This dissertation has been microfilmed exactly as received 69-15,970

SVOBODA, Dean Edward, 1937-
PHASED ARRAY APPLICATIONS OF ANTENNAS WITH INTEGRATED ELECTRONIC CIRCUITS.

The Ohio State University, Ph.D., 1969
Engineering, electrical

University Microfilms, Inc., Ann Arbor, Michigan
PHASED ARRAY APPLICATIONS OF ANTENNAS
WITH INTEGRATED ELECTRONIC CIRCUITS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By


*****

The Ohio State University
1969

Approved by

[Signature]
Adviser
Department of
Electrical Engineering
PLEASE NOTE:

Not original copy. Several pages have indistinct print. Filmed as received.

UNIVERSITY MICROFILMS.
ACKNOWLEDGMENT

I wish to acknowledge the advice and encouragement given by Professor C.H. Walter without which the research reported here might not have been completed. I would like to thank Mr. Niles A. Walker who took care of the greatest part of the hardware details for the experimental array. Other contributors to the research effort who deserve recognition are Messrs. Jonathan D. Young, A. Jennetti and C.H. Boehnker. I would also like to thank Mr. John R. Copeland for the use of his three-dimensional plotting program and other computer software.
VITA

February 17, 1937 . . . . Born, Cleveland, Ohio

1961. . . . . . . . . . . . B.E.E., The Ohio State University
            Columbus, Ohio

1963. . . . . . . . . . . M.Sc., The Ohio State University
            Columbus, Ohio

1961-1969 . . . . . . . Research Assistant,
            Research Associate,
            Assistant Supervisor, ElectroScience
            Laboratory, The Ohio State University
            Columbus, Ohio
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gamma matched dipole: (a) without electronics, (b) with integrated transistor amplifier, (c) with integrated amplifier and gain control, (d) with integrated amplifier, gain control and phase shifter</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Cavity-backed-slot antenna with coaxial transistor mounted in the cavity wall</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Block diagram showing the components which were integrated into the digitally-controlled slot antenna</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Digitally-controlled slot antenna with integrated amplifier, phase and gain control: (a) photograph of an assembled element, (b) sketch showing an expanded view</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Block diagram of receiving array of individually controllable elements arranged for adaptive pattern control</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>Adaptive array of Widrow, Mantey, Griffiths and Goode</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>Two-mode adaptation of Widrow, et al. The 5's are set to simulate the angle-of-arrival of the desired signal</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>Single-mode adaptation of Widrow, et al.</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>Phase-locked loop for obtaining a reference signal for the adaptive processor</td>
<td>23</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>10</td>
<td>Power pattern as a function of the number of adaptations for a 10-element, half-wavelength spaced linear adaptive array of omnidirectional elements. Interfering signal was added at the angle of the first sidelobe after approximately 55 adaptations</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>Power pattern as a function of the number of adaptations for a five-element, half wavelength spaced adaptive array showing the ability of the array to track the desired signal</td>
<td>29</td>
</tr>
<tr>
<td>12</td>
<td>Power pattern as a function of the number of adaptations showing the performance of a five-element half-wavelength spaced adaptive array with a 100 percent modulated AM desired signal and an unmodulated sinusoidal reference</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>Pattern plots for from two through five elements showing the increasing ability of the &quot;trial-and-error&quot; adaptive array to null an interfering signal as the number of elements is increased</td>
<td>34</td>
</tr>
<tr>
<td>14</td>
<td>Pattern plots for the &quot;trial-and-error&quot; five element adaptive array with a desired signal and various amounts of noise showing the decrease in sidelobe level with increasing noise</td>
<td>35</td>
</tr>
<tr>
<td>15</td>
<td>Three-dimensional pattern plot for the &quot;trial-and-error&quot; adaptive array for various conditions of desired signal, interfering signal and noise</td>
<td>37</td>
</tr>
<tr>
<td>16</td>
<td>Block diagram of the experimental digitally-controlled antennafier array</td>
<td>39</td>
</tr>
<tr>
<td>17</td>
<td>Flow diagram for the search-and-track algorithm</td>
<td>40</td>
</tr>
<tr>
<td>18</td>
<td>Sketch showing the parameters for the two-beam interpolation method</td>
<td>45</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>19</td>
<td>DF error as a function of angle-of-arrival for a simulated DF system using two-beam interpolation</td>
<td>47</td>
</tr>
<tr>
<td>20</td>
<td>Scatter diagram of DF error for the simulated DF system with 34 dB S/N using two-beam interpolation</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 20</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>Scatter diagram of DF error for the simulated DF system with 40 dB S/N using two-beam interpolation</td>
<td>51</td>
</tr>
<tr>
<td>23</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 22</td>
<td>52</td>
</tr>
<tr>
<td>24</td>
<td>Scatter diagram of DF error for the simulated DF system with 46 dB S/N using two-beam interpolation</td>
<td>52</td>
</tr>
<tr>
<td>25</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 24</td>
<td>52</td>
</tr>
<tr>
<td>26</td>
<td>Scatter diagram of DF error for the simulated DF system with 54 dB S/N using two-beam interpolation</td>
<td>53</td>
</tr>
<tr>
<td>27</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 26</td>
<td>53</td>
</tr>
<tr>
<td>28</td>
<td>Scatter diagram of DF error for the simulated DF system with 60 dB S/N using two-beam interpolation</td>
<td>53</td>
</tr>
<tr>
<td>29</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 28</td>
<td>54</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>30</td>
<td>Sketch of the electronic scan pattern showing parameters for the three-beam parabolic interpolation</td>
<td>54</td>
</tr>
<tr>
<td>31</td>
<td>DF error as a function of angle-of-arrival for a simulated DF system using the three-beam parabolic interpolation method</td>
<td>56</td>
</tr>
<tr>
<td>32</td>
<td>Scatter diagram of DF error for the simulated DF system with 34 dB S/N using three-beam parabolic interpolation</td>
<td>57</td>
</tr>
<tr>
<td>33</td>
<td>RMS DF error calculated for the scatter diagram of Fig. 32</td>
<td>57</td>
</tr>
<tr>
<td>34</td>
<td>Scatter diagram of DF error for the simulated DF system with 40 dB S/N using three-beam parabolic interpolation</td>
<td>58</td>
</tr>
<tr>
<td>35</td>
<td>RMS DF error calculated for the scatter diagram of Fig. 34</td>
<td>58</td>
</tr>
<tr>
<td>36</td>
<td>Scatter diagram of DF error for the simulated DF system with a 46 dB S/N using three-beam parabolic interpolation</td>
<td>59</td>
</tr>
<tr>
<td>37</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 36</td>
<td>59</td>
</tr>
<tr>
<td>38</td>
<td>Scatter diagram of DF error for the simulated DF system with 54 dB S/N using three-beam parabolic interpolation</td>
<td>59</td>
</tr>
<tr>
<td>39</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 38</td>
<td>59</td>
</tr>
<tr>
<td>40</td>
<td>Scatter diagram of DF error for the simulated DF system with 60 dB S/N using three-beam parabolic interpolation</td>
<td>60</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>41</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 40</td>
<td>60</td>
</tr>
<tr>
<td>42</td>
<td>Photographs of the experimental array: (a) front view, (b) rear view</td>
<td>62</td>
</tr>
<tr>
<td>43</td>
<td>Sketch of the experimental array showing the dimensions</td>
<td>63</td>
</tr>
<tr>
<td>44</td>
<td>Measured beam patterns for the experimental four-element array</td>
<td>66</td>
</tr>
<tr>
<td>45</td>
<td>Angular error of the pattern range</td>
<td>69</td>
</tr>
<tr>
<td>46</td>
<td>Flow diagram for the subroutine RAD</td>
<td>72</td>
</tr>
<tr>
<td>47</td>
<td>Flow diagram for the subroutine CALIB</td>
<td>73</td>
</tr>
<tr>
<td>48</td>
<td>Flow diagram for the subroutine BMDAT</td>
<td>74</td>
</tr>
<tr>
<td>49</td>
<td>Flow diagram for the subroutine SETBM</td>
<td>78</td>
</tr>
<tr>
<td>50</td>
<td>Flow diagram for the search-and-track program. Execution monitoring and control functions are not shown</td>
<td>80</td>
</tr>
<tr>
<td>51</td>
<td>Flow diagram for the DF-accuracy program. Execution monitoring and control functions and data-output functions are not shown</td>
<td>85</td>
</tr>
<tr>
<td>52</td>
<td>Measured DF error as a function of angle-of-arrival for the experimental array using two-beam interpolation</td>
<td>87</td>
</tr>
<tr>
<td>53</td>
<td>Scatter diagram of measured DF error for the experimental array with 34 dB S/N using two-beam interpolation</td>
<td>88</td>
</tr>
<tr>
<td>54</td>
<td>Measured DF error as a function of angle-of-arrival for the experimental array using the three-beam interpolation method with no noise</td>
<td>89</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>55</td>
<td>Scatter diagram of measured DF error for the experimental array with 40 dB S/N using three-beam parabolic interpolation. Data were taken with the pattern range set up used for the data of Fig. 54.</td>
<td>91</td>
</tr>
<tr>
<td>56</td>
<td>RMS measured DF error calculated from the scatter diagram of Fig. 55</td>
<td>92</td>
</tr>
<tr>
<td>57</td>
<td>Average measured DF error calculated from the scatter diagram of Fig. 55</td>
<td>92</td>
</tr>
<tr>
<td>58</td>
<td>Measured no-noise DF error for the experimental array using three-beam parabolic interpolation</td>
<td>93</td>
</tr>
<tr>
<td>59</td>
<td>Scatter diagram of measured DF error for the experimental array with 34 dB S/N using three-beam parabolic interpolation. Data was taken with the pattern range set up used for the data of Fig. 58.</td>
<td>94</td>
</tr>
<tr>
<td>60</td>
<td>RMS DF error calculated from the scatter diagram of Fig. 59</td>
<td>95</td>
</tr>
<tr>
<td>61</td>
<td>Average measured DF error calculated from the scatter diagram of Fig. 59</td>
<td>95</td>
</tr>
<tr>
<td>62</td>
<td>Timing diagram showing the formats of the slot-antennafier element control and clock signals</td>
<td>100</td>
</tr>
<tr>
<td>63</td>
<td>Block diagram showing the relation between the I/O converter, the central processor and the I/O devices</td>
<td>169</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Measured phase shift for phase shifters</td>
<td>65</td>
</tr>
<tr>
<td>2.</td>
<td>Phase shifter settings used for DF experiments</td>
<td>67</td>
</tr>
<tr>
<td>3.</td>
<td>Element phase shift as a function of phase register bits P₁ through P₄</td>
<td>102</td>
</tr>
<tr>
<td>4.</td>
<td>Element gain as a function of amplitude register bits A₁ through A₄</td>
<td>103</td>
</tr>
<tr>
<td>5.</td>
<td>CPU registers</td>
<td>158</td>
</tr>
<tr>
<td>6.</td>
<td>Sense functions</td>
<td>160</td>
</tr>
<tr>
<td>7.</td>
<td>Action taken on overflow</td>
<td>162</td>
</tr>
<tr>
<td>8.</td>
<td>CPU instructions</td>
<td>163</td>
</tr>
<tr>
<td>9.</td>
<td>CPU instruction timing</td>
<td>168</td>
</tr>
<tr>
<td>10.</td>
<td>Control character codes</td>
<td>171</td>
</tr>
<tr>
<td>11.</td>
<td>I/O instructions</td>
<td>174</td>
</tr>
<tr>
<td>12.</td>
<td>Teletype I/O code</td>
<td>176</td>
</tr>
<tr>
<td>13.</td>
<td>Plotter code</td>
<td>177</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................. ii
VITA ........................................................................... iii
LIST OF ILLUSTRATIONS ........................................... iv
LIST OF TABLES ......................................................... x

Chapter

I. INTRODUCTION ......................................................... 1
II. ANTENNAIERS FOR PHASED ARRAYS .................. 4
III. PHASED ARRAY APPLICATIONS FOR ANTENNAIERS ............................... 13
   A. Introduction ........................................... 13
   B. Adaptive Receiving Array .................. 14
   C. Direction Finding Array .................... 38
IV. THE EXPERIMENTAL ARRAY ................................. 61
   A. Introduction ........................................... 61
   B. The Four-Element Array .................. 61
   C. Measured Patterns .......................... 66
   D. Receiver ........................................... 69
   E. Computer Programming .................. 70
   F. Search And Track ......................... 79
   G. Direction Finding .......................... 84
V. SUMMARY AND CONCLUSIONS .............................. 96
<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>CONTROL SIGNAL CODE FOR THE SLOT</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>ANTENNA FIER</td>
<td></td>
</tr>
<tr>
<td>B.</td>
<td>CALRD</td>
<td>104</td>
</tr>
<tr>
<td>C.</td>
<td>SETBM</td>
<td>113</td>
</tr>
<tr>
<td>D.</td>
<td>TWO BEAM INTERP</td>
<td>118</td>
</tr>
<tr>
<td>E.</td>
<td>THREE BEAM INTERP</td>
<td>122</td>
</tr>
<tr>
<td>F.</td>
<td>ADDITIONAL SUBROUTINES</td>
<td>127</td>
</tr>
<tr>
<td>G.</td>
<td>SEARCH AND TRACK PROGRAM</td>
<td>129</td>
</tr>
<tr>
<td>H.</td>
<td>DF TEST PROGRAM</td>
<td>142</td>
</tr>
<tr>
<td>I.</td>
<td>CONTROL COMPUTER CHARACTERISTICS</td>
<td>157</td>
</tr>
<tr>
<td>J.</td>
<td>THE ASSEMBLER</td>
<td>179</td>
</tr>
<tr>
<td>K.</td>
<td>LIBRARY INDEX</td>
<td>188</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>194</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

Over the past decade considerable research effort has been directed toward the investigation of the possible benefit which can be obtained by integrating antennas and active electronic circuits. [1] Most of the current interest in antennas with integrated circuitry has been generated by E. M. Turner, AVWE, who suggested that integrated design of an antenna and its associated electronic circuitry can both simplify an antenna system and improve its performance.

Antennas and circuits may be integrated to the extent that it is impossible to electrically separate the resulting device into a passive antenna and an active circuit or the integration may be only mechanical whereby otherwise unused space on the antenna structure is used in packaging associated circuitry. Integrated antennas and circuits have been variously called antennafiers, antennaverters, antennamitters, antennaceivers (for antenna-amplifier, antenna-converters, etc.) according to the function of the active circuitry.

One of the most natural uses for antennas with integrated electronic circuitry is as phased array elements. Phased arrays are potentially
among the most sophisticated types of antennas and can perform functions which are difficult or impossible for single aperture antennas.

The elements of a phase array can be arranged to form a wide variety of aperture shapes and can be arranged conformally over non-planar surfaces allowing the use of a phased array where a conventional large aperture antenna such as a dish cannot be used.

Uses for phased arrays include radar, communications and direction finding. For radar applications the array can be scanned for surveillance or mapping or it can be made to track a target. If more than one target is to be tracked the array can be time shared between targets by rapidly switching the beam from one target to the next or multiple beams can be formed with a separate beam for each target.

For communications use the array can track the angle-of-arrival of a signal and provide information to retrodirectively position the beam of a transmitting antenna. The receiving pattern can also be adaptively adjusted to maximize signal-to-noise ratio in the presence of interfering signals. Angle tracking techniques applicable to radar and communications can also be used for direction finding. The array can be switched rapidly between different modes which allows the possibility of time sharing the various functions.
Most of the above array functions require individual control of element excitation. This study describes the development of an integrated receiving element having individually controllable gain and phase. Two specific phased array applications are considered that require the individual gain and phase control.

Chapter II briefly describes the work leading up to the development of the integrated receiving element and the element itself. In Chapter III the application of the integrated receiving element to two types of phased array systems is discussed. One type of array, the adaptive receiving array, has the ability to maximize signal-to-noise ratio of a received signal in the presence of interference. The other type is a direction-finding array. For the direction-finding array various methods of processing received signal information to determine angle-of-arrival are considered and weak signal performance is determined. Chapter IV describes an experimental direction-finding system using an array of four of the integrated receiving elements. Experimental verification of the direction-finding concepts of Chapter III is given.
CHAPTER II
ANTENNAFIERS FOR PHASED ARRAYS

Early integrated antenna-circuit work at the Ohio State University
and elsewhere was concerned primarily with modifying or improving
the performance of antennas such as dipoles, conical spirals, and
log-periodic dipole arrays.[2] Initially, very little consideration
was given to use of antennafiers as elements of pattern-controllable
phased arrays.

The first work at the Ohio State University concerned with the
use of antennafiers as phased array elements was done with half-wave
dipoles as shown in Fig. 1.

Figure 1a shows a conventional gamma-matched dipole with a
quarter-wave choke added to the support to suppress unbalanced
currents on the feedline. A phased array of the conventional gamma-
matched dipoles can be formed by connecting the feedlines to a suitable
phasing network. The performance, however, will be degraded by loss
in the feedlines and in the phasing network. For receiving applications
this has the effect of decreasing the received signal-to-noise ratio.
Not only is the signal attenuated but thermal noise generated in the
lossy components is added to the signal. These effects can be overcome
Fig. 1.--Gamma matched dipole: (a) without electronics, (b) with integrated transistor amplifier, (c) with integrated amplifier and gain control, (d) with integrated amplifier, gain control and phase shifter.
by integrating transistor RF amplifiers into the dipoles and thereby increasing the signal strength enough to mask the effects of attenuation and thermal noise in the feedlines and phasing networks. For a given signal strength the signal-to-noise ratio is then determined by the external noise and the noise generated in the RF amplifiers.

A 150 MHz gamma-matched dipole antennafier suitable for use in phased arrays was constructed and evaluated by Copeland and Robertson. [3] The circuit is shown in Fig. 1b. Performance of the dipole antennafier was evaluated by comparison with a similar passive dipole. The circuit gain was determined to be 12 dB over the passive dipole. The noise temperature was 400 °K. This temperature represents a noise figure of about 2 dB better than that obtained with the passive dipole and a conventional amplifier using the same transistor. The improvement was attributed to the effects of eliminating the loss associated with the feedline and the additional matching circuit in the separate RF amplifier.

A simple modification of the above antennafier, as shown in Fig. 1c, allows the amplifier to be gain controlled by means of varying the base bias. The amplitude control function of the feed network is integrated with the element thus simplifying the array hardware requirements.

A further degree of sophistication can be added in the form of an integrated phase shifter. Verstraete has demonstrated this with a
two element gamma-matched dipole array.[4] The circuit used with the elements is shown in Fig. 1d. This circuit is similar to that of Fig. 1c except the output circuit is replaced by a pi network having two voltage variable capacitor (varactor) elements. In addition to providing adjustable phase shift the pi network also functions as a voltage variable matching device. Verstraete obtained phase shifts of 110 degrees and measured patterns with various amounts of phase shift which agreed well with the theoretical patterns calculated for the same phase shifts. Young[5] expanded the array to four elements and modified the phase shifters to give increased phase shift obtainable by adding an additional varactor in parallel with each of the varactors in the output arms of the pi networks. Patterns were measured which showed a beam scanning capability.

The phase-and-amplitude controlled dipole antenna offers quite a bit of flexibility as an array element since it allows amplitudes and phases of individual elements to be adjusted independently. The problem of controlling the amplitude and phase distribution, however, is complicated by the requirement of three nonlinear control voltages for each element. Also the phase shift range is limited. The phase shift range could have been extended by adding more pi-network stages to the phase shifters but this would have further increased the required number of control voltages.
At this point Young and the author working together concluded that digital control of gain and phase shift might be more practical. It was decided that to fully realize the flexibility inherent in individual control of gain and phase shift a stored program digital computer would be needed to control the array. Interfacing of the control computer and the array elements would be simplified by using digital phase shifters and digital gain control circuitry. The use of digital control would also reduce the control wiring required since a coded control signal could be sent in "party-line" fashion to all elements while addressing only the element to which the code would be specific.

About this time Fujimoto and Jennetti were working to develop a 1 GHz cavity-backed-slot antennafier using a coaxial transistor mounted in the cavity wall as shown in Fig. 2. This seemed to be an appropriate element to use in experimenting with a digitally controlled antennafier array since RF circuitry (phase shifter and matching network) could be fabricated using stripline and mounted flush to the cavity wall. A printed circuit digital control board could be mounted similarly with very little increase in the space used.

After evaluating a number of types of phase shifters Young decided to use a diode-switched delay line phase shifter. This type gave potentially broadband performance and was most easily fabricated in stripline. The final design was a four-bit phase shifter with $\frac{\pi}{8}$ phase
Fig. 2.—Cavity-backed-slot antennafier with coaxial transistor mounted in the cavity wall.
increments from 0 to $\frac{15\pi}{8}$ at 1 GHz. The bandwidth over which a non-dispersive time delay could be obtained was two to one.

Figure 3 shows a block diagram of a digitally controlled slot antennafier element. The amplifier gain is controlled by switching various combinations of bias resisters into the emitter of the amplifier transistor according to a four-bit digital code. Sixteen gain settings are possible. The gain control circuitry, essentially a nonlinear digital-to-analog converter, was designed by Young and later modified by Walker.[6]

Fig. 3.--Block diagram showing the components which were integrated into the digitally-controlled slot antennafier.
The digital control signal consists of a serial pulse train containing a four-bit element address, a four-bit phase control code and a four-bit amplitude control code. The control circuitry decodes the element address. If it agrees with the element number, the phase and amplitude codes are stored in the phase and amplitude storage registers, otherwise the control signal is ignored. This allows all control signals to be fed to all elements in parallel. Details of the element construction are given in Reference 6.

Figure 4a is a photograph of a completed element showing the digital control circuit board. The stripline phase shifter is between the control board and the cavity wall as indicated by the expanded view in Fig. 4b.

The amplifier performance was less than spectacular with a noise temperature of 3400*K and a gain of 7 dB. The transistor also seemed to be especially susceptible to burnout and has since been discontinued by the manufacturer. When the available supply of transistors was reduced to about six it was decided to redesign the amplifier using a newer transistor. The array element used in the experimental array to be discussed in Chapter III has no amplifier and is therefore strictly speaking an integrated antenna and phase shifter rather than an antennafier. New amplifiers are being installed at the time this is being written and are described in detail in Reference 7.
Fig. 4.--Digitally-controlled slot antennafier with integrated amplifier, phase and gain control:
(a) photograph of an assembled element,
(b) sketch showing an expanded view.
CHAPTER III
PHASED ARRAY APPLICATIONS FOR ANTENNAFIERS

A. Introduction

An element such as the digitally controlled slot antennafier receiving element described in Chapter II, because it allows both independent control of the gain of individual array elements and independent control of the phase shift of individual array elements, makes an ideal phased array element. In conformal arrays, for example, there will in general be no systematic way of setting amplitude and phase distribution (such as uniform amplitude and linear phase) so individual control of array elements will be required. For conventionally shaped apertures (planar, linear, etc.) where beam shaping or multiple beam capabilities are desired individual element control again may be required. This chapter discusses two array applications for which the antennafier element would be quite useful, an adaptive receiving array and a direction finding array, both of which require individual control of the array elements.
B. Adaptive Receiving Array

A receiving array with a great deal of flexibility of pattern control can be obtained by conventionally summing the signals from individually gain- and phase-controllable array elements. The summing can be done by means of such techniques as a corporate feed or summing amplifiers. Figure 5 shows a block diagram giving the functional relations of the components of a typical system. The elements can be arranged in a circle for omnidirectional coverage or in a straight line or conformally over a surface for sectoral coverage. Amplitude and phase of each element is set by the control computer.

If the received signal is properly processed and fed back to the control computer, adaptive pattern control can be realized. Adaptive pattern control can give a tremendous performance advantage in environments having relatively strong interfering signals, such as are encountered in HF and VHF communications, when compared with simple programmed beam positioning.

With programmed beam positioning, an interfering signal can conceivably be completely out of the main beam and still be only about 13 dB down (the amplitude of the first sidelobe for uniform illumination) from the desired signal. By adaptive pattern control a pattern null can be positioned in the direction of the interfering signal without, in many cases, significantly degrading the directivity
Fig. 5.—Block diagram of receiving array of individually controllable elements arranged for adaptive pattern control.

of the array pattern. This has been shown by Widrow, Mantey, Griffiths and Goode[8] by means of computer simulation of adaptive arrays.

In the case of the conformal array, even with no interfering signals, adaptive phasing techniques used with individually controllable elements would be extremely useful. If programmed phasing were used it would be necessary to store and have available the required amplitude and phase for each element for each beam angle.
for each frequency. For a 40-element array with fifty frequencies
over one hundred thousand pieces of data would be required. The
task of pre-calculating the required amplitudes and phases also
becomes formidable. In a large, regular array mutual coupling
between elements can be taken into account by considering the pattern
of a typical element in the presence of an infinite array of similar
elements. In an irregularly shaped three-dimensional array no
such typical element pattern exists and coupling between individual
elements must be treated on an individual basis. An adaptive technique
can be designed to adjust amplitudes and phases to maximize signal-
to-noise ratio regardless of mutual coupling and external effects such
as structural scattering.

It has been shown that, even for a small linear array of uniformly
spaced elements with individual RF amplifiers with no external noise,
adjusting the phasing to produce a beam in the direction of the signal
source does not necessarily give the best signal-to-noise ratio.[9]
Because of mutual coupling effects, adjusting the phasing to cause the
signals to add in phase does not produce, in general, a beam in the
direction of the signal source. In-phase summing of the signals from
the elements does, however, necessarily give the highest signal-to-
noise ratio since phasing does not affect the RF amplifier noise and
the in-phase condition gives the greatest signal strength. Thus, even
with a very simple array with no external noise where the required
phase control might seem obvious, adaptive control could provide superior performance.

Widrow, et al, discuss several methods of adaptive array control based on minimization of mean-squared error between the array output and a "desired" signal. The basic system shown by Fig. 6 is considered in some detail to illustrate the technique.

![Diagram of adaptive array system](image)

**Fig. 6.** --Adaptive array of Widrow, Mantey, Griffiths and Goode.

This system makes use of the "LMS" (least mean squares) algorithm which is based on the method of steepest descent. Corrections to the weights, \( w \), of Fig. 6, which control amplitude and phase distribution, are made in steps according to the equation
(1) \[ W(j + 1) = W(j) + k_s \nabla(j) \]

where

\[
\begin{align*}
W(j) &= \text{weight vector before adaptation,} \\
W(j+1) &= \text{weight vector after adaptation (one step),} \\
k_s &= \text{scalar constant controlling rate of convergence} \\
&\quad \text{and stability,} \\
\nabla(j) &= \text{estimated gradient of } \varepsilon^2 \text{ with respect to } W, \\
\varepsilon^2(j) &= \text{mean squared error between the reference signal} \\
&\quad \text{and the array output.}
\end{align*}
\]

It is shown that an unbiased estimate of the gradient can be obtained by using

(2) \[ \nabla(j) = -2 \varepsilon(j) X(j) \]

where

\[
\begin{align*}
\varepsilon(j) &= \text{error between the reference signal and the} \\
&\quad \text{array output} \\
X(j) &= \text{vector of signals at the inputs to the weight} \\
&\quad \text{control devices}
\end{align*}
\]

giving

(3) \[ W(j + 1) = W(j) - 2 k_s \varepsilon(j) X(j) \].
If a desired signal which "looks like" the reference signal is incident on the array, after a sufficient number of adaptation steps, the weights of Fig. 6 will be adjusted to give a pattern having a beam in the direction of the desired signal such that the error between the array output and the reference signal is minimum.

One of the problems in implementing an adaptive array is that of obtaining a reference signal that "looks like" the signal that is desired to be received (If an exact replica of the desired signal were available an antenna would not be needed). Widrow, et al, describe two methods of overcoming the problem. These are illustrated in Figs. 7 and 8. In the two-mode method of Fig. 7 the inputs to the adaptive processor, which would normally be connected to the array elements, are first connected to a set of signals which are obtained from a pilot signal generator and have the proper phases to simulate a signal having the angle of arrival of the desired signal. The reference signal is also obtained from the pilot signal generator. The processor is allowed to adapt for a short time after which the processor inputs are switched to the array elements and the reference is switched off (mode II). The processor is again allowed to adapt for a short time. Widrow, et al, have shown by means of a computer simulation that by switching frequently enough so that only small changes in the weights occur the pattern will converge to a pattern
Fig. 7.—Two-mode adaptation of Widrow, et al.

The $\delta$'s are set to simulate the angle-of-arrival of the desired signal.

having a beam in the direction of the desired signal and nulls in the
directions of undesired signals.

In the one-mode system of Fig. 8 both the array elements and
pilot signals are left connected to the adaptive processor while it
adapts. The outputs of the array elements are picked off before the
pilot signals are added and are fed to a slaved processor which
produces the useful array output free of the pilot signal.
Fig. 8.--Single-mode adaptation of Widrow, et al.

Both systems have several short-comings. As pointed out by Widrow, et al, the two-mode system can only receive the desired signal in mode II. In the one-mode system considerable difficulty would be experienced in trying to get the slaved processor to track the adaptive processor accurately enough to realize the deep nulls in the directions of interfering signals predicted by the computer simulations. The most serious short-coming of both of these systems
is, however, the requirement for a predetermined knowledge of phases (and amplitudes, which were not considered by Widrow, et al) produced by the desired signal for each possible frequency and angle of arrival. With this requirement the advantages of conformal placement of elements without a requirement for any detailed knowledge of mutual coupling on structural effects is lost. Widrow, et al, suggest and immediately reject the use of a pilot signal radiated by an antenna in the far-field of the array at the angle of arrival of the desired signal. This would be practical only for a fixed installation with a fixed angle of arrival for the desired signal.

A more practical approach is proposed here. For this approach the desired signal would be approximated to some degree and the approximation used as a reference. Differences between the simulated reference and the desired signal would appear as interference arriving from the direction of the desired signal. It would seem probable that a considerable amount of this "interference" might be tolerated by the adaptive processor. That this is indeed the case has been verified by a computer simulation.

A simulated desired signal can be obtained quite easily, as shown in Fig. 9, if the emitter of the desired signal cooperates by transmitting a pilot carrier. This is not much of a requirement in a communications system. Many communications transmitters already transmit carriers. The system of Fig. 9 extracts the pilot
carrier by means of a phase-locked loop. Phase-locked loops can be made to lock and operate at very low signal-to-noise ratios. The voltage controlled oscillator (VCO) signal has the frequency and phase of the desired-signal pilot carrier and is used for the reference. The array adapts to the desired signal pilot carrier and treats the remainder of the desired signal as interference arriving from the same direction.

One of the disadvantages of this system is that it is susceptible to jamming by a sinusoidal signal of the same frequency as the pilot carrier. The one- and two-mode systems of Widrow, et al, are not as susceptible.

A computer program was written to verify the results of Widrow, et al, and to test the performance of the LMS adaptation algorithm with interference arriving from the direction of the desired signal. In the
calculation, the desired signal and the various interfering signals were considered to be uncorrelated. Only the DC terms of Eq. (2) produced by each of the signals multiplying itself were retained. The signals were also considered to be sinusoidal and complex notation was used. A small amount of narrow band uncorrelated noise (as might be produced by RF amplifiers) was considered to be present at each element and to have the effect of producing an offset in the values of the \( \nabla \)'s.

Thus

\[
\epsilon_m = - E_m \sum_{n=0}^{N-1} w_n e^{-jn \delta_m}, \quad m > 0
\]

and

\[
\epsilon_0 = - E_0 \left[ \sum_{n=0}^{N-1} w_n e^{-j \delta_0} \right] - 1
\]

where

\( \epsilon_m \) = the complex error voltage produced by the \( m \)-th interfering signal,

\( \epsilon_0 \) = the complex error voltage produced by the desired signal (the reference has unity amplitude),

\( E_m \) = the amplitude of the \( m \)-th interfering signal,

\( E_0 \) = the amplitude of the desired signal,

\( w_m \) = the complex weighting coefficient for the \( n \)-th array element,
\( \delta_m = 2\pi d_\lambda \cos \theta_m \) = the phase delay between elements for the \( m \)-th interfering signal where \( d_\lambda \) is the spacing between elements in wavelengths and \( \theta_m \) is the angle of arrival of the signal and,

\[
(6) \quad x_{m,p} = E_m e^{-jp \delta_m}
\]

where

\( x_{m,p} \) = the signal produced at the \( p \)-th element by the \( m \)-th interfering signal (or the desired signal for \( m = 0 \)).

Combining Eqs. (4), (5) and (6) gives

\[
(7) \quad \hat{V}_p = 2 \left[ \sum_{m=0}^{M} \sum_{n=0}^{N-1} E_m e^{j(p-n)\delta_m} - E_0^2 e^{jp\delta_0} \right] + 2 V_{\text{noise}}^2 w_p
\]

where

\( \hat{V}_p \) = the \( p \)-th component of the gradient,

\( V_{\text{noise}} \) = the amplitude of the uncorrelated noise added at each element.

The \( \hat{V}_p \)'s are complex. The real parts when multiplied by an appropriate gain constant provide correction to the real parts of the weighting coefficients. The imaginary parts provide the corrections to the imaginary parts of the weighting coefficients. Thus
(8) \[ W(i+1) = W(i) + k_s \vec{\nabla} \]

where

\[ W(i+1) = \text{the complex weight vector after the i-th} \]

adaptation,

\[ W(i) = \text{the complex weight vector before the i-th} \]

adaptation,

\[ k_s = \text{scalar real gain constant}, \]

\[ \vec{\nabla} = \text{complex gradient vector}. \]

Equation (8) is identical in appearance to Eq. (1) but in Eq. (8) the components are complex and the gradient is obtained by an average rather than as an instantaneous value and represents the exact value rather than an estimate.

Figure 10 is a three-dimensional plot showing the power pattern of a 10-element half-wavelength spaced linear array of omnidirectional elements as a function of the number of adaptations using the LMS algorithm and the calculation method described above. The process was started with the first weighting coefficient set to unity and the rest to zero. With no interference or noise the pattern would remain uniform and unchanged indefinitely since the error would be zero (unity was used for the reference and desired signal amplitudes). The angle-of-arrival of the desired signal was taken to be 45 degrees.
Fig. 10.—Power pattern as a function of the number of adaptations for a 10-element, half-wavelength spaced linear adaptive array of omnidirectional elements. Interfering signal was added at the angle of the first sidelobe after approximately 55 adaptations.

Uncorrelated noise ($V_{\text{noise}}$ of Eq. (7) equal to 0.3) was added at each element. The expected result would be for the array to form a highly directive beam in the direction of the desired signal in order to minimize the error between the array output and the reference. The adaptation process proceeds by keeping the array response approximately constant at unity for 45 degrees and reducing the response at other angles.
After approximately 55 adaptations an interfering signal of amplitude $E_1 = 0.1$ was introduced at the angle of the first sidelobe. The sidelobe was removed within a few adaptations. The interfering signal was removed at about 75 adaptations. The other sidelobes which had begun to grow began to recede again. After sufficient time the missing sidelobe would probably have reappeared.

Figure 11 is a three dimensional plot similar to that of Fig. 10. Twice as many adaptations are shown for a 5-element array. The desired signal was set at 135 degrees. After about 35 adaptations an interfering signal was applied with noticeable effect at the angle of the second sidelobe from where the main beam was forming. After about 20 more adaptations the interfering signal was removed. The angle-of-arrival of the desired signal was then stepped 5 degrees at a time to 90 degrees demonstrating the ability of the array to track a desired signal.

Of particular interest is the plot of Fig. 12 which was calculated to show the performance of the adaptive array system for a 100 percent modulated AM signal and a sinusoidal reference as might be used with the phase-lock technique described earlier and shown in Fig. 9. Sinusoidal modulation was assumed. The two sidebands were represented as two interfering signals arriving from the angle of the desired signal. The angle of arrival was taken to be 135 degrees.
Fig. 11.—Power pattern as a function of the number of adaptations for a five-element, half-wavelength spaced adaptive array showing the ability of the array to track the desired signal.

For the AM signal case the array response in the direction of the desired signal which gives the least mean-squared error will not be unity since the response must provide a compromise between the carrier which correlates with the reference and the error produced by the sidebands which do not correlate with the reference. Therefore, the response should be less than unity. As can be seen from Fig. 12 the processor quickly adjusts to reduce the array response at the angle of arrival of the desired signal at the expense of increasing the
Fig. 12. -- Power pattern as a function of the number of adaptations showing the performance of a five-element half-wavelength spaced adaptive array with a 100 percent modulated AM desired signal and an unmodulated sinusoidal reference.

response at other angles. This behavior is similar to that which was noticed with only a desired signal and a single interfering signal but no noise, i.e., no beam was formed but a null was formed in the direction of the interfering signal. In the present case, however, it can be seen that the noise eventually causes the response at angles other than that of the desired signal to be reduced.

One of the drawbacks of the LMS adaptation method is that it requires a separate receiver channel for each element so that received
signals can be multiplied by the error to obtain corrections. Equipment must also be provided for reading and storing the correction values. The required equipment is essentially multiplied by the number of array elements.

A simpler method using less equipment and which is particularly suited for use with an array of digitally controlled antennafier elements can be used. For this method error detecting circuitry is the same as for the LMS algorithm but weight adjustments are determined by trial and error to give the minimum mean-squared error. Each of the weights (gain and phase in the case of the antennafier array) are sequentially adjusted to minimize the mean-squared error. The process can be continued until no further reduction in error occurs.

Some of the properties which require investigation are the rate of convergence and the parameters which affect the rate of convergence and the stability of the adaptive process. The spectacular ability of the adaptive array to null interfering signals can be expected to be reduced when discrete weight values are used since the exact values required to give perfect nulls will not in general be available. A trade off between the number of elements in the array and the size of phase and amplitude increments can be expected.

A simulation program was written to test the performance of an adaptive system using the trial-and-error method with discrete phase
and amplitude values. The same assumptions with regard to the lack of correlation between the various signals and noise made for the LMS algorithm were made for the trial-and-error method. The contribution of each of the signals to the error was obtained from Eq. (4) or (5). The noise contribution was taken to be proportional to the weighting coefficient (amplitude) giving the expression:

\[
\tilde{\varepsilon} = \sum_{m=0}^{M} |\varepsilon_m|^2 + \sum_{n=0}^{N-1} V_{\text{noise}}^2 A_n^2
\]

where
\[
\varepsilon_m \text{ is given by Eqs. (4) or (5),}
\]
\[
M = \text{the number of interfering signals,}
\]
\[
N = \text{the number of elements,}
\]
\[
V_{\text{noise}} = \text{noise voltage amplitude,}
\]
\[
A_n = \text{amplitude of the } n\text{-th element,}
\]
\[
\tilde{\varepsilon}^2 = \text{the mean-squared error.}
\]

The phase and amplitude were adjusted for minimum error starting with phase of the last element followed by the amplitude of the last element and proceeding in sequence to the first element and then back to the last to repeat the sequence. Each parameter was adjusted for minimum error before proceeding to the next. Phase increments of \(\frac{\pi}{6}\) were used and the amplitude increments were made variable.
Figure 13 shows pattern plots obtained for from two through five elements after the adaptive process had proceeded to the point where no further reduction in error occurred. These plots illustrate the increase in null depth with increasing number of elements. In each case an interfering signal of amplitude 0.5 at an angle-of-arrival of 135 degrees and a desired signal of unity amplitude at an angle-of-arrival of 45 degrees were simulated. No noise was used. The amplitude increments were set to that each increment would produce a change in the desired signal output equal to 1/16 of the reference amplitude. As the number of elements increases both the null depth and the positional accuracy of the null increase. As might be expected, because of the absence of noise, peak responses do not occur at the angle-of-arrival of the desired signal.

Another property of the adaptive array resulting from discrete weight adjustment is illustrated by Fig. 14. The pattern plots of Fig. 14 were made for an angle-of-arrival of the desired signal of 90 degrees, no interfering signals and various amounts of noise. As the amount of noise is increased it can be observed that the null depths increase and the sidelobe levels decrease so as to produce an overall increase in directivity. This is because with a small amount of noise the weights tend to adjust to produced the smallest possible error between the desired signal and the reference. The error cannot in general go to zero because of the discrete nature of the phase and
Fig. 13.—Pattern plots for from two through five elements showing the increasing ability of the "trial-and-error" adaptive array to null an interfering signal as the number of elements is increased.
Fig. 14.--Pattern plots for the "trial-and-error" 5-element adaptive array with a desired signal and various amounts of noise showing the decrease in sidelobe level with increasing noise.
amplitude values. As the noise increases the tendency to maximize directivity predominates at the expense of a somewhat greater error between the desired signal and the reference. It was also observed that for sufficient noise the array simply shut off since this produced the least error.

Figure 15 shows a three-dimensional plot of array power pattern as a function of the number of adaptations for the trial-and-error method using amplitude increments of 0.00625 and phase increments of $\frac{\pi}{6}$. Each pattern represents one iteration through the array in which each of the amplitudes and phases is adjusted for a minimum. The plot is for a ten-element array with five elements initially active with a noise amplitude of $V_{\text{noise}} = 0.4$, a desired signal amplitude of unity and angle-of-arrival of 90 degrees and single interfering signal with an amplitude of 0.5 and an angle-of-arrival of 45 degrees. After 27 adaptations, with no further change in the pattern occurring, the interfering signal was removed. This brought about a small decrease in sidelobe level. At the 51-st adaptation the noise amplitude was increased to unity bringing about a further reduction in sidelobe level as discussed above with regard to Fig. 14. At the 65-th adaptation the angle-of-arrival of the desired signal was changed to 130 degrees. The main beam moved to 130 degrees demonstrating the ability of the array to follow a change in angle-of-arrival. Finally at the 82-nd
adaption the other five elements were activated with a resulting narrowing of the beam.

The results of the above simulations are by no means exhaustive but they are extremely promising. The trial-and-error adaptation method used with a digitally-controlled antennafer array seems especially promising because of its relative simplicity. Also the results indicated that an adaptive array using the trial-and-error
method might be more stable than an array using the LMS algorithm. No instances of instability were noticed during the trial-and-error method simulations while it was observed that with the LMS algorithm instability resulted from excessive gain, excessive noise or excessive interference.

C. Direction Finding Array

A direction finding system has been chosen for study in order to evaluate the performance of an array of digitally-controlled cavity-backed-slot antennafier elements in an actual system. The DF system was picked for evaluation because it could be implemented with the available array hardware and meaningful results obtained. The experimental array consisted of four of the elements described in Chapter II mounted with half wavelength spacing on a ground plane. Details of the equipment will be given in the next chapter.

Figure 16 shows a block diagram of the hardware configuration considered. The element outputs are summed to provide the array output which is fed to a conventional receiver. The receiver output, which is a video signal proportional to the received signal strength, is digitized by an analog-to-digital converter to be read by the control computer. Beam forming and steering information is fed as appropriate serial control signals from the computer to the array elements. Beams are formed conventionally by use of phase progressions
appropriate to the desired beam angles. Phase shift information for beam forming is stored in tables in the computer or calculated to give the proper phase progression along the array for each beam position.

When using a beam forming array for direction finding it may be desirable to both determine the angle-of-arrival of a signal and track the signal if the angle-of-arrival changes. Figure 17 shows

Fig. 16.—Block diagram of the experimental digitally-controlled antennafier array.
Fig. 17. -- Flow diagram for the search-and-track algorithm.

A flow diagram for a computer algorithm which provides both the search and tracking functions.

The search function is shown to the left of the dashed line in Fig. 17. The beam is started at one of the extreme positions and stepped a position at a time to the other extreme. At each beam position the signal strength is read, added to the signal strengths
from the other positions to form an average, and compared with the
maximum signal strength read thus far during the scan. When a signal
strength reading greater than the previous maximum is read its value
becomes the new maximum and the beam position is noted. At the end
of the scan the maximum signal strength is compared with the average.
If the maximum signal strength is a predetermined factor, \( \alpha \), greater
than the average signal strength it is taken as an indication that a
signal is present within the receiver passband and the angle of the beam
which gave the maximum signal strength is taken to be the angle-of-
arrival of the signal. If no signal is detected the scan process is
repeated. When a signal is detected the system switches to the track
mode.

In the track mode only three beam positions nearest the angle of
signal are scanned in order to minimize the time required to deter-
mine the angle-of-arrival. The scanning of only the three beam
positions nearest the angle of the signal also give the best signal
strengths.

In the track mode the continued presence of a signal is deter-
mined by a method similar to the one used in the search mode but
with a different constant, \( \beta \). If the signal is lost the system returns
to the search mode.

The three beam positions scanned in the track mode are selected
so that the beam position giving maximum signal strength is in the
center. If the angle-of-arrival changes so as to change the beam position giving maximum signal strength, the three beam positions are changed accordingly.

The above discussion assumed that the beams would all be the same amplitude. This will not generally be the case since the gain of even an ideal linear array varies with scan angle. In a practical array mutual coupling between elements affects both beam amplitude and position. These effects might be calculated and taken into account but error would be present because of unknown structural effects and irregularities in components. One solution is to simply measure the angles and amplitudes of the beams. The measured beam angles are used in determining the angles-of-arrival of unknown signals and the signal strength readings are corrected by dividing by measured beam amplitudes. This calibration technique was used with the experimental array to test the search and track algorithm and other experiments. Results are given in the next chapter.

The search and track algorithm locates and tracks the signal source to within the nearest beam position. The spacing between beam positions for the four-element experimental array varies between approximately 7 degrees near broadside and 14 degrees near the maximum scan angle (about 60 degrees). This can give rise to errors as great as 3 degrees near broadside and errors as great as
6 degrees near the maximum scan angle. In order to obtain more accurate results some sort of interpolation technique must be used.

One of the more conventional methods of locating the angle-of-arrival of a signal between two antenna beams in angle tracking and DF systems is the sum-and-difference or amplitude-monopulse method. There are a number of variations of the sum-and-difference method but all of these involve comparing the signals received on two antenna beams which are skewed or squinted to slightly different angles so that the responses to the same signal are different. The difference between the two responses is taken to indicate the displacement of the angle-of-arrival from a point midway between the two beams where the difference would be zero. The difference is divided by the sum which approximates a constant proportional to signal strength to compensate for variations in signal strength. For monopulse systems the sum and difference are usually formed at RF but this is not essential. The signals from the two beams can be envelope detected and the sum and difference formed at video. In fact, the signals need not even be present at the same time. The beam can be switched between two positions and the video signals sampled and stored for each beam position. If the beam switching is fast enough so that the signal remains essentially constant, or averaging is employed, no accuracy is lost. This method is well suited for interpolating between beam positions.
Figure 18 shows in sinθ space a sketch of the peaks of the two beams between which the interpolation is to be performed. The quantity X represents the sine of the angle-of-arrival of the signal. Because the area of interest is near the peaks of the beams they may be accurately represented by parabolic curves (the first three terms of a power series expanded about the peak of the beam). The equations for the parabolas are:

\[ y_2 = A \left[ -a(x - x_2)^2 + c \right] \]

and

\[ y_1 = A \left[ -a(x - x_1)^2 + c \right] \]

The difference, \( y_2 - y_1 \), is given by:

\[ y_2 - y_1 = 2Aa(x_2 - x_1) \left( x - \frac{x_2 + x_1}{2} \right) \]

The sum, \( y_2 + y_1 \), is given by

\[ y_2 + y_1 = A \left\{ -2a \left[ \left( x - \frac{x_2 + x_1}{2} \right)^2 + \left( \frac{x_2 - x_1}{2} \right)^2 \right] + 2c \right\} \]

In the above equations a and c are constants controlling the shape of the parabolas. The factor A has been included in Eqs. (10) and (11) to represent signal strength. It can be observed that the signal strength A appears as a factor in both the sum and the difference
Fig. 18.--Sketch showing the parameters for the two-beam interpolation method.

expressions (Eqs. (12) and (13)) and thus will cancel when the quotient is formed making the result independent of signal strength.

Forming the quotient and rearranging terms,

\[
x = \frac{x_2 + x_1}{2} + \frac{y_2 - y_1}{(y_2 + y_1)(x_2 - x_1)}
\]

\[
\left\{ \frac{c}{a} - \left[ \left( x - \frac{x_2 + x_1}{2} \right)^2 + \left( \frac{x_2 - x_1}{2} \right)^2 \right] \right\}
\]
If the beams are close together and \( x \) is restricted to being between the two peaks, the squared terms \( \left( x - \frac{x_2 + x_1}{2} \right)^2 \) and \( \left( \frac{x_2 - x_1}{2} \right)^2 \) will be small and can be neglected as can be demonstrated by simple calculations. Neglecting the squared terms the sine of the angle-of-arrival becomes:

\[
x = \frac{x_2 + x_1}{2} + \frac{c}{a} \frac{y_2 - y_1}{(x_2 - x_1)(y_2 + y_1)}
\]

Figure 19 shows a plot of the DF error as a function of angle-of-arrival at one-degree intervals between -50 degrees and +50 degrees for a simulated DF system obtained by applying Eq. (15) to the two beam positions giving the greatest signal strength. Calculations were based on the theoretical patterns for the experimental array (four omni-azimuthal elements with half-wavelength spacing). The patterns are given by:

\[
F(\theta) = \left| \frac{\sin 4\pi \left( \sin \theta - \frac{k}{8} \right)}{4 \sin \theta \left( \sin \theta - \frac{k}{8} \right)} \right|
\]

where

\( F(\theta) = \) the field pattern amplitude,
\( \theta = \) the angle from broadside in radians,
\( k = \) the beam number \((k = 0; 0 \text{ degrees}, k = 1; 7.2 \text{ degrees}, k = 2; 14.5 \text{ degrees}, \text{ etc.})\).
Fig. 19.--DF error as a function of angle-of-arrival for a simulated DF system using two-beam interpolation.

The constant \( C \) in Eq. (15) was adjusted to minimize the error which examination of Fig. 19 shows to be less than 0.1 degrees.

An important consideration for any system which may have to operate with weak signals is the effect of noise on performance. To test the effects of noise, various amounts of Gaussian pseudo-random noise were added to the pattern function of Eq. (16). This was done by adding successive numbers from a normally distributed pseudo-random number sequence to successive pattern function values. The pseudo-random numbers were multiplied by a constant to adjust the variance.

Also, since the signal-to-noise ratio is affected by the array gain which varies with scan angle, the directivities for the various beams were calculated by pattern integration and used to adjust the
signal strength values by multiplying by the factor \( \sqrt{D_k/D_0} \) (where \( D_k \) is the directivity of the k-th beam) and dividing by the same factor after adding the noise. This corresponds to the amplitude calibration mentioned in the discussion of the search and track algorithm.

Figure 20 shows a scatter diagram of DF error, similar to the plot of Fig. 19, but with ten points taken at one degree, intervals for a noise variance of .0004 corresponding to a video signal-to-noise ratio of 34 dB at broadside. Figure 21 shows the RMS error as a function of angle-of-arrival calculated from the data of Fig. 20. The RMS error varies from about 1.2 degrees at broadside to about 2.3 at the extreme scan angles where, it must be remembered, the signal-to-noise ratio is lower. Figures 22 and 23, 24 and 25, 26 and 27 and 28 and 29 show scatter diagrams and RMS error plots for video signal-to-noise ratios of 40 dB, 46 dB, 54 dB and 60 dB, respectively. By the time the signal-to-noise ratio gets to 60 dB the error becomes negligible.

Nearly all systems designed to measure angle-of-arrival or for homing or angle tracking make use of some sort of nulling or signal difference technique as does the method described above. Not very many systems attempt to get angle information from the peak of an antenna beam (one exception is conical scan). Difference methods generally provide greatest angular sensitivity although the signal-to-noise ratio in the null is less. The greater signal-to-noise ratio at
the peak of the beam might be expected to make up for the lack of angular sensitivity. The interpolation method to be described below makes use of signal strengths measured at three adjacent beam positions to locate the peak of the electronic-scan pattern of the array.

If continuous phase shifters were used in the array, it would be possible to measure, as the beam was swept, an electronic scan pattern for each angle-of-arrival of a signal. With discrete phase increments it is possible by stepping the array beam to measure a number of discrete points on the electronic scan pattern and determine, by curve fitting, the location of its peak corresponding to the angle-of-arrival.

As with the case of the conventional beam patterns, the scan pattern can be approximated by a parabola as shown in Fig. 30.

Using the equation for a parabola

\[ y = ax^2 + bx + c \]

three simultaneous equations corresponding to three points on the parabola sampled by three adjacent beams can be written. Thus

\[ y_{k-1} = ax_{k-1}^2 + bx_{k-1} + c \]

\[ y_k = ax_k^2 + bx_k + c \]

and
Fig. 20.—Scatter diagram of DF error for the simulated DF system with 34 dB S/N using two-beam interpolation.
Fig. 21. -- RMS DF error calculated from the scatter diagram of Fig. 20.

Fig. 22. -- Scatter diagram of DF error for the simulated DF system with 40 dB S/N using two-beam interpolation.
Fig. 23. -- RMS DF error calculated from the scatter diagram of Fig. 22.

Fig. 24. -- Scatter diagram of DF error for the simulated DF system with 46 dB S/N using two-beam interpolation.

Fig. 25. -- RMS DF error calculated from the scatter diagram of Fig. 24.
Fig. 26.—Scatter diagram of DF error for the simulated DF system with 54 dB S/N using two-beam interpolation.

Fig. 27.—RMS DF error calculated from the scatter diagram of Fig. 26.

Fig. 28.—Scatter diagram of DF error for the simulated DF system with 60 dB S/N using two-beam interpolation.
Fig. 29. -- RMS DF error calculated from the scatter diagram of Fig. 28.

Fig. 30. -- Sketch of the electronic scan pattern showing parameters for the three-beam parabolic interpolation.
\[ y_{k+1} = a x_{k+1}^2 + b x_{k+1} + c \]

where

\[ k \] is the beam number,
\[ y_k \] is the signal amplitude measured by the k-th beam,
\[ x_k \] is the position, in \( \sin \theta \) space, of the peak of the k-th beam.

The location of the peak of the parabola (scan pattern) can be found by differentiating Eq. (17) and setting the result equal to zero and is given by:

\[ x_{\text{peak}} = -\frac{b}{2a} \]

Equations (18), (19) and (20) can be solved simultaneously for \( a \) and \( b \) and the results substituted into Eq. (22) to give:

\[ x_{\text{peak}} = \frac{1}{2} \left( \frac{y_{k-1}(x_k^2 - x_{k+1}^2) + y_k(x_{k+1}^2 - x_{k-1}^2) + y_{k+1}(x_{k-1}^2 - x_k^2)}{y_{k-1}(x_k - x_{k+1}) + y_k(x_{k+1} - x_{k-1}) + y_{k+1}(x_{k-1} - x_k)} \right) \]

Figure 31 shows a plot of calculated DF error as a function of angle based on Eq. (22) using the three beams giving the greatest signal strength at each angle-of-arrival. The error is slightly greater than the error obtained using the two beam sum-and-difference method but still less than 0.1 degrees.

Following the method outlined for the two-beam sum-and-difference method, scatter diagrams and RMS plots were calculated
for the three-beam parabolic interpolation method with various amounts of additive Gaussian noise. Figures 32 through 41 show the results for broadside signal-to-noise ratios of 34 dB, 40 dB, 46 dB, 54 dB and 60 dB, respectively. The RMS error for the 34 dB case varies from 0.75 degrees at broadside to 1.5 degrees at the scan limits of ± 50 degrees as compared with an error varying between 1.2 and 2.3 degrees for the two-beam method with the 34 dB signal-to-noise ratio. When the search-and-track algorithm of Fig. 17 is used this improvement is obtained at very little expense in terms of additional time since at least three beams must always be scanned in order to track the signal source.

The above results show the feasibility of using the hardware configuration of Fig. 16. Results obtained with the four-element experimental array are given in the next chapter.

Fig. 31.—DF error as a function of angle-of-arrival for a simulated DF system using the three-beam parabolic interpolation method.
Fig. 32.--Scatter diagram of DF error for the simulated DF system with 34 dB S/N using three-beam parabolic interpolation.

Fig. 33.--RMS DF error calculated for the scatter diagram of Fig. 32.
Fig. 34.--Scatter diagram of DF error for the simulated DF system with 40 dB S/N using three-beam parabolic interpolation.

Fig. 35.--RMS DF error calculated for the scatter diagram of Fig. 34.

Fig. 36.--Scatter diagram of DF error for the simulated DF system with a 46 dB S/N using three-beam parabolic interpolation.
Fig. 37. -- RMS DF error calculated from the scatter diagram of Fig. 36.

Fig. 38. -- Scatter diagram of DF error for the simulated DF system with 54 dB S/N using three-beam parabolic interpolation.

Fig. 39. -- RMS DF error calculated from the scatter diagram of Fig. 36.
Fig. 40.—Scatter diagram of DF error for the simulated DF system with 60 dB S/N using three-beam parabolic interpolation.

Fig. 41.—RMS DF error calculated from the scatter diagram of Fig. 40.
CHAPTER IV
THE EXPERIMENTAL ARRAY

A. Introduction

In order to test the performance of the slot antenna elements described in Chapter II a four element antenna array was built. After a suitable pattern measuring arrangement was worked out patterns were measured and the ability of the array to form and scan beams was verified. Equipment was then assembled into the configuration of Fig. 16 and the search and track and interpolation techniques discussed in Chapter III were tested. This chapter describes the experimental array, equipment for controlling the array, measured patterns of the array, the DF computer programs, and measured DF accuracy results.

B. The Four-Element Array

Figures 42a and b show photographs of the experimental array taken from the front and from the rear. The dimensions of the array are given in Fig. 43. The ground-plane was designed to accept as many as eight elements but only four were built for the present study. The element spacing is a half wavelength at the element center frequency of 1 GHz making the array 1.5 wavelengths long at 1 GHz. The
Fig. 42.--Photographs of the experimental array: (a) front view, (b) rear view.
Fig. 43.--Sketch of the experimental array showing the dimensions.

Ground-plane extends to 4.6 wavelengths and has semi-cylindrical end pieces about a wavelength in diameter to reduce pattern scalloping caused by scattering from the discontinuity at the edge of the ground-plane.

Power supplies required to operate the digital logic and phase-shifter diode drivers are mounted at the rear of the ground plane support structure and can be seen in Fig. 42b. The RF amplifier power supplies are also mounted although RF amplifiers were not installed for the measurements to be described and phase control only was used.
The power and control signal distribution cabling and the signal distribution amplifier are also mounted at the rear of the array and can be seen on the trapezoidal panel in Fig. 42. The control signal distribution amplifier is near the center of the panel and the control signal and clock input fittings are just above it.

An eight-way (to allow for eight elements) isolated power divider is used as an RF signal combining network. Although the isolated power divider increased the signal combining loss by 9 dB (because of the isolation feature) it was decided that for experimental studies the 9 dB loss could be tolerated since transmitted power could be easily increased and the isolation eliminates the complications of mismatch and possible interaction between elements which would occur with a reactive power divider. The power divider can be seen in Fig. 42 mounted at the top of the ground-plane support structure and connected to the element output by RG-9 cables.

Before any measurements were made the elements were each carefully tuned for an impedance match, as indicated by maximum power transfer, by means of the adjustable cavity shorts and probe tuning capacitors. Phase shifts for each phase increment from 0 through $337\frac{1}{2}$ degrees were measured for each element. The phase shift measurements were made using a Hewlett Packard model 8405A vector voltmeter. Each of the elements was excited individually by means of a short (less than $\lambda/10$) probe at its aperture and the vector
voltmeter used to measure the phase difference between the voltage at the probe and the voltage at a 50 ohm load which was used to terminate the phase-shifter output. Measurements were made at 1 GHz. The measured phase shift values are given in Table 1. The values differ somewhat from the design values but this did not seem to seriously affect the experimental results. Details of the control signal code used to set the element to given phase shift values are given in Appendix A.

### TABLE 1
**MEASURED PHASE SHIFT FOR PHASE SHIFTERS**

<table>
<thead>
<tr>
<th>Design Phase Shift at 1 GHz</th>
<th>Measured Phase Shift at 1 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Element 0</td>
</tr>
<tr>
<td>0.0</td>
<td>0</td>
</tr>
<tr>
<td>22.5</td>
<td>25.0</td>
</tr>
<tr>
<td>45.0</td>
<td>42.0</td>
</tr>
<tr>
<td>67.5</td>
<td>66.0</td>
</tr>
<tr>
<td>90.0</td>
<td>97.0</td>
</tr>
<tr>
<td>112.5</td>
<td>118.0</td>
</tr>
<tr>
<td>135.0</td>
<td>132.0</td>
</tr>
<tr>
<td>157.5</td>
<td>154.0</td>
</tr>
<tr>
<td>180.0</td>
<td>177.0</td>
</tr>
<tr>
<td>202.5</td>
<td>202.0</td>
</tr>
<tr>
<td>225.0</td>
<td>219.0</td>
</tr>
<tr>
<td>247.5</td>
<td>244.0</td>
</tr>
<tr>
<td>270.0</td>
<td>275.0</td>
</tr>
<tr>
<td>292.5</td>
<td>296.0</td>
</tr>
<tr>
<td>315.0</td>
<td>310.0</td>
</tr>
<tr>
<td>337.5</td>
<td>331.0</td>
</tr>
</tbody>
</table>
C. Measured Patterns

A large number of array patterns was measured to check the operation of the array. A typical set of measured patterns showing fifteen beam positions is shown in Fig. 44. The phase shifter settings used to form the beams are given in Table 2.

**Fig. 44.**—Measured beam patterns for the experimental four-element array.
The phase shifters were designed so that at 1 GHz the phase shift steps would be an integral subdivision of 360 degrees, i.e. $22\frac{1}{2}$ degrees. The original beam steering program made use of this fact and generated phase shifter settings by adding, for each successive element, the beam number (from -7 to 7) of the desired beam using 2's complement arithmetic modulo $2^4$. Thus, for example, beam 2 would have phase shifter settings of 0, 2, 4, 6 corresponding to phase shifts of 0, 45, 90, 135 degrees. Beam position -1 would have phase shifter
settings of 15, 14, 13, 12 corresponding to phase shifts of 337.5, 315.0, 292.5, 270.0 degrees.

For uniform phase shift between elements sixteen different phasing combinations can be obtained by adding numbers from 1 through 15 (modulo $2^4$) to the phase shifter settings determined for a given beam position by the method described above. Since the phase shifters are not ideal it might be expected that there would be some variation in beam positions and shapes for various phase shifter setting combinations which theoretically would produce identical beams. Pattern measurements showed this to be the case. The phase shifts of Table 2 were used for the DF experiments to be described later in this chapter.

Another important consideration with regard to measured patterns was the angular error between the antenna rotator and the polar recorder turntable. The calibrations on the turntable were used to indicate the angle of the array with respect to the transmitting antenna for the DF accuracy measurements. The error was measured by placing a transit on the pattern rotator and sighting a vertical pole in the distance. Transit and recorder turntable readings were compared at one-degree intervals. The error was found to be 0.3 degrees or less and is shown plotted as a function of angle in Fig. 45.
D. **Receiver**

The block marked receiver in the block diagram of the DF system in Chapter III (Fig. 20) should be a receiver of suitable bandwidth and sensitivity for the signal to be received and should have an output level proportional to received signal strength. For the purposes of the experimental DF system a Scientific Atlanta Wide Range Receiver, designed to be used as a pattern range receiver, was available and was found to have adequate gain stability to allow calibration of the array beam amplitudes.

The Scientific Atlanta receiver had no output level which was proportional to the received signal amplitude. It did, however, have a 1000 Hz bolometer output proportional to the square of the received signal amplitude. This output was intended to drive a pattern recorder. A Hewlett-Packard Model 416E standing-wave-ratio meter, which had a 0-1 volt DC output proportional to signal strength and was
designed to accept 1000 Hz bolometer input, was used to convert the Scientific Atlanta receiver output to drive the computer analog-to-digital converter input. Since the output of the standing-wave-ratio meter was still proportional to the square of the signal amplitude rather than to the signal amplitude, it was necessary to take the square root of the analog-to-digital converted reading in the computer. This made it necessary, for experiments involving noisy signals, to add the noise digitally within the computer after taking the square root rather than adding analog noise externally since the square root operation would change the signal statistics.

E. Computer Programming

In the process of conducting experiments with the array it was necessary to write a number of computer programs. To simplify the task of making modifications, programs were written in modular form with subroutines performing some of the major functions. As an example, a subroutine was written to perform the interpolation in the DF system to find the angle of arrival given the signal amplitudes and angles of the three beams giving the greatest received signal strength. This subroutine was modified a number of times to test various interpolation methods. The programs and subroutines were
written in assembler language and listings are given in Appendices B, C, D and E. Computer and software characteristics are given in Appendices I, J and K. Floating point number representation and arithmetic (performed by subroutine) were used extensively for programming convenience.

The major subroutines were arranged into three separately assembled subprograms containing subroutines having interrelated functions.

The first subprogram, the listing of which is given in Appendix B, was named CALRD and has five subroutine entries: RAD, CALIB, BMDAT, PRTBL and CRINIT. Flow diagrams for the first three of these subroutines are given in Figs. 46 through 48.

The subroutine RAD (read A/D) reads the analog-to-digital converter to obtain a 10-bit integer representing output of the standing-wave-ratio meter, converts this number to a floating point number between 0 and 1.0, takes the square root to compensate for the square-law characteristic of the bolometer output of the Scientific Atlanta receiver and returns the result. This subroutine also has provisions for averaging a specified number of readings. Because of the narrow bandwidth of the standing-wave-ratio meter (< 10 Hz) a time delay has been inserted before the reading of the analog-to-digital converter to allow time for the standing-wave-ratio meter output to stabilize after the array beam is changed. The subroutine also tests a simulation
Fig. 46.—Flow diagram for the subroutine RAD.
Fig. 47.--Flow diagram for the subroutine CALIB.
Fig. 48. -- Flow diagram for the subroutine BMDAT.
indicator and if the indicator is "true" returns a precalculated value from a common storage memory location without reading the analog-to-digital converter to provide for simulating the array output with theoretical readings.

The subroutine CALIB (Fig. 47) provides for the filling and updating of the array beam amplitude and position calibration table. The subroutine reads beam numbers and angles which are entered manually via the I/O teletype. As soon as the beam number is entered the phasing information for the corresponding beam is sent to the array by means of the subroutine SETBM which will be described later. The beam angles are in degrees from broadside and must be determined by the operator who rotates the array to give a maximum reading on the standing-wave-ratio meter and reads the corresponding angle from the pattern recorder turntable. As soon as the angle is entered, RAD is called to average ten readings from the standing-wave-ratio meter. Beam angles and amplitudes are stored in table locations corresponding to the beam numbers.

After all the desired beam data entries are made as indicated by the typing of a "control D" on the I/O teletype the updated table is written on the computer disk memory so that the data may be recovered at another time. The subroutine then returns to the calling program.

The subroutine BMDAT (Fig. 48) accepts as input data a beam number plus 7 (i.e., 0 indicates beam number -7, 1 indicates beam
number -6, etc.) and returns the corresponding angle and amplitude data from the calibration table. If the simulation indicator is "true", theoretical calibration data which have been determined from the pattern function of Eq. (16) and the directivity, as discussed in Chapter III, are returned.

The subroutine entry PRTBL causes the calibration table to be printed on the I/O teletype. The measured data is printed regardless of the state of the simulation indicator. The subroutine CRINIT causes the calibration table data previously saved on the disk memory to be read back into core memory.

The second important subprogram, SETBM, has one subroutine entry also named SETBM. The subroutine accepts as input data a beam number plus 7, as was the case with BMDAT, and assembles and writes (to the array) control words, using data from a table, to cause the appropriate beam to be formed. If the simulation indicator is "true" the array response is calculated from Eq. (16) of Chapter III for the specified beam position using an angle in degrees supplied to a common memory location by the calling program. The result is multiplied by a correction factor depending on the beam number and supplied from a table to account for the decrease in directivity as the beam is scanned from broadside as described in Chapter III. The corrected result is placed in a common storage location for use by
the simulation part of the subroutine RAD as described above. A flow diagram of SETBM is given in Fig. 49. The symbolic listing is in Appendix C.

The third subprogram INTERP also has a single subroutine entry of the same name. The subroutine accepts as input data the amplitude of the signal received at three beam positions and the corresponding beam angles in degrees. The sines of the beam angles are taken and one of the interpolation formulas of Chapter III is applied to the resulting data. The arcsine of the result is taken and this value is returned as the angle of arrival.

A number of different versions of this subroutine using different interpolation formulas have been tried. Symbolic listings of two of these corresponding to the two-beam sum-and-difference interpolation of Eq. (15) and the three-beam parabolic interpolation of Eq. (22) are given in Appendices D and E, respectively.

In addition to the subprograms CALRD, SETBM and INTERP a number of other subroutines which were not specifically written for the DF system have been used. These subroutines are described in Appendix F. Also, a number of utility and math subroutines such as the input-output routines, floating point arithmetic, and SIN, COS, etc. were used and are described in Appendix K.
Fig. 49.--Flow diagram for the subroutine SETBM.
F. Search and Track

A search-and-track algorithm was discussed in Chapter III. The flow diagram was given in Fig. 17. The purpose of the search-and-track algorithm was to determine the presence of a signal and cause the array beam to angle track the signal if present. The signal strength information for the three beams involved in tracking would be available for interpolation to more precisely determine the angle-or-arrival of the signal.

A computer program was written along the lines of Fig. 17 to evaluate the search-and-track algorithm. A more detailed flow diagram is given in Fig. 50. A symbolic listing of the program is given in Appendix G. The program is fairly lengthy and complicated but most of the complication is concerned with provisions for controlling and monitoring the execution of the program through the teletypewriter and the plotter. These program execution control and monitoring functions are not shown in Fig. 50 and will not be discussed further.

Referring to Fig. 50, for the search part of the program the maximum signal amplitude and the sum of the signal amplitudes (used to compute the average signal amplitude) are initialized to zero. The current beam position is initialized to -7. The beam amplitude is then obtained from the subroutine BMDAT, the beam is scanned to the current beam position by SETBM and the signal amplitude is read by
Fig. 50.--Flow diagram for the search-and-track program.
Execution monitoring and control functions are not shown.
RAD. The value of the signal amplitude read is adjusted by dividing by the beam amplitude obtained from BMDAT. The corrected signal amplitude is added to the sum of the beam amplitudes. It is then compared with the maximum signal amplitude. If the corrected signal amplitude is greater than the maximum signal amplitude the corrected signal amplitude replaces the maximum signal amplitude. Also the current beam position replaces the previous maximum beam position in this case.

The above process is repeated for each beam position from -7 through +7. At the end of the scan the sum of the signal amplitudes is divided by fifteen to obtain the average signal amplitude which is compared with the maximum signal amplitude. If the maximum signal amplitude exceeds the average by a factor $\alpha$, it is taken to indicate the presence of a signal and the program switches to the track mode. Otherwise it remains in the search mode and the scan is repeated.

For the track mode the value of the maximum signal amplitude and the sum of the amplitudes are initialized to zero and the current beam position is set to one less than the maximum amplitude position as determined by the search or track routine whichever was last executed. If the position number is less than -7 or greater than 5 the source is considered to have moved beyond the angular tracking limits of the array and the program returns to the search mode.
The track loop is nearly identical to the search loop. The differences are that only three beam positions are scanned instead of fifteen, and the three beam angles corresponding to the beam position numbers, as determined by BMDAT, and the corrected signal amplitudes are saved in data arrays. At the end of each scan the maximum signal amplitude is compared with $\beta$ times the average to determine the presence of a signal. The program returns to the search mode if the signal is absent as indicated by the maximum being greater than $\beta$ times the average. Otherwise, the track routine is repeated. If the maximum amplitude beam position has changed, the new maximum amplitude beam position will be tested to see that it is within the angular scan limits and a different set of three beam positions will be scanned. At the end of each scan during the track mode the two data arrays containing the three corrected signal amplitudes and the corresponding beam angles provide sufficient information to interpolate and locate the angle-of-arrival of the signal.

The search-and-track program was tested and evaluated qualitatively. For the search mode a value of the parameter $\alpha$ of 1.5 gave good results. The presence of a signal was detected by the search routine at the same time it became noticeable above the noise on the monitor oscilloscope of the Scientific Atlanta receiver. The track mode had no trouble tracking the angle-of-arrival as the array was rotated on the pattern rotator but the method of comparing maximum
and average signals to determine the presence of a signal was not completely successful. Since only three beams were used and the crossover points between beams are only 0.2 dB down not much difference between maximum and average signal amplitudes can occur. If the parameter $\beta$ was set too high the array would not track. A value of 1.01 for $\beta$ came closest to working. What happened frequently when the signal strength was gradually reduced was the beam position would shift from the angle-of-arrival with no indication that signal had been lost. The beam would then drift to one of the scan limits which would cause the program to switch back to the scan mode. Because of the amplitude calibration the system always has a tendency to drift to the nearest scan limit with no signal present. This is because in compensating for the reduced gain when the beam is scanned from broadside the noise amplitude is increased. This means that, for each group of three beams scanned during the track routine, the beam nearest the closest scan limit appears to have the greatest signal strength and becomes the center beam for the next scan. The process continues until the beam reaches the scan limit.

Except for the trouble in detecting a signal loss during track the search-and-track program performed its intended functions well.
G. **Direction Finding**

A program similar to the search and track program was written to test the two-beam sum-and-difference and the three-beam parabolic interpolation methods discussed in Chapter III. This program was, in fact, used to obtain the simulated DF accuracy results of Chapter III by setting the simulation indicator to "true" so that the various subroutines described earlier in the section on programming would use theoretical amplitude and calibration data instead of actual data from the array. The flow diagram for the program is shown in Fig. 51. The symbolic listing is given in Appendix H.

As was the case with the search-and-track program the DF accuracy program is considerably more elaborate than indicated by Fig. 51. Most of the program is concerned with execution control via the I/O teletype and plotting output data on the plotter. Figure 51 shows only the part of the program concerned with direction finding in detail. The control portion is shown as a single block.

The search-and-track modes were not implemented. The program scans the beam once through all fifteen beam positions as was the case with the search routine in the search-and-track program. A pseudo random number from a population having a normal distribution is added to the value of the signal amplitude returned by the subroutine **RAD** at each beam position to simulate a noisy signal. At the end of the scan the beam position giving maximum signal strength will have
Fig. 51.—Flow diagram for the DF-accuracy program. Execution monitoring and control functions and data-output functions are not shown.
been determined. The corrected amplitudes and the beam angles for this beam position and for the two adjacent beam positions are fed to the subroutine INTERP to determine an angle-of-arrival. The program then returns to the control routine to process and store the error data.

A plot of measured DF error as a function of angle-of-arrival at one-degree intervals for the two-beam sum-and-difference method with no noise is shown in Fig. 52. A scatter diagram with 10 points at each degree interval from -50 degrees to +40 degrees is shown in Fig. 53. Unfortunately, an error in the two-beam sum-and-difference interpolation equation (Eq. (15)) such that the factor \((x_1 - x_2)\) occurred in the numerator of the second term rather than in the denominator caused the measured results to be inaccurate. The significance of this error is that irregularities in beam positions will not be properly taken into account. This has probably caused the error indicated in Fig. 52 to be greater than it would have been had the correct formula been used.

In spite of the error the data of Figs. 52 and 53 were included to show the point spread caused by noise which agrees well with the simulated results of Figs. 19 and 20 calculated for the same signal-to-noise ratio in Chapter III. The interpolation subroutine given in Appendix D has the correct formula but time did not permit repeating the experiments.
Fig. 52.--Measured DF error as a function of angle-of-arrival for the experimental array using two-beam interpolation.
Fig. 53.---Scatter diagram of measured DF error for the experimental array with 34 dB S/N using two-beam interpolation.
Direction finding error data were also taken for the three-beam parabolic interpolation method. Because of the difficulty of reproducing pattern range conditions from one day to the next, no-noise DF error data were taken each time the equipment was set up. Figure 54 shows a typical plot of DF error as a function of angle-of-arrival. This can be compared with Fig. 58 which gives the no-noise DF error measured on a different day. The curves are similar in shape and the magnitudes of the errors are about the same but the data are definitely not identical. The array was recalibrated for each set-up. It was found that without recalibration the error generally increased about 50 percent from the previously measured error.

Fig. 54.—Measured DF error as a function of angle-of-arrival for the experimental array using the three-beam interpolation method with no noise.
A scatter diagram of DF error for a 40 dB signal-to-noise ratio is shown in Fig. 55. The corresponding RMS error is shown in Fig. 56. The average error determined from the scatter diagram of Fig. 55 is shown in Fig. 57 and can be seen to approximate the no-noise data quite well. Except for the no-noise error the measured results of Figs. 55 and 56 agree well with the corresponding results of the simulations of Chapter III. Even considering the no-noise error the RMS error of Fig. 56 does not differ a large amount from the RMS error for the corresponding simulated results. (Fig. 33).

Data were also taken for a 34 dB signal-to-noise ratio. Figures 58, 59, 60 and 61 give the no-noise DF error (which differs slightly from that of Fig. 54 because of the different pattern range set-up), the scatter diagram of the DF error with noise, the RMS DF error and the average error, respectively. These measured results also agree (except for the no-noise error) with the simulated results of Chapter III.

The deviation of the no-noise error in the measured results from the noise-error of the simulation results is, of course, due to the deviation of the beam shapes and position from the ideal beam shapes and positions of Eq. (16). The errors are not serious and the accuracy is quite good for a 1.5 wavelength aperture considering the possible errors introduced by the supporting structure of the array which is
fairly large in terms of wavelength (6.6 wavelengths). Since practical
arrays will in general be mounted on even large and more complex
structures further investigation will be required to determine the
effects of these structures on accuracy and on the value of the beam
calibration in overcoming these effects.

Fig. 55.--Scatter diagram of measured DF error for the
experimental array with 40 dB S/N using three-beam
parabolic interpolation. Date were taken with the
pattern range set up used for the data of Fig. 54.
Fig. 56. -- RMS measured DF error calculated from the scatter diagram of Fig. 55.

Fig. 57. -- Average measured DF error calculated from the scatter diagram of Fig. 55.
Fig. 58:--Measured no-noise DF error for the experimental array using three-beam parabolic interpolation.
Fig. 59. -- Scatter diagram of measured DF error for the experimental array with 34 dB S/N using three-beam parabolic interpolation. Data was taken with the pattern range set up used for the data of Fig. 58.
Fig. 60.--RMS DF error calculated from the scatter diagram of Fig. 59.

Fig. 61.--Average measured DF error calculated from the scatter diagram of Fig. 59.
CHAPTER V
SUMMARY AND CONCLUSIONS

Many phased array applications require control of amplitude and phase of the individual elements. The development of an element having individually controllable gain and phase and two specific phased array applications of such an element have been discussed. One system, the adaptive receiving array, was discussed briefly while the other system, the direction finding array, was discussed in more detail including a description of an experimental array and results of DF accuracy measurements.

The adaptive receiving array showed considerable promise as a system for forming a directive beam in the presence of noise and for minimizing interference arriving at the array from directions other than that of the desired signal. The ability of an adaptive array to adjust when an interfering signal is added and then readjust when the signal is removed has been demonstrated by computer simulation first using the LMS algorithm of Widrow, Mantey, Griffiths and Goode and then using a trial-and-error adaptation method suitable for use with an array of the individually controllable elements which were developed. It has also been shown that the array can adjust to
track a changing angle-of-arrival of the desired signal. It has furthermore been shown that the reference signal to which the array output is compared in order to optimize performance need not be identical to the desired signal. This obviates the need for a pilot signal to be injected at each element to simulate a signal arriving from the angle of the desired signal and thus greatly simplifies the design of the array.

The direction finding system was chosen for study because it could be implemented with available hardware which consisted of four cavity-backed slot elements with controllable phase shift. The DF system has demonstrated the use of these elements in a system with closed loop control of an array.

The DF system was designed to form any one of fifteen discrete beams and determine approximately the angle-of-arrival of a signal by determining the beam giving the greatest signal strength and taking the angle of that beam to be the angle-of-arrival. Provisions were made to track a signal with a changing angle-of-arrival by repeatedly scanning the three adjacent beams nearest the angle-of-arrival of the signal. The signal strengths at each of the three beams are available for processing to refine the angle-of-arrival information and thus improve the DF accuracy of the system.

Two methods of improving the DF accuracy have been studied. They are the two-beam sum-and-difference method and the three-beam
parabolic-interpolation method. The two-beam method is definitely more susceptible to errors caused by noise than the three beam method as was indicated by both the results of a computer simulation and measurements using the experimental array. For a 34 dB video signal-to-noise ratio the RMS error varied between 0.75 degrees and 1.5 degrees for the three-beam method as compared with 1.2 and 2.3 degrees for the two-beam method.

The results of the investigation of the direction-finding system show that it performs adequately at a single frequency. To extend the operation to wide range of frequencies a number of sets of calibration data are necessary. The calibration technique used (calibration of beam angles and amplitudes) can reduce the amount of calibration data required compared with the amount of data required by the more conventional technique of calibrating indicated angle-of-arrival against actual angle-of-arrival.

The two arrays which were investigated, the adaptive array and the direction finding array, demonstrate the degree of flexibility which can be obtained from a phased array with individually controllable elements. Both systems operate with practically the same set of hardware. This includes the array elements, the signal combining network and the control computer. The only difference is in the way the receiver processes the received signal. Since all that is required of the receiver for the DF system is that it have an output proportional
to the received signal, the adaptive array system can be made to function as a DF system simply by adding an amplitude output to its receiver. Changing the control computer program then changes the system from the adaptive array environment to the DF environment. This illustrates the versatility of a phased array of individually controlled elements.
APPENDIX A
CONTROL SIGNAL CODE FOR THE SLOT ANTENNA FIER

The digitally-controlled slot antennafiers required a serial NRZ digital control signal and a clock signal to change their gain and phase shift settings. The formats of the two signals are shown in Fig. 62

![Fig. 62. Timing diagram showing the formats of the slot-antenna Fier element control and clock signal.](image)

The \( r_1 \) bit is required to be a one and functions to start the element control circuitry. Bits \( a_1 \) through \( a_4 \) are the binary element address. If the element address agrees with the element number the remainder of the control signal is processed. Otherwise, it is ignored.

Assuming the address agrees and if \( r_2 \) is a one, bits \( p_1 \) through \( p_4 \) replace the contents of the phase register. If bit \( r_2 \) is zero the phase register contents remain unchanged. Table 3 shows the phase
shifter setting as a function of phase register bits $P_1$ through $P_4$.

The order of the bits is inverted from natural binary. $P_1$ is the
low-order bit representing $22\frac{1}{2}$ degrees and $P_4$ is the high order bit
representing 180 degrees.

If bit $r_3$ is a one (assuming again that the address agreed),
bits $A_1$ through $A_4$ replace the contents of the amplitude register to
effect a gain change. If bit $r_3$ is a zero the contents of the phase
register remain unchanged. Table 4 shows the nominal values of the
amplifier gain as a function of the contents of the amplitude register.

In Fig. 62 the clock is shown as symmetric and the bit times
as uniform. This need not be the case (and in fact is not) so long as
the falling clock transition occurs no closer than a microsecond to
the time of a signal transition.
<table>
<thead>
<tr>
<th>Phase Register Bits</th>
<th>Nominal Phase Shift - Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1  P_2  P_3  P_4</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>22.5</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>45.0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>67.5</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>90.0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>112.5</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>135.0</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>157.5</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>180.0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>202.5</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>225.0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>247.5</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>270.0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>292.5</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>315.0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>337.5</td>
</tr>
<tr>
<td>Amplitude Register Bits</td>
<td>Nominal Amplifier Gain - dB</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>7.0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>6.0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>5.0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>4.0</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>3.0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>2.0</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>1.0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0.0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>-1.0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>-2.0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>-3.0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>-4.0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>-5.0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>-6.0</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>-7.0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>-8.0</td>
</tr>
</tbody>
</table>
APPENDIX B
CALRD

-PRINT

NAME: CALRDS

* THIS SUBPROGRAM CONTAINS THE SUBROUTINES "CALIB", "RAD", "PRTEBL", "BMDAT" AND "CRINIT". THE CALLING
* PROCEDURES ARE:
* TRL $CALIB
* TRL $RAD,1,N
* WHERE "N" IS THE NUMBER OF A/D READINGS TO BE AVERAGED,
* TRL $PRTEBL
* TRL $BMDAT,1,TEMPS
* WHERE TEMPS IS THE FIRST OF TWO LOCATIONS SET ASIDE TO
* RECEIVE THE BEAM ANGLE AND AMPLITUDE OF THE BEAM
* SPECIFIED BY C(IX3)-7.
* TRL $CRINIT
$CALIB MOV (PCS),SAVS
TRL REGSAV
TRL CRINIT
WAT CAL,21
BNUM WAT BPN,5
TRL READ
CLA BEAM
BNUM1
TRL $PRI
TRU BNUM1
TRL $FLINT
BNUM1 STR BEAM
ADD [7]
TRN ERR
STR (IX3)
SHL 1,,7
STR (IX4)
SUB [29]
TRP ERR
TRL $SETBM
WAT BA,3
TRL READ
CLA TABLE,4
TRL $PRFL
TRU *+2
STR TABLE,4
TRL RAD,1,10
STR TABLE+1,4
WAT BAM,3
TRL $PRFL
TRU BNUM
READ MOV (PCS),RSAVS
MOV (Z),STRING
MOV (Z),STRING+1
MOV (Z),STRING+2
MOV (Z),STRING+3
MOV (Z),STRING+4
RAT STRING,5
TIU *
MOV [5J,(IX2)
MOV [6J,(IX1)
LOD STRING-1,2,(Q)
CLA (Z)
SLL 6
SUB ['57']
TRZ ERR
TRX *-4,4
TRX *-7,1
CLA STRING
TRZ NODATA
SHR 24
SUB [4]
TRZ DONE
CLA STRING
SHR 30
TRZ ERR
TRL $CNVTF,1,STRING
TRU ERR
ADB RSAVS,, 3
CLA (Q)
MOV RSAVS,(PC)

ERR WAT ["405277520000", 1
TRU BNUM
NODATA MOV RSAVS,(PC)
#RAD MOV (IX2), RADS2
MOV (PCS), RSAVS
MOV (IX1), RADS1
CLA '7777'
TRZ ++3
CLA '7775'
TRS
CLA (Z)
MOV [4095], (IX2)
TRX *+1
RAN ADL, 1, 6
TIU *
ADD ADL
TRX *-5, 4
ADB (Z)
DVL RADS1
CLA (Q)
TRL $INTFL
TRL $FDVD,1,["6000000000412"]
TRL $SQRT
USE MATH1,LIB
MOV RADS2,(IX2)
MOV RADS1,(IX1)
MOV RSAVS,(PC)
DONE
MOV SAVS,(PCS)
SNS *+1,,(ISN)
WAD TABLE,32
TRU REGRST
USE UTIL,LIB
USE SETBM,DEAN
CAL
TAB "1500120012"
ALPHA 3 CALIBRATION -
TAB "1500120012"
ALPHA 7 ADJUST FOR BEAM MAX BEFORE ENTERING
ALPHA 2 -BEAM ANGLE.
TAB "150012"
ALPHA 7 END CALIBRATE MODE BY "CTRL" D.
BPN
TAB "1500120012"
ALPHA 4 BEAM POSITION:
BA
ALPHA 3 ANGLE:
BAM ALPHA 3 AMPLITUDE:

SAVS BS 1

BEAM BS 1

TABLE BS 32

RSAVS BS 1

STRING BS 5

RADS1 BS 1

RADS2 BS 1

ADL BS 1

REGSAV MOV (A),REG

MOV (Q),REG+1

MOV (IX1),REG+2

MOV (IX2),REG+3

MOV (IX3),REG+4

MOV (IX4),REG+5

TRS

REGRST MOV REG,(A)

MOV REG+1,(Q)

MOV REG+2,(IX1)

MOV REG+3,(IX2)

MOV REG+4,(IX3)

MOV REG+5,(IX4)

TRS

REG BS 6
PRTBL MOV (PCS), PRSAVS

TRL REGSAV

WAT HEAD, 14

LDX 15, 3, 28

LOOP WAT ['1500124040', 1]

CLA (IX4)

SUB [8]

TRL $PRI

WAT ['404040404040', 1]

WAT ['404000000000', 1]

CLA TABLE, 3

TRL $PRFL

WAT ['404040400000', 1]

CLA TABLE+1, 3

TRL $PRFL

TRX LOOP, 3, -2

MOV PRSAVS, (PCS)

TRU REGRST

PRSAVS BS 1

HEAD TAB '1500120012'

ALPHA 2 BEAM DATA -

TAB '1500120012'

ALPHA 7 NUMBER ANGLE

ALPHA 4 - AMPLITUDE
#BMDAT MOV (PCS), SAVS
TRL REGSAV
CLA '7777'
TRZ ++7
CLA TBL2, 3
STR 1, 1
CLA TABLE1, 3
STR 0, 1
MOV SAVS, (PCS)
TRU REGRST
CLA (IX3)
SHL 1
STR (IX3)
CLA TABLE, 3
STR , 1
CLA TABLE+1, 3
STR 1, 1
MOV SAVS, (PCS)
TRU REGRST
#CRINIT CAM (D)
SUB [D8800]
TRZ ++2
WKD [D8800]
SNS ++1, (ISN)
<table>
<thead>
<tr>
<th>RDB</th>
<th>TABLE1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIU</td>
<td>*</td>
</tr>
<tr>
<td>TRS</td>
<td></td>
</tr>
<tr>
<td>TBL2</td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td>TABLE1</td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td></td>
<td>TAB</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C
SETBM

NAME: STBM2S
* THIS SUBPROGRAM CONTAINS THE SUBROUTINE "SETBM".
* THE ARRAY BEAM IS SET TO THE POSITION SPECIFIED BY
* C(Ix3)-7.

#SETBM MOV (A),REG
  MOV (Q),REG+1
  MOV (IX2),REG+2
  MOV (IX3),REG+3
  MOV (IX4),REG+4
  CLA '7777'
  TRZ +/-2
  TRU SIM
  LDX 4,3,0
LOOP CLA (IX3)
  SHL 31
  LG A ['410000000000']
  STR WD
  CLA REG+3
  SHL 2
  ADD (IX3)
  STR (IX2)
CLA PHASE, 2
CYS 33
MOV (Z), (Q)
MOV (41), (IX2)
SRL 1
CYS 2
TRX \(*-2,1\)
SLL 30
LGA WD
STR WD
WAN WD, 1, 7
SEN HALT, \((SW4)\)
TRX LOOP, 3, 1
CLA REG
MOV REG+1, (Q)
MOV REG+2, (IX2)
MOV REG+3, (IX3)
MOV REG+4, (IX4)
TRS
HALT MOV WD, (DISP)
HLT
TRX LOOP, 3, 1
TRS
REG BS 5
PHASE	TAB 5,14,7,0
       TAB 14,8,2,12
       TAB 0,11,6,1
       TAB 0,12,8,4
       TAB 11,8,5,2
       TAB 10,8,6,4
       TAB 10,9,8,7
       TAB 0,0,0,0
       TAB 13,14,15,0
       TAB 10,12,14,0
       TAB 7,10,13,0
       TAB 4,8,12,0
       TAB 1,6,11,0
       TAB 10,0,6,12
       TAB 11,2,9,0

WD	BS 1

SIM	MOV (IX1),SAV1
       MOV (PCS),SAVS
       CLA '7774' THETA
       TRL $FMLY,1,['435750652373'] PI/180
       TRL $SIN
       TRL $FMLY,1,['622077327402'] PSI
       CLA (IX3)
SUB 7
TRL $INTFL
TRL $FMLY,1,'622077327377'  PIS8
CLS (A)
TRL $FADD,1,PSI
TRL $FDVD,1,'4000000000402''
STR PSI
TRL $SIN
STR DENOM
CAM (A)
TRL $FSUB,1,'40000000364''
TRP **3
CLA ['40000000401']
TRU SIM2
CLA PSI
TRL $FMLY,1,'40000000403''
TRL $SIN
TRL $FDVD,1,DENOM
TRL $FDVD,1,'40000000403'
CAM (A)
SIM2 STR '7775'
CLA TBL2,3
TRL $FMLY,1,'7775'
STR '7775'
MOV SAV1, (IX1)

MOV SAVS, (PCS)

TRU HALT-6

SAVS BS 1

PSI BS 1

DENOM BS 1

SAV1 BS 1

TBL2 TAB '620566471400', '642700522400', '673123036400'

TAB '722451571400', '744772770400', '762541137400'

TAB '774232156400', '400000000401', '774232156400'

TAB '762541144400', '744772776400', '722451602400'

TAB '673123045400', '642700533400', '620566476400'

USE UTIL, LIB

USE MATH1, LIB

END
APPENDIX D
TWO BEAM INTERP

The subroutine listed below evaluates the formula

\[
\theta = \arcsin \left[ \frac{\sin \theta_2 + \sin \theta_1}{2} + 0.16 \frac{y_2 - y_1}{(\sin \theta_2 - \sin \theta_1)(y_2 + y_1)} \right]
\]

-PRINT

NAME: INTER2

* THIS SUBROUTINE TAKES THREE BEAM ANGLES FROM THE
* LOCATION SPECIFIED BY IX4 AND SUCCESSIVELY HIGHER
* LOCATIONS AND THE CORRESPONDING SIGNAL AMPLITUDES
* FROM THE LOCATION SPECIFIED BY IX3 AND SUCCESSIVELY
* HIGHER LOCATIONS AND RETURNS IN THE ACCUMULATOR
* AN APPROXIMATE ANGLE-OF-ARRIVAL USING THE TWO-BEAM
* SUM-AND-DIFFERENCE INTERPOLATION METHOD. ANGLES ARE
* IN DEGREES.
#INTERP MOV (PCS),SAVS

CLA 0,4
TRL $FMLY,1,"435750652373"
TRL $SIN
STR X3

CLA 1,4
TRL $FMLY,1,"435750652373"
TRL $SIN
STR X2

CLA 2,4
TRL $FMLY,1,"435750652373"
TRL $SIN
STR X1

CLA 0,3
STR Y3

CLA 1,3
STR Y2

CLA 2,3
STR Y1

CLA Y1
TRL $FSUB,1,Y3
TRP *+5
MOV Y2,Y1
MOV Y3,Y2
MOV X2,X1
MOV X3,X2
CLA Y1
TRL $FADD,1,Y2
STR DENOM
CLA X2
TRL $FSUB,1,X1
TRL $FMLY,1,DENOM
STR DENOM
CLA Y2
TRL $FSUB,1,Y2
TRL $FDVD,1,DENOM
TRL $FMLY,1,C
TRL $FADD,1,X1
TRL $FADD,1,X2
TRL $FMLY,1,['4000000000400']
MOV (Q),QSAV
STR A
TRL $FMLY,1,A
TRL $FSUB,1,['40000000401']
TRL $SQRT
STR TEMP
CLA A
TRL $FDVD,1,TEMP
STR (Q)
CLA ['40000000401']
TRL $ATAN2
MOV QSAV,(Q)
TRL $FMLY,1,('712273404406')
MOV SAVS,(PC)
C TAB '5100000000377'
SAVS BS 1
TEMP BS 1
NUM BS 1
DENOM BS 1
X1 BS 1
X2 BS 1
X3 BS 1
Y1 BS 1
Y2 BS 1
Y3 BS 1
USE UTIL,LIB
USE MATH1,LIB
USE ATAN2,DEAN
A BS 1
QSAV BS 1
END
APPENDIX E
THREE BEAM INTERP

The subroutine listed below evaluates the formula

\[
\theta = \arcsin \frac{\frac{1}{2} \left( y_1(x_2^2 - x_3^2) + y_2(x_3^2 - x_1^2) + y_3(x_1^2 - x_2^2) \right)}{y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_3(x_1 - x_2)}
\]

where

\[
(24) \quad x_1 = \sin \theta_1 \\
(25) \quad x_2 = \sin \theta_2 \\
(26) \quad x_3 = \sin \theta_3
\]

NAME: INTER3

* THIS SUBROUTINE TAKES THREE BEAM ANGLES FROM THE
* LOCATION SPECIFIED BY IX4 AND SUCCESSIVELY HIGHER
* LOCATIONS AND THE CORRESPONDING SIGNAL AMPLITUDES
* FROM THE LOCATION SPECIFIED BY IX3 AND SUCCESSIVELY
* HIGHER LOCATIONS AND RETURNS IN THE ACCUMULATOR
* AN APPROXIMATE ANGLE-OF-ARRIVAL USING THE THREE-BEAM
* PARABOLIC INTERPOLATION METHOD. ANGLES ARE IN DEGREES.

\#INTERP MOV (PCS),SAVS

CLA 0,4
TRL $FMLY,1,[435750652373]
TRL $SSIN

122
STR X3
CLA 1,4
TRL $FMLY,1,['435750652373']
TRL $SIN
STR X2
CLA 2,4
TRL $FMLY,1,['435750652373']
TRL $SIN
STR X1
CLA 0,3
STR Y3
CLA 1,3
STR Y2
CLA 2,3
STR Y1
MOV (Z),NUM
MOV (Z),DENOM
CLA Y2
TRL $SUB,1,Y3
TRL $FMLY,1,X1
STR TEMP
TRL $FMLY,1,X1
TRL $FADD,1,NUM
STR NUM
CLA DENOM
TRL $FADD,1,TEMP
STR  DENOM
CLA  Y3
TRL  $FSUB,1,Y1
TRL  $FMLY,1,X2
STR  TEMP
TRL  $FMLY,1,X2
TRL  $FADD,1,NUM
STR  NUM
CLA  DENOM
TRL  $FADD,1,TEMP
STR  DENOM
CLA  Y1
TRL  $FSUB,1,Y2
TRL  $FMLY,1,X3
STR  TEMP
TRL  $FMLY,1,X3
TRL  $FADD,1,NUM
STR  NUM
CLA  DENOM
TRL  $FADD,1,TEMP
STR  DENOM
CLA  NUM
TRL  $FDVD,1,DENOM
TRL  $FMLY,1,[400000000400]
MOV (Q),QSAV
STR  A
TRL  $FMLY,1,A
TRL  $FSUB,1,['4000000000401']
TRL  $SQRT
STR  TEMP
CLA  A
TRL  $FDVD,1,TEMP
STR  (Q)
CLA  ['4000000000401']
TRL  $SATAN2
MOV  QSAV,(Q)
TRL  $FMLY,1,['712273404406']
MOV  SAVS,(PC)
SAVS  BS  1
TEMP  BS  1
NUM  BS  1
DENOM  BS  1
X1  BS  1
X2  BS  1
X3  BS  1
Y1  BS  1
Y2  BS  1
Y3  BS  1
USE UTIL.LIB
USE MATH1.LIB
USE ATAN2.DEAN

A BS 1
QSAV BS 1,

END
APPENDIX F
ADDITIONAL SUBROUTINES

-PRINT

NAME: ARSUBS

THE FOLLOWING SUBROUTINES ARE USED IN THE
EXPERIMENTAL ARRAY PROGRAMS:

TRL $SATAN2
CALCULATES THE POLAR ANGLE (FOUR QUADRANTS) GIVEN
X AND Y IN THE (A) AND (Q) REGISTERS RESPECTIVELY.
OUTPUT RANGE IS +PI TO -PI.

TRL $MARK,1,X
TRL $MARK2,1,X
TRL $MARK3,1,X

CAUSE A SMALL CIRCLE, TRIANGLE OR SQUARE, RESPECTIVELY,
TO BE PLOTTED ON THE PLOTTER AT THE X COORDINATE GIVEN
IN LOCATION "X" AND THE Y COORDINATE GIVEN IN THE
LOCATION FOLLOWING LOCATION "X".

TRL $RAND

OBTAINS A PSEUDO-RANDOM NUMBER FROM A NORMALLY
DISTRIBUTED POPULATION HAVING ZERO MEAN AND
STANDARD DEVIATION UNITY. THE RESULT IS RETURNED IN
THE (A) REGISTER.

TRL $READF,1,X
TRL $READI,1,X

READ FLOATING POINT NUMBERS AND INTEGERS, RESPECTIVELY, FROM THE I/O TELETYPEx. THE NUMBERS ARE PLACED IN LOCATION "X" AND IN THE (A) REGISTER. IF NO NUMBER IS ENTERED ON THE TELETYPEx KEYBOARD, THE NUMBER IN LOCATION "X" IS PRINTED AND PLACED IN THE (A) REGISTER.
APPENDIX G
SEARCH AND TRACK PROGRAM

NAME: DFS

*  THIS PROGRAM IS DESIGNED TO TEST THE DIRECTION
*  FINDING SYSTEM "SEARCH" AND "TRACK" MODES.

#BEGIN TRL $CRINIT
    MOV [1], BKIND
    ADB (Z), 2, SCAN
    TRU $TTY4

#SCAN
    MOV ["SEARCH"], MODE
    WAT ['150012'], 1
    WAT ['SCAN'], 1
    MOV (Z), MAX
    MOV (Z), SUM
    LDX 15, 3, 0
    TRL $BMDAT, 1, TEMP1
    TRL $SETBM
    TRL $RAD, 1, 1
    TRL $FDVD, 1, TEMP1+1
    STR SCNDAT, 3
    STR TEMP
    TRL $FADD, 1, SUM
    STR SUM

129
CLA MAX
TRL $FSUB,1,TEMP
TRP ++3
MOV TEMP,MAX
MOV (IX3),MAXPOS
TRX SCAN+6,3,1
CLA SUM
TRL $FDVD,1,['7400000000404']
STR AVE
TRL $FMLY,1,ALPHA
TRL $FSUB,1,MAX
STR ASAV
TRL $TTYCT,2,++2
TRU CON
CLA MAXPOS
SUB [7]
CAM (A)
SUB [6]
TRP SCAN+3
CLA ASAV
TRN TRACK
TRU SCAN+3
TTYDC STR ASAV
CLA (Q)
SHR  24
SUB  [7]
TRZ  $EXEC
CLA  (Q)
SUB  ['"BMTBL"']  7
TRZ  TABLE
SUB  ['"BRK"-"BMTBL"']  7
TRZ  BRK
SUB  ['"CAL"-"BRK"']  7
TRZ  CAL
SUB  ['"CON"-"CAL"']  7
TRZ  CON
SUB  ['"PRINT"-"CON"']  7
TRZ  PRINT
SUB  ['"SCNPAT"-"PRINT"']  7
TRZ  SCNPAT
SUB  ['"SCNTBL"-"SCNPAT"']  7
TRZ  SCNTBL
SUB  ['"SEARCH"-"SCNTBL"']  7
TRZ  SCAN
SUB  ['"SET"-"SEARCH"']  7
TRZ  SET
SUB  ['"SIM"-"SET"']  7
TRZ  SIM
SUB ["TRACK"="SIM"], 7
TRZ TRACK
SUB ["TRKTBL"="TRACK"], 7
TRZ TRKTBL
SUB ["UNBRK"="TRKTBL"], 7
TRZ UNBRK
SUB ["UNSIM"="UNBRK"], 7
TRZ UNSIM
WAT ['405277520000'], 1
TRU STTY4

SIM MOV (1), '7777'
TRU CON

UNSIM MOV (2), '7777'
TRU CON

TRKTBL WAT BMAPL, 4
LDX 3, 3
WAT ['150012'], 1
CLS (IX4)
ADD MAXPOS
SUB (5)
TRL $PRI
WAT ['404040000000'], 1
CLA MAGS-1, 4
TRL $PRFL
TRX TRKTBL+2,3
TRU CON
SAV2 BS 1
TRACK MOV ["TRACK"],MODE
WAT ['150012'],1
WAT ['"TRACK"'],1
CLA MAXPOS
SUB [7]
CAM (A)
SUB [6]
TRP SCAN
MOV (Z),MAX
MOV (Z),SUM
CLA MAXPOS
SUB [1]
STR (1X3)
MOV [3],(1X4)
TLOOP TRL $BMDAT,1,TEMP1
CLA TEMP1
STR ANGLES-1,4
TRL $SETBM
TRL $RAD,1,1
TRL $FDVD,1,TEMP1+1
STR TEMP
STR MAGS-1,4
TRL $FADD,1,SUM
STR SUM
CLA MAX
TRL $FSUB,1,TEMP
TRP **5
MOV TEMP,MAX
MOV (IX3),MAXPOS
MOV (IX4),RELMAX
TRX TLOOP,3,1
CLA SUM
TRL $FDVD,1,['6000000000402']
STR AVE
TRL $FMLY,1,BETA
TRL $FSUB,1,MAX
STR ASAV
TRL $TTYCT2,++2
TRU CON
TRP SCAN
TRU TRACK+3
RELMAX BS 1
MAGS BS 3
ANGLES BS 3
CAL TRL $CALIB
TRU $TTY4
TABLE TRL $PRBL
TRU CON
PRINT WAT ['150012'], 1
WAT VN 3
RAT (A), 1
WAT ['40750000000'], 1
SUB ['AVE'], 7
TRZ P2
SUB ['MAX'-'AVE'], 7
TRZ P1
SUB ['MAXANG'-'MAX'], 7
TRZ P4
SUB ['MAXPOS'-'MAXANG'], 7
TRZ P3
SUB ['MODE'-'MAXPOS'], 7
TRZ P5
WAT ['405277520000'], 1
TRU PRINT
XY TAB 1700,
VN ALPHA 3 VARIABLE NAME:
P1 CLA MAX
TRU FLPRNT
P2 CLA AVE
TRU FLPRNT

P3 CLA MAXPOS
SUB [7]
TRU INTPR

P4 MOV MAXPOS,(IX3)
TRL $BMDAT,1,TEMP1
CLA TEMP1
TRU FLPRNT

P5 WAT MODE,1
TIU *
TRU CON

BRK MOV [1],BKind
TRU CON

UNBRK MOV (Z),BKind

CON CLA BKind
TRZ *+2
TRU $TTY4

CON1 CLA ASAV
MOV (IX2),(PC)

BKind BS 1
FLPRNT TRL $PRFL
TRU CON

INTPR TRL $PRI
TRU CON
SET  WAT ['150012'],1
WAT  VN,3
RAT  (A),1
WAT ['407540000000'],1
STR  TEMP
CLA  (Z)
RPT  4,1
STR  STRING
RAT  STRING,5
TIU *
TRL $CNVTF,1,STRING
TRU  SETQ
STR  TEMP1
CLA  TEMP
SUB  ['"ALPHA"]
TRZ  SETA
SUB  ['"BETA"-"ALPHA"'],7
TRZ  SETB
SUB  ['"SIMANG"-"BETA"'],7
TRZ  SETS
SETQ  WAT ['405277520000'],1
TRU  SET
SETS  CLA ['7774']
TRU *+2
SETA CLA AA
RPA SETSTR+1
TRU SETSTR

SETB CLA AB
RPA SETSTR+1

SETSTR CLA TEMP1
STR **
TRU CON

SCNTBL WAT BMAPL,4
LDX 5,3,0
WAT ['150012'],1
CLA (IX3)
SUB [7]
TRL $PRI
WAT ['404040000000'],1
CLA SCNDAT,3
TRL $PRFL
TRX SCNTBL+2,3,1
TRU CON

BMAPL TAB '1500120012'
ALPHA 3 BEAM AMPL.

SCNPAT TRL $PLOT,1,XY1
TRL $PLOT,1,XY2
TRL $UP
TRL $SPL\text{O}\text{T}, 1, XY3
TRL $SPL\text{O}\text{T}, 1, XY4
TRL $\text{SUP}
TRL $SPL\text{O}\text{T}, 1, XY5
LDX 15, 3, 0
PLOOP TRL $SBM\text{D}\text{A}\text{T}, 1, TEMP1
CLA TEMP1
TRL $FADD, 1, ['550000000407']
TRL $FM\text{L}\text{Y}, 1, ['54343434403']
TRL $FADD, 1, ['620000000406']
TRL $FL\text{I}\text{N}\text{T}
STR X+1
CLS SCNDAT, 3
TRL $FDV\text{D}, 1, MAX
TRL $FM\text{L}\text{Y}, 1, ['620000000412']
TRL $FADD, 1, ['620000000412']
TRL $FL\text{I}\text{N}\text{T}
STR X
TRL $SPL\text{O}\text{T}, 1, X
TRX PLOOP, 3, 1
TRL $SPL\text{O}\text{T}, 1, XY4
TRL $\text{SUP}
TRL $SPL\text{O}\text{T}, 1, XY6
TRL $SOR\text{I}\text{G}, 1, ZZ
TRL $UP
TRU CON
X BS  2
ZZ TAB ,
XY6 TAB 650,0
XY1 TAB 0,10
XY2 TAB 650,10
XY3 TAB 800,0
XY4 TAB 800,1010
XY5 TAB 800,10
MODE BS  1
MAX BS  1
SUM BS  1
USE CALRD,DEAN
TEMP1 BS  2
USE UTIL,LIB
SCNDAT BS  16
TEMP BS  1
MAXPOS BS  1
AVE BS  1
ALPHA TAB '400000000402'
SSAV BS  1
ASAV BS  1
USE EXRET,LIB
USE PLOT,LIB

STRING BS 5

AA HLT ALPHA

AB HLT BETA

BETA TAB '5000000000401'

USE TTYCT,DEAN

END
APPENDIX H
DF TEST PROGRAM

NAME: DFTS

* THIS PROGRAM IS DESIGNED TO MEASURE DIRECTION FINDING SYSTEM ACCURACY.

HLT $BEGIN

$BEGIN TRL $SCRINIT

CLA [-1]
RPT 449 , 1
STR AVETAB
MOV (Z) , AVE
MOV (Z) , MSQ

BEGIN TRL $PLOT, 1 , XY1
TRL $PLOT, 1 , XY2
TRL $SUP
TRL $PLOT, 1 , XY3
TRL $PLOT, 1 , XY4
TRL $SUP

MLOOP SBB RPCNT , 1
TRZ **2
TRN **3
CLA ANG1
TRU ML2
CLA RECIND
TRZ ML5
LDX 150,3,0
ML4 CLA AVETAB,3
ADD [1]
TRZ ML3
CLA AVETAB,3
SUB ANG1,.7
TRZ ML3
TRX ML4,3,3
WAT ['150012'],1
WAT SUU,3
TRU ML1
ML3 CLA ANG1
STR AVETAB,3
CLS AVE
TRL $FDVD,1,FRP1
STR AVETAB+1,3
CLA MSQ
TRL $FDVD,1,FRP1
STR AVETAB+2,3
MOV (Z),AVE
MOV (Z),MSQ
ML5 MOV (Z),RECIND
SBB PNTCNT,,1
TRP *+2
TRU ML1
CLA ANG1
TRL $FADD,1,['400000000401']
MOV RP1,RPCNT
STR ANG1
TRU MLOOP

ML1
WAT M1,3
CLA (Z)
RPT 4,,1
STR BUF
RAT BUF,5
TRL $RAND
TIU *-1
CLA BUF
SUB ['700000000']
TRZ EXEC
TRL $CNVT,1,BUF
TRU NONUM
STR ANG1
MOV RP1,RPCNT
TRU MLOOP

ML2
STR '7774'
CAM (A)

TRL $FSUB,1,'S500000000407'

TRP ERR

MOV (Z),MAX

MOV [1],RECIND

LDX 15,3,0

SLOOP TRL $BMDAT,1,TEMP1

TRL $SETBM

TRL $RAND

TRL $FMLY,1,SIGMA

STR TEMP

TRL $RAD,1,1

TRL $FADD,1,TEMP

STR UCDAT,3

TRL $FDVD,1,TEMP1+1

STR SCNDAT,3

STR TEMP

CLA MAX

TRL $FSUB,1,TEMP

TRP ++3

MOV TEMP,MAX

MOV (IX3),MAXPOS

TRX SLOOP,3,1

CLA MAXPOS
SUB [1]
STR (IX3)
TRN PLOW
SUB [12]
TRP PHIGH
MOV [3], (IX4)
TLOOP TRL $BMDAT, 1, TEMP1
CLA TEMP1
STR ANGLES-1, 4
CLA SCNDAT, 3
STR MAGS-1, 4
CLA UCDAT, 3
STR UCMAGS-1, 4
TRX TLOOP, 3, 1
LDX ANGLES, 3, MAGS
TRL $INTERP
STR TEMP
CLA PIND
TRZ TB1
WAT ['150012'], 1
LDX 3, 3
TBLOOP WAT ['150012'], 1
CLS (IX4)
ADD MAXPOS
SUB [5]

TRL $PRI

WAT ['404040000000']

CLA MAGS-1,4

TRL $PRFL

WAT ['404040000000']

CLA UCMAGS-1,4

TRL $PRFL

TRX TBLOOP,3

WAT ['1500120012']

CLA TEMP

TRL $PRFL

TB1 CLA ANG1

TRL $FSUB,1,TEMP

STR TEMP

TRL $FADD,1,AVE

STR AVE

CLA TEMP

TRL $FMLY,1,TEMP

TRL $FADD,1,MSQ

STR MSQ

CLA TEMP

TRL $FMLY,1,['620000000406']

TRL $FLINT
ADD [400]
STR X
TRN XH
SUB [850]
TRP XL
XR CLA ANG1
TRL $FMLY,1,'500000000403'
TRL $FLINT
ADD [500]
STR Y
TRL $MARK,1,X
TRU MLOOP
XL MOV [844],X
TRU XR
XR MOV [6],X
TRU XR
PHIGH MOV [12],(IX3)
TRU TLOOP-1
PLOW MOV (Z),(IX3)
TRU TLOOP-1
ERR WAT ['405277520000'],1
TRU ML1+1
NONUM CLA BUF+1
TRZ ++2
TRU ERR
CLA BUF
SUB ["AVE"], 7
TRZ AVE2
SUB ["BMTBL"-"AVE"], 7
TRZ TABLE
SUB ["CAL"-"BMTBL"], 7
TRZ CAL
SUB ["GAIN"-"CAL"], 7
TRZ GAIN
SUB ["NOPR"-"GAIN"], 7
TRZ NOPR
SUB ["PAGE"-"NOPR"], 7
TRZ PAGE
SUB ["PR"-"PAGE"], 7
TRZ PR
SUB ["RMS"-"PR"], 7
TRZ RMS
SUB ["RPT"-"RMS"], 7
TRZ RPT
SUB ["SCAT"-"RPT"], 7
TRZ SCAT
SUB ["SIGMA"-"SCAT"], 7
TRZ SIG
SUB ["SIM"-"SIGMA"] 7
TRZ SIM
SUB ["STAT"-"SIM"] 7
TRZ ERR
SUB ["STDEV"-"STAT"] 7
TRZ STDEV
SUB ["UNSIM"-"STDEV"] 7
TRZ UNSIM
TRU ERR
TABLE TRL $SPRTBL
TRU MLOOP
CAL TRL $SCALIB
TRU MLOOP
PAGE TRL $MOVE,1,OR1
TRL $ORIG,1,OR2
TRU BEG2
OR1 TAB 1250,50
OR2 TAB 400,50
SIM MOV [1], '7777'
TRU MLOOP
UNSIM MOV [2], '7777'
TRU MLOOP
EXEC TRL $MOVE,1,NOR
TRU $SEXEC
NOR  TAB  850,0
SIG  WAT ['407540000000']1
TRL $READF1,SIGMA
TRU MLOOP
PR  MOV [1],PIND
TRU MLOOP
NOPR  MOV [Z],PIND
TRU MLOOP
RPT  WAT M2,2
TRL $READI1,RP1
STR RPCNT
TRL $INTFL
STR FRP1
TRU ML1
SCAT  WAT M2,2
TRL $READI1,PNT1
STR PNTCNT
MOV RP1,RPCNT
CLA ANG1
TRL $FADD1,['4000000000401']
STR ANG1
TRU MLOOP
GAIN LDX 4095,3,7
TRL $SETBM
G2 RAN (A), 1, 6
TIU *
STR (DISP)
SUB ['1400']
TRN 4
SNS ++1, (SL2)
SNR ++1, (SL1)
TRU ++3
SNR ++1, (SL2)
SNS ++1, (SL1)
TRL $FDVD
TRX G2.3
TRU MLOOP

AVE2 TRL TEST
CLA 1, 3
TRL FIXUP
TRL $MARK1, x
TRU ST2

RMS TRL TEST
CLA 2, 3
TRL $SQRT
TRL FIXUP
TRL $MARK2, 1, x
TRU ST2
STDEV  TRL  TEST
CLA  1,3
STR  TEMP
TRL  $FMLY,1,TEMP
STR  TEMP
CLA  2,3
TRL  $FSUB,1,TEMP
TRL  $SQRT
TRL  FIXUP
TRL  $MARK3,1,X
TRU  ST2

ST1  WAT  CLMSG,4
CLA  (Z)
RAT  (A),1
TIU  *
TRZ  */+2
TRU  MLOOP
CLA  [-1]
RPT  449,,1
STR  AVETAB
TRU  MLOOP

FIXUP  CLS  (A)
MOV  (PCS),SAVE
TRL  $FMLY,1,['620000000406']
TRL $FLINT
ADD [400]
STR X
MOV SAVE,(PC)

TEST MOV (PCS),SAVE2
LDX 150,3,AVETAB
CLA 0,3
ADD [1]
TRZ ST1
CLA 0,3
TRL $FMLY,1,['5000000000403']
TRL $FLINT
ADD [500]
STR Y
MOV SAVE2,(PC)

ST2 TRX TEST+2,3,3
TRU ST1

CLMSG ALPHA 5 CLEAR?
SUU ALPHA 3 STORAGE USED UP.

RECIND BS 1
AVE BS 1
MSG BS 1
FRP1 TAB '4000000000401'

AVETAB BS 450
SAVE BS 1
SAVE2 BS 1
PNT1 BS 1
PNTCNT BS 1
RP1 BS 1
RPCNT BS 1
M2 ALPHA 2 COUNT:
PIN1 BS 1
M1 TAB '1500120012', '1200127540',
BUF BS 5
ANG1 BS 1
MAX BS 1
TEMP1 BS 2
UCDAT BS 15
SCNDAT BS 15
TEMP BS 1
MAXPOS BS 1
ANGLES BS 3
MAGS BS 3
UCMAGS BS 3
XY1 TAB 400,50
XY2 TAB 400,950
XY3 TAB 0,500
XY4 TAB 850,500
SIGMA BS 1
X BS 1
Y BS 1
USE INTERP,DEAN
USE CALRD,DEAN
USE UTIL,LIB
USE EXRET,LIB
USE PLOT,LIB
USE MARK,DEAN
USE READ1,DEAN
USE RAND,DEAN
END
APPENDIX I
CONTROL COMPUTER CHARACTERISTICS

The purpose of this appendix is to describe the hardware and software associated with the digital computer used to control the four-element array. The description is given in sufficient detail that programs and hardware discussed elsewhere in the report are understandable. The basic computer was manufactured by IBM and called the Minimal Informer. It is a 37-bit parallel-binary stored program computer and uses the Fielddata instruction set.

Registers: The central processos has a total of 12 programmable registers and a core memory of 4096 37-bit (plus parity) words. The registers and their functions are tabulated in Table 5. Note that register lengths are not all the same. High-order bits are lost when data are transferred from memory or a long register to a shorter register. When data are transferred from a short register to a longer register, high-order zeros are dummied-in.

Number system: The number system used by the computer is 36-bit binary fractional magnitude with sign as shown below.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>26</td>
<td>----</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c|c}
S & N \\
\hline
\text{Sign} & \text{Magnitude} \\
\end{array}
\]
TABLE 5  
CPU REGISTERS

<table>
<thead>
<tr>
<th>Name</th>
<th>Uses</th>
<th>Length</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero register</td>
<td>source of zero words, always zero, read only</td>
<td>37 bits</td>
<td>70000(_8)</td>
</tr>
<tr>
<td>index register 1</td>
<td>counting, address modification</td>
<td>12 bits</td>
<td>70001(_8)</td>
</tr>
<tr>
<td>index register 2</td>
<td>counting, address modification</td>
<td>12 bits</td>
<td>70002(_8)</td>
</tr>
<tr>
<td>index register 3</td>
<td>counting, address modification</td>
<td>12 bits</td>
<td>70003(_8)</td>
</tr>
<tr>
<td>index register 4</td>
<td>counting, address modification</td>
<td>12 bits</td>
<td>70004(_8)</td>
</tr>
<tr>
<td>A register</td>
<td>accumulator, addition subtraction multiplication division, shifting, logical operations</td>
<td>37 bits</td>
<td>70010(_8)</td>
</tr>
<tr>
<td>Q register</td>
<td>quotient register, multiplication division, shifting</td>
<td>37 bits</td>
<td>70011(_8)</td>
</tr>
<tr>
<td>program counter</td>
<td>holds address of next instruction to be executed</td>
<td>15 bits</td>
<td>70013(_8)</td>
</tr>
<tr>
<td>program counter store</td>
<td>holds return address from subroutines</td>
<td>37 bits</td>
<td>70014(_8) also mem. loc. 14(_8)</td>
</tr>
<tr>
<td>display register</td>
<td>console indicator light display, write only</td>
<td>37 bits</td>
<td>70016(_8)</td>
</tr>
<tr>
<td>switch register 1</td>
<td>console switch register, read only</td>
<td>37 bits</td>
<td>70020(_8)</td>
</tr>
<tr>
<td>switch register 2</td>
<td>console switch register, read only</td>
<td>37 bits</td>
<td>70021(_8)</td>
</tr>
</tbody>
</table>
The largest number is $1-2^{-36}$. The smallest is $-(1-2^{-36})$. Minus zero is a valid number and in fact results when a positive number in the accumulator (A) register is reduced to zero by an addition or subtraction operation.

**Addressing and instruction format:** Memory and registers can be referenced by 15-bit addresses either directly or by means of indexing. If indexing is specified, the contents of the specified index register are added to the address part of the instruction word to obtain an effective address. The instruction format is shown below.

<table>
<thead>
<tr>
<th></th>
<th>37</th>
<th>36</th>
<th>31</th>
<th>30</th>
<th>28</th>
<th>27</th>
<th>16</th>
<th>15</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP Code</td>
<td>$\gamma$</td>
<td>$\beta$</td>
<td>$\alpha$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit positions 31 through 36 contain a 6-bit operation code. Bit positions 1 through 15 ($\alpha$) contain the 15-bit address of an operand in memory reference instructions. Bit positions 28 through 30 ($\gamma$) specify the index register (1, 2, 3 or 4) whose contents are to be added to $\alpha$ before the instruction is executed. A $\gamma$ of zero specifies no indexing.

**Sense instructions:** The $\beta$ part of the instruction word (bits 16 through 27) performs different functions depending on the instructions. One of the functions is to specify an indicator or condition to be set, reset or tested by the sense instructions. Table 6 gives the sense functions of corresponding $\beta$ codes. The SEN (sense), SNS (sense and set) and SNR (sense and reset) instructions have operation codes 05a,
<table>
<thead>
<tr>
<th>Octal β code</th>
<th>Function</th>
<th>SNR</th>
<th>SNS</th>
<th>SEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-76</td>
<td>even numbers-I/O converter in use</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>100</td>
<td>overflow alarm</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>interpret sign mode</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>continue on I/O error</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>sense switch 1</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>sense switch 2</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>sense switch 3</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>sense switch 4</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>114</td>
<td>sense light 1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>115</td>
<td>sense light 2</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>I/O converter alarm</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>136</td>
<td>break occurred</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>allow interrupts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>141</td>
<td>allow I/O interrupts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>allow CPU interrupts</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>153</td>
<td>write EOF</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>memory alarm</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>bit error</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>I/O converter deselect*</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>174</td>
<td>generate bit error*</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>complement memory parity</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Transfer is forced
06₈ and 07₈, respectively. If indexing is specified by γ, the contents of the specified index register are added to α to form an effective address. For the SEN instruction, if the condition being tested is met or the indicator being tested is set, the effective address is placed in the program counter causing a transfer to occur. Otherwise the program counter is incremented by 1 and the program continues in sequence. For the SNS and SNR instructions the specified indicators or conditions are set and reset, respectively. For these instructions if a change in state of the specified indicator occurs the computer transfers to the effective address, otherwise it continues in sequence.

**Overflow:** For instructions where accumulator overflow is possible, that is, the result of executing the instruction is greater than or equal to unity and will not fit in the A register, the action taken by the computer is controlled by the β part of the instruction word as specified in Table 7. Two exceptions are the add β (ADB) and subtract β (SBB) instructions where the equivalent of β = 5₈ is forced.

Table 8 describes the operation of the central processor instructions. In order to keep the table reasonably short the following conventions have been used:

1) α means the value of the α part of the instruction word.

   If the instruction has been indexed, it means the sum of the α part of the instruction word and the contents of the specified index register.
### TABLE 7
ACTION TAKEN ON OVERFLOW*

<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit 0 Action</th>
<th>Bit 1 Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>clear OA before instruction execution</td>
<td>no action</td>
</tr>
<tr>
<td>17</td>
<td>set OA on overflow</td>
<td>no action</td>
</tr>
<tr>
<td>16</td>
<td>set OA on overflow and halt</td>
<td>continue on overflow</td>
</tr>
</tbody>
</table>

* ADB and SBB force equivalent of $101_2$ for bits 18-16
**TABLE 8**  
**CPU INSTRUCTIONS**

<table>
<thead>
<tr>
<th>Inst &amp; Code</th>
<th>Instruction</th>
<th>Repeatable</th>
<th>Overflow Possible</th>
<th>Function</th>
<th>A</th>
<th>Q</th>
<th>Index Range</th>
<th>Address e</th>
<th>Overview Conditions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD 24</td>
<td>Add Data</td>
<td>Yes</td>
<td>Yes</td>
<td>A, Q</td>
<td>C(A) + β</td>
<td>C(A)</td>
<td>17 = [C(A) + β]</td>
<td>1-12</td>
<td>C(β) positive and 1 carry from bit 30 of adder</td>
<td>Bits 16-18 do not control Action on overflow, C.A. In case and computer continues in sequence.</td>
</tr>
<tr>
<td>ADD 12</td>
<td>Add</td>
<td>Yes</td>
<td>Yes</td>
<td>A, Q</td>
<td>C(A) + C(e)</td>
<td></td>
<td></td>
<td></td>
<td>Same sign and 1 carry from bit 30 of adder</td>
<td>Bits 16-18 control action on overflow.</td>
</tr>
<tr>
<td>AIM 13</td>
<td>Add</td>
<td>Yes</td>
<td>Yes</td>
<td>A, Q</td>
<td>C(A) + [C(e)]</td>
<td></td>
<td></td>
<td></td>
<td>Sign of A positive and 1 carry from bit 30 of adder</td>
<td>Bits 16-18 control action on overflow.</td>
</tr>
<tr>
<td>CAM 11</td>
<td>Clear and Add Magnitude</td>
<td>Yes</td>
<td></td>
<td>A, Q</td>
<td>+ [C(e)]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sign included; if e = 0, contents of A and Q unchanged.</td>
</tr>
<tr>
<td>C'LA 10</td>
<td>Clear and Add</td>
<td>Yes</td>
<td></td>
<td>A, Q</td>
<td>C(e)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sign is not included; if e = 0, contents of A unchanged.</td>
</tr>
<tr>
<td>CLN 14</td>
<td>Clear and Subtract</td>
<td>Yes</td>
<td>1-36: C(A) - 36 17: Complement 1e</td>
<td>A, Q</td>
<td>C(A) - C(e)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSM 15</td>
<td>Clear and Add, Magnitude</td>
<td>Yes</td>
<td></td>
<td>A, Q</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVL 35</td>
<td>Cycle Left Long</td>
<td>Yes</td>
<td>A, Q cycled left a mod 128 places</td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVL 34</td>
<td>Cycle Left Short</td>
<td>Yes</td>
<td>A cycled left a mod 128</td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVI 22</td>
<td>Divide</td>
<td>Yes</td>
<td>Remainder</td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVS 76</td>
<td>Divide Fast</td>
<td>Yes</td>
<td>22-26: Quotient 1-15: Remainder 16-25: Zero</td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVL 23</td>
<td>Divide Long</td>
<td>Yes</td>
<td>Remainder</td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HIT 00</td>
<td>Hit</td>
<td>Yes</td>
<td></td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDE 53</td>
<td>Load Index Register</td>
<td>Yes</td>
<td></td>
<td>A, Q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In 4 index register computers: 1 = 4, 2 + 1 = 1</td>
<td></td>
</tr>
<tr>
<td>LGA 02</td>
<td>Logical Add</td>
<td>Yes</td>
<td>Logical Sum</td>
<td>C(β) + C(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 x 0 = 0 0 x 1 = 1 1 x 0 = 1 1 x 1 = 1 bits 1-37</td>
<td></td>
</tr>
<tr>
<td>LGM 02</td>
<td>Logical Multiply</td>
<td>Yes</td>
<td>Logical Product C(β) x C(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 x 0 = 0 0 x 1 = 1 1 x 0 = 1 1 x 1 = 1 bits 1-37</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Indirect</td>
<td>Repeatable</td>
<td>Overline</td>
<td>Donable</td>
<td>Function</td>
<td>A</td>
<td>Q</td>
<td>Stack</td>
<td>Address</td>
<td>Conditions</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>----------</td>
<td>---------</td>
<td>-----------------------------------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>HM 04</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Logical Negation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HM 05</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>EQ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HM 06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Memory Test (special sequence)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HM 27</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Multiply</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLR 21</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Multiply and Add</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLR 22</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDW 25</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HM 37</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Normalize</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HFA 24</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Replace Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HFA 01</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDI 26</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Subtract</td>
<td>C(A) - C(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDI 27</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Subtract</td>
<td>C(0) - 32 - C(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNX 26</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Sense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLI 20</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Shift Left</td>
<td>C(0) shifted left and 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* = Logical "AND"  
+ = Logical "OR"
<table>
<thead>
<tr>
<th>Inst. Code</th>
<th>Condition</th>
<th>Repeatability</th>
<th>Overflow Possible</th>
<th>Function</th>
<th>A</th>
<th>Q</th>
<th>Indx Reg</th>
<th>Address</th>
<th>Overflow Conditions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift 32</td>
<td>Yes</td>
<td>Yes</td>
<td>Shift Right</td>
<td>CIA(A) shifted right $a$ mod 128</td>
<td>$1$ is shifted out of position 38 of A register</td>
<td>Sign not included, inject 0's to left.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shift 31</td>
<td>Yes</td>
<td>Yes</td>
<td>Shift Left Long</td>
<td>CIA(A,Q) shifted left $a$ mod 128</td>
<td>$(\beta) = 1, a - PC$ and $0 &lt; \beta$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\beta) = 0, 1 - PC - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRK 07</td>
<td>Yes</td>
<td></td>
<td>Sense and Reset</td>
<td></td>
<td>$(\beta) = 0, a - PC$ and $1 - \beta$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$(\beta) = 1, 1 - PC - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRL 33</td>
<td>Yes</td>
<td></td>
<td>Shift Right Long</td>
<td>CIA and Q</td>
<td>Shift right $a$ mod 128 places</td>
<td>Sign excluded, inject 0's to left of A register.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STN 50</td>
<td>Yes</td>
<td>Yes</td>
<td>Store</td>
<td>CIA(A)</td>
<td>Different sign and 1 carry from bit 36 of adder</td>
<td>Bits (16-15) control overflow action.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB 16</td>
<td>Yes</td>
<td>Yes</td>
<td>Subtract</td>
<td>CIA(A) - CIA($#$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRC 47</td>
<td>Yes</td>
<td>Yes</td>
<td>Compare</td>
<td>CIA(A)</td>
<td>$H:\text{RPT, TRC}$ then $P$ contains repeat count remaining</td>
<td>$C(#) &gt; CIA(A), PC + 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$C(#) &lt; CIA(A), PC + 1$</td>
<td>$C(#) = CIA(A), PC + 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRL 41</td>
<td>Yes</td>
<td></td>
<td>Load PC+ Register and Transfer</td>
<td></td>
<td>$PC + 1 - PCS reg, a - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRN 46</td>
<td>Yes</td>
<td></td>
<td>Transfer on Negative</td>
<td></td>
<td>$(\alpha) = 1, a - PC$</td>
<td>$(\alpha) = 0, (PC + 1) - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRP 44</td>
<td>Yes</td>
<td></td>
<td>Transfer on Positive</td>
<td></td>
<td>$(\alpha) = 0, a - PC$</td>
<td>$(\alpha) = 1, (PC + 1) - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRS 42</td>
<td>Yes</td>
<td></td>
<td>Transfer to PCS Register</td>
<td></td>
<td>$C(PCS) reg - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRU 40</td>
<td>Yes</td>
<td></td>
<td>Transfer Unconditional</td>
<td></td>
<td>$a - PC$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRZ 43</td>
<td>Yes</td>
<td></td>
<td>Transfer on Index</td>
<td></td>
<td>$H(# + 1) &lt; 0; (# + 1) = 1$</td>
<td>$</td>
<td>a</td>
<td>&gt; 0, a - PC$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRZ 45</td>
<td>Yes</td>
<td></td>
<td>Transfer on Zero</td>
<td></td>
<td>$</td>
<td>a</td>
<td>&gt; 0, a - PC$</td>
<td>$</td>
<td>a</td>
<td>&gt; 0, PC + 1$</td>
</tr>
</tbody>
</table>
2) $C(\alpha)$ means the contents of address $\alpha$.

3) $C(x)$ where $x$ is a register name means the contents of the register $x$. $C(A)$, for example, means the contents of the accumulator (A) register.

4) If a register column is left blank the corresponding register is unaffected by the instruction. For instructions referencing two index registers, $I\gamma$ and $I(\gamma+1)$, if $\gamma$ is 4, $\gamma+1$ is 1.

Several instructions are not completely specified in Table 8. Three of these, the sense instructions, have already been described. Another which needs further discussion is the repeat (RPT) instruction. The repeat instruction causes the instruction immediately following it to be executed $\alpha+1$ times. After each execution of the instruction its $\alpha$ is increased by $\beta$ of the repeat instruction. $\beta$ of the repeat instruction is also placed in index register 4. If the repeated instruction calls for indexing, it is indexed normally before the first execution. Both instructions remain unchanged in memory. For the sequence RPT, TRC, the TRC (transfer or compare) instruction is repeated until the contents of the accumulator are less than or equal to the contents of address $\alpha$ for a maximum of $\alpha$ (of the RPT) $+ 1$ times. If the TRC is repeated the specified number of times, one instruction is skipped and the computer continues in sequence. If the contents of address $\alpha$ are equal to the contents of the accumulator, two instructions are skipped. If the contents of the accumulator are less than the
contents of address a the computer continues in sequence. The remaining repeat count is placed in index register 3. For the RPT, MOV (move) sequence the γβ address is indexed by the contents of index register 2. When the sequence is completed the Q register contains the last address where data was extracted.

Timing for instructions is given in Table 9.

**Interrupt:** A number of conditions can cause an interrupt. When an interrupt occurs the contents of the program counter are placed in memory location 15₈ and 00200₈ is placed in the program counter so that the computer transfers to location 200₈. For an interrupt to occur the allow—interrupts indicator (140₈) must be set. When an interrupt occurs the allow-interrupts indicator is reset to prevent any further interrupts until the indicator is set again by the program. In addition, for interrupts to occur, at least one of two other indicators must be set. The allow-CPU indicator must be set for memory parity or bit errors to cause interrupts. Setting the allow-I/O indicator enables a number of I/O conditions to cause interrupts.

**Input/Output:** Input/output operations are handled by a separate processor called the I/O converter. The relationship between the I/O converter, the central processor and the I/O devices is shown in Fig. 63. When an I/O instruction is recognized by the central processor it is transmitted to the I/O converter for decoding and processing. The central processor normally continues with the succeeding instructions
### TABLE 9

**CPU INSTRUCTION TIMING**

<table>
<thead>
<tr>
<th>OP Code</th>
<th>Mnemonic</th>
<th>Average Time (sec.)</th>
<th>OP Code</th>
<th>Mnemonic</th>
<th>Average Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>MLY</td>
<td>22.67</td>
<td>50</td>
<td>STB*</td>
<td>22.67</td>
</tr>
<tr>
<td>01</td>
<td>NLY</td>
<td>22.67</td>
<td>51</td>
<td>LOD</td>
<td>33.33</td>
</tr>
<tr>
<td>02</td>
<td>MLL</td>
<td>22.67</td>
<td>52</td>
<td>MOV*</td>
<td>33.33</td>
</tr>
<tr>
<td>03</td>
<td>NLL</td>
<td>22.67</td>
<td>53</td>
<td>LDX</td>
<td>18.67</td>
</tr>
<tr>
<td>04</td>
<td>DLY</td>
<td>22.67</td>
<td>54</td>
<td>RPA*</td>
<td>28</td>
</tr>
<tr>
<td>05</td>
<td>DLL</td>
<td>22.67</td>
<td>55</td>
<td>MSR*</td>
<td>48</td>
</tr>
<tr>
<td>06</td>
<td>NLL</td>
<td>22.67</td>
<td>56</td>
<td>SRA*</td>
<td>28</td>
</tr>
<tr>
<td>07</td>
<td>MLF</td>
<td>22.67</td>
<td>57</td>
<td>SRE*</td>
<td>28</td>
</tr>
<tr>
<td>08</td>
<td>SLL</td>
<td>22.67</td>
<td>58</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>09</td>
<td>STL</td>
<td>22.67</td>
<td>59</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>SNL</td>
<td>22.67</td>
<td>60</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>SML</td>
<td>22.67</td>
<td>61</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>12</td>
<td>ADD*</td>
<td>22.67</td>
<td>62</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>AOM*</td>
<td>22.67</td>
<td>63</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>CLS*</td>
<td>22.67</td>
<td>64</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>DSN</td>
<td>22.67</td>
<td>65</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>SUB*</td>
<td>22.67</td>
<td>66</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>SIM*</td>
<td>22.67</td>
<td>67</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>MLY</td>
<td>22.67</td>
<td>68</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>NLY</td>
<td>22.67</td>
<td>69</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>20</td>
<td>NLL</td>
<td>22.67</td>
<td>70</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>21</td>
<td>MLF</td>
<td>22.67</td>
<td>71</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>22</td>
<td>SLL</td>
<td>22.67</td>
<td>72</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>23</td>
<td>DSL</td>
<td>22.67</td>
<td>73</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>24</td>
<td>DLY</td>
<td>22.67</td>
<td>74</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>25</td>
<td>DLY</td>
<td>22.67</td>
<td>75</td>
<td>SLE*</td>
<td>28</td>
</tr>
<tr>
<td>26</td>
<td>DLY</td>
<td>22.67</td>
<td>76</td>
<td>SLE*</td>
<td>28</td>
</tr>
</tbody>
</table>

* Repeatabl instructions. If instruction is repeated, subtract 1.33 x sec from average time for each repetition after the first. Indexing, if any, applies to first repetition only.

**Non-indexable instruction.** If instruction is indexable and it is indexed, add 2.67 x sec to average time.
Fig. 63. --Block diagram showing the relation between the I/O converter, the central processor and the I/O devices.

while the I/O converter independently processes the I/O instruction.

If the I/O converter is already in use when the central processor encounters an I/O instruction, the central processor is held up until the I/O converter is free. Data are transferred through the central processor between the core memory and the I/O converter as 37-bit words.

I/O instruction format: The I/O instruction word format is shown below.

<table>
<thead>
<tr>
<th>37</th>
<th>36</th>
<th>31</th>
<th>30</th>
<th>22</th>
<th>21</th>
<th>16</th>
<th>15</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP Code</td>
<td>K</td>
<td>J</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The α part (bits 1 through 15) specifies the starting memory address for data transfer. The K part specifies the number of words to be transferred. For write (output) instructions, bits 22 through 30 are used. For read (input) instructions bits 22 through 29 are used to specify either a word count or block count as determined by bit 30 being a zero or a one. Blocks are made up of arbitrary numbers of words and are separated by block marks. Block marks will be explained below. The J part of the instruction word is the device address used by the device-selection multiplexer.

**I/O registers:** Two registers and a memory location are available for programming. The I/O instruction register contains the op-code and J address of the current or last I/O instruction. The K part contains the current contents of the work or block counter and the α part contains the memory address of the next memory location from which data are to be taken or into which data are to be placed. When the I/O instruction is completed the address is one higher than the last location accessed. The address of the I/O instruction register is 70030₈. The maintenance register (address 7002₄₈) contains information on the current status of the I/O converter equipment and is not generally used for programming. Memory location 10₈ contains the current or last I/O instruction executed.

**Data character format:** Data are transferred between the I/O converter and the address-selection multiplexer as 8-bit characters.
The 8 bits are designated, from high order to low order, as P (parity), C (control), I₂, I₁, D₃, D₂, D₁, D₀. For output the P-bit is generated by the I/O converter to give the character odd parity. For input the I/O converter tests for odd parity. A C-bit of 1 indicates a data character. A C-bit of 0 indicates a control character. Meaningful control characters are BLS (block start), BLE (block end), EOF (end of file) and STOP. The codes for the control characters are given in Table 10.

<table>
<thead>
<tr>
<th>BIT</th>
<th>P</th>
<th>C</th>
<th>I₂</th>
<th>I₁</th>
<th>D₃</th>
<th>D₂</th>
<th>D₁</th>
<th>D₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLS</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BLE</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EOF</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>STOP</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE 10
CONTROL CHARACTER CODES

Each write instruction causes a BLS character to be transmitted at the beginning of the output operation. Data characters are generated by breaking down the words to be output. The data characters are followed by two BLE's. EOF's can be substituted for BLE's by executing a SNS 153₈ prior to execution of the output instruction.
For input, 8-bit data characters are assembled into 37-bit computer words. The input data may be divided into blocks for a read-by-blocks instruction (bit 30 of the read instruction word = 1) by BLS's at the beginning of each block and two BLE's or EOF's at the end of each block. If EOF's are used and the I/O interrupt has been enabled by execution of SNS 140 and SNS 141 instructions, an interrupt will occur when the EOF's are read. For a read-by-words (bit 30 = 0) instruction, block marks may or may not be present. The specified number of words is read regardless of how the data are divided into blocks. If a STOP character is read, the input operation is terminated without regard for word or block counts.

Data characters are transferred between the I/O converter and the address selection multiplexer in one of three modes; as octal data, as alphanumeric data in the interpret-sign mode, and as alphanumeric data in the not-interpret-sign mode. For octal output 37-bit computer words are broken into thirteen data characters corresponding to the alphanumeric representations of the sign and the twelve octal digits making up the remainder of the word. The alphanumeric representation is obtained by setting the $I_2$, $I_1$ and $D_3$ bits equal to $110_2$ and the $D_2$, $D_1$ and $D_0$ bits equal to the octal number. Transmission starts at the sign bit and provides to the low order end of the word. For alphanumeric output in the interpret-sign mode the 37-bit word is broken into seven data characters. The sign bit is output as an
alphanumeric zero or one as for octal output. The remainder of the word is output six bits at a time. Each group of six bits forms I₂ through D₀ of the corresponding data character. The interpret-sign mode is set prior to the execution of an alphanumeric I/O instruction by the execution of a SNS 102₈ instruction. In the not-interpret-sign mode the sign bit is ignored and the 37-bit word is broken into six data characters.

For input, the process is reversed and data characters are assembled into 37-bit computer words. For input in the not-interpret-sign mode the sign bit is made zero. Blocks of data to be read need not form an integral number of characters. If block mark characters are encountered, the I/O converter completes incomplete words by supplying low-order zeros. Incomplete words are similarly completed if a STOP character is read.

I/O instructions: A list of input-output instructions is given in Table 11. The search instruction performs an automatic search for data defined by up to thirty descriptors. It hasn't been used in any of the programs published in this work and will not be described. Magnetic tape instructions have been omitted since the present computer system does not have magnetic tape.

Disk memory: The disk memory has a capacity of 20.5 million 8-bit characters. The data are divided into ten thousand tracks of 2050 characters each. The data are written and read by a single pair
TABLE 11
I/0 INSTRUCTIONS

<table>
<thead>
<tr>
<th>Octal OP code</th>
<th>Mnemonic</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>RAN</td>
<td>read alphanumeric</td>
</tr>
<tr>
<td>72</td>
<td>ROK</td>
<td>read octal</td>
</tr>
<tr>
<td>73</td>
<td>SCH</td>
<td>search</td>
</tr>
<tr>
<td>74</td>
<td>WAN</td>
<td>write alphanumeric</td>
</tr>
<tr>
<td>76</td>
<td>WOK</td>
<td>write octal</td>
</tr>
</tbody>
</table>

of heads which are mechanically positioned to the desired track.

This is accomplished by the execution of a write octal instruction with

a J address of 30₈ and a word count of one. The J address specifies

the location of a four-digit BCD track address. Access time is between

100 and 800 milliseconds. Data can be written by a write alphanumeric

instruction with a J address of 30₈. Old data are effectively erased.

Similarly, data may be read by a read alphanumeric instruction. If

more data than can be contained on one track is specified by the word

or block count the heads are automatically moved to the next sequential

track and the input/output operation is continued.

**Teletypewriter**: The teletypewriter uses a modified ASCII code

(American Standard Code for Information Interchange) as shown in

Table 12. The ASCII code bits actually used by the teletypewriter are
given as column headings and down the left margin in the table. The
internal code representation is given in octal along with each character.
For output, bits I₂ through D₀ of the output character are used directly
to form bits b₆ through b₁ of the ASCII character. Bit b₇ is formed by
either complementing or duplicating bit b₆. The mode is set to "com-
plement" at the beginning of the output operation and does not change so
long as non-zero (bits I₂ through D₀) characters are output. A zero-
character is not transmitted to the printer but switches the mode to
"duplicate". The first non-zero character is transmitted to the printer
in the duplicate mode and the complement mode is restored. Trailing
zero-characters are effectively ignored. Control characters are ignored.

For input, characters from columns 0 through 5 are converted
in a way analogous to the output conversion. Columns 0 and 1 produce
two input characters for each keystroke, a zero-character and a
character having I₂ through D₀ equal to b₆ through b₁. Columns 2, 3,
4 and 5 produce single characters with I₂ through D₀ equal to b₂ through
b₁. Columns 6 and 7 produce control-characters with I₂ through D₀
equal to b₆ through b₁. The only characters from columns 6 and 7
that can be generated from the keyboard are the RUBOUT and the
STOP, the latter of which is generated by the "CLR DYBD" key. The
"CLR KYBD" key also generates an interrupt pulse and will cause an
interrupt if the I/0 interrupt has been enables by SNS 140₈ and
SNS 141₈ instructions.
<table>
<thead>
<tr>
<th>ASCII Control Characters</th>
<th>Printing characters</th>
<th>Informer Control Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0020 00 00 20 00</td>
<td>00 00 00 20 10 10</td>
<td>(00)40 (00)60</td>
</tr>
<tr>
<td>0001 0001 0021 00 01 01</td>
<td>00 00 00 00 10 10</td>
<td>(00)41 (00)61</td>
</tr>
<tr>
<td>0002 0022 00 02 02</td>
<td>00 00 00 00 10 10</td>
<td>(00)42 (00)62</td>
</tr>
<tr>
<td>0003 0023 00 03 03</td>
<td>00 00 00 00 10 10</td>
<td>(00)43 (00)63</td>
</tr>
<tr>
<td>0100 0004 0024 00 04 04</td>
<td>00 00 00 00 10 10</td>
<td>(00)44 (00)64</td>
</tr>
<tr>
<td>0101 0005 0025 00 05 05</td>
<td>00 00 00 00 10 10</td>
<td>(00)45 (00)65</td>
</tr>
<tr>
<td>0110 0006 0026 00 06 06</td>
<td>00 00 00 00 10 10</td>
<td>(00)46 (00)66</td>
</tr>
<tr>
<td>0111 0007 0027 00 07 07</td>
<td>00 00 00 00 10 10</td>
<td>(00)47 (00)67</td>
</tr>
<tr>
<td>1000 0010 0030 00 10 10</td>
<td>00 00 00 00 10 10</td>
<td>(00)50 (00)70</td>
</tr>
<tr>
<td>1001 0011 0031 00 11 11</td>
<td>00 00 00 00 10 10</td>
<td>(00)51 (00)71</td>
</tr>
<tr>
<td>1010 0012 0032 00 12 12</td>
<td>00 00 00 00 10 10</td>
<td>(00)52 (00)72</td>
</tr>
<tr>
<td>1011 0013 0033 00 13 13</td>
<td>00 00 00 00 10 10</td>
<td>(00)53 (00)73</td>
</tr>
<tr>
<td>1100 0014 0034 00 14 14</td>
<td>00 00 00 00 10 10</td>
<td>(00)54 (00)74</td>
</tr>
<tr>
<td>1101 0015 0035 00 15 15</td>
<td>00 00 00 00 10 10</td>
<td>(00)55 (00)75</td>
</tr>
<tr>
<td>1110 0016 0036 00 16 16</td>
<td>00 00 00 00 10 10</td>
<td>(00)56 (00)76</td>
</tr>
<tr>
<td>1111 0017 0037 00 17 17</td>
<td>00 00 00 00 10 10</td>
<td>(00)57 (00)77</td>
</tr>
</tbody>
</table>
Plotter: The plotter has four basic pen motions: 0.01 inches in the plus-x direction, 0.01 inches in the minus-x direction, 0.01 inches in the plus-y direction and 0.01 inches in the minus-y direction. In addition the pen can be raised or lowered. Each data character output to the plotter produces one or more functions as indicated by Table 13. X- and y-motions can be combined to produce 45-degree diagonal motions. Zero-characters and control-characters are ignored.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2</td>
<td>I1</td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
**Parallel/serial converter:** In the parallel-to-serial converter, bits $I_2$ through $D_0$ of the data characters are converted to serial NRZ data at a 500 KHz rate. A clock is provided which makes a one-to-zero transition approximately 1µs after a change in the data line and at least 1µs before the next change in the data line. Data are not transmitted continuously but are limited by the maximum character rate of the I/O converter which is about 75 KHz. Control-characters are suppressed.

**A/D converter:** The analog-to-digital converter accepts a 0 to 1.0 volt analog signal and converts it to 10-bit binary 0 to 17778. The conversion is started when the first data character is read. Four zero-characters are sent followed by a character containing the two high-order octal digits and a character containing the two low-order octal digits. Data may be read at approximately a 5 KHz word rate but the input bandwidth has been limited to about 50 Hz to minimize noise.
APPENDIX J
THE ASSEMBLER

A. Introduction

The assembly-language programming system used with the IBM minimal Informer computer is documented in detail in an internal publication of the ElectroScience Laboratory, [10] and the description contained in this appendix is only sufficient to guide an experienced assembly-language programmer in reading the computer programs contained in this report.

The instruction format in this language is less rigid than in most assemblers in that the various fields of the instructions are not required to start or end in specific columns. The fields are separated by one or more spaces.

B. Instruction Fields

The four fields of an instruction are:

a) Label (optional) - one to six alphanumeric characters beginning with an alphabetic character

b) Operation (always required) - a two-, three-, or five-letter alphabetic code specifying either a machine operation, an extended mnemonic to be interpreted in terms of machine
functions, or an assembler command causing one of the following actions:

i) assignment of data storage

ii) assignment of data storage

iii) linkage to external files

iv) termination of assembly.

c) Operand (required except for op-codes of HLT, TRS, or END) - a variable length field composed of subfields separated by commas. The subfields are:

i) Alpha - address

ii) Gamma - index register reference, if used

iii) Beta - increment or second address, if used

iv) K - word or block count

v) J - device number, if used

vi) textual material

Zeros are supplied for any missing subfields of the operand.

d) Comments (optional) - all text material following the third field on any line.

C. Instruction Types

In the following examples of various types of instruction formats, the label field, if used, must not be preceeded by a space character.
A central processor instruction has the form:

Label OP Alpha, Gamma, Beta comments

The move instruction is an exception to this form because the length of its second address requires that the gamma and beta portions of the address be taken together as a single subfield. It has the form:

Label MOV Alpha, Gamma-Beta

An input or output instruction has the form:

Label OP Alpha, K, J comments

A block storage command assigning N words of storage has the form:

Label BS N comments

A tabulation command assigning values to successive words has the form:

Label TAB N1, N2, N3, ... comments

A command to assign N words of alphanumeric data with six characters per word has the form:

Label ALPHA N XXX... comments

A command to designate that an external file is required by a program, where that file is indexed in the disk file directory by files name and user name, has the form:

USE file, name comments
The command which indicates to the assembler that the end of the symbolic program has been reached has the form:

END

D. Special Symbols

The following special symbols are recognized by the assembler:

* In column 1 - entire line is comment
# In column 1 - entry point for this program
$ Prefix to address - external routine
* As address - current location
** As address - to be supplied by program
D Prefix to number - disk address
Unmodified number - decimal constant
" " Enclosing characters - alphanumeric constant
' ' Enclosing number - octal constant
[ ] Enclosing number - decimal literal from literal table
[ ' ' ] Enclosing number - octal literal from literal table
[ " ' ' ] Enclosing characters - alphanumeric literal from literal table
( ) Enclosing characters - machine register or sense indicator

An address reference may be composed of one of the above addresses plus or minus a constant to reference unlabeled instructions or storage.
### E. Hardware Operation Codes

The following mnemonics are recognized as machine operation codes:

<table>
<thead>
<tr>
<th>Misc. Class</th>
<th>Transfer Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT Halt</td>
<td>TRU Transfer Unconditional</td>
</tr>
<tr>
<td>RPT Repeat</td>
<td>TRL Transfer and Load PCS</td>
</tr>
<tr>
<td>LGM Logical Multiply</td>
<td>TRS Transfer to PCS</td>
</tr>
<tr>
<td>LGA Logical Add</td>
<td>TRX Transfer on Index</td>
</tr>
<tr>
<td>LGN Logical Negation</td>
<td>TRP Transfer on (+) A Reg.</td>
</tr>
<tr>
<td>SEN Sense</td>
<td>TRZ Transfer on Zero A Reg.</td>
</tr>
<tr>
<td>SNS Sense and Set</td>
<td>TRN Transfer on (-) A Reg.</td>
</tr>
<tr>
<td>SNR Sense and Reset</td>
<td>TRC Transfer on Compare</td>
</tr>
<tr>
<td>Add Class</td>
<td>Multiply Class</td>
</tr>
<tr>
<td>CLA Clear and Add</td>
<td>MLY Multiply</td>
</tr>
<tr>
<td>CAM Clear and Add Magnitude</td>
<td>MLR Multiply and Round</td>
</tr>
<tr>
<td>ADD Add</td>
<td>DVD Divide</td>
</tr>
<tr>
<td>ADM Add Magnitude</td>
<td>DVL Divide Long</td>
</tr>
<tr>
<td>CLS Clear and Subtract</td>
<td>DVF Divide Fast</td>
</tr>
<tr>
<td>CSM Clear and Subtract Magnitude</td>
<td>MLF Multiply Fast</td>
</tr>
<tr>
<td>SUB Subtract</td>
<td>SBM Subtract Magnitude</td>
</tr>
<tr>
<td>ADB Add Beta</td>
<td>SBB Subtract Beta</td>
</tr>
<tr>
<td>Shift Class</td>
<td>I/O Class</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>SHL     Shift Left</td>
<td>RAN Read Alphanumeric</td>
</tr>
<tr>
<td>SLL     Shift Left Long</td>
<td>ROK Read Octal</td>
</tr>
<tr>
<td>SHR     Shift Right</td>
<td>SCH Search</td>
</tr>
<tr>
<td>SRL     Shift Right Long</td>
<td>WAN Write Alphanumeric</td>
</tr>
<tr>
<td>CYS     Cycle Short</td>
<td>WOK Write Octal</td>
</tr>
<tr>
<td>CYL     Cycle Long</td>
<td></td>
</tr>
<tr>
<td>NRM     Normalize</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Store Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR Store</td>
</tr>
<tr>
<td>LOD Load</td>
</tr>
<tr>
<td>MOV Move</td>
</tr>
<tr>
<td>LDX Load Index</td>
</tr>
<tr>
<td>RPA Replace Address</td>
</tr>
<tr>
<td>MSK Mask</td>
</tr>
</tbody>
</table>

A complete description of these hardware functions is presented in Appendix I.
F. **Extended Operation Codes**

The following mnemonics are recognized as extended operation codes in which additional information such as device number or sense indicators is supplied by the assembler:

- **TBR** - Transfer if break occurred
- **TIU** - Transfer if I/O in use
- **TL1** - Transfer if sense light 1 on
- **TL2** - Transfer if sense light 2 on
- **TOV** - Transfer if overflow indicator set
- **RAB** - Read alphanumeric by blocks
- **RAD** - Read alphanumeric from disk
- **RAT** - Read alphanumeric from typewriter
- **RDB** - Read disk by blocks
- **RTB** - Read typewriter by blocks
- **WAT** - Write alphanumeric on typewriter
- **WAD** - Write alphanumeric on disk
- **WKD** - Write octal disk (reposition R/W heads)

G. **Machine Registers**

The following symbols are recognized as machine registers:

- **(Z)** - Zero register
- **(IX1)** - Index register 1
(IX2) - Index register 2
(IX3) - Index register 3
(IX4) - Index register 4
(A) - A register
(Q) - Q register
(PC) - Program counter
(PCS) - Program counter store
(IPCS) - Interrupt program counter store
(DISP) - Display register
(SR1) - Switch register 1
(SR2) - Switch register 2
(M) - I/O maintenance register
(I0) - I/0 Instruction register
(D) - Disk address register

H. Sense Codes

The following symbols are recognized as sense indicators:

(IU) - I/0 in use
(OVA) - Overflow alarm
(ISN) - Interpret-sign mode
(CIO) - Continue on I/0 error
(SW1) - Sense switch 1
(SW2) - Sense switch 2
(SW3) - Sense switch 3
(SW4) - Sense switch 4
(SL1) - Sense light 1
(SL2) - Sense light 2
(IOA) - I/O alarm
(BRK) - Break occurred
(AI) - Allow interrupt
(AIO) - Allow I/O interrupt
(ACPU) - Allow CPU Interrupt
(WEF) - Write end of file

I. **Data Formats**

Floating point numbers are represented as a signed binary fraction with a nine-bit exponent. The least-significant nine bits of the word represent the power of two multiplier plus 4008. Thus the octal floating point representation of -1.0 would be written as -'40000000401' where the primes denote octal notation.

ASCII control characters are represented internally as 12-bit characters with zeros for the high-order six bits. This is done to distinguish them from the ASCII printing characters which are represented as six-bit characters.
APPENDIX K
LIBRARY INDEX

The system library contains a large number of useful subroutines which may be called from user programs to perform a variety of commonly-needed functions.

These subroutines are grouped in several files under the user name "LIB", and the subroutines in a given file share some common features or applications. For example, the file "PLOT,LIB" contains all of the library subroutines which pertain to the plotter.

Most subroutines are called through a TRL instruction (transfer and load PCS register), which utilizes the PCS register (program counter storage) as explained in Appendix I, to indicate where the return from the subroutine should be directed.

In addition, if an index register is specified in the TRL instruction, that index register will be loaded with the address specified in the beta part of the instruction. This feature is used in many subroutine calls to indicate to the subroutine where it must find or place additional data beyond that which it finds or places in the A- and Q-registers. Index register 1 has been chosen for this purpose in all library subroutines which require such additional data. The remaining three
index registers are undisturbed by library subroutines, but the contents of index register 1 may be lost if used in the calling sequence.

The following list of available library subroutines is arranged according to the grouping within the several files of the library. This list shows the appropriate calling sequence for each library subroutine, along with a brief description of the action taken by the subroutine. The following calling sequences may be used with UTIL, LIB:

TRL $FADD,1,ADDEND
Floating add ADDEND to accumulator.

TRL $FSUB,1,SUBTR
Floating subtract SUBTR from accumulator.

TRL $FMLY,1,MPLR
Floating multiply accumulator by MPLR, return result in accumulator.

TRL $FDVD,1,DVSR
Floating divide accumulator by DVSR, return result in accumulator.

TRL $AFTR,1,TRADD
Arm floating trap. Spill will force transfer to TRADD, location 16 will contain address of instruction causing spill.
**TRL $DFTR**
Disarm floating trap.

**TRL $TFTR**
Test floating trap. Return skips one instruction if trap is armed.

**TRL $PRFL**
Print floating point number in accumulator, using format `-1.0000000E-00`

**TRL $PRI**
Print integer in accumulator, using format `-1234`

**TRL $CNVTF,1,BUFFER**
Convert 5 words of teletype code beginning with BUFFER into a floating point number in accumulator. Return skips one instruction if conversion was successful, continues in sequence if not.

**TRL $CNVTI,1,BUFFER**
Same as $CNVTF, except an integer is returned in the accumulator.

**TRL $INTFL**
Convert integer in accumulator to a floating point number.

**TRL $FLINT**
Convert floating point number in accumulator to an integer.
For debugging purposes, if sense switch 1 is on, floating point arithmetic routines will halt just prior to the return to the calling program. Note that other library routines may use floating arithmetic and have several halts if sense switch 1 is on.

The following calling sequences may be used with CMPLX,LIB:

TRL $CLOD,1,Z
Load A&Q registers with the complex number in location Z and the next following location.

TRL $CSTR,1,Z
Store the complex number in A&Q registers in location Z and the next following location.

TRL $CADD,1,ADDEND
Complex floating add ADDEND pair to A&Q registers.

TRL $CSUB,1,SUBTR
Complex floating subtract SUBTR pair from A&Q registers.

TRL $CMLY,1,MPLR
Complex floating multiply A&Q registers to MPLR pair.

TRL $CDVD,1,DVSR
Complex floating divide A&Q registers by DVSR pair.

TRL $CMAG
Return the magnitude of the complex number in the A&Q registers in the accumulator.
TRL  $CEXP
Return the complex exponential of the complex floating
point number in the A&Q registers.

TRL  $PRCX
Print the complex number in A&Q registers, using
format -1.00000000E-00 -J 1.00000000E-00

The following calling sequences may be used with MATH1, LIB:

TRL  $SQRT
Return the square root of the magnitude of the floating
point number in the accumulator.

TRL  $SIN
Return the sine of the floating point number (in radians)
in the accumulator.

TRL  $COS
Return the cosine of the floating point number (in radians)
in the accumulator.

TRL  $SINCOS
Return the sine of the floating point number (in radians)
in the accumulator, and the cosine of the same number
in the Q-register.

TRL  $EXP
Return the exponential of the floating point number in
the accumulator.
TRL  $LN
Return the natural logarithm of the floating point number
in the accumulator.

The following calling sequence may be used with PLOT, LIB:

TRL  $UP
Lift pen of plotter if it was down.

TRL  $PLOT, 1, X
Move pen in straight line from present position to the
point \((X, Y)\) represented by the integers in location \(X\)
and the next following location, provided fewer than 4096
increments are required.
The pen is lowered at the new position if it was up
previously.

TRL  $MOVE, 1, X
Same as $PLOT, except pen is not lowered.

TRL  $ORIG, 1, X
Reset the plotter origin of that the present position of
the pen is the point \((X, Y)\) specified by the integers in
location \(X\) and the next following location.

TRL  $WHERE, 1, X
Return present position of pen as the point \((X, Y)\) repre-
sented by the integers in location \(X\) and the next following
location.

The following calling sequence may be used with EXRET, LIB:

TRU  $EXEC
Return to executive system.
REFERENCES


   (AD 409 017)

5. Young, J.D., "Antennafiers for Beam-Steering Arrays,"
Abstracts of the 14th Annual Symposium USAF Antenna Research
and Development Program, Monticello, Illinois, October 1964.

Steering and Shaping of Slot Antennafier Arrays," Report 2142-9,
27 November 1967, ElectroScience Laboratory, The Ohio State
University Research Foundation; prepared under Contract
AF 33(615)-3384 for Air Force Avionics Laboratory, Wright-
Patterson Air Force Base, Ohio. (AD 823 944).

7. Interim Technical Report (Techniques for Integrating Solid
State Circuitry into Antennas), 1 December 1967 to 29 February
1968, Report 2142-14, 28 March 1968, ElectroScience Laboratory
The Ohio State University Research Foundation; prepared under
Contract AF 33(615)-3384 for Air Force Avionics Laboratory,
Wright-Patterson Air Force Base, Ohio. (AD 829 428).

8. Widrow, B., Mantey, P.E., Griffiths, L.J. and Goode, B.B.,
"Adaptive Antenna Systems," Proceedings of the IEEE, Vol. 55,
No. 12, December 1967.

of Mutual Coupling in Antenna Arrays," Report 2142-6,
15 September 1966, Antenna Laboratory, The Ohio State University
Research Foundation; prepared under Contract AF 33(615)-3384,
Air Force Avionics Laboratory, Wright-Patterson Air Force Base, Ohio.