in degrees. The operator selects a range or bearing entry, the sensor being defined and the numeric measurement for that entry.

Example:

"R1 = 5000" Range to Sensor 1 is 5000 yards.

"B3 = 17" Bearing to Sensor 3 is 17 degrees.

Remember, all references are from a common origin known to the operator - such as the center of a particular drop pattern.

The CRT maintains the current position display on the right side under the heading:

"RANGE BRN"
YARDS DEG

The display shows bearing (BRN) using standard Navy format - (0 to 359 degrees). Bearing can be entered in any convenient format, e.g., "Bi = 270" or"Bi = -90" or"Bi = -450" or "Bi = 630" will generate "Bi = 270" internally. Bearing inputs may contain non-zero fractional components which are retained for all calculations but not displayed. Likewise, the range expression may involve non-zero fractional components. These also are retained but do not appear on the display.

2.3.5 Line Tracking in the Frequency Domain

Specifying the Line's Domain (Low Frequency/High Frequency Limits)

The ALI arrays are frequency oriented lists of energy values covering the selected frequency bandwidth (FW). A "window" can be defined by specifying a lower limit and a higher limit such that a subarray is defined. This scheme selects a portion of the ALI array (a window) and selects the peak energy

cell from this subset. The doppler differential fixing algorithms require a set of frequencies which are produced by calculating the frequency corresponding to the bin containing this peak energy.

The operator can specify the low and high frequency limits thru the LF = XXX and HF = XXX commands where LF defines the LOWER LIMIT and HF defines the HIGHER LIMIT in hertz. These directives require a positive decimal integer operand.

Example:

"LF = 50"

"HF = 60"

will establish a 10-hertz window such that line tracking will be accomplished by auto-selecting the frequency bin of peak energy within this window. The values for "LF" and "HF" will be displayed on the CRT, second-line center.

In operation, the value of LF is checked, and if it is less than the lowest frequency in the ALI array then the window is defined to start at the beginning of the ALI array. Then the HF value is tested, if it is less than, or equal to, the lower window frequency then the upper window defaults to the highest ALI array frequency. Initially both LF and HF are zero which will define a window over the entire ALI array.

2.4 NDDT and CDF Fixing Algorithms

2.4.1 Introduction to the General Theory

There are two doppler fixing programs in the Acoustic Software System. The first is N-Buoy Doppler Differential Tracking (NDDT) developed at NADC which Boeing has implemented in FORTRAN. The Correlated Doppler Fixing (CDF) program was developed from the NDDT by applying statistical methods proven on other Boeing projects. Figure 2-6 shows the real-time operational system incorporating a doppler fixing program.

The fixing algorithm requires that multiple sonobuoy sensors simultaneously detect some frequency emitted by a moving object. The fixing algorithm calculates the relative position of the target within the sensor field by resolving the small frequency differences between sensor observations into a common frequency with individual doppler shifts due to the geometry of the problem.

This solution is iteratively evolved by superimposing a grid through the sensor field and evaluating the doppler shift which would result if the target were at a specific grid location. This is repeated for each grid position and for different values of speed and heading. A comparison of these test cases with the observations is used to select a best estimate. Another cycle using a tighter grid centered about this best estimate further reduces the error. The solution is evolved thru such refinements.

2.4.2 NDDT

The NDDT program selects a primary sensor and constructs the ratios of observed frequencies.

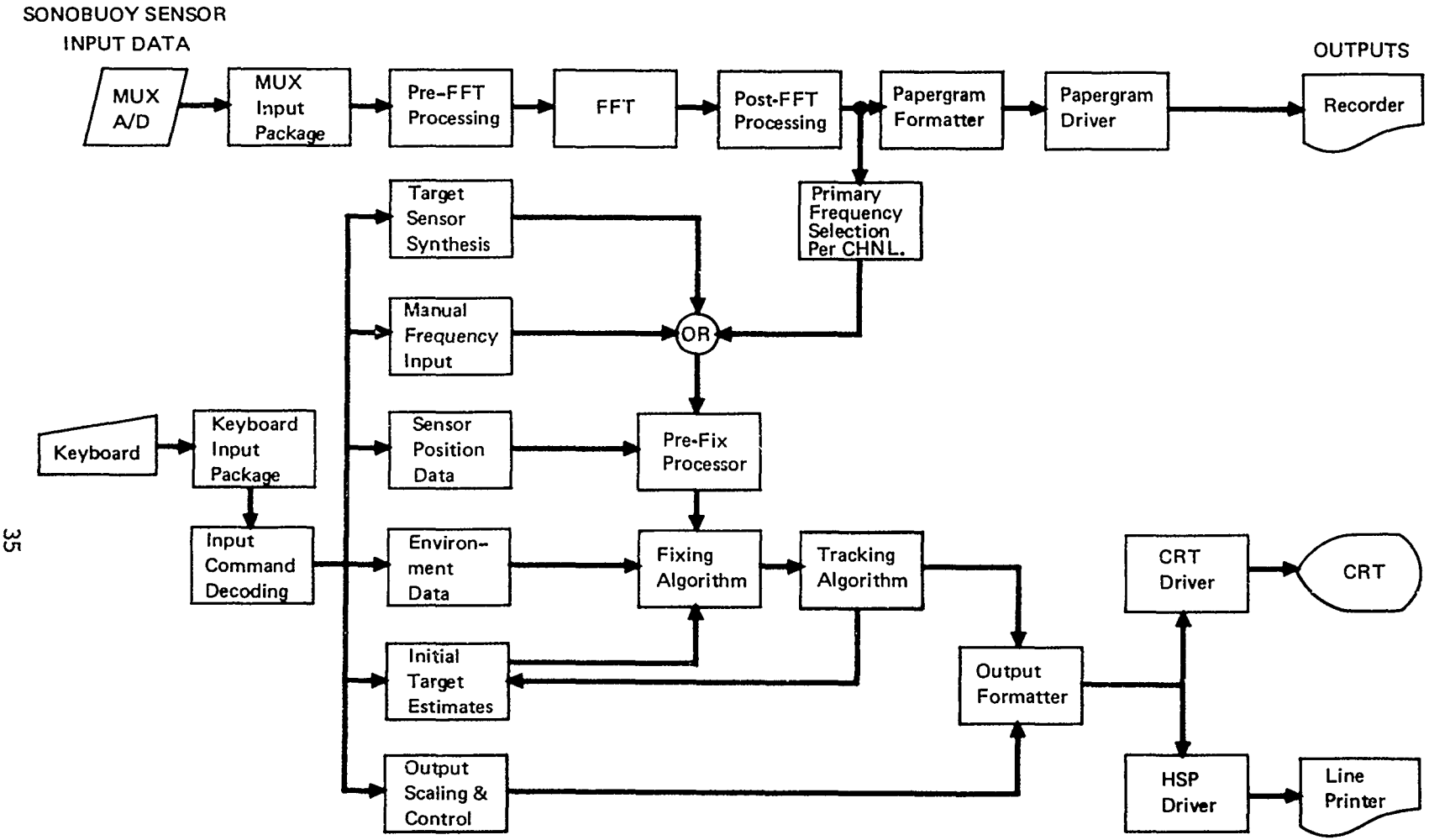


Figure 2-6 : Test Acoustic System Real-Time Program Structure

Example: Let F_i be the frequency observed by sensor i -- consider 5 simultaneous detections:

F_1, F_2, F_3, F_4, F_5,

Construct 4 ratio values:

$$R_1 = \frac{F_2}{F_1},\ R_2 = \frac{F_3}{F_1},\ R_3 = \frac{F_4}{F_1},\ R_4 = \frac{F_5}{F_1} \quad .$$

The "goodness of fit" measurement is defined as:

$$\Sigma\,(R_i - T_i)^2 \ ,$$

where T_i is the equivalent R_i for the test case.

The "n" smallest goodness values are combined to form the resulting estimate. This combination is via a weighted averaging using inverse "goodness of fit" values as coefficients.

The operator is required to enter:

- the number of iterations.
- the number of "X" steps in the grid.
- the number of "Y" steps in the grid.
- the number of Speed steps to be tried.
- the number of Heading steps to be tried.

The Operator must also enter all step sizes:

- for "X" increment.
- for "Y" increment.
- for Speed increment.
- for Heading increment.
- for iteration closing factor.

The Operator must also enter an initial estimate:

- for "X" position.
- for "Y" position.
- for Speed.
- for Heading.

The desirability of fully automating this process is obvious. An immediate goal is to remove the requirement for an initial estimate. This could be done by exhaustively stepping all parameters over a large range. This stepping of parameters could also establish the step sizes and step counts thus greatly reducing the operator inputs by exponentially increasing the computation size.

Investigations along these lines has led to the CDF program which "deduces" the speed without any speed iterations and rotates heading thru 180 degrees and uses back azimuth, thus covering all possible headings.

Another major improvement is the nature of the test evaluation. NDDT checks a "Ratio" vector of length "n-1" while CDF correlates a vector of length "n".

2.4.3 Correlated Doppler Fixing (CDF) Program

ASW systems include acoustic sensors which can be deployed over an area of the sea. Each sensor is equipped with an acoustic hydrophone whose function is to listen for submarines and transmit this data by radio to the ASW aircraft flying overhead. Acoustic processing on board the aircraft transforms this information from the time domain into the frequency domain through an FFT algorithm. The resulting power spectral density is displayed on the papergram recorders. The CDF program accepts this same power spectrum data and by computing a primary frequency for each sensor and recognizing the doppler shift due to the submarine's motion, estimates the

target's position relative to the sensor field. Successive fixes constitute a track.

How CDF Works

The "pre-fix" routine accepts input data from the FFT processor or from a synthetic "checkout" target generator or directly from an input keyboard. In any mode, the pre-fix routine produces a vector of observed frequencies, one per reporting sensor.

The "fixing" routine uses this vector of observed frequencies as input data. Test vectors are constructed for each point on an internal grid. These test vectors are correlated with the input vector of observations and estimates of position (X and Y) and velocity (heading and speed) are computed from those test vectors which most closely correlate with the observation vector. This estimate is then used to establish a finer grid and another iteration of test vectors is correlated with the observation vector. With each iteration, the quality of the resulting estimate improves to yield a fix. This fix is then passed to the tracking routine.

Why It Works:

The input vector of observed frequencies:

$$\overline{F} = (f_1, f_2, f_3 \ldots f_n),$$

$$f_1 = f_o + f_o * SR * COS(\text{ASPECT ANGLE TO SENSOR 1}) = f_o + A*X_1,$$

$$f_2 = f_o + f_o * SR * COS(\text{ASPECT ANGLE TO SENSOR 2}) = f_o + A*X_2,$$

.

.

.

$$f_n = f_o + f_o * SR * COS(\text{ASPECT ANGLE TO SENSOR n}) = f_o + A*X_n.$$

Where SR is the Speed Ratio: $\frac{\text{SPEED OF OBJECT}}{\text{SPEED OF SOUND}}$,

And A is $f_o * SR$,

And X_i = A COS (ASPECT ANGLE TO SENSOR i),

$$A\overline{X} = \frac{A}{n} \sum_{i=1}^{n} Xi .$$

Normalizing this vector we get:

$$\overline{F}_n = A(X_1-\overline{X})/B,$$
$$A(X_2-\overline{X})/B,$$
.
.
.
$$A(X_n-\overline{X})/B.$$

WHERE: $B = A \sum_{i=1}^{n} (X_i-\overline{X})^2$

(At this time "A" is unknown.)

Consider test vectors with A = 1:

$$\overline{T} = \text{COS (ASPECT ANGLE TO SENSOR 1)} = Y_1,$$
$$\text{COS (ASPECT ANGLE TO SENSOR 2)} = Y_2,$$

.

.

.

$$\text{COS (ASPECT ANGLE TO SENSOR n)} = Y_n.$$

Normalizing this vector we get:

$$\bar{T}_n = (Y_1-\bar{Y})/C \ ,$$

$$(Y_2-\bar{Y})/C \ ,$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$(Y_n-\bar{Y})/C \ .$$

WHERE: $C = \sum_{i=1}^{n} (Y_1-\bar{Y})^2$.

Defining "GRADE" to be $\bar{F}_n.\bar{T}_n$ and selecting the "m" best test vectors (highest correlation), we can compute an estimation vector $\bar{E}$ by combining these "m" best test cases and taking the weighted average of the aspect angles.

Now we normalize this estimate:

$$\bar{E}_n = (Z_1-\bar{Z})/D \ ,$$

$$(Z_2-\bar{Z})/D \ ,$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$(Z_n-\bar{Z})/D \ ,$$

WHERE "D" = $\sum_{i=1}^{n} (Z_1-\bar{Z})^2$.

And correlate it with the observed vector:

"GOODNESS" = $\bar{E}_n.\bar{F}_n$.

The "GOODNESS" approaches unity when:

$$\frac{(Z_i-\bar{Z})}{D} \approx \frac{(X_i-X)A}{B} \ .$$

Thus D and B are proportional, i.e., D = B/A. They are different because the normalized estimate vector $\bar{E}_n$ has a "Speed Ratio" multipled by $F_o = 1$ and the normalized observed vector $\bar{F}$ does not. Dividing these vector lengths $\frac{B}{D}$ yields "A". NOTE: $A = F_o \frac{\text{SPEED OF VEHICLE}}{\text{SPEED OF SOUND}}$. Since the limit of the doppler shift is less than 2%, then taking the average observed frequency as $\approx f_o$ introduces an error < 2%.

Thus SPEED $\approx \frac{A}{f}$ *SPEED OF SOUND = $\frac{f_o \text{ *SPEED*SPEED OF SOUND}}{f \text{ *SPEED OF SOUND}}$.

So much for calculating speed. Now to clarify how position (both X and Y) and heading are calculated, consider the test vector composed of aspect angle cosines. The aspect angle is defined as the difference between bearing to a specific sensor (obviously a function of position) and heading.

ASPECT ANGLE = $BR_j - H = ATAN \frac{(\Delta X)}{(\Delta Y)} - H$.

Note that for a given position and heading, the aspect angles to different sensors varies as ΔX and ΔY change with the geometry.

Test vectors are synthesized by construction a grid in 3 dimensions (X, Y, H). Typically grids contain six "X" points (NX=6) separated by 10,000 yards (DX=10000), and six "Y" points (NY=6) separated by 10,000 yards (DY=10000), 36 heading orientations separated by 10 degrees (DH=10). Note: By considering "back azimuths" only 18 Heading need be computed. A test vector is constructed for each combination of "X", "Y", and "H" and graded. Note that these particular grid control parameters allow coverage of 50,000 yds. by 50,000 yds. of sea surface and encompass all headings. A second iteration is performed using a grid centered about the estimated X, Y, and H produced in the first iteration. This grid is similar (NX=6, NY=6, NH=18) but the step sizes

are reduced to one-fifth (DI=0.2), i.e., DX=2000 yards, DY=2000 yards, and DH=2 degrees. The estimate produced by this second iteration is considered sufficiently accurate as to constitute a "FIX". Other combinations (e.g. 3 or more iterations, NX=5 or 9, NY=8 or 11, NH=7 or 12, DX= 50000, etc.) can easily be set by the operator but it remains to be shown that an increase in accuracy can be achieved without incurring excessive computational costs. Operational accuracy is corrupted by noise but a goal of ± 0.5 degrees heading is expected when six or more sensors detect the same target and that target is moving at 20+ knots.

CDF "Correlated Doppler Fixing"

Suggest Using NX=6, DX=10000., XI=0
NY=6, DY=10000., YI=0
NH=18 DH=10.0, HI=0
NI=2, DI=0.1

DESIGN

Three Nested Loops.

- Outer Loop: Varies "Y", calculates YL(J) = YS(J)-Y.
- Mid Loop: Varies "X", calculates XL(J)=X-XS(J), then computes XL(J) = ATAN (XL(J))/YL(J).
- Inner Loop: Varies "H", calculates:

  $$HL(J) = \cos(XL(J)+H).$$

  then normalizes this vector and computes:

  $$\text{"GRADE"} = \overline{HL}_n \quad \overline{FOBS}_n,$$

  saves "X", "Y" and "H" only for top "NBEST" positive grades.
- End of all loops:
  Compute "X", "Y", "H" based upon weighted average of NBEST cases (uses "GRADE" as weighing measure).

Calculates a "GOODNESS" grade for this "X", "Y", "H" by constructing a normalized vector $\bar{V}$ and taking the dot product with $\overline{FOBS}$.

Then divides the length of vector $\overline{FOBS}_n$ by ($\bar{f}$* length of $\bar{V}$) to get the Speed Ratio $\frac{S}{SMAX}$.
Then "S" = $\frac{S}{SMAX}$ *Speed of sound in water.

Optimization

Note that in the design (for one iteration):
Outer Loop (Y-Loop) done six times/sensor,

YL(J) = YSENS(J)-Y.

Mid Loop (X-Loop) done 36 times/sensor,

XL(J)=ATAN2($X_i/Y_{(J)}$) where X=X-$XSENS_J$.

Inner Loop (H-Loop) done 1296 times/sensor,

HL(J)=COS(XL(J)+H)

Note that for each sensor:
36 ARCTANS are computed, and
1296 COSINES are computed.
e.g. for eight sensors (max)= 288 ARCTANS + 10368 COSINES.
Thus top priority is given to optimizing "COSINE's."

Scheme:

(1) Theory: COS(B+H)=COSB*COSH-SINB*SINH
Application: If "B" is bearing and "H" is heading we can precalculate COS "H" and SIN "H" for all 36 possible headings prior to entering the outer loop. Then, in the mid loop, instead of computing ARCTAN(X(J))/Y(J), calculate XL(J)=COS (ARCTAN $\frac{X(J)}{Y(J)}$)
and
SL(J)=SIN(ARCTAN(X(J))/Y(J)
Then in the inner loop, COS (BEARING+HEADING) becomes:
HL(J)=XL(J)*COS"H"-SL(J)*SIN"H"

Where COS"H" and SIN"H" are simple single entries from the two precalculated tables.

Result:

36 cosines + 36 sines are calculated prior to entering the outer loop.

36j "SIN (ATAN$\frac{X}{Y}$)" and 36j "COS(ATAN$\frac{X}{Y}$)" are calculated during the mid loop and 1296j inner loop cosines are replaced by: "BC(j)*HC(j)-BS(j)*HS(j)" which can be done with two multiplies and one subtract each.

Also the 36j SIN(ATAN(XL(j), YL(j)) and 36jCOS(ATAN(XL(j),YL(j)) calculations can be performed by:

done 36 times {
Z=1./SQRT(XL(j)*XL(j)+YL(j)*YL(j)),
SL(j)=XL(j)*Z,
XL(j)=YL(j)*Z.

which involves four multiplies, one add, one divide and one square root each.

Thus, 36j ATANS and 1296j COSINES become: 36 sines, 26 cosines, 36j square roots, 36j divides, 2736j multiplies, and 1332j adds; and provides an order of magnitude speed improvement.

Tables 2-4 through 2-9 contain system control command tables.

MASTER CONTROL COMMANDS

CONTROL CODES WITHOUT OPERANDS

HA	HALT ALL EXECUTION
AM	SELECT AUTOMATIC MODE
FM	SELECT FREQUENCY MODE
TM	SELECT CHECKOUT OR TARGET MODE
GØ	EXECUTE RUN
SU	SCALE-UP THE DISPLAY
SD	SCALE-DOWN THE DISPLAY
RS	RESTART
DT	CENTER DISPLAY ON TARGET
DZ	CENTER DISPLAY AT X=0, Y=0

TABLE 2-4

LOOP CONTROL AND LOGICAL UNIT SELECTION

CONTROL CODES WITH INTEGER OPERANDS

NB=8	NUMBER OF "BESTS" COMBINED FOR A FIX
NI=3	MAXIMUM NUMBER OF INTERATIONS/FIX
NX=6	NUMBER OF GRID "X" POSITIONS/ARRAY
NY=6	NUMBER OF GRID "Y" POSITIONS/ARRAY
NS=3	NUMBER OF "SPEED" VARIATIONS/ITERATIONS (only for NDDT)
NH=18	NUMBER OF "HEADING" VARIATIONS PER INTERATION

NOTE: VALUES SHOWN ARE THE SYSTEM'S INITIAL VALUES

TABLE 2-5

SENSOR PARAMETERS

CONTROL CODES REQUIRING "REAL" NUMBERS

X1 = 0.0	GRID X POSITION FOR SENSOR 1 IN YARDS.
X2 =	" " " " " 2 " "
X3 =	" " " " " 3 " "
X4 =	" " " " " 4 " "
X5 =	" " " " " 5 " "
X6 =	" " " " " 6 " "
X7 =	" " " " " 7 " "
X8 =	" " " " " 8 " "
Y1 =	GRID Y POSITION FOR SENSOR 1 IN YARDS.
Y2 =	" " " " " 2 " "
Y3 =	" " " " " 3 " "
Y4 =	" " " " " 4 " "
Y5 =	" " " " " 5 " "
Y6 =	" " " " " 6 " "
Y7 =	" " " " " 7 " "
Y8 = 0.0	" " " " " 8 " "
F1 = 0.0	THE OBSERVED FREQUENCY AT SENSOR 1 IN HZ.
F2 =	" " " " " 2 " "
F3 =	" " " " " 3 " "
F4 =	" " " " " 4 " "
F5 =	" " " " " 5 " "
F6 =	" " " " " 6 " "
F7 =	" " " " " 7 " "
F8 = 0.0	" " " " " 8 " "

TABLE 2-6

POSITION AND VELOCITY PARAMETERS

CONTROL CODES REQUIRING "REAL" NUMBERS

XI = 0.0	INITIAL ITERATION GUESS FOR "X" POSITION IN YARDS.
XI = 0.0	" " " " "Y" POSITION IN YARDS.
*SI = 20.0	" " " " SPEED IN KNOTS.
*HI = 0.0	" " " " HEADING IN DEGREES.
DX = 10000.0	INITIAL DISTANCE BETWEEN GRID POSITIONS ALONG X-AXIS IN YARDS.
DY = 10000.0	INITIAL DISTANCE BETWEEN GRID POSITIONS ALONG Y-AXIS IN YARDS.
*DS = 10.0	INITIAL SPEED STEP SIZE IN KNOTS.
DH = 10.0	INITIAL HEADING STEP SIZE IN DEGREES.
XT = 0.0	SYNTHETIC TARGET'S INITIAL "X" POSITION IN YARDS.
YT = 0.0	" " " "Y" POSITION IN YARDS.
ST = 28.0	" " " SPEED IN KNOTS.
HT = 180.0	" " " HEADING IN DEGREES.

TABLE 2-7

*Not Required for CDF (Used for NDDT Only)

Note: XI & YI are not normally required for CDF

DISPLAY PARAMETERS

CONTROL CODES REQUIRING REAL NUMBERS

CX = 0.0	DISPLAY CENTER, X-AXIS, IN YARDS.
CY = 0.0	DISPLAY CENTER, Y-AXIS, IN YARDS.
RX = 10000.0	DISPLAY RANGE, X-AXIS, CENTER TO EDGE, IN YARDS.
RY = 10000.0	DISPLAY RANGE, Y-AXIS, CENTER TO EDGE, IN YARDS.
RR = 10000.0	DISPLAY RANGE, CENTER TO EACH EDGE, IN YARDS.

NOTE: RR = nnn IS THE SAME AS:
RX = nnn.
RY = nnn.

TABLE 2-8

CONTROL CONSTANTS

CONTROL CODES REQUIRING REAL NUMBERS

ZS = 1500.0	ZERO SUBSTITUTE, "CLOSE" DISTANCE SAFETY FACTOR IN YARDS.
SS = 4895.0	SPEED OF SOUND IN WATER IN FEET PER SECOND.
FO = 50.0	CENTER FREQUENCY IN HZ (MUST BE LARGER THAN 0.04 HZ).
QT = 5000.0	POSITION CRITERION FOR TRACKING (RSS OF X&Y ERRORS IN YARDS) (MUST BE LARGER THAN 9.0).
CT = 0.5	TRACKING FILTER, WEIGHT GIVEN TO TRACK HISTORY. (1.0 > DI > 0.0).
SP = 25.6	SYSTEM PERIOD IN SECONDS (FOR CHECKOUT MODE ONLY).

TABLE 2-9

2.5 DIFAR Program

The DIFAR computer program is presently designed for experimentation and development of detection and bearing estimate algorithms. It should not be construed to be an operational program for real-time in flight tests. As it is an experimental program, and the program is continually being modified. Hence, this program description will describe only those general characteristics of the program which are expected to remain stable.

The general data flow of the program is as follows:

2.5.1 Main Routine

This routine performs the initialization of the program. It accepts input commands from the operator and controls the calling of other routines to manipulate the input data.

2.5.2 Input Data Routine

Sonar data input is normally available through the receivers or from analog tapes in an off-line mode. This routine is written to sample the selected frequency-translated band from three input channels assumed to be the DIFAR omni, north-south, and east-west channels. The sample rate is program controlled through an operator input. The sample data is stored in arrays to be operated upon by other subroutines.

2.5.3 Frequency Transformation

Each of the three channel inputs is processed through an FFT routine. These outputs are then processed to obtain the power spectra of the omni channel and the cross power spectra of the north-south and east-west channels with the omni channel. The omni power spectra is used later for detection of lines while the cross power spectra are used for bearing estimation.

2.5.4 Spectral Equalization

This present routine has three options which allow the operator to discriminate against targets which have energy bands several lines wide in the frequency domain. The routine removes the mean-value trend from the omni power spectra data. The algorithm for removing the mean-value trend is expected to undergo change as experimental data are evaluated.

2.5.5 Automatic Line Integration (ALI) Routine

The output of the spectral equalization filter is line integrated to provide additional signal-to-noise (S/N) gain. The variance of the omni power spectra is calculated to provide a reference level for thresholding in the detection routine. The cross power spectra are also line integrated to improve bearing estimates.

2.5.6 Detection

The presence of target line frequencies is detected by testing each frequency cell of the line integrated omni power spectra against an operator selected threshold level. The selected threshold is relative to the calculated omni power variance. Each frequency cell found to exceed this selected threshold will be further processed to obtain S/N and bearing estimates. The S/N estimate is based upon the selected frequency cell power level relative to a set or adjacent, or near adjacent, frequency cells. The chosen set is a function of the spectral equalization option chosen by the operator.

2.5.7 Bearing Estimation

The bearing estimate of each frequency line is calculated as the arctangent of the cross-power spectra ratio of the selected frequency. The cross-power spectra are first corrected for channel-to-channel noise correlation. This is done by estimating the mean cross-power spectra in a set of adjacent

or near adjacent frequency lines and subtracting this level from the selected frequency cross-power. This is done for each of the two cross-power spectra prior to calculating the arctangent of the ratio.

2.5.8 Data Display

The present data display uses the Hazeltine 2000 CRT/keyboard to list the frequencies which were detected and to list the S/N and bearing estimates calculated for the associated line-frequencies. A confidence factor of the bearing estimate is also provided. This confidence factor is presently calculated by an empirical equation using the estimated S/N.

Significant modification of some of the algorithms is anticipated. As of the time of this writing, none of the program has been tested against actual DIFAR buoy data. The operator inputs and data display presently being used are designed for experimentation, and hence, may be different for operational programs.

3.0 Group 2 -- ACCOUSTIC SUPPORT SOFTWARE

3.1 PROPLOSS Program

3.2 TASDA Program

3.0 GROUP 2 -- ACOUSTIC SUPPORT SOFTWARE

3.1 PROPLOSS Program

The PROPLOSS program calculates a ray-path emanating from a transducer situated somewhere in the ocean. Inputs to the program include ocean characteristics -- dividing the ocean depths into layers of constant gradients -- transducer depth, initial starting angle of the ray, maximum X-distances for ray-path calculations, and an incremental distance along the X-axis. The X-axis is assumed to be parallel to the surface of the ocean; the Y-axis refers to depth.

Outputs of the program are a ray plot on the Hazeltine 2000 video display terminal and an optional tabular print-out on either the Hazeltine or a high-speed printer. The tabular print-out displays X- and Y-coordinates of the ray at selected intervals. In addition, at each X-Y coordinate the angle the ray makes with the horizontal is shown along with the intensity at that point.

Given: Three Layers.

Depth	Temperature
0	80
-200	76
-120000	76
-150000	60

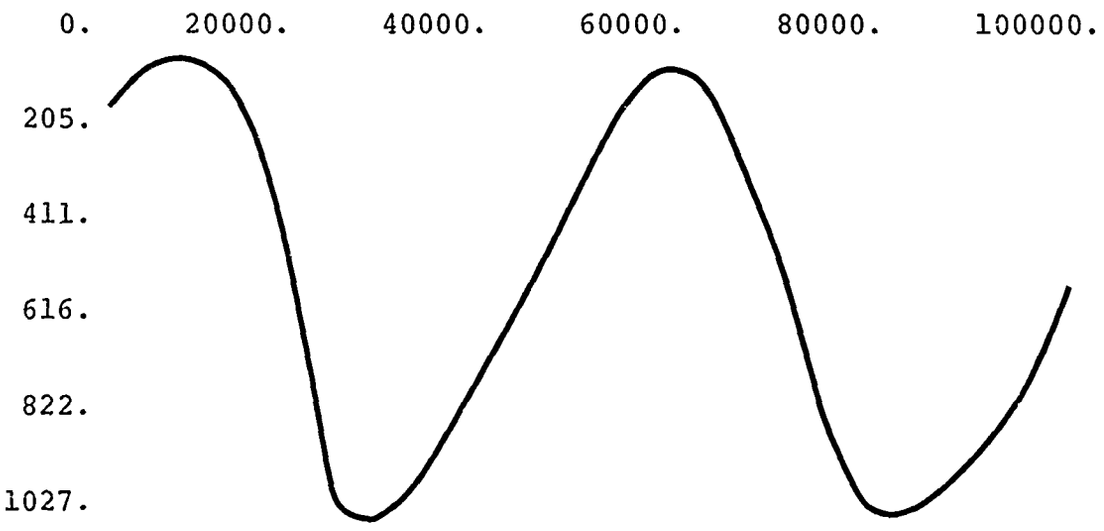
(0,-100)

Transducer depth = -100
Starting angle of ray = 80
X-limiting distance = 100000
X-axis increment = 2500

Example of Tabular Print Out:

X	Y	ANGLE	INTENSITY
2500	-10.	.9921	-58.4164
5000	-13.375	-1.1432	-74.9897
7500	-109.875	-3.2801	-80.7950
10000	-287.375	-4.1839	-82.6230
92500	-878.5625	1.8723	-91.0695
95000	-785.5625	2.3887	-92.2565
97500	-669.875	2.9052	-93.24
100000	-531.875	3.422	-94.0917

Example of Hazeltine Plot:



At the start of execution, the words "PROPOGATION LOSS PROGRAM" will appear on the Hazeltine screen. The user should hit the carriage return on the keyboard and prepare to input the following items:

(1) Input source FORMAT (I1)

Response: 1 for Hazeltine,
any other integer for card reader.

NOTE: All responses followed by carriage return.

(2) Number of BT (bathythermal) inputs FORMAT (I1).

(3) New BT data desired? FORMAT (I1).

Response: 0 means no,
any other integer means yes.

(4) Depth FORMAT (12.4).

(5) Velocity or temperature at that depth FORMAT (F12.4).

NOTE: Inputs 3 + 1 + 5 are repeated for each BT input, if input 3 = 0 then inputs 4 + 5 are skipped.

Initial default values are:

DEPTHS	TEMPERATURES
0.	80.0
-200.	76.0
-12000.	76.0
-15000.	60.0

(6) New maximum X or maximum Y? FORMAT (I1).

Response: 0 means no,
any other integer means yes.

(7) XMAX FORMAT (F8.0).

(8) YMAX FORMAT (F8.0).

NOTE: If input 6 to 0 then inputs 7 and 8 are skipped.

Initial default values are:

XMAX = 100000.

YMAX = -12000.

(9) New X increment value of ALPHA? FORMAT (I1).

Response: 0 means no,
any other integer means yes.

NOTE: ALPHA is angle to be added to get top ray.

(10) X increment FORMAT (F8.0).

(11) New ALPHA FORMAT (F9.3).

NOTE: If input 9 is 0 then inputs 10 and 11 are skipped.
Initial default values are:
XINC = 2500.
ALPHA = 0.

(12) New DEG or ANGLI? FORMAT (I1).

Response: 0 means no,
any other integer means yes.

NOTE: DEG is angle to be subtracted to get split ray.
ANGLI is incremental angle to be added (top ray) or subtracted (bottom ray) for intensity.

(13) DEG FORMAT (F9.3)

(14) Temperature IVEL - velocity code FORMAT (I1)

Response: 1 means BT cards are depth-velocity,
2 means BT cards are depth-temperature.

(15) ANGLI FORMAT (F5.3).

NOTE: If input 12 is 0, then inputs 13-15 are skipped.
Input 14 refers to inputs 4 and 5.
Initial default values are:
DEG = .002
IVEL = 2
ANGLI = .02

(16) New transducer depth? FORMAT (I1).

Response: 0 means no,
any other integer means yes.

(17) Transducer depth FORMAT (F9.3).

NOTE: If input 16 is 0 then input 17 is skipped.

Initial default value is:
YTDR = -100.

(18) New THETA? FORMAT (I1).
Response: 0 means no,
any other integer means yes,
NOTE: THETA is initial angle of ray path with horizontal.

(19) THETA FORMAT (F10.0).
NOTE: If input 18 is 0, then input 19 is skipped.

The words "OUTPUT READY" will appear almost immediately on the Hazeltine screen. The user should hit the carriage return and a plot of the ray path will appear. When finished examining the plot, the user should again hit the carriage return.

(20) Repeat results in tabular form? FORMAT (I1)
Response: 0 means yes,
any other integer means no.
NOTE: When Hazeltine screen becomes full, the tabular print-out will cease. To continue, the user must hit the carriage return.

(21) Process another set of data? FORMAT (I1)
Response: 0 means no,
any other integer means yes.
NOTE: If response is 0, the program will execute. If response is not 0, program will continue processing at input.

3.2 TASDA Program (Formal description to be provided.)

Current Status:

Items that need to be accomplished before the TASDA program will be operable:

(1) A random number generator giving values between 0 and 1 is needed.

(2) A buoy geometry tape must be created (doesn't necessarily need to be a tape -- a FORTRAN block data program would probably suffice).

(3) An input/output package to enable the TASDA program to communicate with the Hazeltine 2000 CRT/keyboard. The routines to do this are available, it is only a matter of incorporating these routines into the program.

One possible problem in implementing the above is a memory limitation. Only about 1500_{16} bytes are available for the I/O and random number generator. If this memory limitation does become a problem, it may be necessary to utilize the overlay subroutines provided by Interdata. Assuming it will not be necessary to resort to overlays, approximately one man-month of effort should result in the TASDA program being in a near operable condition.

The TASDA program predicts behavior of passive sonobuoy configurations. There are three problem options:

(1) Threshold -- Sonobuoy configurations are found in which the probability of submarine detection exceeds a specified threshold over a fixed area of ocean.

(2) Spacing optimization - Sonobuoy configurations are found for which the probability of submarine detection is maximized over a fixed area of ocean.

(3) Area maximization - Sonobuoy configurations are found for which the area of coverage is maximized, subject to the constraint that the probability of submarine detection exceeds a specified threshold level.

Inputs to the program include the following:

(1) Target characteristics -- This includes the type of submarine (nuclear or conventional), submerged and snorkling velocities, and the type target of motion (transiting or holding).

(2) Aircraft characteristics - This includes the amount of time necessary for airplane to arrive on station and the number of information processing channels the airplane has available.

(3) Ocean characteristics - This is basically propagation loss data.

(4) Other - This includes figure-of-merit values and buoy geometries (configuration of buoys with schedules for monitoring subsets of buoys).

Outputs from the program include:

(1) A plot of the buoy configuration.

(2) Monitoring duration and schedule.

(3) Probabilities of one, two and three simultaneous detections.

(4) Mean time to detection.

(5) Mean holding time of 1, 2 and 3 simultaneous detections.

At this time the exact formats of the inputs and outputs have not been decided upon. However the Hazeltine 2000 video display terminal will be the prime source of I/O.

4.0 GROUP 3 -- ACOUSTIC DEVELOPMENT SOFTWARE

4.1 The Checkout Monitor

4.0 GROUP 3 - ACOUSTIC DEVELOPMENT SOFTWARE

4.1 The Checkout Monitor

A collection of useful software routines has been integrated into a system called "the checkout monitor." This package is appended to the interactive control group such that the command decoder will parse messages to the appropriate processing group. As with all other command processing ,the operator interrupts current computations with each keystroke. The carriage return key causes the input character string to be analyzed and the operation field interpreted. Illegal operators produce an "UNDEFINED" response. Legal operators may involve operands which are decoded and subjected to validity checks. Invalid operands generate a "REJECT" response. The real-time software and the application packages are directed through appropriate control vectors--often in FORTRAN COMMONS. These system routines reference control vector parameters to direct their internal logic. Each factor is checked for consistency with respect to local criteria at time of use.

The checkout monitor is different from other elements in the interactive control monitor, it performs on command, i.e., it doesn't set flags/values to be used at some future time of need, but instead, directly executes a specified task independently of whatever computations were in process at the instant of operator intervention. As with all elements in the interactive control group, a recovery sequence is established which will resume the computational work suspended when interrupted by the operator. The checkout monitor is unique in that it is capable of terminating the suspended program and redirecting the resources to execute elsewhere.

Operator control is through the Haseltine 2000 CRT/keyboard. Commands are typed in and initiated upon receipt of a carriage

return. All commands follow the following format:

OPERATOR DELIMITER VALUE

Operators are two-symbol command codes, either a single blank or an equal sign is used as a delimiter to separate the operator field from an operand field. To evoke a command which does not require an operand, a carriage return is typed immediately after the two character operator code.

The CRT will display "UNDEFINED" if the operator code is not recognized, "REJECTED" is displayed when a legal operator is given, but with an invalid operand. A null operator is defined as a single carriage return and displays "READY" to signify that the interactive operator control monitor is alive and active.

The checkout monitor package is logically divided into six classes:

(1) On-line hexadecimal/decimal basic
(2) Inspect and change memory
(3) Set registers and execute
(4) Search under mask
(5) Snapshot and test
(6) Breakpoints and patches

For On-line Hexidecimal/Decimal Basic, the user is provided with a four-function arithmetic basic capability with a single accumulator and multiple storage registers. Sixteen commands are defined:

LH	LOAD ACCUMULATOR WITH A HEXADECIMAL VALUE.
LD	LOAD ACCUMULATOR WITH A DECIMAL INTEGER.
AH	ADD TO THE ACCUMULATOR A HEXADECIMAL VALUE.
AD	ADD TO THE ACCUMULATOR A DECIMAL INTEGER.
SH	SUBTRACT FROM THE ACCUMULATOR A HEXADECIMAL VALUE.
SD	SUBTRACT FROM THE ACCUMULATOR A DECIMAL INTEGER.
MH	MULTIPLY THE ACCUMULATOR BY A HEXADECIMAL VALUE.
MD	MULTIPLY THE ACCUMULATOR BY A DECIMAL INTEGER.
DH	DIVIDE THE ACCUMULATOR BY A HEXADECIMAL VALUE.
DD	DIVIDE THE ACCUMULATOR BY A DECIMAL INTEGER.
SA	SWAP THE ACCUMULATOR WITH CONTENTS OF STORAGE.
RA	RELOAD THE ACCUMULATOR WITH CONTENTS OF STORAGE.
+A	ADD TO THE ACCUMULATOR THE CONTENTS OF STORAGE.
-A	SUBTRACT FROM THE ACCUMULATOR THE CONTENTS OF STORAGE.
*A	MULTIPLY THE ACCUMULATOR BY THE CONTENTS OF STORAGE.
/A	DIVIDE THE ACCUMULATOR BY THE CONTENT OF STORAGE.

The current contents of the accumulator is displayed in a hexadecimal or decimal integer format corresponding to the format defined by the operation code.

```
e.g.   ACUM(H)      =  FFE3
       or ACUM(D)   =  314
```

Since these commands manipulate a single accumulator, hexadecimal and decimal operands can be intermixed providing hexadecimal to decimal and decimal to hexadecimal conversions. The ten commands which end with an "H" or "D" (LH, LD, AH, AD, SH, SD, MH, MD, DH, DD) are immediate instructions where the operand is directly specified by the user.

Example:

COMMANDS	DISPLAY
LH = FF00	ACUM(H) = FF00
AD = 0	ACUM(D) = -256
AH = -F	ACUM(H) = FEF1
MD = -1	ACUM(D) = 271

NOTE: Any product or quotient which exceeds a 16-bit binary representation generates the following display:

"ABORTED BY OVERFLOW"

and the accumulator is unchanged. However, overflows due to addition or subtraction are ignored, i.e., the accumulator retains the 16 lower order binary bits discarding the most significant bit.

The six commands which end with "A" (SA, RA, +A, -A, *A, /A) interpret the user provided operand as a storage designator. The operation will be performed using the current contents of that selected storage cell displaying the resulting accumulator's value in hexadecimal format.

OPERAND CODES	OPERATION
"X"A = HHHH	"X" = S, R, +, -, *, /
HHHH = -1	Selects accumulator's previous (last) value.
HHHH = 1, ..., 9	Selects special accumulator reserved storage register #1,...9.
HHHH = A	Uses the contents of the cell whose address is now in the accumulator.
HHHH = B	Uses the "BIAS" pointer
HHHH = C	Uses the "CURRENT" address pointer
HHHH = 0	Uses the contents of the "CURRENT" address

Otherwise, "HHHH" is the halfword address where the operand is to be found.

The SA command swaps the accumulator with the contents of a cell and RA recalls the accumulator value previously saved in a storage cell.

Example:

COMMAND	DISPLAY
SA = 1	ACUM(H) = XXXX
LD = 100	ACUM(D) = 100
RA = -1	ACUM(H) = YYYY

NOTE: "XXXX" represents initial contents of special storage register #1. "YYYY" represents initial contents of the accumulator.

Nine backup accumulator storage cells have been established to facilitate mathematical manipulations. These are separate from the Interdata hardware registers and are meant to provide the type of storage associated with a desktop calculator. Operand codes of 1, 2, 3, 4, 5, 6, 7, 8, 9, identify specific storage registers.

The six command codes allow load, swap (exchange), add, subtract, multiply and divide operations in the obvious way.

Example:

COMMAND	DISPLAY
RA = 3	ACUM(H) = XXXX
RA = A	ACUM(H) = YYYY
+A = 1000	ACUM(H) = ZZZZ

Where "XXXX" is the contents of accumulator storage register #3 "YYYY" is the contents of location "XXXX," "ZZZZ" is "YYYY" plus contents of location "1000."

For Inspect and Change Memory, the following operator commands are defined:

"IM"	Inspects the contents of memory.
"IN"	Inspects the contents of the next memory cell.
"BI"	Sets the bias register.
"IB"	Inspects the contents of memory relative to bias.
"SM"	Stores into memory cell.
"SN"	Stores into the next memory cell.
"+M"	Adds to the contents of the current cell.
"EB"	Evaluates addresses by block modules plus displacement.
"IE"	Inspect floating point number.
"NE"	Inspect next floating point number.
"SE"	Store floating point number into memory.
"DI"	Disassemble instruction in memory.
"AI"	Assemble an instruction into memory.

Because each operator involves only a single operand, the problem of putting "X" into memory cell "Y" is accomplished with two commands. A concept of "current" cell is used in which storage instruction operands are placed into the "current" cell.

Example:

```
IM = YYYY,
SM = XXXX,
SN = ZZZZ,
```

will put "XXXX" into location "YYYY" and then put "ZZZZ" into the next cell.

The "BI" and "IB" commands establish a relative addressing scheme.

Example:

```
BI = XXXX,
IB = YYYY,
SM = ZZZZ,
```

will put "ZZZZ" into the location "XXXX" + "YYYY+."

The "IN" and "SN" commands allow inspecting and setting of consecutive memory cells.

Example:

IM = XXXX,
SM = YYY1,
SN = YYY2,
SN = YYY3,

will put "YYY1" into location "XXXX," "YYYY2" into the next cell (XXXX+2), and "YYYY3" into the next cell (XXXX+4).

COMMAND

IM = hhhh* Inspects the contents of memory cell located at hhhh.

Displays: (hhhh) = hhhh
(hhhh): Address; hhhh: Contents

e.g.:IM = 3FC0,
(3FC0) = 0A91,

IN = hhhh Inspects the contents of the next memory cell plus hhhh cells.

Displays: (hhhh) = hhhh
(hhhh): Effective Address; hhhh: Contents

e.g.:IN = 1,
(3FC4) = 003F,

NOTE: The operand is a "halfword" or cell reference.

+M = hhhh Adds hhhh to the contents of this memory cell. If hhhh is zero then (BIAS) is added to this memory cell.

Displays: (hhhh) = hhhh
↑ Effective Address
↑ Resulting Contents

SB = hhhh — Sets hhhh into each cell of the block defined by (LOW) and (HIGH)

NOTE: (HIGH) cannot be less than (LOW), (see "LA" and "HA" directives).

*NOTE: A byte address, the least significant bit (LSB) is ignored.

Displays: "REJECTED" when (HIGH) is less than (LOW). "ACCOMPLISHED" when successfully completed.

MB = hhhh — Masked block. This directive provides the capability of logically modifying each cell in the block defined by (LOW) and (HIGH). Specific bits can be set, others cleared and still others unchanged in each word within the block.

The bits which will set are specified by "hhhh." The bits to be kept unchanged are specified by (MASK), while all other bits are cleared.

Displays: "REJECTED" when (HIGH) is less than (LOW). "ACCOMPLISHED" when completed.

BI = hhhh* — Sets the bias to hhhh.

Displays: (BIAS) = hhhh
↑ New BIAS value

Example:

BI = 3F88.

IB = hhhh* Inspects the contents of the location defined by (BIAS) plus hhhh

Displays: (hhhh) = hhhh
(hhhh): Effective Address
hhhh: Contents

Example:

IB = 1000,

(4F88) = E605.

SM = hhhh Sets the contents of the current cell to hhhh

Displays: (hhhh) = hhhh
(hhhh): Effective Address
hhhh: New Contents

Example:

SM = 83,

(4F88) = 0083.

NOTE: To clear a cell, type "SM" carriage return.

SN = hhhh Sets the contents of the next cell to hhhh.

Displays: (hhhh) = hhhh
(hhhh): Effective Address
hhhh: New Contents

Example:

SN = 3EA1,

(4F8A) = 3EA1.

The sequence of operators SM, SN, SN, +M will allow the user to manually enter a machine-language program in a relative mode. Other directives allow the user to set initial values into the register file, set the PSW control bits and execute from memory.

Recall that the on-line hexadecimal-decimal package can manipulate both the "CURRENT" pointer and the "BIAS" pointer via the SWAP ACCUMULATOR command.

The ability to convert an address into a base plus displacement is very handy when studying systems composed of multiple modules. Consider a fault at address "XXXX," this is most meaningful when expressed as "MODULE ABC +158." The "EB" directive allows the user to establish a reference directory identifying system modules and where they start.

COMMAND

"EB = -1"	Clears the reference directory Displays "ACCOMPLISHED."
EB = hhhh = name	Adds the named module starting at hhhh to the directory. Displays "REJECTED" if "name" has more than six characters or if the reference directory is full. "hhhh" is a halfword address, see NOTE 1. Displays "ACCEPTED" when successfully accomplished.
EB = hhhh	Evaluates hhhh against the reference directory. "hhhh" is a byte address, see NOTE 1. Displays "REJECTED" if a format error is detected.

Displays:

hhhh	=	"name"	+	hhhh
↑		↑		↑
Address Being Evaluated		Module Identifier		Displacement Address into Module

The module identifier (name) selected is that for which the displacement is the minimum positive value. In case the address is below the lowest module in the reference directory, (i.e., all displacements would be negative) or if the directory is empty, the display is:

hhhh	=	"*ZERO*"	+	hhhh
↑		↑		↑
Address Evaluated		Origin of Memory		Displacements (same as the address evaluated)

The inspect and change package is augmented by three floating point commands "IE," "NE," and "SE."

COMMAND

IE = hhhh — Will display the contents of the full word stored at "hhhh" as a floating point number.

NE = hhhh — Will display the contents of the fullword stored at "CURRENT" plus hhhh plus one fullwords as a floating point number.

SE = XXXXX.XXXX — Will set the current fullword to the floating point representation corresponding to the real number "XXXX.XXXX." The current pointer will then be advanced to point to the storage word immediately following the real number just stored.

The Display common to "IE," "NE," and "SE" Commands is:

(1234)	=	-1.23456E-12
↑		↑
Effective Address		Decimal Value

The inspect and change package also supports direct assembly and disassembly in standard assembly-language for the mainframe.

COMMAND

"EI = hhhh" — Will display the assembly language equivalent of the instruction at location "hhhh."

"DI = 0" Will disassemble the instruction located at "CURRENT" plus one halfword cell.

Displays:

1234 ABCD F, F "RR" FORMAT,
1234 ABCD F, FFFF (F) "RS, RX" FORMATS,
1234 DC FFFF = -12345 "UNDEFINED INSTRUCTIONS".

Where: 1234 is the effective location, ABCD is the op-code mnemonic, F is a hexadecimal register, FFFF is a hexadecimal operand, -1234 is the decimal equivalent of the hexadecimal data value.

NOTE: The "RS, RX" format will advance the current pointer to point to the operand field, "RR" and "DC" formats leave the current pointer pointing to the address as shown in the display. The current pointer always points to the halfword cell immediately prior to the start of the next instruction.

"AI = ABCD R1, R2" RR FORMAT.
"AI - ABCD R1, Y(X1)" RX, RX FORMAT.

To facilitate character conversions between printing symbols and "ASCII" hexadecimal code representations, the following two directives are provided:

C'α and 'C = hh

"C'α" will display the hexadecimal code for the symbol α".

Displays: "ASCII α IS FF".
ASCII Character
Hexadecimal Code

"'C = hh" will convert the hexadecimal byte "hh" to an ASCII symbol.

Displays: "REJECT" if "hh" out-of-range,
"NON PRINTING ASCII" if,
"hh" < 20 or "hh" > 5F .

ASCII α IS hh.
ASCII ↑ ↑
Character Hexadecomal
Code

The "AI" directive will directly assemble INTERDATA ASSEMBLY LANGUAGE STATEMENTS and store into memory location "CURRENT" plus one halfword. The display is identical to that for the "DI" instruction.

NOTE: The R1 & R2 (or X1) fields may be hexadecimal or decimal, may or may not be preceded by R or X. The operand field "Y" on "RS, RX" instructions may begin with "*" meaning relative addressing followed by "+" or "-" and "X'FFFF'" for hexadecimal or "1234" for decimal values.

To define program labels, the following subdirective is provided:

"AI = < name = term string" ,
which is equivalent to "name EQU term" .

Displays: "name = X'hhhh'"
↑
Hexadecimal value assigned to this name

Other displays:

"DUPLICATE LABEL",
"SYMBOL TABLE FULL",
"FORWARD REFERENCE",
All cause the input to be rejected.

Operand strings may contain:

(1) Decimal numbers,

(2) Hexadecimal numbers X'··',

(3) Character codes C' ..' (2 symbols max),

(4) Symbolic names,

which can be algebraically combined using the "+" and "-" operators.

Register designations may be decimal or hexadecimal, and may or may not be preceeded by "R" or "X".
e.g., "LHI R14, CAT + C'F' -9 + X'F' (Xb)".

Assemble instruction subdirectives:

The origin function "ORG" is provided by:

"AI = * = hhh"	Which will assemble into hexadecimal address "hhhh".
"AI = * = A"	Will origin to the address in the accumulator.
"AI = * = B"	Will origin to the BIAS address.
"AI = * = O"	Will origin to the contents of the current address.
"AI = * = C" "AI = * = *" "AI = *"	Will origin to the current address.

All the instructions above generate the following response:

"(START) = hhhh"
↑
Hexadecimal address into which the next instruction will be assembled.

To clear the symbol table issue the following command:
"AI =< ", which will display "all symbols cleared".
NOTE: This must be done to initialize the symbol directory.

To Set Registers and Execute, the following directives are supported:

LM	Load registers R_0 thru R_{15}.
LP	Load program status word.
IR	Inspect the contents of a specific register.
EX	Transfer control to a specific location.
E$	Set an "End" command into a specific location.
SR	Stores registers 0 thru 15 and the PSW into memory.
LG	Loads registers 0 thru 15 and the PSW from memory and goes;
LM = hhhh*	Load R_0 thru R_{15} from the contents of the sixteen memory cells beginning at location hhhh. "hhhh" is a halfword address (See NOTE 1). Displays: ACCOMPLISHED.
LP - hhhh	Load Program Status Bits. Displays: (PSW)= hhhh. Example: LP = 3100 (PSW)= 7900 NOTE: The bits for "AIO" and Machine Fault are marked in.
E$ = hhhh*	Sets a "STOP" command into location hhhh. Displays: (END) = hhhh Example: E$ = 7000, (END) = 7000. NOTE: If hhhh is zero, the stop command is stored into the next cell. When #7000 is executed, the following is displayed - "(STOP) = 7000" and control is given to the operator.
IR = hhhh	Inspects the contents of the selected register.

Values of hh		Display code
0 ... F	Register 0 thru 15	0 ... F
10	Program Status Word	S
11	Program Counter	P
12	"BIAS" Register	R
13	"CURRENT" Address Register	X
14	Mask Register	M
15	Low Address Register	L
16	High Address Register	H
17	Basic Accumulator	T

Example:

COMMAND	DISPLAY
IR = 5	(REG 5) = hhhh
IR = F	(REG F) = hhhh
IR = 10	(REG S) = hhhh
IR = 17	(REG T) = hhhh

If a "SM" command follows an "IT" command, that register's contents take the value of the SM's operand.

Example:

IR = 0
SM = 1
SN = 2
SN = 3

will set Register "0" to "1", Register "1" to "2", and Register "2" to "3".

SR = hhhh* Store Registers 0 thru 15 and the PSW into memory beginning at location "hhhh." Eighteen half-words are copied to memory ("hhhh" is a half-word address, See NOTE 1).

Displays: "ACCOMPLISHED".

EX = hhhh* — Execute command, transfers control to the instruction located at hhhh. All registers (R0 thru R15 and the PSW) are initialized prior to passing control. If hhhh* is zero, then the contents of the stores "PSW" determines the starting address. If "hhhh" = D then restart DOS. No display.

LG = hhhh* — Loads Registers 0 thru 15 and the PSW from memory beginning at location hhhh, then execution resumes using this PSW. "hhhh" is a halfword address, (See NOTE 1).

NOTE 1: For operation codes "LM", "SR", "EX", "LG", and "EB", the address "hhhh" is a halfword designator.

If:
- "hhhh" = "A" then use the contents of the accumulator.
- "hhhh" = "B" then use the "BIAS" pointer.
- "hhhh" = "C" then use the "CURRENT" pointer.
- "hhhh" = 0 then use the contents of the current address.

Otherwise, "hhhh" is the address operand.

For Search Under Mask, the user is provided with a search memory capability. Five commands are defined:

COMMAND	
MA	Set up a mask register.
LA	Lower or starting address.
HA	Higher or terminal address.
FI	Find or initiate a search under the mask.
CO	Continue the search for the next find.
MA = hhhh	Sets the mask register. (Note: If the operand is zero, all bits are set.) Displays: (MASK) = hhhh.
LA = hhhh*	Sets the low or starting address. Displays: (LOW) = hhhh.
HA = hhhh*	Sets the high or terminal address. Displays: (HIGH) = hhhh.
FI = hhhh	Initiates the search, beginning at "LA" and advancing through memory stopping only upon a "FIND" or at "HA". NOTE: "LA" need not be less than "HA", Displays: (FAIL = hhhh for no find (hhhh: High address) Displays: (hhhh) = hhhh upon a find (hhhh: Address of a find; hhhh: Contents of this address)
CO = hhhh	Will resume searching, beginning at the next cell plus h cells, this continues the previous successful find or "CA" command. The displays are identical with those associated with the "find" or "CA" operation.

For Snapshot and Check, there comes a time during software integration when the human operator is forced to act as a "computer" by studying "dumps" to discover what went wrong. To facilitate this type of investigation using the computer to perform a snapshot comparison to find all elements which have undergone change. The following two directives are defined:

"CT" Copy a block of information to another location in memory.

"CA" Check a block of information against another copy, noting all changes.

The formats are:

CT = hhhh* Will copy the block of information defined by "LA" and "HA" to another location in memory starting at hhhh. The "HIGH ADDRESS" (HA) cannot be less than the "LOW ADDRESS" (LA) and the duplicate block beginning at hhhh must lie either totally above or entirely below the source block. If these conditions are not met then the operation is aborted displaying "REJECTED". A successful termination is noted by displaying:

(THRU) = hhhh
↑
Top of duplicate block

NOTE: Duplicate block starts at the location given as the operand of the "CT" directive and extends thru the location displayed as "(THRU) = hhhh".

CA = hhhh* Will check the source block defined by (LA) and (HA) against another block starting at hhhh. The constraints imposed upon (LA), (HA) and hhhh are identical to those specified in the "CT" directive above. The test starts at (LA) and advanced to (HA). The comparison is under (MASK), see "MA" directive.

Displays:

"REJECTED" When the operation is aborted due to (LA), (HA) and hhhh incompatibility.

"ALL DONE" When the entire block has been checked.

"*(hhhh)	=	hhhh*	*hhhh"
↑ Address of mismatch		↑ Contents of source	↑ Contents of duplicate

Example:

LA = 1000	(LOW)	= 1000
HA = 2000	(HIGH)	= 2000
CT = 4000	(THRU)	= 5000
.		.
.		.
.		.
.		.
CA = 4000	*(1735)	= 2165**2160

Means: The contents of 1735 is 2165 while location 4735 contains 2160.

Testing begins with (LA) and advances to (HA) stopping either when all done or when the two values being checked differ by one or more bit(s) under the field defined by (MASK).

Concerning Breakpoints and Patches, often during software integration, there are occasions when the author wishes to modify a program in memory without reverting to source revisions and the associated re-assembly and reloading.

The following directives are defined:

B$ Sets a breakpoint.

P$ Establishes a patch (branch).

Z$ Zaps (removes) breakpoint(s)/patch(es).

C$ Continues execution from last breakpoint.

In general, a breakpoint will interrupt processing and display "(BREAK) = hhhh* where hhhh is the location of the breakpoint executed. The registers R_0 thru R_{15} and the PSW reflect the processing state when the breakpoint instruction was encountered. These may be examined via the "IR" directive and altered using the "SM", "SN" ... directives. The "C$" directive removes the breakpoint just encountered and resumes execution. Using the stored (?modified?) state parameters. No functional differences can be detected between regular execution of a program and the encounter of a breakpoint followed by a "C$" (resume execution) command.

Patches on the other hand alter the execution sequence, i.e. effects a branch. Typical applications fall into two categories: (1) to jump over or skip part of the regular execution sequence, and (2) to substitute alternate logic at some point in the existing program.

Patches and breakpoints can be removed as easily as they can be inserted, in fact, the program can be restored to its original configuration with a single command removing multiple patches and breakpoints.

FORMATS

B$ = hhhh*

Establishes a breakpoint at location hhhh.

Displays: "(BRK.) = hhhh"
↑Address at which breakpoint is set.

(See note below.)

P$ = hhhh*

Establishes a patch at the current location to effect a branch to location hhhh if hhhh is zero, the (BIAS) is used. The "current cell" is updated to point hhhh.

Displays: "(PATCH) = hhhh"
↑Location patched

Examples:

IM = 1000	(1000)	= XXXX
P$ = 2000	(PATCH)	= 1000
SM = XXXX	(2000)	= XXXX

Explanation:

Instruction XXXX at location 1000 has been converted to a branch to location 2000 and at location 2000 the patched instruction is duplicated. The user is responsible for insuring that there will be an intelligent instruction sequence at the "PATCHED TO" location.

NOTE: Applicable to both B$ and P$. Breakpoints and patches must be set in a program stream, i.e., they can be placed anywhere a short or long instruction could be located.

When the program counter selects a patch instruction or a breakpoint code the processing state is recorded in registers R_0 thru R_{15} and the PSW.

C$ = hhhh

Resumes processing by restoring the breakpointed instruction and processor status, then continue execution from the breakpoint plus hhhh cells.

NOTE: hhhh specified a halfword count, not a byte reference.

If not preceeded by a breakpoint detection, then this command is aborted displaying "REJECTED". There is no display when this command is successfully executed.

Z$ = hhhh

Removes the breakpoint or the patch at location hhhh and restores the program instruction which was originally at that location.

Displays: "REJECTED" if there was not a patch or breakpoint at the specified location.

Displays: "(hhhh) = hhhh" after completion.
(hhhh: Address; hhhh: Restored instruction)

NOTE: If hhhh = 0, then all outstanding patches and breakpoints are restored.
If hhhh = 1, then all outstanding breakpoints are restored.
If hhhh = 2, then all outstanding patches are restored.

Displays: "ACCEPTED" after completion.

The power of this checkout monitor becomes apparent only through applying various combinations of directives against real applications programs.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: D189-90022-2	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE *(and Subtitle)*: Advanced Acoustic Sensor System Software	5. TYPE OF REPORT & PERIOD COVERED: Final 1973
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Greg Scallon and Charles Cretin	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Airborne Anti-Submarine Warfare and Surveillance Military Systems Division, The Boeing Company - P.O. Box 3999, Seattle, Washington 98124	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: April 3, 1974
	13. NUMBER OF PAGES: 93
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

[redacted] The Boeing Company without the prior approval of the Airborne Anti-Submarine Warfare and Surveillance Program organization.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

A Boeing IR&D Study

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Acoustic	Fixing Algorithms
Airborne ASW	Low Frequency Analysis
Directional Frequency Analysis	Real-Time
Doppler Fixing	Software
Fast Fourier Transform	Sonobuoy

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

As a result of acoustic sensor analyses, trade studies and evaluations completed during 1973, an acoustic sensor system concept was developed and components were selected, procured and installed in a Boeing ASW test airplane.

The applications software, designed and programmed for operation in the Interdata Model 85 computer, controls data acquisition, processing and

OVER

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Block 20. ABSTRACT (Continued)

display. The nucleus of this software is the real-time program which controls data acquisition from the analog-to-digital converter, performs the Fast Fourier Transform, line integration and line tracking algorithms, and communicates with the operator through the Hazeltine 2000 CRT/keyboard and the electrographic recorder displays.

In November 1973, an end-to-end test of the Acoustical Signal Processing and Analysis System (ASPAS) was run. The test employed an analog test tape, first with a sine wave with no acoustic noise, and then with a sine wave with noise added for approximately thirty minutes. The tests clearly demonstrated the capability of the ASPAS to detect and process signal-to-noise levels lower than -20 decibels.

The acoustic sensor system was exercised during six flight tests with varying degrees of success during 1973. This flight testing culminated in an operable system of hardware and software with the capability of recording acoustic data and subsequently playing it back on the ground for detail study and analysis.