USING CONTINUOUS VOICE RECOGNITION TECHNOLOGY AS AN INPUT MEDIUM TO THE N. (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA  J P LOMBARDO JUN 84

UNCLASSIFIED

END

DATE ISSUED  JUN 84

DIG
USING CONTINUOUS VOICE RECOGNITION TECHNOLOGY AS AN INPUT MEDIUM TO THE NAVAL WARFARE INTERACTIVE SIMULATION SYSTEM (NWISS)

by

John Philip Lombardo

June, 1984

Thesis Advisor: G. K. POOCK

Approved for public release, distribution unlimited
A great deal of research has been conducted in the past 20 years concerning the use of voice recognition equipment with computers. The goal of this research has been to improve the man-machine interface. With the breakthrough from discrete to continuous voice recognition technology in the 1970's, a large step toward that goal was taken.
This thesis attempts to show that continuous voice recognition technology can be effectively applied in a highly interactive, computer-aided wargaming environment. Through analysis of the strictly-formatted command syntax of the Naval Warfare Interactive Simulation System (NWISS) and use of commercially available, innovative, continuous speech hardware and software, a new input medium was created for the user of that wargame. The true effectiveness of this application of voice recognition technology must still be tested. Plans for such testing are being made and, to that extent, the thesis objectives are partly met.
Using Continuous Voice Recognition Technology as an Input Medium to the Naval Warfare Interactive Simulation System (NWISS) by

John Philip Lombaro
Civilian, National Security Agency
B.S., Loyola College (Baltimore), 1958
M.B.A., Loyola College (Baltimore), 1979

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Command, Control and Communications)

from the

NAVAL POSTGRADUATE SCHOOL
June 1984

Author: ____________________________

Approved by: _______________________
Thesis Advisor

_______________________________
Mary R. Porter
Second Reader

_______________________________
Chairman, Command, Control and Communications Academic Group

_______________________________
Academic Dean
ABSTRACT

A great deal of research has been conducted in the past 20 years concerning the use of voice recognition equipment with computers. The goal of this research has been to improve the man-machine interface. With the breakthrough from discrete to continuous voice recognition technology in the 1970's, a large step toward that goal was taken.

This thesis attempts to show that continuous voice recognition technology can be effectively applied in a highly interactive, computer-aided wargame environment. Through analysis of the strictly-formatted command syntax of the Naval Warfare Interactive Simulation System (NWISS) and use of commercially available, innovative, continuous speech hardware and software, a new input medium was created for the users of that wargame. The true effectiveness of this application of voice recognition technology must still be tested. Plans for such testing are being made and, to that extent, the thesis objectives are partly met.
# TABLE OF CONTENTS

I. INTRODUCTION ........................................... 9
   A. REVIEW OF VOICE RECOGNITION TECHNOLOGY ........... 9
      1. General ........................................... 9
      2. Continuous Voice Recognition ....................... 11
   B. VERBEX 3000 SPEECH APPLICATION DEVELOPMENT
      SYSTEM (SPADS) .................................... 13
   C. NAVAL WARFARE INTERACTIVE SIMULATION
      SYSTEM (NWISS) ..................................... 14
   D. PURPOSE ............................................. 16
      1. A Past Study of Voice Recognition
         Applied to Wargaming .............................. 16
      2. Thesis Objectives ................................ 16

II. SOFTWARE DESIGN .................................... 19
   A. NWISS REQUIREMENTS ................................. 19
      1. Command Syntax ................................... 19
      2. NWISS Data Requirements ............................ 25
   B. GRAMMAR DESIGN .................................... 26
      1. Overview .......................................... 26
      2. Verbex Grammar Design Tools ....................... 27
      3. NWISS Grammars .................................... 32
   C. PASCAL PROCEDURE DESIGN ......................... 43
      1. Verbex Predefined Functions and Types ........... 44
      2. Programming Techniques ......................... 45

III. USER'S GUIDE .................................... 47
   A. LEARNING OPERATION OF THE VERBEX 3000
      USER'S CONSole .................................. 47
   B. ENROLLING THE NWISS VOCABULARY .................... 49
   C. TRAINING THE GRAMMARS ............................. 50
   D. TESTING ............................................ 51
LIST OF TABLES

I. Permitted NWISS "FOR" Commands .......................... 20
II. Sea of Japan Scenario Blue Force-names .............. 23
III. Enrollment-Ordered NWISS Vocabulary ............. 49
LIST OF FIGURES

2.1 Examples of "FOR" Command Syntax ............ 21
2.2 Example of Complete Launch Command .......... 24
2.3 Graphics Display and Other Player Commands .. 25
2.4 NWISS Position Grammar .................. 29
2.5 NWISS Nwisgram1 Grammar .................. 34
2.6 NWISS Display Grammar .................... 37
2.7 NWISS Airorders Grammar ................... 38
2.8 NWISS Shiporder Grammar ................... 40
2.9 NWISS Launch Grammar ..................... 41
2.10 NWISS Fire Grammar ....................... 43
3.1 NWISS Test Commands ....................... 52
I. INTRODUCTION

As computer science has advanced into the era of human interactive design, one technology which has received increasing attention and has already demonstrated many practical results is that of speech recognition. It has the potential to vastly change the state of man-machine interaction by allowing humans to use their most natural communications output mode, speech, and thus freeing them from the constraints of the keyboard. This thesis is concerned with the application of speech recognition technology to military wargaming, specifically a computer-aided simulation of the naval warfare environment known as the Naval Warfare Interactive Simulation System (NWISS). Because computer-aided wargames entail a very high degree of human interaction, they are excellent candidates for application of voice recognition technology. The remainder of this chapter will cover what speech recognition technology is and can do, describe a specific implementation of this technology in a product named the Verbex 3000, introduce NWISS in more detail and close with a summary of the thesis objectives.

A. REVIEW OF VOICE RECOGNITION TECHNOLOGY

1. General

In the discussion of automatic speech recognition, the distinction between recognition and understanding is sometimes unclear. W.A. Lea [Ref. 1: p. 40] has defined voice recognition by machine "generally as the process of transforming the continuous human acoustic signal into discrete representations which may be assigned proper..."
meanings and which may be comprehended to affect responsive
behavior." For the purposes of this thesis, the process
will be understood as the conversion of human speech into
recognizable text, i.e. words and symbols. Due to idiosyn-
cracies of the human voice, as exhibited in such individual
variations as sex, race, geographic origin, age, emotion and
numerous other factors which impact the human acoustic
signal, this is by no means a simple task. Machine under-
standing of speech, on the other hand, is a closely related
activity which follows recognition and applies artificial
intelligence to invoke parsing rules and to make logical
inferences from the semantic content of the spoken (reco-
nized) message. This task is also very difficult as hu-
man know from daily experience (not so with recognition which
is generally taken for granted). While speech recognition and
understanding are not separate functions for the human (we
often use semantics to reconstruct/complement a sentence we
did not fully hear or listen to), they have tended as
computer technologies to develop in a parallel but partly
separated fashion. [Ref. 2]

It has long been recognized that speech is the
human's highest capacity, most natural output communications
channel. However it has only been during the past thirty
years that it has been possible to create machines which
begin to take advantage of this fact. In terms of human
input to computers, the keyboard has had superior speel,
error correction capability and overall versatility. How-
long the keyboard will retain this superiority is open to
debate. Already commercial word recognizers have been
effectively used for such jobs as package-sorting systems
and inspection and quality control, where keyboards or
numeric keypads served before. Military uses include
cartography, computer-assisted training of air traffic
controllers and aircraft cockpit communications.
[Ref. 3: p. 28]
These uses of speech recognition technology have benefited from several advantageous properties inherent in speech input to machines in addition to high speed and naturalness. Automatic speech recognition is unique in its ability to free the user's mind and eyes for such tasks as viewing graphics screens or other decision aids in a command post, overseeing an operations center, or just reading from a data source without having to remove the eyes to find and ensure the correct key is being struck. While skilled typists can read from a data source and input data at a rapid rate, such proficiency is not achieved without a good deal of training. By comparison training in the use of voice recognition equipment can be minimal. [Ref. 4]

A further advantage of automatic speech recognition is mobility. With a lightweight, wireless, microphone headset, a person is free to roam about and attend to other duties, such as an air traffic controller monitoring radar screens and speaking simultaneously to pilots and machine, either for transcript purposes or to control navigation and landing aids. Finally machine access can be controlled through spoken codeword authentication using voice recognition in combination with a speaker verification system. [Ref. 4]

2. **Continuous Voice Recognition**

Historically, almost all commercial applications of voice recognition technology have fallen in the category known as isolated (discrete) word recognizers. Typically this class of speech recognizers has demonstrated the ability to recognize limited vocabularies (up to 300 words) where the speaker is required to pause perceptibly between each word or utterance (string of words constrained to a specific time frame such as 1.5 or 2.0 seconds). The pauses provide boundaries for the machine processing of the voice.
message and allow the machine to "catch up" to the speaker. Discrete speech recognition detracts from the naturalness of human speech and imposes an artificial constraint on the speaker which requires training to adapt to such a speaking mode.

Sparked by the five year, $15 million research effort of the Department of Defense's Advanced Research Projects Agency in the mid-1970's known as Speech Understanding Research (SURT), the recognition of continuous speech was proven possible in the laboratory. As other advances in microcomputer and memory technologies came about, the first commercial continuous voice recognition products have come on the market in the past three years. While speaker-dependence and limited vocabulary (300 or so words) are still the rule, clearly naturalness is enhanced as well as input rate when continuous speech is used. "One can continuously speak at a rate of about 150 to 300 words or more per minute, but when words must be individually separated by pauses, the rate drops to less than 125 (usually to around 50 to 80) words per minute."

[Ref. 1: p. 66]

In general, continuous voice recognizers rely on the definition of grammars to limit the number of words from which the machine must choose at any instant based on previously recognized words. A grammar is a representation of the allowable word sequences in a state diagram composed of nodes representing a word or group of words and the possible transitions between the nodes. While it has been shown that finite state grammars cannot properly characterize major subsets of English sentences, unless sentence complexity is severely limited, they are quite appropriate for applications involving strictly-formatted sequences of words.

[Ref. 1: p. 52]
B. VERBEX 3000 SPEECH APPLICATION DEVELOPMENT SYSTEM (SPADS)

One product in particular which resulted from continuous voice work of the 1970's and which is the basis for this thesis is the Verbex 3000 voice terminal, marketed by a subsidiary of Exxon Corporation. The Verbex 3000 [Ref. 5: p. 8] "is a continuous speech, voice data entry terminal. It can operate as an input/output peripheral that adds voice entry capability to other computer systems or, in some applications, as a stand-alone data handling system. It is designed for use in industrial and commercial environments with either high or low noise levels, and allows operators to input data and commands in a naturally spoken stream of numbers, words, or phrases, without pausing." With its maximum number of four speech processing boards, the Verbex 3000 can recognize up to 360 different words spread over as many as 20 grammars. A finite limit to grammar size, based on total number of words and complexity of the node transition network, is necessary to allow the device to remain "real time" in terms of computation speed and memory (stored voice patterns) requirements. Thus the total application may involve up to 360 words, but at any instant the recognizer is dealing with only a subset (grammar).

The Speech Application Development System (SPADS) is a hardware/software adjunct to the Verbex 3000 voice terminal which allows the user to program the voice terminal to run a particular application. In other words, the design and definition of grammars, the control of transitions between grammars, the processing of text output, and the design of the terminal's visual and aural interface (feedback) with

---

1 This information was obtained at a SPADS training course attended by the author and conducted by Mr. Thomas DiGennaro, Verbex Senior Software Engineer, 7 - 9 November 1983. Future references to this source will simply be denoted "SPADS Training Course".

13
the operator are accomplished through SPADS. Verbex has designed SPADS so that Verbex 3000 users can write their own applications and update them as required, thus purchasing customer engineering services from Verbex. Currently in beta test status (as delivered to NPS), the SPADS is intended to be user friendly such that the building of applications and grammars is done through menu-driven editors. A procedure must be written in the Pascal programming language to control the application. Verbex supplies about 20 predefined functions to ease this process.

C. NAVAL WARFARE INTERACTIVE SIMULATION SYSTEM (NWISS)

As noted earlier, the candidate system for application of continuous voice recognition technology was the NWISS developed at the Battle Group Training Computer Support Facility within the Naval Ocean Systems Center at San Diego. Per the NWISS user's manual, NWISS "is a real-time, man-interactive, discrete event, time step computer-aided simulation of the naval warfare environment...for the purpose of supporting the training of senior naval officers in force-level tactical decision making and management in command and control."

In addition to ICSC, the NWISS is resident at the NPS Wargaming Analysis and Research laboratory where it is used primarily to introduce wargaming to students and to expose them to tactical, force-level decision making problems in command and control. The NWISS supports a two-sided (Blue vs. Orange) scenario in which opposing sides can define, structure, and dynamically control forces with the support of an umpire-like function called Control. Normally, at NPS, the force building and structuring phases are accomplished by the instructor and the students begin the wargame phase with a predefined scenario and force structure. (The
most often used (at NPS) set of scenarios is based on operations in the Sea of Japan. Since the author's NWISS experience was as a Blue player, this thesis uses the Blue force levels and names from the Sea of Japan scenario set.)

During the wargame phase the two sides must position and equip their forces and sensors to be ready for whatever the scenario entails, staying within the rules of engagement, and to engage the opposing forces in combat when appropriate. The NWISS can support various "views" of the action, representing the current tactical situation known to a user through the various sensors organic to his controlled force elements. Thus the Blue side may for example consist of several views representing different warfare commanders.

Typically a player has at his disposal an alphanumeric display capable of showing various information status boards, a color graphics display showing force positions (with Naval Tactical Data System symbology) and sensor information superimposed over a map of the area of operation, and an alphanumeric terminal for entering the player commands. Via keyboard, the player may enter strictly-formatted commands to change the graphics display characteristics, to equip and move forces, to control sensors, to engage enemy forces and, in general, to command and control the battle from his view. The alphanumeric terminal can also be used to send and receive messages between players (views) of the same size and/or Control.

The purpose of this brief description of NWISS is to provide a sense of the system's overall capability and, more importantly, to emphasize that, as currently configured at NPS (but not NOSC), all input to the NWISS from its players is via keyboard. Although a fair degree of user friendliness has been incorporated such that prompts and help for entering the next field of a command are readily provided and only enough characters need be typed to guarantee
uniquness, and errors are easily corrected, command entry can still be a laborious, error prone, time consuming task for many users. Precise entry of force names, weapon identifiers, latitude-longitude, bearings, ranges, altitudes, etcetera is required and errors are not always easily forgiven by NWISS.

D. PURPOSE

1. A Past Study of Voice Recognition Applied to Wargaming

The predecessor to NWISS at NPS was the Warfare Environmental Simulator (WES), also developed at NOSC, with many of the same capabilities as NWISS but to a lesser degree (particularly in system response time). In 1981, W. J. McSorley published his master's thesis at NPS on the subject of using voice recognition technology to run WES. Using a discrete voice recognizer (Threshold 600), he compiled a set of typical WES commands and conducted an experiment with 12 subjects of varied typing abilities and voice (microphone) experience to determine which input medium was superior. Based on lengthy statistical analysis of speed and error results, McSorley [Ref. 6: p.65] concluded that "the subjects were able to input WES commands faster and with fewer total errors using the manual typing mode than with voice mode." Since McSorley's subjects were using discrete voice and had an average typing ability better than 35 words per minute, the results are not too surprising.

2. Thesis Objectives

One reason for summarizing McSorley's work is to show that interest in applying voice recognition technology to computer-aided wargames is not new. Such wargames are
highly interactive and hence invite attempts to improve the interaction (and the participants are usually willing subjects in new or experimental undertakings). Given the inherent advantages to voice input stated earlier, it seems only natural to want to apply speech recognition technology to a wargame such as NWISS.

The more cogent reason for reporting McSorley's results is to establish the grounds for this thesis: to make use of the progression from discrete to continuous voice recognition technology and build a voice input medium to NWISS which has the potential to compete effectively with the keyboard both in speed and error results. With such an input medium, NWISS players will be able to spend their time more profitably in monitoring the graphics display and status boards and in commanding and controlling their forces with more natural voice commands as opposed to being tied to a keyboard. However this application of continuous voice recognition technology is not intended to, nor can it, completely replace the player's alphanumeric keyboard. Rather it is intended to substantially improve the man-machine interface for NWISS and allow the player to perform all but a very small part of his input with voice vice the keyboard.

Given the commercial state of the art (as represented by the Verber 3000), the challenge is to thoroughly scrutinize the subject application (NWISS) and design grammars and a grammar transition network in software such that grammar boundaries (which tend to be "discrete") occur in places where natural pauses occur or where the user can be easily induced to pause with minimal disruption to speech patterns.

Thus the purpose of this thesis is to show that continuous voice recognition technology can be effectively applied to a computer-aided wargame through demonstration of
the software design process (Chapter 2) and the user's operating instructions (Chapter 3). To be precise, the NWISS Blue player commands, force names, weapon identifiers, et cetera for the Sea of Japan scenario (totalling about 150 words or symbols) have been placed into 10 different (but overlapping) grammars in an application which uses about 800 lines of Pascal to control the network flow between grammars and restructure textual output to NWISS requirements. The application is designed to be user friendly and require a minimum of player involvement with the mechanics of the process.
II. SOFTWARE DESIGN

As noted in Chapter 1, the application of continuous voice recognition technology requires careful study of the man-machine interactive process being substituted for or supported. In the case of NWISS, such questions as the following must be answered:

- Where will natural input pauses occur?
- What feedback or prompting will the user require?
- What means must be provided for error control?
- How must the data be structured for the host computer and what special data characters, if any, are needed?
- How will the overall process be controlled, particularly in the time domain?

The answers to these and other related questions lead to the initial design of the grammars and the application control program. This chapter will define the NWISS requirements in terms of input command syntax and data structures, describe the grammars which resulted from the design process, and provide some insights to the structure of the application program code.

A. NWISS REQUIREMENTS

1. Command Syntax

There are approximately 37 commands which the NWISS player can issue to forces under his control for maneuvering and launching platforms, controlling sensors, and engagement [Ref. 7]. In this paper these are referred to as "FCS"
commands of which about 23 are used with any frequency at NPS and have been included in the continuous voice recognition application (see Table I). There are approximately another 20 commands an NWISS player can use to control the graphics display characteristics, obtain bearings and accomplish other actions [Ref. 7]. Again these 20 commands are not all used at NPS and so the 10 commands used most often have been included in the voice application and are addressed later. The primary reasons for excluding unused commands are grammar and application code efficiency and reduced user training time; these design factors and others will be addressed throughout this chapter. What follows now is a description of the general syntax and some examples of these commands.

### TABLE I

**Permitted NWISS "FOR" Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTITUDE</td>
<td>DECM</td>
</tr>
<tr>
<td>DECM</td>
<td>MISSION</td>
</tr>
<tr>
<td>MISSION</td>
<td>SPEED</td>
</tr>
<tr>
<td>BARRIER</td>
<td>DEPTH</td>
</tr>
<tr>
<td>DEPTH</td>
<td>ORDERS</td>
</tr>
<tr>
<td>ORDERS</td>
<td>STATION</td>
</tr>
<tr>
<td>BINGO</td>
<td>EMCON</td>
</tr>
<tr>
<td>EMCON</td>
<td>PERISCOPE</td>
</tr>
<tr>
<td>PERISCOPE</td>
<td>SURFACE</td>
</tr>
<tr>
<td>BLIP</td>
<td>FIRE</td>
</tr>
<tr>
<td>FIRE</td>
<td>PROCEED</td>
</tr>
<tr>
<td>PROCEED</td>
<td>TAKE</td>
</tr>
<tr>
<td>COURSE</td>
<td>LAUNCH</td>
</tr>
<tr>
<td>LAUNCH</td>
<td>PBOC</td>
</tr>
<tr>
<td>PBOC</td>
<td>WEAPONS</td>
</tr>
<tr>
<td>COVER</td>
<td>LOAD</td>
</tr>
<tr>
<td>LOAD</td>
<td>REFUEL</td>
</tr>
</tbody>
</table>

#### a. "FOR" Commands

The primary NWISS command syntax is quite simple and consists of the following standard format:

```
FOR <addressee> <command> [TIME <start-minute>] <CR>
```
where the following conventions apply:

1. Capitalized words are command keywords which may be entered in abbreviated form (enough letters to insure uniqueness).
2. Lowercase words inside parentheses are prompts from NWISS received when the space or escape character is struck after the preceding field.
3. Lowercase words inside arrows are command arguments which must be specified precisely.
4. Keywords and arguments inside brackets are optional.
5. "FOR <addressee>" is not required for subsequent commands directed to the same addressee.

Some specific command formats and examples are shown in

```plaintext
FOR <addressee> ALTITUDE <feet>  
"FOR VA024 ALTITUDE 4000"

FOR <addressee> COVER <track *>  
"FOR MP604 COVER ES002"

FOR <addressee> PROCEED POSITION <latitude> <longitude>  
"FOR KNOX PROCEED POSITION 36-30N 134-55E"

FOR <addressee> FIRE <number> <name> TORPEDO (at) <track *>  
"FOR CMAHA FIRE 2 MK48 TORPEDO (at) ES006"
```

Figure 2.1 Examples of "FOR" Command Syntax.

Figure 2.1 where <CR>, carriage-return, is assumed and "TIME" is not used. As convention #4 implies, the keyword
"TIME" and its following argument are optional: most player commands are entered without specifying a desired game time so that execution by NWISS occurs in real time.

The "addressee" and "force-name" fields have the same domain, i.e. all legitimate force-names are specified in the NWISS force-building and scenario definition phases. For the Sea of Japan scenarios, the addressable Blue forces number 9 ships, 1 shorebase and potentially 100 aircraft (aircraft do not have callsigns assigned and hence are not addressable until launched). Aircraft callsigns consist of 2 alphabetic characters followed by 3 digits whereas shipnames and shorebases may be abbreviated to the first 5 characters, e.g. RATHOrn. In addition, task force designators may be used to address collective segments of the Blue forces or individual units (not aircraft). Thus "FOR 1.1" catches the entire Blue task group including all aircraft while "FOR 1.1.0.0" catches the Kittyhawk but not her aircraft. Other than "1.1", these designators are seldom if ever used and are in a sense redundant. For that reason plus a technical problem with defining periods as part of object names in the Verbex 3000, only "1.1" is permitted in this application. See Table II for all allowable force-names.

b. "LAUNCH" Command

The longest and most difficult "FOR" command to learn how to enter properly to NWISS is that for launching aircraft. Ability to correctly launch aircraft is of course indispensable to game play. To avoid too much complication and be in conformity with how the command is most often used, its simplest syntax will be discussed and is shown below:
TABLE II

Sea of Japan Scenario Blue Force-names

<table>
<thead>
<tr>
<th>Ships</th>
<th>Aircraft</th>
<th>Shorebase</th>
</tr>
</thead>
<tbody>
<tr>
<td>KITTYhawk</td>
<td>MP600-608</td>
<td>MISAWa</td>
</tr>
<tr>
<td>KNOCX</td>
<td>SH100-103</td>
<td></td>
</tr>
<tr>
<td>LOSAngeles</td>
<td>WA000-033</td>
<td>Task Group</td>
</tr>
<tr>
<td>MCCOnick</td>
<td>VE000-003</td>
<td>1.1</td>
</tr>
<tr>
<td>OMAHA</td>
<td>VE100</td>
<td></td>
</tr>
<tr>
<td>RATHbourne</td>
<td>VF000-019</td>
<td></td>
</tr>
<tr>
<td>SPRUAnce</td>
<td>VH000-005</td>
<td></td>
</tr>
<tr>
<td>WICHita</td>
<td>VK000-003</td>
<td></td>
</tr>
<tr>
<td>WILSON</td>
<td>VS000-009</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VT000-003</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VW000-003</td>
<td></td>
</tr>
</tbody>
</table>

FOR <addressee> LAUNCH <number> <aircraft-type> (event name) <name> (course) <degrees> (speed) <knots> (altitude) <feet>

Upon accepting this "first-level" command, NWISS responds with the prompt "FLT PLAN:" on a new line. In theory the user may now specify any of about 25 commands applicable to aircraft. However the normal NPS practice is to provide a "MISSION" for the aircraft, "LOAD" the aircraft with expendables, possibly specify a "PROCEED POSITION", and signify the completion of the launch command with "STOP". Except for "STOP", any of the other commands is followed by the prompt "FLT PLAN:" on a new line. Figure 2.2 shows a complete launch command.
"FOR KITTY LAUNCH F14A (event name) F14A1 (course) 210 (speed) 650 (altitude) 2000"

FLT PLAN: "MISSION CAP"

FLT PLAN: "PROCEED POSITION 36-30N 137-45E"

FLT PLAN: "LOAD (equipment) 2 PHENX 2 SPAR 2 SWDR"

FLT PLAN: "STOP"

Figure 2.2 Example of Complete Launch Command.

c. Graphics Display and Other Commands

In addition to the "FOR" commands, there is a large repertoire of commands for controlling the characteristics of the NWISS graphics output. In general, the geographical area of operation can be centered about any force-name, track, or position specified and its radius can be made as small or as large as desired. An xmark, circle or grid (set of concentric circles plus 12 lines of bearing spread 30 degrees apart) can be placed over any force, track or position. NWISS generated lines of bearing (LOB) for passive sonar or ESM (electronic sensor) may be erased or turned back on. Finally there are other non-graphics commands for executing a preplanned launch (the Sea of Japan scenario provides five "canned" Blue launch plans which allow the player to get many aircraft up at once), or obtaining bearing and range information, or overriding the NWISS generated NIDS assignments for friendly, neutral or enemy platforms. Figure 2.3 shows examples of some of these commands' syntax together with an example.
PLACE (a) CIRCLE (around) FORCE <force-name> GRID TRACK <track #> POSITION <lat> <long>
(radius) <nautical-miles>
"PLACE (a) CIRCLE (around) FORCE MP604 (radius) 60"

CENTER (plot at) FORCE <force-name> TRACK <track #> POSITION <latitude> <longitude>
"CENTER (plot at) FORCE KITTY"

<CTRL-F> <plan-name>
"<CTRL-F> F14STRCAP.PRE"

BEARING (and range from) FORCE <force-name> TRACK <track #> POSITION <lat> <long>
(to) FORCE <force-name> TRACK <track #> POSITION <lat> <long>
"BEARING (and range from) FORCE KNOX (to) TRACK BS004"

DESIGNATE (as) FRIENDLY <track #> NEUTRAL ENEMY
"DESIGNATE (as) ENEMY BU007"

Figure 2.3 Graphics Display and Other Player Commands.

2. *NWISS* Data Requirements

As shown in Figure 2.3, execution of preplanned launches is accomplished with a "control-f" followed by the plan name. Of greater importance is the "control-k" character with which the NWISS player may cancel any command prior to its complete entry and acceptance by NWISS. There
are word and character erase functions also in NWISS but they are not pertinent to a continuous voice application which outputs buffered strings of data vice keyboard character by character output. This means that the output from the voice recognizer must be at least syntactically correct and thus in conformity with the syntax examples shown above. In general output strings must have spatial separation of command keywords and arguments and spaces must be kept out of digit strings which are meant to be contiguous since NWISS can interpret intervening spaces as completion of the field (e.g. launching an F14 at 5 vice 5000 feet). Another requirement is to signify completion of command entry by a \textless CR\textgreater (carriage return).

B. GRAMMAR DESIGN

1. Overview

As noted earlier, ten grammars have been defined for the NWISS continuous speech application. While the strictly-formatted structure of the NWISS commands combined with the Verber upper limit on grammar size helps to determine the overall grammar design, there are numerous factors to consider in building the software (both grammars and Pascal procedure) for any application. According to Verber [Ref. 8: p.60], the grammar design goals are:

- to improve recognition accuracy
- to allow for feedback
- to allow continuity of speech
- to allow natural pauses
- to reduce response time
- to allow error correction
As with most sets of goals, there are some incompatibilities among them and tradeoff decisions must be made. This chapter section will briefly describe the Verbex tools used to define grammars, describe the major grammars built for NWISS in detail, and, wherever appropriate, discuss grammar design goals and tradeoffs vis-a-vis the NWISS application.

2. Verbex Grammar Design Tools

a. Verbex Standard Notation (VSN)

In a takeoff on Backus-Naur Form (BNF), Verbex has created a very logical and understandable means for defining grammars which must first be described in order to define the NWISS grammars. The basic element of VSN is the object, which may be either simple or complex. A simple object is a word that the user actually speaks. A compound object is a category, or group of objects, and is so denoted by placing a period in front of it. Note that a compound object can represent a group of compound objects as well as simple objects. Within a compound object definition, alternative objects are arranged vertically and consecutive objects are arranged horizontally. Thus to define a compound object which is used in every NWISS grammar, we write

```
digit ::= 0
```

to represent the fact that any one of the ten digits may be spoken wherever .digit appears in a higher level compound object or grammar definition. Thus
.course ::= .digit .digit .digit
could suffice as the VSN definition of a compound object used frequently in NWISS. It may be noted that sometimes a course may be specified with only 1 or 2 digits. VSN defines optional objects with brackets. Hence a better definition of course might be

.course ::= .digit [.digit] [.digit]
to allow such variation. However this definition of "course" implies a time constraint: the Verbex 3000 will consider the above string to be complete after the first digit plus a half-second of silence. Thus the user needs to know this so that, if a course is to be specified with 3 digits, it is spoken as a continuous string without pauses. The tradeoff here is whether users must enter "course" as 3 digits always (in which case the Verbex 3000 waits until its timeout threshold, approximately one minute, to receive all 3 digits) or whether they can enter "course" as 1, 2, or 3 digits with no intermittent pauses.

Finally VSN allows an unspecified amount of repetition of the same object through use of the "+". In NWISS, aircraft altitudes can vary by several digits in length. Thus the following definition

.altitude ::= .digit+
accomplishes the same task as several consecutive, bracketed, identical objects. Again, however, there is the same time constraint imposed on the user as above with brackets: no intermittent pauses. On the other hand, a user could speak digits continuously up to the Verbex 3000 buffer limit of 258 digits² with the above definition.

------------------
²SPADS Training Course
According to Verbex [Ref. 8: p. 64], "a grammar is complete when every compound object included in its definition has been completely defined in terms of simple objects. There is no limit to the number of levels this definition can take, nor on the scope of complexity of any level." To see how this definition applies, the VSN definition of the NWISS grammar "position" is shown in Figure 2.4. This grammar is confusing at first glance since it seems to provide for both latitude and longitude but without specifying which. That is precisely the job of the application code: it works in cohesion with the grammar design and calls Position twice for recognition, the first time with the prompt "LATITUDE" and the second time with the prompt "LONGITUDE". Here grammar efficiency is achieved because the application code takes advantage of the natural pause which occurs between latitude and longitude expressions.

```
Position ::= [1] .digit .digit [ .minutes ] . direction
          CONTROL_K

  .minutes ::= - .6digit .digit

  .direction ::= \n
        .digit ::= 0 1 2 3 4
        \n
  .6digit ::= 0 1 2 3 4 5 6 7 8
```

Figure 2.4 NWISS Position Grammar.
With that understanding, we may note that Position has an optional complex object, .minutes, which the user is required to begin with the sentinel "dash". Since NWISS requires the dash from the keyboard as well, this does not seem burdensome. Less complexity and greater accuracy are achieved with the object .digit defined as part of .minutes.

Note also that CONTROL_K is an alternative in this grammar as well as all other NWISS grammars: the user may cancel the NWISS command at any point. More rationale for the CONTROL_K will appear in the discussion of the grammar NWisgram1. Further, the Verbex will output N, S, E, or W in accordance with the voice signals which the user trains for those symbols. In other words, the user may train and speak these as "North", "South", "East" and "West". Finally, note that the grammar will not prevent incorrect expressions: one could easily say "120N" for latitude or "95S" for longitude. The primary purpose of grammar definition is to define what can be said legally. Preventing illegal expressions is a "side" benefit which, if pursued too far, can cost too much in complexity, a subject of the next section.

1. Verbex Grammar Editor

The Verbex 3000 SPADS has a menu-driven facility, called GRID, for creating grammars which is basically user-friendly. With GRID, the designer inputs potential grammars for an application in VSN form, pushes the SPADS "application generate" function key, and sits back. The result, if the grammars are not too complex, is a complete application less the Pascal control code. In other words, SPADS automatically interprets the GRID-built VSN file and builds recognition instructions for the Verbex 3000 telling it when and where to look for acoustic input
including long and short silences for the grammars in that application. It also creates all the files necessary for
user voice training and testing of the grammars.

Regarding complexity, SPADS generates a report
for each grammar which states the number of individual words
in that grammar, the percentage of machine vocabulary
capacity that total represents, and, most importantly, a
determination of the grammar's complexity expressed as a
percentage of machine capacity. Complexity is based on both
number of words and number of node transitions in the
grammar. As this complexity percentage nears 100, the
Verbex 3000 ability to remain "real time" is reduced
[Ref. 5: pp. 72-75]. However SPADS generally (in the
author's experience) will not generate a grammar with
complexity higher than 90% for the maximum capacity model
3000 (four speech processing boards). The formula used by
Verbex to compute complexity is too cumbersome to describe
here and is fully explained in [Ref. 8]. However there are
three important factors in the computation:

1. Total number of distinct simple objects or words in
   the grammar;
2. Total number of words that may occur as the first
   word of a legal path through the grammar;
3. The average length of all possible paths through the
   grammar.

To provide some yardstick measure of complexity for compar-
ison with other grammars, the report for the grammar
Position is:

   Total vocabulary is 16 words
   Vocabulary is 5% of capacity
   Complexity is 41% of capacity.

The average path length is the major factor in the
complexity of the Position grammar. Most of the NWISS gram-
mars fall in the 40 to 60% range for complexity.
3. **NWISS Grammars**

a. **Nwisgram1**

The first grammar from which the Verbex 3000 attempts to recognize NWISS commands is Nwisgram1. This grammar is called in for recognition by the Pascal application code as soon as the previous command has been output to NWISS. Its purpose is to allow the user to begin any legal NWISS command and then, based on the recognition result, allow the application code to call in the next appropriate grammar. The number of ways that one can begin a command are numerous:

```
FOR <addressee>
Graphics command
Other non-graphics command
"FOR" command without "FOR <addressee>"
(when directed to same addressee as before)
```

Based on trial and error experience, these requirements are too large to be handled by any single grammar on the Verbex 3000. Using two or more grammars will not solve the problem since at any instant only one can be recognized and there is no prior knowledge as to which is needed. The only possible solution is a compromise with the requirements:

1. **Eliminate the possibility of beginning a "FOR" command without the "FOR <addressee>"**. This seems only a slight inconvenience to the user.

2. **Create a sentinel for graphics commands so that a separate grammar can be called in upon recognition of the sentinel.** This was done using the sentinel word "display". Hence the user must say "display" and pause for a half-second before entering any graphics commands. This is an inconvenience but in practice the author had little difficulty in adapting to it.
3. A third possibility is to divorce "FOR" from the addressee so that the user is required to pause after "FOR" while the application loads a grammar containing all the Blue force-names. This was tried and proved difficult as it violates the rule of placing grammar boundaries where natural pauses occur. "FOR KITTY" is much more natural than "FOR", pause, "KITTY".

Another design question was how to handle the "control_f" for executing predefined Blue launch plans. Should the user say "control_f" or some more meaningful word such as "execute"? The latter alternative was chosen as being easier to associate with plans and not being confused with the other control character, "control_k", used for cancelling commands in mid-stream. This character is quite prominent in use in NWISS, has a distinctive acoustic pattern, and is shorter than some phrase such as "cancel command". Hence "control_k" is used as it looks. In both cases, the application code converts the recognized string to the proper ASCII output character for NWISS.

Similarly, the application code can make the user's task easier by not requiring "pre" to be said after each plan name. Thus the plans are specified as A6STRIKE, SCAP1, et cetera and the application code takes the recognized plan name and attaches ".PRE" to it as required for NWISS.

The requirements for the NWissgram1 grammar have been refined well enough that the grammar can be specified as in Figure 2.5. As noted earlier, objects such as "1.1" cannot be entered into GRID. The period is illegal except as the first character of a complex object. Hence the application code must convert "!point1" to "1.1" for output to NWISS. Another application code conversion occurs with .aircraft because the Vertex 3000 outputs a space between
N wisgram1 ::= FOR .force
    DISPLAY
    EXECUTE .preplan
    DESIGNATE .status
    CONTROL_K
    BEARING TRACK
    BEARING FORCE
    FORCE
    TRACK
    POSITION

    .status ::= FRIENDLY
               NEUTRAL
               ENEMY

    .preplan ::= A6STRIKE
                SCAP1
                SCAP2
                F14STRCAP

    .force ::= 1point1
               .aircraft
               .ship/base

    .aircraft ::= MP8 0 .digit
                 SH1 0 .digit
                 VFO .digit .digit
                 V20 0 .digit
                 V51 0 .digit
                 VFO .digit .digit
                 VBO 0 .digit
                 VFO .digit .digit
                 VBO 0 .digit
                 VFO .digit .digit
                 VFO 0 .digit

    .ship/base ::= KITY
                  KNCX
                  LCSAN
                  NCCOR
                  NISA4
                  OTARA
                  RACME
                  SFU1
                  WICHI
                  WILSO

Figure 2.5  NWISS Nwisgram1 Grammar.

each object it recognizes. Hence "M2604" is placed in a
buffer as "M2604" where the program then removes the
offending spaces prior to output to NWISS. Too many conver-
sions such as "1.1",",PRE" and aircraft callsigns add to
the length and complexity of the application code and poten-
tially can slow the real time capability of the Vertex 3000.
Hence the effort should be made to place objects in grammars exactly as they are to be output where possible. The SFADS report for NWISgram1 is:

- Total vocabulary is 49 words
- Vocabulary is 14% of capacity
- Complexity is 48% of capacity.

Here, by comparison, the complexity is somewhat higher than that (41%) for Position, due to the driving factor of total number of words.

e. Force, Track, and Position

The last three objects of NWISgram1's top level definition are not just objects but also the names of three individual grammars. Their appearance in NWISgram1 is necessitated by the syntax of the "BEARING" command as shown in Figure 2.3. Because FORCE, TRACK and POSITION are used often as command keywords and the size of their respective argument domains precludes lumping them together or inside some other grammar and the argument can be made that there is a natural pause after these keywords but before specification of their arguments, they have been specified as individual grammars. Since Position has already been examined (see Figure 2.4), the Force and Track grammars will be described.

Force ::= 1point1
   .aircraft .orngbases ::= FCNSA
   .ship/base ALEKS
   .orngbases PETRO
   CONTROL K VLAD

The Force grammar is quite similar to the complex object .force in NWISgram1. The only difference is the addition of .orngbases because NWISS allows the Blue player to request bearings on Orange bases by name as well as position for
convenience. Hence the redundancy between *Nwisgram* and Force is considerable but necessary: one grammar can not be devised to do the job of both. This grammar overlap has a cost in terms of storage space and user training time but is unavoidable. The SPADS report for Force is:

- Total vocabulary is 37 words
- Vocabulary is 11% of capacity
- Complexity is 58% of capacity.

The number of initial path words (27) is the major factor in this complexity figure.

The Track grammar is relatively small and simple but is called quite often:

```
Track ::= EAO .digit .digit
      | EEO .digit .digit
      | BPO .digit .digit
      | ESO .digit .digit
      | BUO .digit .digit
      | CONTROL_K
```

The SPADS report for Track is:

- Total vocabulary is 15 words
- Vocabulary is 5% of capacity
- Complexity is 24% of capacity.

To see how the application code comes into play, consider the `BEARING` command: when the user says "BEARING FORCE", for example, a complete path through *Nwisgram* exists and the result is placed in a buffer. The application code analyzes the buffer contents and determines that the buffer contents should be sent unchanged to NISS with a space after "FORCE", then calls in grammar Force for recognition, checks the contents again for necessary conversions, outputs the converted string with a trailing space, returns to *Nwisgram* to see what the next keyword in the command will be (a choice of FORCE, TRACK, or POSITION), outputs
this keyword followed by a space, calls in the keyword-specified grammar, performs any necessary conversions and outputs the string followed by a <CR>.

c. Display

As previously discussed, NWISgram1 contains the sentinel "DISPLAY" to allow the application control software to call in the grammar with that name shown in Figure 2.6.

```
Display ::= RADIUS .digit+
         DROP TRACK
         CENTER FORCE
         CENTER POSITION
         PLACE .what .where
         CANCEL .what .where
         PLOT LOB ESM
         PLOT LOB SONAR
         ERASE LOB ESM
         ERASE LOB SONAR
         CONTROL_K

   .what ::= CIRCLE
           GRID
           XMARK

   .where ::= FORCE
             TRACK
             POSITION
             ALL
```

Figure 2.6 NWISS Display Grammar.

to call in the grammar with that name shown in Figure 2.6. The SPALIS report on Display is:

- Total Vocabulary is 28 words
- Vocabulary is 37% of capacity
- Complexity is 49% of capacity.

37
Airorders ::= ALTITUDE .digit+  
BARRIER  
BINGO  
CONTROL_K  
COURSE .digit+  
COVER  
EMCON .plan  
FIRE  
MISSION .mission  
ORDERS  
PROCEED POSITION  
REFUEL VK00 .digit  
SPEED .digit+  
TAKE  
WEAPONS FREE .how  
WEAPONS TIGHT  

.mission ::= ASK  
AIR  
AIRTANKER  
AV  
CAP  
SEARCH  
SEARCHCAP  
STRIKE  
SURFACE  

.plan ::= SEN  
AIRS  
STLEN  
SONAE  
SURF  

.digit ::= 0  
1  
2  
3  

.how ::= AIR  
ALL  
ENEMY .enemy  
SUBMARINE  
SURFACE  

.enemy ::= AIR  
ALL  
SUBMARINE  
SURFACE  

Figure 2.7  NDISS Airorders Grammar.
d. Airorders and Shiporder

These two grammars are the largest and most complex developed for NWISS. The difference of singular vs. plural in the names is due to GRID not accepting grammar names longer than nine characters. As the names indicate, there is sufficient difference between the types of orders that aircraft and ships receive to justify individual grammars for reasons of reduced complexity and increased accuracy. One of these two grammars is called from NWisgram1 every time "FOR <addressee>" is recognized (both 1.1 and MISAW call Shipordier). Figure 2.7 contains Airorders. Here is the only instance of command arguments being excluded from a grammar because of lack of use at NPS and need for efficiency: there are actually 14 possible aircraft missions in NWISS but only 8 have been included in Airorders.

Figure 2.8 contains Shiporder. Comparison with Airorders shows that the two grammars have nine commands in common. SPEED is defined differently in Shiporder because ship speeds can be narrowly defined. If ships ever go faster than 39 knots, the grammar must be changed. The SPADS report for Shiporder is:

- Total Vocabulary is 42 words
- Vocabulary is 12% of capacity
- Complexity is 53% of capacity.

e. Launch

Because of its difficult and lengthy syntax (see Figure 2.2), and a large vocabulary requirement due to the large number of aircraft types which can be launched and the large number of expendables which can be loaded on the aircraft, the LAUNCH command merits its own grammar. The launch grammar can only be called from Shiporder when that
Shiporder ::= BLIP .on/off
CONTROL_K
COURSE .digit+
DECM .on/off
DEPTH .digit+
EMCON .plan
FIRE
LAUNCH
ORDERS
PERISCOPE
PROCEED POSITION
R3OC .on/off
SPEED .digit
SPEED .3digit .digit
STATION
SURFACE
TAKE
WEAPONS FREE .now
WEAPONS TIGHT

.on/off ::= ON
OFF

.plan ::= AW,
AIR,
AIRS
PADI.
PADD
SILEN
SIGNAR
SURF

how := AIR
ALL
ENEMY .enemy
SURFACE
SUBMARINE

.enemy ::= AIR
ALL
SUBMARINE
SURFACE

Figure 2.8  NWISS Shiporder Grammar.

command is stated by the user. As noted earlier, only a
simplified version of the command syntax is supported in
this application and hence the grammar is smaller than might
otherwise be required. For instance, the <event-name> argument can only be specified as the repeated aircraft type plus a single digit. The choice of "FLT PLAN" commands is limited to MISSION, LOAD, PROCEED POSITION, and STOP. Here is an instance where the application program is used to save grammar storage space and complexity: the MISSION command and its eight arguments are not part of Launch. Instead the application code calls in Airorders for recognition at that point, provides a display message to the user (feedback and guidance to users is described more fully in the next chapter) asking for the mission, outputs accordingly and returns to the Launch grammar for the next recognition. The same is true of the course, speed and altitude arguments for
LAUNCH wherein a "utility" grammar, Digits (exactly what its name implies), is called in three times with a different display prompt each time.

The implication above is that grammar design and application program design are intertwined so that consideration cannot be given one without the other. A further implication is that the user is forced to follow the application program sequence of directions in entering the launch command. From one viewpoint this simplified syntax and controlled sequence of input eases the user's job. From a different point of view, it costs the user in terms of the flexibility afforded by the keyboard. (Of course there is always the option to increase grammar and program complexity to provide such flexibility.) The SPADS report for Launch is:

- Total vocabulary is 41 words
- Vocabulary is 12% of capacity
- Complexity is 40% of capacity

f. Fire

The FIRE (TORPEDO) command syntax appears in Figure 2.1. In addition to torpedoes, NWISS players can FIRE cruise missiles using the following syntax:

```
FIRE <number> <name> CRUISE (missile)
AT <shore-base>
BEARING <degrees> RANGE <nautical-miles>
```

"FIRE SPRUA FIRE 3 HRPON CRUISE (missile) AT ALEKS"

The Fire grammar can be called from either Airorders or Shiporder and appears in Figure 2.10. Although the FIRE command is much less complex than LAUNCH, it still has a large enough vocabulary requirement to merit a separate grammar. As with LAUNCH, the FIRE (CRUISE) command is
broken into parts. However the application code does not branch to other grammars but simply recalls Fire until the command is complete. The SPADS report for Fire is:

Total vocabulary is 26 words
Vocabulary is 8% of capacity
Complexity is 42% of capacity.

Fire ::= .digit .cruistype CRUISE
      .digit .torptype TORPEDO
      AT .base
      BEARING .digit+
      RANGE .digit+
      CONTROL_K

    .cruistype ::= TCMHK
                 HAP0N
                 .torptype ::= ASROC
                               MK46
                               MK46A
                 .base ::= ALEKS
                           PETEO
                           VLAD
                           WCNSA

Figure 2.10 NWISS Fire Grammar.

C. PASCAL PROCEDURE DESIGN

Numerous references to the "application code" in preceding sections have already indicated what its purposes are:

- to provide for the control of the interactive process between the user and the Verbex 3000
- to control data output to the host process, NWISS.
Hence calling the correct grammars in sequential order is not enough: feedback must be provided the user so that he knows the machine's status at all times. In part this is accomplished automatically by indicator lights on the Verbex 3000 User's Console. In part it is accomplished by the system response of the host process to which the user is inputting commands. Finally, with regard to the subject at hand, it is also accomplished in part by the visual and aural messages which the application program generates to the user through the User's Console. (No aural feedback is used in the NWISS application).

Appendix A of this thesis contains the approximately 600 lines of Pascal code used to control the NWISS continuous voice application. This chapter section will describe the Verbex predefined functions which appear repetitively throughout Appendix A and predefined types and explain the reasons underlying the programming techniques used.

1. **Verbex Predefined Functions and Types**

   It was noted in Chapter 1 that Verbex has created a library of about 20 predefined functions to ease the programmer's task. However the SPADS is still in beta test status and many of these functions do not work yet, though they are defined. Of those that work, only three are used in the NWISS application and are defined below:

1. **Recognize (grammarname, buffername)** is the workhorse function. It tells the Verbex 3000 to begin listening for acoustic signals matching the named grammar and to place the output result in the named buffer (of type "string"). All Verbex functions return a value of type "short" to indicate success or failure or some other appropriate result. Recognize can return four values: 1) **Voicein** means success
2) **recognize_bad** means failure 3) **timeout** means no input which could be classified as recognizable or unrecognizable was received for approximately one minute 4) **keypadin** means the recognition cycle was interrupted by the user from the User's Console keypad.

2. **Hostwrite** *(buffername)* tells the Verdex 3000 to write the contents of the named buffer *(string)* to the host computer *(WISS in this case)* and simply returns success or failure.

3. **DisplayMessageClear** *(display_primary, message)* tells the Verdex 3000 to write the message *(of type string)* to the 32 character display on the User's Console and simply returns success or failure.

Some other functions were investigated but found not to work. This was unfortunate as it made the programming task for WISS a rather tedious one in terms of comparing and manipulating character strings character by character.* Wordcount*(string)*, *Wordfind*(string)*, *Stringcopy*(string1, string2)*, and *Stringcompare*(string1, string2)* are fairly descriptive names of functions which are defined in [Ref. 5] and are expected to work with the next SPADS software release. Thus the Appendix A software will need a fair amount of revamping in order to take advantage of these new functions when they are available.

2. **Programming Techniques**

The job of the voice application programmer is to write a Pascal procedure with the name "application". Only that name will suffice as the procedure is included by the SPADS compiler into the standard operating software for the Verdex 3000 where it is called at the appropriate time. The
NWISS application procedure in Appendix A uses a high number of labels (35) and a proportionate number of GOTO statements. This is due in part to the fact that it was not possible to define functions of the predefined type "string" such as will be available from Verbex, and in part to the fact that the GOTO statement is efficient and saves one from indenting off the right side of the page in a highly nested environment which easily results when jumping from grammar to grammar. The program is 25,000 bytes long and hence close to the Verbex 3000 upper limit of 30,000 bytes. For this reason and to allow room for growth, the comments have been kept to a minimum but are intended to be adequate for the purpose of future updates and maintenance.
III. USER'S GUIDE

To use the NWISS continuous voice application properly, one must invest approximately three hours both in learning the operation of the Verbex 3000 voice terminal and in training voice patterns to the grammars. This investment may be two or three times more than that required to become proficient at inputting NWISS commands from the keyboard (given that one already has a fair amount of keyboard experience). Nevertheless, it is the author's opinion that the investment in voice input will more than pay for itself in time saved and reduced aggravation when several lengthy NWISS sessions are to be played. The reasons for this opinion have already been stated in several places in this thesis and stem from the inherent advantages of voice input: naturalness, speed, hands-free and eyes-free (relative to a keyboard) input. Further it is difficult to output mistakes in the sense that the NWISS grammars only have "correct" objects to be output and substitution errors are exceedingly rare if a person has taken the necessary time to train voice patterns properly. This chapter will move sequentially through the steps which a prospective user of the NWISS continuous voice application should follow in becoming proficient.

A. LEARNING OPERATION OF THE VERBEX 3000 USER'S CONSOLE

Verbex has published a very readable, illustrated operating manual which is called the Supervisor's Manual [Ref. 9]. This manual should be skimmed and referred to during the user's first login to the voice terminal. The only amplifying instructions are that after the system is powered up and completes its self-boot, the user should type
nwiiss/nwiiss

on the Verbex 3000 associated V2102 keyboard. This will cause the NWISS application to be loaded.

B. ENROLLING THE NWISS VOCABULARY

The first step in training one's voice patterns on the Verbex 3000 is to enroll the entire vocabulary (NWISS vocabulary is 151 words) at one time. This means that the Verbex 3000 will automatically step the user through all 151 words, requesting each to be spoken twice, occasionally three times, to get an initial set of voice patterns for each word. This process should take only fifteen minutes.

In order to make the enrollment process go smoothly, the user should take time beforehand to look at the vocabulary and decide what pronunciation will be given each word. As many of the NWISS words are really just symbols put together to make an aircraft type, weapon type, track number, callsign, et cetera, it is important to do this beforehand. See Table III for a complete NWISS vocabulary ordered (column by column) the same way as the Verbex 3000 will present it.

Suggested rules of thumb for pronunciation are:

1. In general, use the most natural pronunciation which comes to mind.

2. Pronounce numbers which appear as part of fixed identifiers naturally, e.g. "F14A" as "F fourteen A" or "MK43" as "Mark forty-eight".

3. Do pronounce digits which appear as part of variable strings, e.g. callsigns, track numbers, bearings, et cetera, as individual digits, e.g. "ALTITUDE 2500" as "ALTITUDE two five zero zero".

48
<table>
<thead>
<tr>
<th>Control_K</th>
<th>VK0</th>
<th>BCAP1</th>
<th>CIRCLE</th>
<th>AT</th>
<th>STRIKE</th>
<th>ON</th>
<th>S3A</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOR</td>
<td>VS0</td>
<td>BCAP2</td>
<td>XMARK</td>
<td>ALEKS</td>
<td>ASW</td>
<td>DECM</td>
<td>SH2F</td>
</tr>
<tr>
<td>KITTY</td>
<td>VTO</td>
<td>DESIGNATE</td>
<td>RADIUS</td>
<td>PETRO</td>
<td>AIRTANKER</td>
<td>OFF</td>
<td>SH3H</td>
</tr>
<tr>
<td>KNOX</td>
<td>VWO</td>
<td>NEUTRAL</td>
<td>POSITION</td>
<td>WONGSA</td>
<td>STNCAP</td>
<td>BLIP</td>
<td>LOAD</td>
</tr>
<tr>
<td>LOSAN</td>
<td>1POINT11</td>
<td>FRIENDLY</td>
<td>-</td>
<td>VLAD</td>
<td>FINGO</td>
<td>WEAPONS</td>
<td>HARP</td>
</tr>
<tr>
<td>MCCOR</td>
<td>0</td>
<td>ENEMY</td>
<td>N</td>
<td>RANGE</td>
<td>REFUEL</td>
<td>TIGHT</td>
<td>MK82</td>
</tr>
<tr>
<td>MISAW</td>
<td>1</td>
<td>BEARING</td>
<td>E</td>
<td>TORPEDO</td>
<td>VK00</td>
<td>FREE</td>
<td>MK83</td>
</tr>
<tr>
<td>OMAHA</td>
<td>2</td>
<td>FORCE</td>
<td>W</td>
<td>MK48</td>
<td>ORDERS</td>
<td>ALL</td>
<td>MK84</td>
</tr>
<tr>
<td>RATHB</td>
<td>3</td>
<td>DISPLAY</td>
<td>S</td>
<td>ASHCOC</td>
<td>ENCON</td>
<td>AIR</td>
<td>PHENX</td>
</tr>
<tr>
<td>SPRUA</td>
<td>4</td>
<td>CENTER</td>
<td>TRACK</td>
<td>MK46</td>
<td>SILEN</td>
<td>SUBMARINE</td>
<td>SHRIK</td>
</tr>
<tr>
<td>WICHI</td>
<td>5</td>
<td>PLACE</td>
<td>BAO</td>
<td>MK46A</td>
<td>SURF</td>
<td>LAUNCH</td>
<td>SPAR</td>
</tr>
<tr>
<td>WILSO</td>
<td>6</td>
<td>GRID</td>
<td>BEO</td>
<td>SPEED</td>
<td>AEW</td>
<td>A6E</td>
<td>SWDR</td>
</tr>
<tr>
<td>MP6</td>
<td>7</td>
<td>PLOT</td>
<td>BPO</td>
<td>STATION</td>
<td>RADIA</td>
<td>A7E</td>
<td>SQ538</td>
</tr>
<tr>
<td>SH1</td>
<td>8</td>
<td>LOB</td>
<td>BSO</td>
<td>COURSE</td>
<td>ATMS</td>
<td>E2C</td>
<td>SQ47</td>
</tr>
<tr>
<td>VAO</td>
<td>9</td>
<td>SONAR</td>
<td>BUO</td>
<td>PROCEED</td>
<td>TAKE</td>
<td>EA3B</td>
<td>SSQ53</td>
</tr>
<tr>
<td>VEO</td>
<td>EXECUTE</td>
<td>ERASE</td>
<td>FIRE</td>
<td>PERISCOPE</td>
<td>COVER</td>
<td>EA6B</td>
<td>SSQ62</td>
</tr>
<tr>
<td>VE1</td>
<td>DCAIR</td>
<td>ASW</td>
<td>ESM</td>
<td>CRUISE</td>
<td>DEPTH</td>
<td>FARRIER</td>
<td>F14A</td>
</tr>
<tr>
<td>VFO</td>
<td>A6STRIKE</td>
<td>DROP</td>
<td>HRON</td>
<td>SURFACE</td>
<td>ALTITUDE</td>
<td>F14T</td>
<td>STOP</td>
</tr>
<tr>
<td>WHO</td>
<td>F14STNCAP</td>
<td>CANCEL</td>
<td>TOHHK</td>
<td>MISSION</td>
<td>RBOC</td>
<td>KA6D</td>
<td></td>
</tr>
</tbody>
</table>
4. Pronounce -, N, E, W, and S as "dash", "north", "east", "west" and "south".

5. Do not use the phonetic alphabet for individual letter pronunciation. It is not natural or necessary. For example, pronounce "VA024" as "V A zero two four" not "victor alpha ..."

6. Give the full pronunciation to abbreviated ship names, shore base names, and weapon names, e.g. "Spruance", "Tomahawk", "Harpoon", "Sidewinder", "Misawa". ("Kitty" is perfectly acceptable for "Kittyhawk").

C. TRAINING THE GRAMMARS

The next step after enrollment is to train the NVISS grammars. This means that the Verbex 3000 will automatically step the user through a large number of triplets (3 words in a phrase). This training can be tedious especially with the number of digits used in NVISS. However it is very important to accomplish this training properly to get good recognition results. (After the enrollment phase, one could choose to test his or her recognition accuracy on the Verbex 3000 and would find scores ranging around 50 or 60. After the training phase, testing is automatically invoked and should show recognition scores in the 90's.)

To make this training less tedious, the following change has been made to the Verbex 3000 scheme of training: the 32 character display on the User's Console will ask whether it is desired to train

ALL GRAMMARS?

The only correct response is NO. It will then ask the user which grammars to train individually. This is the desired
mode. It allows the user to take a break in between grammars for as long as desired. With "allgrammars", the machine retains no memory of where one leaves the ordered list of triplets and, in essence, a marathon training session is implied. For this reason, and to conserve space, the NWISS "allgrammars" is merely a shell with but a few entries to please SPADS' expectations. Training the grammars individually should take about 15 or 20 minutes each, depending on size. The Digits grammar is last on the list and may be left untrained as the user will get to train many digits in the other grammars. However if digits ever sees a problem, then it may be worthwhile to train Digits as well as the other grammars.

D. TESTING

After each grammar has been trained, the Verbex 3000 will ask the user if testing is desired. This is a worthwhile two-minute exercise in which the Verbex 3000 displays complete legal paths (not just triplets) through the grammar and, after the user has spoken each, displays the recognition score for that utterance. Scores should generally be in the 90's with a few 80's. Scores in the 70's and below may indicate retraining is needed.

However complete paths through grammars are not complete NWISS commands. Users should test their "feel" for grammar boundaries, where pauses are required, by testing on NWISS itself (after training all grammars and prior to beginning operation). Figure 3.1 contains a fairly representative sample of NWISS commands which the user should attempt. Pauses are indicated by "..." and prompts are inside parentheses.
"DISPLAY ... RADIUS 250"
"EXECUTE P14STRCAP"
"DISPLAY ... CENTER FORCE ... KITTY"
"EXECUTE BAIRASH"
"DISPLAY ... PLACE CIRCLE FORCE ... MP602 ... (radius) 60"
"FOR 1.1 ... WEAPCONS FREE ENEMY ALL"
FOR KITTY ... LAUNCH ... 3 A6E A6E1 ... (course) 270 ... (speed) 450 ...
     (altitude) 2000"
     FLT PLAN: "MISSION STRIKE"
     FLT PLAN: "PROCEED POSITION ... 37N ... 135-30E"
     FLT PLAN: "LOAD ... (equipment) 2 SHRIK ... 9 MK82 ... 3 HAAR ... 7 SWDR"
     FLT PLAN: "STOP"
"LESEIGNATE ENEMY ... (track) BS005"
"FOR VA012 ... WEAPONS FREE ENEMY SURFACE"
"FOR VA012 ... TAKE ... (track) BS005"
"FOR CMAHA ... FIRE ... 4 MK48 TORPEDO ... (track) BU003"
"FOR SH103 ... BARRIER ... (position) 35-30N ... 129-45E ... (bearing) 180 ...
       (distance) 60 ... (using) SSQ47 ... (spacing) 3"
"BEARING FORCE ... KNOX ... (to) TRACK ... BS019"
"FOR KNOX ... FIRE ... 2 HRPON CRUISE ... BEARING 270 ... RANGE 15"

Figure 3.1 NWISS Test Commands.
E. OPERATIONAL USE

After the enrollment and training phases are over, the interesting phase begins: actual input to NWISS. What follows are a few suggestions to make this phase easier and hopefully trouble-free. In general one should enter the NWISS commands in accordance with the display messages (which appear on the User's Console every time the Verbex 3000 must leave one grammar and call in another) and with the "feel" for those grammars obtained from training the triplets. Speech is continuous within a grammar but discrete across grammar boundaries. A few guidelines are:

1. A pause is always required after saying "FOR <addressee>", "DISPLAY", "FORCE", "TRACK", or "POSITION". (Wait for the appropriate display prompt before continuing).

2. When speaking a field of digits, prepare ahead of time what they are and speak them continuously without pausing in the middle. However this is not true of positions (latitude and longitude), aircraft callsigns, or track numbers (i.e. FORCE, TRACK, and POSITION) which are defined as fixed length fields and may be entered as discretely or continuously as desired.

3. Simple commands which have only one argument of digits should be spoken continuously, e.g. "SPEED 35", "RADIUS 250", or "ALTITUDE 2000".

4. For the more difficult commands (i.e. FIRE, LAUNCH, BARRIER, and STATION) the command keyword itself serves as an entry point to other grammars and application code and hence a pause is required after it.
5. Under "LAUNCH", <event-name> must be specified as the aircraft type plus a single digit, e.g. "F14A1" or "F3C2".

6. Conversation with other players can easily trigger recognition in the Verbex 3000 and cause unwanted output to NWISS. This can be prevented by swinging the headset microphone away and covering it with the hand or pressing the "STOP" button on the User's Console. This latter method is most effective as it stops the Verbex 3000 from listening and is easy to clear: simply press the "YES" key in response to the "CONTINUE?" message on the display.

7. Don't panic if the above happens. "CONTROL_K" can be issued from anywhere and will return the process back to NWisgram! ("ENTER NWISS COMMAND PLEASE" is displayed).
IV. CONCLUSIONS AND RECOMMENDATIONS

The scope of this thesis was confined to the analysis of NWISS command syntax and the creation of continuous voice application software to meet the requirements developed from that analysis. The result is that NWISS users can input their commands using their most natural mode of communication, speech, as the means of input. While the thesis objectives are therefore satisfied in this practical sense, whether the higher goal of achieving a truly effective improvement in man-machine interaction has been achieved cannot be known until independent, operational testing is conducted. At this time no one but the author has trained their voice patterns on the Verdex 3000 and exercised the NWISS continuous voice application.

Hence the principal recommendation is that the NWISS voice application be used, tested, refined and improved to ensure that the above goal is achieved. Only through use will the pitfalls be found and corrected. The requirements, as analyzed here, will change over time and be reinterpreted several times as well. New NWISS scenarios are certain to be developed and require new force names, base names, etcetera. The Orange side needs to have its continuous voice application too: hopefully that will not be difficult to do in light of this thesis. Thus, like any original software work, the NWISS voice application will need to be modified, in all likelihood extensively, and toward that end it has been defined and organized through this thesis.

Some thought should perhaps be given to approaching the problem from the other end: how should NWISS command syntax be changed to reflect the needs of voice input? One suggestion is that NWISS not handle the incoming data character by
character, as from a keyboard, but instead "buffer" and parse the entire command: the user can then "send" the buffer if it is satisfactory or cancel it. Prompting can be adequately provided by the Verbex 3000 in such a setup.

Some "requirements" have not been met. In particular the ability to specify the "TIME" of a command is not available. This is the result of a design judgment that it would be too difficult to provide and is seldom used. Another lack is the ability to get help (keyboard "?" at any place in a command. The problem here is that to get the desired effect, every grammar must have a multitude of legal paths defined which end in ". However, with careful study, one might be able to redefine some of the NWISS grammars to allow "help" to be spoken in the middle of a command where it might most be needed. Here is a situation where perhaps NWISS modification, such as having a separate "HELP" command whereby one would specify the command and/or command argument where help is needed, might be easier to accomplish. One or two additional grammars would be required on the Verbex 3000 but there is room for that. Creation of such grammars would also facilitate implementing the "CANCEL" command of NWISS which is not implemented currently.

There will be some who criticize the grammar boundaries as either being misplaced or just too "discrete". Either criticism could well be valid: misplacements can be corrected to some extent but the length of pauses for grammar boundaries will be more difficult. Only faster processors and faster, larger memories can solve this problem. The Verbex 3000 represents the state of the art (commercially) today in terms of affordable continuous voice recognition technology.

Sometime in the foreseeable future men will talk naturally to machines and machines will talk back in clear, understandable prose. Obviously the NWISS continuous voice
application is far removed from that scenario, but it represents one of many steps which must be taken. To the extent that computer-aided wargames such as NWISS entail a high degree of man-machine interaction and model the real world where military decision makers must make real time decisions in consort with a computer, they serve the purpose of promoting man-computer symbiosis. Hopefully this demonstration of continuous voice recognition technology will form a foundation for further study by others into its full possibilities in advancing the state of man-machine interaction.
LIST OF REFERENCES


APPENDIX A

PASCAL CODE FOR NEWSS CONTINUOUS VOICE APPLICATION

PROCEDURE application;
LABEL 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
25,26,27,28,29,30,31,32,33,34,35;
VAR
buf, space, null, dash, period, lf, ctrl_f, ctrl_k : string;
one, e, f, g, h, k, n, o, p, r, t, x, display$ : string;
shiporder, airorders, fire, display, track : grammar;
lauch, digits, nwisgram, position, force : grammar;
status, status1 : short;
dispcall, launchcall, launchloop, firecall : boolean;
barriercall, barrierloop, barrier2loop, barrier3loop : boolean;
lltrauto, nwiscall, station1call, station2call : boolean;
l, m : integer;
BEGIN
status := hostinitialize(baud_9600,eight_bits,no_parity,one_stop_bit);
nwisgram1 := "nwisgram1";
poshion := "position";
force := "force";
track := "track";
fire := "fire";
digits := "digits";
launch := "launch";
shiporder := "shiporder";
airorders := "airorders";
lf := "$n$";
lf.characters[1] := chr(10);
ctrl_f := "$m$";
ctrl_f.characters[1] := chr(6);
ctrl_k := "$m$";
ctrl_k.characters[1] := chr(11);
period := "$n$";
period.characters[1] := chr(46);
dash := "$m$";
dash.characters[1] := chr(45);
space := "$n$";
space.characters[1] := chr(32);
nul := "$n$";
nul.characters[1] := chr(0);
one := "$n$";
one.characters[1] := chr(49);
c := "$m$";
c.characters[1] := chr(67);
e := "#";
characters[1] := chr(69);

f := ";
characters[1] := chr(70);
g := ";
characters[1] := chr(71);
h := ";
characters[1] := chr(72);
k := ";
characters[1] := chr(75);
n := ";
characters[1] := chr(78);
o := ";
characters[1] := chr(79);
p := ";
characters[1] := chr(80);
q := ";
characters[1] := chr(82);
r := ";
characters[1] := chr(84);
x := ";
characters[1] := chr(88);

1: dispmsg := "ENTER NISS COMMAND PLEASE";
    GOTO 3;
2: dispmsg := "FORCE, TRACK, OR POSITION";
3: nwisecall := false;
station1call := false;
station2call := false;
displaycall := false;
hardircall := false;
barrier1loop := false;
barrier2loop := false;
barrier3loop := false;
launcheall := false;
launcheall := false;
tirecall := false;
status1 := timeout;
REPEAT
    status := displaymessageclear(display_primary, dispmsg);
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.characters[1]) THEN
        GOTO 5
    ELSE
        GOTO 6
    END IF
    status1 := timeout;
    UNTIL status1 = voicein;
        GOTO 4
    ELSE IF (buf.characters[4] = p.characters[1]) AND \{display\}
        (buf.characters[8] = hull.chars...
ELSE IF (buf.characters[4] = c.characters[1]) AND
(buf.characters[9] = t.characters[1]) THEN
(bearing track)
BEGIN
  status := hostwrite(buf);
  status := hostwrite(space);
  nvisicall := true;
  GOTO 6
END
ELSE IF (buf.characters[5] = y.characters[1]) OR
(buf.characters[1] = t.characters[1]) THEN
(designate)
BEGIN
  status := hostwrite(buf);
  status := hostwrite(space);
  GOTO 6
END
ELSE IF (buf.characters[4] = c.characters[1]) AND
(buf.characters[9] = f.characters[1]) THEN
(track)
BEGIN
  status := hostwrite(buf);
  status := hostwrite(space);
  nvisicall := true;
  GOTO 7
END
ELSE IF (buf.characters[4] = c.characters[1]) AND
(buf.characters[5] = c.characters[1]) THEN
(bearing force)
BEGIN
  status := hostwrite(buf);
  status := hostwrite(space);
  GOTO 7
END
(position)
BEGIN
  status := hostwrite(buf);
  status := hostwrite(space);
  GOTO 8
END
execute .preplan
GOTO 11
ELSE
(control_k)
BEGIN
  status := hostwrite(ctrl_k);
  status := hostwrite(lf);
  GOTO 1
END;

4: IF (buf.characters[8] = space.characters[1]) AND
(buf.characters[10] = space.characters[1]) THEN
.aircraft
BEGIN
  buf.characters[ 8 ] := buf.characters[ 9 ];
  buf.characters[ 9 ] := buf.characters[ 10 ];
  buf.characters[ 10 ] := null.characters[ 1 ];
  status := hostwrite (buf);  
  status := hostwrite (space);
  GOTO 12
END
BEGIN
  buf.characters[ 6 ] := period.characters[ 1 ];
  buf.characters[ 7 ] := one.characters[ 1 ];
  buf.characters[ 8 ] := null.characters[ 1 ];
  status := hostwrite (buf);
  status := hostwrite (space);
  GOTO 13
END
ELSE IF (buf.characters[ 1 ] = c.characters[ 1 ]) AND
         (buf.characters[ 9 ] = k.characters[ 1 ]) THEN
BEGIN
  status := hostwrite (ctrl_k);
  GOTO 1
END
ELSE
BEGIN
  status := hostwrite (buf);
  status := hostwrite (space);
  GOTO 14
END
END;

5: displaymsg := "DISPLAY COMMAND PLEASE";
FOR i := 7 to 22 DO
  buf.characters[i] := null.characters[1];
status1 := timeout;
REPEAT
  status := displaymessageclear (display_primary, displaymsg);
  status1 := recognize (display, buf);
  UNTIL status1 = voicein;
  GOTO 35
ELSE IF (buf.characters[10] = k.characters[1]) OR
         (buf.characters[16] = k.characters[1]) OR
         (buf.characters[17] = k.characters[1]) OR
         (buf.characters[19] = k.characters[1]) OR
         (buf.characters[20] = k.characters[1]) THEN
BEGIN
  status := hostwrite (buf);
status := hostwrite(space);

IF ((buf.characters[1] = f.characters[1]) AND
    (buf.characters[4] = c.characters[1]) AND
    (buf.characters[7] <> x.characters[1])) THEN
  displaycall := true;  {xmark}
ELSE displaycall := false;
GOTO 6
END

ELSE IF (buf.characters[8] = f.characters[1]) OR
         (buf.characters[12] = f.characters[1]) OR
         (buf.characters[13] = f.characters[1]) OR
         (buf.characters[14] = f.characters[1]) OR
         (buf.characters[15] = f.characters[1]) THEN
   status := hostwrite(buf);
   status := hostwrite(space);
   IF ((buf.characters[1] = p.characters[1]) AND
       (buf.characters[4] = c.characters[1]) AND
       (buf.characters[7] <> x.characters[1])) THEN
     displaycall := true;  {xmark}
ELSE displaycall := false;
GOTO 7
END

ELSE IF (buf.characters[8] = p.characters[1]) OR
         (buf.characters[12] = p.characters[1]) OR
         (buf.characters[13] = p.characters[1]) OR
         (buf.characters[14] = p.characters[1]) OR
         (buf.characters[15] = p.characters[1]) THEN
   status := hostwrite(buf);
   status := hostwrite(space);
   IF ((buf.characters[1] = p.characters[1]) AND
       (buf.characters[4] = c.characters[1]) AND
       (buf.characters[7] <> x.characters[1])) THEN
     displaycall := true;  {xmark}
   ELSE displaycall := false;
GOTO 8
END

ELSE
  [plot/erase LOB sonar/ESM, cancel grid/xmark/circle all]
END

ELSE BEGIN
  status := hostwrite(buf);
  status := hostwrite(lf);
GOTO 1
END

b: displaymsg := "WHAT TRACK PLEASE";
status1 := timeout;
REPEAT
  status := displaymessageclear(display_primary, displasmg);
  status1 := recognize(force, buf);
UNTIL status1 = voicein;
  BEGIN
    status := hostwrite(ctrl_k);
    GOTO 1
  END
ELSE
  BEGIN
    buf.characters[5] := buf.characters[7];
  END;
  status := hostwrite(buf);
IF displaycall THEN
  BEGIN
    status := hostwrite(space);
    GOTO 2
  END
ELSE IF displaycall THEN
  BEGIN
    status := hostwrite(space);
    GOTO 24
  END
ELSE
  BEGIN
    status := hostwrite(1f);
    GOTO 1
  END;
END;

7: IF station1call THEN
  displasmg := "WHAT GUIDE PLEASE"
ELSE
  displasmg := "WHAT FORCE PLEASE"
ELSE
  status1 := timeout;
REPEAT
  status := displaymessageclear(display_primary, displasmg);
  status1 := recognize(force, buf);
UNTIL status1 = voicein;
  BEGIN
    buf.characters[2] := period.characters[1];
    buf.characters[3] := one.characters[1];
BEGIN
END
ELSE IF (buf.characters[1] = c.characters[1]) AND
          (buf.characters[9] = k.characters[1]) THEN
BEGIN
  status := hostwrite(ctrl_k);
  GOTO 1
END
ELSE
BEGIN
  status := hostwrite(buf);
  IF nuis1call THEN
  BEGIN
    status := hostwrite(space);
    GOTO 2
  END
ELSE IF displaycall THEN
BEGIN
  status := hostwrite(space);
  GOTO 24
END
ELSE IF station1call THEN
BEGIN
  status := hostwrite(space);
  station2call := true;
  GOTO 20
END
ELSE
BEGIN
  status := hostwrite(1f);
  GOTO 1
END;

8: dispmsg := "LATITUDE PLEASE";
latloop := true;
GOTO 10;
9: dispmsg := "LONGITUDE PLEASE";
10: status1 := timeout;
REPEAT
  status := displaymessageclear(display_primary, dispmsg);
  status1 := recognize(position, buf);
UNTIL status1 = voicein;
IF (buf.characters[1] = c.characters[1]) AND (buf.characters[9] = k.characters[1]) THEN [control_k]
BEGIN
status := hostwrite(ctrl_k);
GOTO 1
END
ELSE
BEGIN
i := 0;
REPEAT
i := i + 1;
UNTIL buf.characters[i] = null.characters[1];
FOR j := 2 to (i DIV 2) DO buf.characters[i] := buf.characters[2*j-1]; buf.characters[i DIV 2 + 1] := null.characters[1]
END;
status := hostwrite(buf);
IF latlcmloop THEN
BEGIN
status := hostwrite(space);
latlcmloop := false;
GOTO 9
END
ELSE IF displaycall THEN
BEGIN
status := hostwrite(space);
GOTO 24
END
ELSE IF barriercall THEN
BEGIN
status := hostwrite(space);
GOTO 25
END
ELSE IF launchcall THEN
BEGIN
status := hostwrite(lf);
launchcall := false;
GOTO 32
END
ELSE
BEGIN
status := hostwrite(lf);
GOTO 0
END;
END;

11: status := hostwrite(ctrl_lf);
i := 0; [execute .preplan]
REPEAT
  i := i + 1;
  UNTIL buf.characters[1] = null.characters[1];
  FCH j := 1 to i-9 DO
    buf.characters[1] := buf.characters[j+8];
    buf.characters[1-8] := period.characters[1];
    buf.characters[1-7] := p.characters[1];
    buf.characters[1-6] := r.characters[1];
    buf.characters[1-5] := e.characters[1];
    buf.characters[1-4] := null.characters[1];
    status := hostwrite(buf);
    status := hostwrite(lf);
GOTO 1;

12: IF lauchcall THEN
  displaysg := "STATE 'MISSION ______'",
ELSE
  displaysg := "AIRCRAFT ORDERS PLEASE";
  status1 := timeout;
  REPEAT
    status := displaymessageclear(display_primary, displaysg);
    status1 := recognize(airorders, buf);
    UNTIL status1 = voicein;
    IF (buf.characters[1] = c.characters[1]) AND
        (buf.characters[9] = k.characters[1]) THEN
      BEGIN
        status := hostwrite(ctrl_k);
        GOTO 1
      END
    ELSE IF (buf.characters[1] = i.characters[1]) AND
              (buf.characters[3] = r.characters[1]) THEN
      BEGIN
        status := hostwrite(buf);
        status := hostwrite(space);
        GOTO 16
      END
    ELSE IF (buf.characters[9] = o.characters[1]) AND
              (buf.characters[7] = n.characters[1]) THEN
      BEGIN
        status := hostwrite(buf);
        status := hostwrite(lf);
        IF lauchcall THEN
          GOTO 3;
        ELSE
          GOTO 1
        END
      ELSE IF ((buf.characters[1] = c.characters[1]) AND
        (buf.characters[3] = r.characters[1]) AND
        (buf.characters[7] = n.characters[1]);
BEGIN status := hostwrite(buf);
status := hostwrite(spc);
GOTO 6
END
ELSE IF \( (\text{buf} \cdot \text{characters}[3] = \text{p} \cdot \text{characters}[1]) \) AND \( (\text{buf} \cdot \text{characters}[9] = \text{p} \cdot \text{characters}[1]) \) THEN
BEGIN status := hostwrite(buf);
status := hostwrite(spc);
GOTO 8
END
ELSE IF \( ((\text{buf} \cdot \text{characters}[2] = \text{e} \cdot \text{characters}[1]) \) OR \( ((\text{buf} \cdot \text{characters}[3] = \text{e} \cdot \text{characters}[1]) \) OR \( ((\text{buf} \cdot \text{characters}[4] = \text{r} \cdot \text{characters}[1]) \) OR \( ((\text{buf} \cdot \text{characters}[5] = \text{t} \cdot \text{characters}[1]) \) AND \( (\text{buf} \cdot \text{characters}[3] = \text{r} \cdot \text{characters}[1]) \) AND \( (\text{buf} \cdot \text{characters}[4] = \text{t} \cdot \text{characters}[1]) \) AND \( (\text{buf} \cdot \text{characters}[5] = \text{t} \cdot \text{characters}[1]) \) THEN
GOTO 35
ELSE
BEGIN status := hostwrite(buf);
status := hostwrite(\text{lf});
GOTO 1
END;

13: \text{displmsg} := \text{"ORDERS FOR 1.1 PLEASE"}:
BEGIN IF (buf_characters[1] = t_characters[1]) AND (control_k) THEN [fire]
END
BEGIN IF (buf_characters[1] = t_characters[1]) AND (take) THEN [launch]
END
BEGIN IF (buf_characters[1] = t_characters[1]) AND (control_k) THEN [station]
GO TO 40
END
ELSE IF ((buf\_characters[2] = p\_characters[1]) AND
(buf\_characters[3] = p\_characters[1]) OR
(buf\_characters[4] = t\_characters[1]) OR
(buf\_characters[2] = o\_characters[1]) AND
(buf\_characters[4] = r\_characters[1])) THEN
GO TO 35
ELSE
BEGIN
[eriscope, surface, orders, emcon plan or weapons f/t] [or rdoc/decem/blip on/off]
status := hostwrite(buf);
status := hostwrite(tf);
END;
16: displaymsg := "NR, TYPE, TORPEDO OR CRUISE";
GO TO 19;
17: displaymsg := "STATE 'BEARING ___' OR 'AT ___'";
GO TO 19;
18: displaymsg := "STATE 'RANGE ____'";
19: status1 := timeout;
REPEAT
status := displaymessageclear(display\_primary, displaymsg);
status1 := recognize(fire, buf);
UNTIL status1 = voicein;
IF ((buf\_characters[1] = c\_characters[1]) AND
(buf\_characters[9] = k\_characters[1])) THEN
BEGIN
status := hostwrite(ctrl\_k);
GO TO 1
END
ELSE IF ((buf\_characters[8] = t\_characters[1]) AND
(buf\_characters[11] = p\_characters[1]) OR
(buf\_characters[9] = t\_characters[1]) AND
(buf\_characters[12] = p\_characters[1])) OR
BEGIN
status := hostwrite(buf);
status := hostwrite(space);
GO TO 6
END
ELSE IF (buf\_characters[2] = t\_characters[1]) AND
(buf\_characters[3] = space\_characters[1]) THEN
BEGIN
status := hostwrite(buf);
status := hostwrite(tf);
GO TO 1
END
ELSE IF (buf.characters[9] = c.characters[1]) AND (buf.characters[10] = r.characters[1]) THEN
BEGIN
status := hostwrite(buf);
status := hostwrite(space);
GOTO 17;
END
BEGIN
firecall := true;
GOTO 35
END
ELSE
GOTO 35;

20: displaymsg := "RANGE";
GOTO 29;
21: displaymsg := "DISTANCE";
barriercopy := true;
GOTO 29;
22: displaymsg := "TYPE OF SONOBUOY";
GOTO 29;
23: displaymsg := "SPACING";
barriercopy := true;
GOTO 29;
24: displaymsg := "CIRCLE OR CIRCLE RADIUS IN NM";
GOTO 29;
25: displaymsg := "BEARING";
GOTO 29;
26: displaymsg := "COURSE";
GOTO 29;
27: displaymsg := "SPEED";
lauchcopy := true;
GOTO 29;
28: displaymsg := "ALTITUDE";
29: status1 := timeout;
REPEAT
status := displaymessageclear(display_primary, displaymsg);
status1 := recognize(digits, buf);
UNTIL status1 = voicein;
IF (buf.characters[1] = c.characters[1]) AND (buf.characters[1] = k.characters[1]) THEN
BEGIN
status := hostwrite(ctrl_k);
GOTO 1
END
BEGIN
i := 0;
REPEAT
i := i + 1;
UNTIL buf.characters[i] = null.characters[1]; FOR j := 2 to (i DIV 2) DO
buf.characters[j] := buf.characters[2*j-1];
buf.characters[i DIV 2 + 1] := null.characters[1]
END
ELSE
barrier2loop := true;
status := hostwrite(buf);
IF displaycall OR station2call OR barrier3loop THEN
BEGIN
status := hostwrite(id);
GOTO 1
END
ELSE IF station1call THEN
BEGIN
status := hostwrite(space);
GOTO 7
END
ELSE IF barrier2loop THEN
BEGIN
status := hostwrite(space);
GOTO 23
END
ELSE IF barrier1loop THEN
BEGIN
status := hostwrite(space);
GOTO 22
END
ELSE IF barriercall THEN
BEGIN
status := hostwrite(space);
GOTO 21
END
ELSE IF launchcall THEN
BEGIN
status := hostwrite(space); launchcall := false;
GOTO 27
END
ELSE IF launchloop THEN
BEGIN
status := hostwrite(space);
launchloop := false;
GOTO 28
END
ELSE
BEGIN
status := hostwrite (lf);
launchcall := true;
GOTO 12
END;

30: displaymsg := "NR, ACFTYPE, ACFTYPE + DIGIT";
launchcall := true;
GOTO 34;
31: displaymsg := "PROCEED POSITION, LOAD OR STOP";
launchcall := false;
GOTO 34;
32: displaymsg := "LOAD OR STOP";
GOTO 34;
33: displaymsg := "NUMBER AND WEAPONTYPE, OR STOP";
34: status1 := 'timeout;
REPEAT
status := displaymessageclear(display_primary, displaymsg);
status1 := recognize(launch, buf);
UNTIL status1 = 'voicein;
IF {buf.characters[1] = c.characters[1]} AND
{buf.characters[9] = k.characters[1]} THEN
BEGIN
status := hostwrite (ctrl_k);
GOTO 1
END 
ELSE IF launchcall THEN
BEGIN
i := 0;
REPEAT
i := i + 1;
UNTIL buf.characters[i] = null.characters[1];
buf.characters[i-2] := buf.characters[i-1];
buf.characters[i-1] := null.characters[1];
status := hostwrite (buf);
status := hostwrite (space);
GOTO 28
END
ELSE IF (buf.characters[2] = o.characters[1]) AND
(buf.characters[5] = null.characters[1]) THEN
BEGIN
status := hostwrite (buf);
status := hostwrite (space);

}
GOTO 33
END
BEGIN
    status := hostwrite(buf);
    status := hostwrite(space);
    launchcall := true;
GOTO 8
END
BEGIN
    status := hostwrite(lf);
    status := hostwrite(buf);
    status := hostwrite(lf);
GOTO 1
END
ELSE IF (buf.characters[4] = space.characters[1]) THEN (wpnsload)
BEGIN
    i := 0;
    REPEAT
        i := i + 1;
        UNTIL buf.characters[i] = null.characters[1];
    FOR j := 2 to i-1 DO
        buf.characters[j] := buf.characters[j+1];
    status := hostwrite(buf);
    status := hostwrite(space);
GOTO 33
END
ELSE
BEGIN
    status := hostwrite(buf);
    status := hostwrite(space);
GOTO 33
END;

35: i := 0;
    m := 0;
    REPEAT
        i := i + 1;
        UNTIL buf.characters[i] = null.characters[1];
    REPEAT
        m := m + 1;
        UNTIL buf.characters[m] = space.characters[1];
    FOR j := m+2 to (i+m) DIV 2 DO
        buf.characters[j] := buf.characters[2*j-m-1];
**INITIAL DISTRIBUTION LIST**

<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Name and Address</th>
</tr>
</thead>
</table>
| 1.  | 2      | Defense Technical Information Center  
Defense Technical Information Center  
Cameron Station  
Alexandria, Virginia 22314 |
| 2.  | 2      | Library, Code 0142  
Naval Postgraduate School  
Monterey, California 93943 |
| 3.  | 15     | G. Poock, Code 55PK  
Naval Postgraduate School  
Monterey, California 93943 |
| 4.  | 1      | CDR G. Porter, Code 55PT  
Naval Postgraduate School  
Monterey, California 93943 |
| 5.  | 1      | John P. Lombardo  
10542 Moss Gate Ct.  
Ellicott City, MD 21043 |
| 6.  | 1      | Dr. Azad Madni  
Perceptronics, Inc.  
6271 Variel Avenue  
Woodland Hills, CA 91367 |
| 7.  | 1      | Hal Miller  
Code 8302  
Naval Ocean Systems Center  
San Diego, CA 92152 |
| 8.  | 1      | Lon Davenport  
9C-62  
Boeing Computer Services  
55 Andover Pkwy.  
Tukwila, WA 98188 |
| 9.  | 1      | Tice DeYoung  
U.S. Army Engineering Topographic  
Lab Research Institute  
Ft. Belvoir, VA 22060 |
| 10. | 1      | CPT Naughton  
Naval War College  
Newport, RI 02840 |
| 11. | 1      | Chief of Naval Creations  
Attn: Commodore Armstrong  
GP953  
The Pentagon  
Washington, DC 20350 |
| 12. | 1      | M. Sovereign, Code 74  
Naval Postgraduate School  
Monterey, CA 93943 |
| 13. | 1      | Curricular Officer, Code 39  
Naval Postgraduate School  
Monterey, CA 93943 |