THE INTEGRATION OF AUTOMATIC SPEECH RECOGNITION INTO THE AIR TRAFFIC CONTROL SYSTEM

by

JOAKIM KARLSSON

B.S.E., Mechanical and Aerospace Engineering
Princeton University, Princeton, N.J.
(1988)

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of
The Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

January 1990

Signature of Author [illegible]
Department of Aeronautics and Astronautics
January, 1990

Certified by ____________________
Prof. Robert W. Simpson
Thesis Supervisor

Accepted by ____________________
Prof. Harold Y. Wachman
Chairman, Department Graduate Committee

THE INTEGRATION OF AUTOMATIC SPEECH RECOGNITION INTO THE AIR TRAFFIC CONTROL SYSTEM

by

JOAKIM KARLSSON

Submitted to the Department of Aeronautics and Astronautics
on January 17, 1989 in partial fulfillment of
the requirements for the Degree of Master of Science in
Aeronautics and Astronautics

ABSTRACT

Today, the Air Traffic Control (ATC) system relies primarily on verbal communication between the air traffic controllers and the pilots of the aircraft in the controlled airspace. Although a computer system exists that processes primary radar, secondary radar, and flight plan information, the information contained within the verbal communications is not retained. The introduction of Automatic Speech Recognition (ASR) technology would allow this information to be captured for processing.

The research presented in this paper examines the feasibility of using ASR technology in the Air Traffic Control environment. The current status of the technology is assessed. Problems that are unique to ATC applications of voice input are identified. Since ASR technology is inherently a part of the man-machine interface between the user and the system, emphasis is placed on the relevant human factors issues. A man-machine model is presented which demonstrates the use of mixed input modalities, automatic error detection and correction techniques, and the optimal use of feedback to the controller.

Much of the potential benefit of introducing ASR technology into the Air Traffic Control system is a result of the highly constrained language used by air traffic controllers. Consequently, the information content of the ATC language must be determined, and methods must be designed to process the various levels of knowledge inherently available in ATC communications. The man-machine model adopted in this paper demonstrates techniques to utilize syntactic, semantic, and pragmatic information to improve overall recognition accuracy. An intelligent, adaptive voice input parser is presented.

Thesis Supervisor: Dr. Robert W. Simpson

Title: Professor of Aeronautics and Astronautics

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor Robert W. Simpson, my thesis advisor, as well as to all staff and student members of the Flight Transportation Laboratory.

Many thanks to Professor David L. Akin for kindly supplying me with the necessary computing equipment required to finish this work on time.

The research presented within this paper was sponsored by the FAA/NASA Joint University Program for Air Transportation Research. I would like to thank the individuals associated with this program, not only for their support, but also for an invaluable educational experience.

Most of all, I would like to thank my wife Karen without whom none of this would have been possible.

CONTENTS

LIST OF FIGURES

List of Tables

CHAPTER ONE

INTRODUCTION

1.1 MOTIVATION

The use of Automatic Speech Recognition (ASR) equipment in classical applications such as package sorting, quality control, cartography, or receiving inspection, is typically justified by one or more of the following factors [1, 2]:

- Voice input allows operators to remain mobile.
- Hands and eyes can be directed towards the primary task while providing commands and input through speech.
- Input data rate is higher for speech than other human based modalities.
- Use of voice input requires little operator training.

It has also frequently been pointed out that ATC employs strictly constrained and well defined vocabulary and syntax, making it an amenable environment for the introduction of ASR technology [3, 4, 5, 6, 7, 8, 9]. However, it must be recognized that this alone cannot be the justification for introducing ASR technology into the ATC system. Furthermore, the classical benefits of ASR technology listed above do not necessarily apply to the ATC environment.

Before addressing the motivation for using ASR technology in ATC applications it is thus necessary to identify the fundamental differences between ATC and other applications where the use of voice input may be beneficial. The most significant of these differences is that voice is currently the primary communication channel in ATC. Controllers are already mobile (when required, as in the tower cab environment), and hands and eyes are almost exclusively directed to the primary task of ensuring aircraft separation. The higher information throughput capacity of speech is already being used advantageously. The issue of introducing new technology into the ATC system is further complicated by the possibility of errors and failures leading to fatal results, and the critically high workload already present during peak traffic situations. Hence, the only possible justifications for introducing ASR technology are those that imply an improvement in safety and efficiency, while maintaining or reducing the air traffic controller workload.

The fundamental potential benefit of using ASR equipment in ATC systems lies in the technology's capability to capture information that is currently transmitted in verbal form between the controllers and the participating aircraft. These transmissions typically consist of vectors, traffic advisories, requests for information about the aircraft and its intentions, or the provision of information useful to the crew of the aircraft. Thus, the information contained within these verbal communications contains knowledge about the state of the airspace, as well as knowledge about its future state. The underlying motivation for the introduction of ASR technology into the ATC environment is to acquire this knowledge in a form suitable for automatic processing.

The ability to capture the information contained in the verbal messages delivered by ATC would open up a host of new capabilities that could improve both safety and efficiency:

- Post-processing of the input could be used to predict the future state of the airspace, providing an early warning capability to avoid separation criteria violations or other hazards.
- A history of clearances could be presented to the controller as a memory aid.
- Mode-S equipped aircraft could receive a copy of the captured information, for use as a backup or for on-board processing.
- Routine or anticipated clearances could be prestored during periods of little or no activity.
- Controller strategies could be monitored, providing useful information for evaluation, planning, and training.
- The use of keyboards and trackballs to designate aircraft and change data tag information could be reduced, as much of the information entered by these modalities is already present in the voice communications.

Hence, the possibility exists to acquire and process strategically significant knowledge. This can be achieved without extensive modifications to the current operating practices of the air traffic controller, which is significant not only from a convenience standpoint, but also to ensure that the controller workload is not increased.

Given this potential benefit for using ASR technology, specific preliminary research requirements can be defined. These requirements constitute the basic areas of investigation of the research presented in this paper. The most basic of these is an evaluation of the current state of ASR hardware. ATC specific performance criteria must be selected to set up a set of specifications against which currently existing equipment can be assessed. Limitations of the hardware must be identified, and an attempt should be made to predict any improvements that can be expected to be introduced in the near future. Evaluation procedures must be designed, to test the equipment under the specific conditions present in the ATC environment.

Despite the significance of the performance of the ASR hardware, the success of introducing speech recognition technology is contingent on another issue – the interaction between man and machine. The National Research Council has concluded that "human factors issues are central to the successful deployment of voice interactive systems." [3] This is inherent from the nature of the technology, as it represents an interface between the user and the system under control. Human factors are particularly significant in ATC applications, where levels of high workload intermixed with occasional periods of boredom are common. The possibility of loss of life in the event of human factors related failures and deficiencies underscores the importance of these issues.

Human factors issues are of significance in all applications utilizing ASR, and hence considerable research has already been completed in this area, identifying key issues such as background noise, microphone placement, speech variations, user acceptability and so on [10, 11]. However, since ATC differs from classical ASR applications, it must be determined to what extent these issues have an impact. Also, human factors issues unique to the ATC environment must be identified. A model for a functional man-machine interface should be designed and evaluated, to demonstrate the feasibility of advanced techniques to overcome human factors related problems. In order to minimize these problems, it is necessary to implement an iterative design cycle, that should be continued until the needs of the system users are met [10]. A human factors perspective has been adopted throughout this paper, and the research effort it represents should be considered as one step in that design cycle.

Another issue that is directly related to the use of voice input technology is the nature of the ATC language. Once again, there is a major difference between ATC and other ASR applications in this respect. In ATC, the vocabulary and syntax are strictly defined and cannot easily be altered [12, 13]. In other ASR applications, the definition of the permissible vocabulary and syntax is an element controlled by the designer. Hence, it is necessary to examine the information content of the ATC language, to determine its suitability for speech recognition, and to evaluate the extent of controller deviation from the prescribed syntax in today's ATC system. Techniques must be developed to either ensure conformance to the syntax or to cope with occasional deviations.

Other authors have correctly concluded that current ASR technology falls short of the performance necessary to be used operationally for ATC applications [8]. However, the hardware is sufficiently capable that it can be used for the preliminary research necessary before operational use. The potential benefits of the technology warrant that such preliminary work should be undertaken at this point. Although this paper does not present any components or systems intended for operational use, it constitutes an effort to investigate the hardware, human factors, and language issues that must be successfully solved in order to capitalize on the potential of ASR technology.

1.2 APPLICATION AREAS

Although this paper concerns itself primarily with the *operational* use of ASR technology in ATC, it should be recognized that there is a secondary application area – ATC *training and simulation*. The operational environment and the simulation environment each have distinct requirements that demand unique solutions for the issues related to introducing ASR technology.

In the operational environment, the motivation is based on a potential increase in safety and efficiency. However, requirements on robustness and recognition accuracy will be very stringent, disqualifying current ASR technology from being introduced. Furthermore, issues such as mental workload, design of user feedback, and ability to cope with syntax deviations will be critical. Controller acceptance of the new technology, development of new procedures, and initial training will also be significant issues. Overall, there is a substantial amount of outstanding research that must be undertaken prior to the operational use of ASR systems.

In the training and simulation environment, the use of ASR has the potential of eliminating so called *pseudo pilots* or *blip drivers* who simulate aircraft controlled by the trainee. Speech recognition could be used to recognize clearances, which could be processed by the simulator to generate appropriate aircraft actions accompanied by simulated pilot acknowledgements using voice synthesis technology. The result would be an overall reduction in cost and complexity. In this case, recognition accuracy is of secondary importance, since recognition errors can be treated as simulated pilot errors. In the simulation environment one could insist that the controllers adhere to the prescribed syntax. Also, errors caused by human factors issues may be admissible due to the relatively benign nature of errors during simulations.

Hence, it is reasonable to assume that ASR technology will be introduced to the ATC training and simulation environment before it is used operationally. In fact, development of training suites using ASR technology is already underway [15, 16, 17, 18]. This should be considered a desirable development, since it will provide needed experience for the design of operational systems and will condition new controllers to the presence of ASR technology and possibly reduce the problem of syntax deviation.

1.3 The Future of Air Traffic Control

The research effort presented in this work is based on the ATC system in operation during the late 1980's. However, the introduction of ASR technology cannot be justified if its use is not warranted in the future ATC environment. The current National Airspace System (NAS) plan does not include a provision for the use of ASR equipment within its timeframe, from 1985 to the year 2000. It is likely that voice input technology will not be introduced within the next decade, and thus an attempt must be made to identify the fundamental differences between today's ATC system and that of the near future.

There is an ongoing trend to further automate ATC operations, especially in the terminal area. This is embodied by the current Advanced Automation System (AAS) program and the introduction of the Mode-S transponder, which will provide a digital datalink between ATC and participating aircraft. The result will be the introduction of new man-machine interface elements including high resolution color graphics, and new input modalities such as touch screens. Increased automation coupled with the Mode-S datalink and onboard Electronic Flight Information Systems (EFIS) will allow aircraft to accept four dimensional[1] clearances in the terminal area. Work has already been completed on the feasibility of providing such clearances as a function of dynamic constraints [19].

Hence, it appears that a trend exists towards increased automation, digital transmission of ATC commands directly to aircraft, and the use of other channels of communication than voice. The history of ATC demonstrates however, that care must be taken to introduce new technology in such a way that users employing old technology can still be serviced. One cannot expect all aircraft to instantaneously have Mode-S equipment installed. Nor will all aircraft have EFIS systems on board. Aircraft that are capable of taking advantage of advanced

[1] The four dimensions are as follows: latitude, longitude, altitude, and time.

automation services will occasionally experience equipment failures. Hence, verbal communication must be retained as an information channel. In light of this, it may be most practical to retain voice is the *primary* input modality, even when issuing automatically generated four dimensional clearances. This would allow for a smooth transition to cases where Mode-S is not available, and would require less modification of existing practices. Thus, a system can be envisioned where the controller is in constant dialog with the system, using voice input both to communicate with the aircraft and to control the automation tools being used.

1.4 History

The general history of Automatic Speech Recognition has already been thoroughly documented [20, 24], and need not be repeated here. However, the literature specifically covering speech recognition applied to ATC applications is much less extensive. A brief history of this field is presented below.

Although the first speaker dependent digit recognizer was devised in 1952, preliminary research dedicated to ATC applications did not commence until the mid-1970's. Early work was conducted by the U.S. Naval Training Equipment Center (NAVTRAEQUIPCEN) [21]. The first FAA sponsored research was concluded in 1979 by Donald W. Connolly [4]. That research effort, conducted at the FAA's National Aviation Facilities Experimental Center[2] (NAFEC), constituted an initial attempt to assess the performance of ASR technology given an ATC data entry task, and to determine how voice data entry compares with keyboard data entry. Several experiments were performed using isolated speech recognition and varying amounts of auditory feedback. Connolly concluded that applications of the technology to ATC should be given serious consideration, but that performance improvements should be awaited before adoption of any major upgrading of existing ATC systems.

During the early and mid-1980's, the possibility of using speech recognition technology in ATC simulators was investigated by the Human Interface Technology Group at Bracknell in the United Kingdom [6, 7]. It was recognized that the strict syntax used in ATC and the nonfatal penalty of failure during simulations made ATC training a suitable application for introducing ASR technology. The research showed that a small vocabulary speaker

[2] The predecessor of the FAA Technical Center.

dependent system would be sufficient, but that it must recognize continuous speech. Speech input and output equipment was adapted to a commercially available ATC simulator for evaluation purposes. The importance of designing feasible correction strategies was identified. It was concluded that the major limiting factor was the performance of the speech recognition technology.

It was not until the mid-1980's that major U.S. research efforts dedicated to civil ATC applications of ASR technology were initiated. During 1983-1987, several projects were undertaken within the FAA's Small Business Innovation Research (SBIR) program [14]:

- *Computer Voice and Speech Data Entry and Recognition*
  (Speech Systems, Inc., contract no. DTRS-57-87-00016): This study demonstrated the feasibility of continuous speech recognition by computers, evaluated how machines should recognize large human vocabularies and complex syntaxes, and researched the human factors involved in the speech input and machine feedback of the man-machine interaction.

- *Computer Aided Reasoning Technology*
  (University Faculty Associates, Inc., contract no. DTRS-57-C-0119/00103): This project intended to produce an Intelligent Tutoring System, including the use of ASR equipment, to assist training development of ATC personnel.

- *Expert Systems Applications for Air Traffic Control: A Feasibility Study*
  (University Faculty Associates, Inc., contract no. DTRS-57-84-C-00124): This research demonstrated the feasibility of using an expert system in the training of ATC controllers using ASR technology.

- *Speech Recognition in Air Traffic Control*
  (Emerson and Stern Associates, contract no. DTRS-57-85-00122): This research was aimed at producing a reliable, real-time speech recognition capability in the ATC environment. The grammar used was based on data obtained from 20 hours of audiotapes from a wide variety of ATC speakers and situations. The research suggested that the environmental challenges of ATC and its psychological or physical stresses can be overcome, while meeting the needs of robust operation.

Speech Systems, Inc. (SSI), and University Faculty Associates (UFA), Inc. are cooperating in an effort to develop an ATC training expert system, using SSI's ASR system [15, 16]. By using a large vocabulary, speaker independent speech recognition system, it is believed that problems related to enrollment templates, coarticulation, stress, and noise can be overcome.

An internally funded research effort by MITRE has investigated potential applications of ASR technology in the context of advanced ATC automation aids [8, 22]. Candidate functions which could benefit from the introduction of speech input and output technology have been identified as the following:

- Controller data entry.
- Controller training.
- Composition of flight advisories.
- Control of voice switching systems.
- Digital data link in the cockpit.

The research intended to estimate the operational suitability of applying ASR to ATC in terms of human factors issues such as controller workload and input/output modality compatibility. The study concluded that today's speech synthesis hardware is adequate, but the speech recognition hardware is not. Preliminary research should be conducted now however, while awaiting further improvement of ASR technology.

Research at academic institutions has been mostly in the form of Masters thesis work. In 1987, Thanassis Trikas of M.I.T.'s Flight Transportation Laboratory (FTL) completed a Masters thesis that demonstrated the feasibility of using off-the-shelf ASR equipment in conjunction with an existing ATC simulator [9]. Two speech recognition systems were evaluated, and problem areas such as speech variations were identified. A simple speech input interface was designed for use in a simulation environment, and basic operational environment issues were discussed. Another Masters thesis, completed in 1988 by Robert F. Hall at Arizona State University, evaluated an ASR system in an operational environment at Williams Air Force Base [5]. The possible use of Artificial Intelligence techniques were investigated, and potential postprocessing applications of captured ATC communications were identified. A verb centered language model was adopted, and used as a basis for the vocabulary and syntax definition.

The tangential issue of assessing the linguistic properties of the ATC language has been addressed in a recent paper completed by Steven Cushing [23]. This research effort constituted part of a larger investigation of the feasibility of using voice input/output technology in aviation. The general problems of natural language understanding, as well as specific ambiguities that occur in the ATC environment were investigated. Several accidents that have resulted from language confusions were described. A solution was proposed including a voice interface using an intelligent voice input component and an intelligent voice output component. The voice input component would have as subcomponents a voice word recognizer and a language filter, employing a lexicon, a grammar, and a knowledge base.

In addition to the research efforts described above, government sponsored work is being conducted at the NASA Langley Research Center and at the Department of Transportation's Transportation Systems Center. Furthermore, the U.S. Navy is developing an ATC training facility incorporating speech input/output technology in conjunction with Logicon, Inc., and the ITT Defense Communications Division. In France, the two companies Crouzet and Steria have recently been awarded contracts to develop a prototype version of an automatic voice recognition and synthesis system to be used in the training of air traffic controllers [18].

1.5 OUTLINE

The purpose of this research effort is to continue some of the work described above, especially the research already conducted at M.I.T.'s Flight Transportation Laboratory. However, the emphasis will be placed on the operational environment. The focus of this paper are the human factors problems involved, as these are believed to be the chief limiting factors.

In Chapter Two, a brief introduction is given to the speech recognition technology. The underlying techniques used to recognize speech by machine are discussed, and the state of the art of the technology is described. Parameters that classify the different types of speech recognizers are identified, as well as the levels of knowledge present in speech.

The preliminary discussion is continued in Chapter Three with a description of the ATC environment and its relevance to speech recognition. ATC language issues are described, including a discussion of the information content of the language. Human factors issues such as background noise and cognitive workload that have a direct impact on speech recognition performance are also discussed.

Chapter Four presents a method to adopt an analytical approach to the human factors issues of introducing ASR technology into the ATC environment. The differences in the man-machine interface in the case of ATC applications vs. classical ASR applications are discussed. The fundamental human factors concepts behind the design of the suggested man-machine interface are presented.

The results from evaluating two commercially available ASR systems are described in Chapter Five. Problems inherent in the current technology are identified. A technology demonstrator using an off-the-shelf ATC simulator is presented.

In Chapter Six, a model for a suitable interface centered around ASR technology to be used between the air traffic controller and the ATC system is presented. The use of mixed input modalities, automatic error detection and correction techniques, and adaptive training is discussed. An expert system used to process higher levels of knowledge such as semantics and pragmatics is described. The results of simulations using the man-machine interface model are used to demonstrate the feasibility of these techniques.

Finally, Chapter Seven contains a summary of the research presented in this paper, as well as suitable conclusions based on the achieved results. Recommendations for future research are presented, as well as guidelines for the developers of future ASR technology.

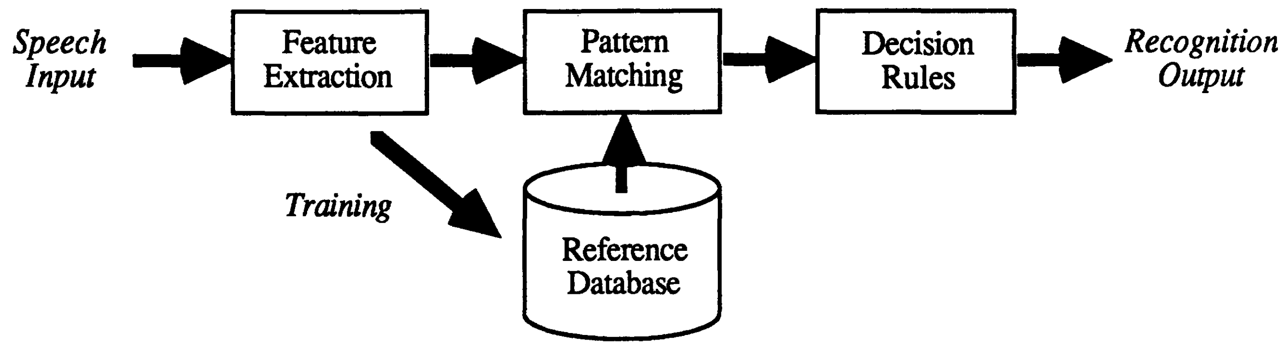
CHAPTER TWO

AUTOMATIC SPEECH RECOGNITION

2.1 THE TECHNOLOGY

All speech recognition devices are based on a three step process [24]. The first step is a data reduction and signal representation process (feature extraction). After this pre-processing, there is a pattern recognition process where the extracted features are compared to a database of prestored patterns. Finally, there is a set of decision rules based on scores obtained from the pattern matcher to decide which phrase was recognized. This process is described schematically in Figure 2.1. The process of training (also known as enrollment) can be represented as adding signal features of known phrases to the reference database.

Figure 2.1: The speech recognition process.



2.1.1 FEATURE EXTRACTION

The first step of the recognition process consists of pre-processing the raw speech input signal. Typically, the frequency band of interest is the range from 0 to 8 kHz, with most of the information present below 4 kHz. Thus, the sampling rate is typically 8-16 kHz. The resulting high data rates[3] require the use of data reduction techniques in order to represent the signal in a manageable form. The objective of these techniques is to reduce the amount of data

[3] If one byte (8 bits) of information is obtained for each sample with a sampling rate of 8 kHz, then the input data rate would be 64,000 baud. Hence, a couple of seconds of speech input would result in a sample equivalent to the size of this document.

necessary for processing, while retaining as much meaningful information about the speech signal as possible.

Several standard signal processing techniques exist to achieve this desired result. One of the simplest is filter bank analysis. A set of filters is used to determine the signal energy at a discrete number of frequencies. This takes advantage of the fact that most phonemes (a basic unit of speech) exhibit a set of fundamental frequencies known as *formants*. This filter technique attempts to identify which phonemes are present by assessing the signal energy distribution across the formants. Another spectral technique is to represent the signal by a set of Fourier coefficients.

An alternative to these spectral techniques is the so called *cepstral* analysis of speech, which is defined such that the Z-transform of the complex cepstrum is the logarithm of the Z-transform of the input signal [25]. Cepstral analysis can be used for pitch extraction and formant tracking. Thus, many speech recognizers represent the speech signal as a set of cepstral coefficients, computed at a fixed frame rate. Time derivatives of the cepstral coefficients have also been used. Some systems employ Mel-scale cepstral coefficients to mimic auditory processing. The Mel frequency scale is linear up to 1000 Hz, and logarithmic thereafter.

One of the most commonly adopted techniques is Linear Predictive Coding (LPC). An all-pole zero mean model of speech generation is assumed [26]. The model output (i.e. the speech signal) is represented as a linear combination of previous outputs. The speech transfer function consists simply of a pure gain **G** and a polynomial denominator with coefficients $\mathbf{a_k}$. The determination of the set of coefficients $\{\mathbf{a_k}\}$ reduces to a set of linear equations obtained by minimizing the total prediction error. The gain **G** is found through an energy conservation criterion. Using LPC coefficients to represent the speech input signal, the data rate can typically be reduced to 100 Hz.

Vector quantization is a technique commonly used to further reduce the amount of data required to represent the speech signal. Features from data samples obtained during training sessions are represented as vectors and stored in a codebook. Then, each new input vector is compared to these standardized reference frames, and is assigned to a class with similar features. This classification procedure allows the speech data to be represented as a set of indices into the reference codebook.

2.1.2 Pattern Matching

In the pattern matching phase, the extracted speech features of interest are classified into categories (e.g. words). There are two types of pattern matching problems:

- *Supervised:* Classes are known in advance, and samples of each class (i.e. training data) are available.
- *Unsupervised:* Classes are not known a priori, and must be inferred from the data.

Practically all current speech recognizers employ some form of supervised pattern matching, where a set of reference patterns have been obtained through the process of user training.

The first step in the pattern matching process is usually some method to achieve *time normalization.* This ensures that an attempt to match will not fail simply because the input data and the reference data are of different temporal lengths. A commonly used technique is *Dynamic Time Warping* [27]. First, the endpoints of the data sample are detected. Then, the isolated template is time distorted by a warping function in order to minimize the distance between it and the reference template. This procedure is then repeated for all reference templates.

There are several algorithms that can be adopted for the pattern matcher [28]. The simplest ones are based on *Bayes Decision Theory.* Given two classes $\mathbf{C_1}$ and $\mathbf{C_2}$, and a feature $\mathbf{x}$, then an intuitive decision rule would be to choose $\mathbf{C_1}$ if $\mathbf{P(C_1 \mid x)}$ is greater than $\mathbf{P(C_2 \mid x)}$; else choose $\mathbf{C_2}$. This approach can be generalized to a set of output classes. Another parametric approach based on probability theory is the *Gaussian Classifier.* It uses a cost function known as the *Mahalanobis Distance*, which is based on the feature vector, a mean vector, and a covariance matrix.

Non-parametric techniques exist that rely on geometric interpretations of the features. The most common of these is the *K Nearest Neighbor (KNN)* classifier. KNN assigns a feature $\mathbf{x}$ to the class most frequently represented among the $\mathbf{k}$ nearest tokens in the training set. First, the distances between $\mathbf{x}$ and *all* the training tokens are determined. Then, the $\mathbf{k}$ nearest neighbors are found, and $\mathbf{x}$ is classified by majority rule. When the amount of training data is infinite, KNN approaches the generalized Bayes Decision Theory classifier. The performance of KNN is dependent on the choice of **k** which is often chosen to be $\mathbf{a}\sqrt{\mathbf{n}}$ where **a** is a constant and **n** is the total number of training tokens.

One of the most successful recognition techniques used in speech recognition has been to model the speech using *Hidden Markov Models (HMM)* [29]. An HMM is an extension of a *Finite State Machine (FSM)*, in which states are associated with features and branches with probabilities of passing from one state to another. Typically, each recognizable word is represented by an HMM, which has been established through a set of a priori training tokens. When a new token **x** is recognized, the resulting word **w** is the one which has the highest probability $\mathbf{P(x \mid w)}$. This optimization is typically achieved by using a search procedure such as the Viterbi algorithm [30].

2.1.3 DECISION RULES

The result of the pattern matching phase is typically a set of scores which indicate how well the recognized token matched each reference token. Barring any post-processing such as syntax checking, the recognized phrase then consists of the words corresponding to the highest scores. Normally a selectable threshold value is used as a cut-off limit. If the highest score falls below the threshold, then the recognized token is rejected and no word is returned. Some algorithms, such as the Viterbi algorithm used in conjunction with HMM representations cannot determine scores for all reference tokens, but simply finds the highest scoring word. This is undesirable as the complete set of scores is often useful for post-processing such as examining the syntax and the semantic content of the entire recognized phrase.

2.2 CHARACTERISTICS OF SPEECH RECOGNIZERS

Specific criteria for selecting an ASR system depend on the application in which it is to be used. However, all ASR systems can be categorized by the following fundamental characteristics:

- Speaker dependent vs. speaker independent recognition.
- Isolated, connected, or continuous speech recognition.
- Vocabulary size.

Other parameters of significance to ATC applications include baseline recognition accuracy, design of training procedures, accessibility to the recognition software, noise robustness, sensitivity to variations in speech, recognition delay, and the availability of speech playback or speech synthesis. The following sections describe the characteristics of ASR systems in detail, and set criteria for selection of a system suitable for ATC applications.

2.2.1 Speaker Dependent vs. Speaker Independent Recognition

The ideal ASR system is speaker independent, that is, it will recognize any user's voice with no prior training. However, as significant differences exist in the quality of separate individuals' voices, it is difficult to achieve speaker independent recognition with reasonable recognition accuracy. Hence, most available systems are speaker dependent and require some training by each individual user prior to being used operationally. For ATC applications it would be desirable to use speaker independent ASR equipment to avoid user enrollment and to allow for flexible and quick replacement of a controller at any given station. Realistically however, it is more likely that a speaker dependent system will be used as speaker independence usually implies a reduction in recognition accuracy. Table 2.1 outlines the differences in characteristics between speaker dependent and speaker independent systems.

Table 2.1: Speaker dependent vs. speaker independent recognition.

Characteristic	Speaker Dependent Recognition	Speaker Independent Recognition
Convenience	Requires time consuming user training. Flexibility in changing users is reduced.	Requires no a priori user training.
Accuracy	Accuracy is higher due to available information on user's voice.	Accuracy suffers from lack of specific user data.
Robustness	Performance deteriorates as user's voice changes from training tokens.	Speaker independent recognition is robust to variations in speech.
Availability	Low cost speaker dependent systems are available today.	Very few speaker independent systems are available and the prices are relatively high.

2.2.2 Isolated, Connected, or Continuous Speech Recognition

The rate at which the user of an ASR system can speak depends on whether the system is capable of isolated, connected, or continuous speech recognition. In an *isolated* word recognition system, each spoken word must be preceded and succeeded by silence. Only single utterances can be recognized. In a *connected* speech recognition system words must also be separated by periods of silence. However, the recognizer is capable of processing sequences of words with periods of silence as short as 0.1 s between them. Hence, entire sentences can be recognized, provided that the user pauses briefly between words. A *continuous* speech recognition system requires no pauses, and the user can speak at a natural rate.

It has been established previously that the constraints on speech rate imposed by isolated and connected speech recognizers are unacceptable in ATC applications [7, 9]. Furthermore, this has become less of an issue than in the past, when the acceptable speech rate was limited by the available technology. Continuous speech recognizers do have some limitations, including higher costs, and relatively small vocabularies. Furthermore, with some continuous speech recognizers, output is only generated after the recognition of a full sentence. When a rejection error occurs, the whole sentence may be lost, instead of just a single word. Recognition accuracy may be lower for a continuous speech system than for an equivalent isolated or connected speech unit. Despite these drawbacks, ATC applications, whether in an operational or a simulation environment, require the use of continuous speech technology.

2.2.3 Vocabulary Size

As with the other characteristics of ASR systems described above, there is typically a trade-off between vocabulary size and performance measures such as recognition accuracy and processing speed. In a large vocabulary system, there is inherently a greater probability of misrecognition, and hence overall recognition accuracy can be expected to decrease with vocabulary size. Furthermore, processing time tends to increase with vocabulary size. Although there is no standard nomenclature, a system capable of recognizing 100 words or less can be considered a small vocabulary system, whereas systems capable of vocabularies with 1000 words or more can be considered large vocabulary systems. Units with vocabulary sizes between 100 and 1000 words are medium size vocabulary systems.

Previous research has demonstrated that vocabularies with less than 100 words are sufficient to implement most ATC commands [9]. However, an operational system is expected to require 300-500 words [4, 9]. If all possible aircraft callsigns are to be recognized, as well as uncommon ATC commands, the required vocabulary size could well exceed 1000 words. ATC applications are typically dominated by a few commands which occur the majority of the time, interspersed with a large number of other commands which occur only infrequently.

2.2.4 Baseline Recognition Accuracy

The *baseline recognition accuracy* of an ASR system refers to the rate of correct recognition prior to the application of automatic error correction techniques. Although such techniques can be introduced to improve the recognition accuracy, the baseline recognition accuracy imposes a limit on overall performance. Most producers of ASR equipment claim word recognition rates of 95% or greater. However, these figures are usually based on ideal conditions and are artificially high for marketing purposes. Furthermore, a word recognition rate of 95% implies a phrase recognition rate of 60% for a ten word sentence. Baseline recognition rates must be measured empirically in operational conditions before employing the equipment in question in an operational ATC environment. Word recognition rates of 98% and above are desirable.

2.2.5 Design of the Training Procedures

Classical training procedures typically consist of repeating every word in the vocabulary several times. For recognition systems that employ HMM techniques, a large number of training tokens are required. Therefore, as the vocabulary grows in size, several hours of training may be required. This is inconvenient and costly, and should therefore be avoided. Furthermore, if the user becomes bored during training, the voice patterns may assume a form different from those which could be expected in an operational setting. Techniques to reduce the amount of training include adaptive training during actual operation, and the use of another user's voice patterns as an underlying basis. The problem of lengthy training procedures is one of the major disadvantages of speaker dependent recognition systems.

2.2.6 Accessibility to the Recognition Software

When using an ASR system for research on speech input applications in ATC, it is desirable to have some control over the recognition software. This allows for more flexibility in creating an intelligent parser. In particular, most ASR systems report only the score for the best match when a word is recognized. It would be useful to obtain scores for *all* words in the vocabulary. These could be used to evaluate a different alternative in case the recognized phrase does not parse syntactically or semantically. Access to the extracted features and the ability to train during recognition are other software issues of interest. In general, the more open and accessible the recognition software is, the higher the effective recognition accuracy after post-processing of the recognized input.

2.2.7 Noise Robustness

The presence of background noise and its influence on recognition accuracy remains a challenging problem facing ASR technology. The problem is twofold [31]:

- As background noise levels increase, the signal-to-noise ratio of the speech input signal decreases, which results in recognition errors. Peaks in the background noise, especially when non-random, may be recognized as words.
- As background noise levels increase, the signal-to-noise ratio in the user's auditory feedback decreases, which causes a speech variation which in turn may result in recognition errors.

An additional problem is posed by variable (vs. constant) noise levels. Techniques to increase robustness to background noise include the use of noise cancelling microphones, isolating headsets with amplified feedback, inclusion of training tokens with background noise present, and the adoption of speech enhancing pre-processors.

2.2.8 Sensitivity to Variations in Speech

Background noise has been mentioned above as one of the factors which may alter the user's voice. Other causes include fatigue, stress, boredom, colds, temporal drift, user idiosyncrasies, changes in intonation, and microphone placement. Furthermore, it appears that speech recognizers work better with some people than others. Users that obtain high recognition accuracies are often termed *sheep*, whereas those with consistent problems are termed *goats* [32]. The so called *training effect* is the problem of users speaking more slowly and carefully during training than during operational usage. Another speech variation problem is that of *coarticulation*, the change in pronunciation of words due to the presence of its neighbors. For example, the fricative sound "s" at the end of the word "this" all but disappears in the compound "this ship", when part of a full sentence spoken at a natural rate.

The lack of robustness to variations in speech is one of the greatest limitations of current speech recognition technology. The problem is of special concern to ATC applications where periods of high stress or boredom are frequent. A promising technique seems to be adaptive or on-the-fly training. By continuously updating the reference database with correctly recognized phrases, variations in speech are reflected in the training data. However, some speech variations may occur so rapidly that adaptive training will not cause a sufficiently quick response. The problem of coarticulation, which is particularly serious in the case of isolated and connected speech recognizers, has been mostly overcome by continuous speech recognizers using HMM representations. Other speech variation problems, such as inter-speaker variations, remain largely unsolved, although the constantly improving performance of speaker independent recognizers offers some hope.

2.2.9 Recognition Delay

The term *recognition delay* refers to the time delay between the completion of an utterance and the time recognition output is produced by the ASR unit. The delay is a function of utterance length, vocabulary size and complexity, hardware capability, and software complexity. Excessive recognition delays are unacceptable, as they prohibit the user from obtaining feedback and slow down the application. For ATC applications, recognition delays of 1-4 s have been recommended [9]. However, in an operational setting, employing extensive post-processing techniques of the recognition output, a recognition delay of less than 0.5 s is desired.

2.2.10 Speech Playback and Speech Synthesis

The research in this paper is focused on the recognition of speech. However, in some of the potential ATC applications of ASR technology a speech output capability is also desired. This is particularly true for ATC training and simulation environments, where speech output is required to simulate aircraft acknowledgements and readbacks. Speech output may also be desired for auditory feedback purposes in operational applications.

Several speech recognition units are delivered with the ability to generate speech output. Two types of speech output generators exist:

- *Speech playback:* Actual voices of speakers are recorded digitally and then reproduced. This offers the most natural sounding pronunciation of individual words, but flexibility is reduced, and sentence intonation may be unnatural if words are used in a sequence other than initially recorded.

- *Speech synthesis:* Speech is synthesized through a procedure similar to the inverse of the feature extraction and data reduction procedures described previously. The resulting voice typically exhibits a robotic sound quality, but speech synthesis offers the capability of generating unforeseen sentences.

The speech synthesis technology is sufficiently advanced that it can be employed using programs executed on personal computers, requiring no dedicated hardware. However, such software driven systems are usually computationally intensive and prohibit the processor from handling other tasks. For this reason, dedicated auxiliary hardware is usually desired. In general however, it is recognized that the speech synthesis technology is more mature than the speech recognition technology [3].

Because many ATC applications would benefit from the use of speech output as well as speech input, it is desirable to select an ASR system capable of generating speech output. Speech synthesis offers the greatest flexibility, as sound quality is becoming more and more natural sounding. Although it is possible to use separate equipment to handle the speech generation, this is usually less desirable as it complicates issues such as handshaking between voice input and output, and usually degrades portability and efficiency.

2.2.11 MICROPHONE ISSUES

When selecting an ASR system one must also choose a suitable microphone for the application. The choice of microphone has a marked influence on performance, convenience, and user acceptability. Characteristics of microphones that must be selected include microphone element type, directionality, and mounting [33]. In ATC applications, a noise cancelling element may be desired, although these are usually sensitive to microphone placement. The microphone should be mounted on a headset similar or superior to those in use by controllers today. Mobility and comfort must be ensured. A push-to-talk (PTT) switch should be included, as currently used, so that recognition will only occur when desired.

2.3 SOURCES OF KNOWLEDGE IN SPEECH

Uncertainties and ambiguities resulting from the speech recognition process can often be resolved by subjecting the conflicting word hypothesis to post-processing. This is similar to the way humans recognize and understand speech. This process takes advantage of several levels of knowledge available to the recognizer [20, 34, 35]:

- *Acoustic analysis:* Feature extraction from the speech input signal.
- *Phonetic analysis:* Identifying basic units of speech such as vowels, consonants, and phonemes.
- *Prosodic analysis:* Using intonation, rhythm, or stress to identify linguistic structures.
- *Lexical analysis:* Matching words by comparing sequences of extracted features with reference templates.
- *Syntactic analysis:* Applying constraints specified by a predefined syntax.
- *Semantic analysis:* Testing the meaningfulness of the recognized phrase.
- *Pragmatic analysis:* Predicting likely future words based on the past and on the state of the system.

The first four steps compose the actual speech *recognition*, whereas the last three perform the speech *understanding* function. As such, the first four steps are usually incorporated in the ASR hardware, whereas the last three are left to the designer to implement. In practice however, little, if any, prosodic analysis is performed. Furthermore, it is becoming increasingly common to have some of the syntactic analysis be performed by the ASR unit itself.

Processing these levels of knowledge consists of several phases, including acquisition, representation, and implementation. The problem of processing the lower levels of knowledge, such as acoustics, phonetics, and lexical analysis, is well understood and is performed adequately by most ASR systems. Methods to acquire prosodic information are much less common, however. Although it is clear that the intonation, stress, and changes in rhythm of the controller's voice contain valuable information for the pilot, most ASR systems work best when the speech pattern is void of such variations. In the case of the higher levels of knowledge, implementation of syntactic processing is common, whereas semantic and pragmatic analysis is well understood, but rarely implemented. Since these higher sources of knowledge must usually be implemented by the designer, more detailed examination is warranted [34].

2.3.1 Syntactic Analysis

The use of syntax involves defining a grammar which can be used to parse sentences. The prime desired qualities of a syntax are *simplicity*, *generality* (ability to parse all acceptable sentences), and *specificity* (the ability to diagnose and reject all unintelligible sentences). A syntax rule consists of *non-terminals* (possible phrase types), *terminals* (possible words), *starting symbols* (patterns designating complete sentences), and rules relating these components. The uses of syntactic knowledge include *recognition*, *prediction*, *enumeration* (generation of a set of alternate, specific predictions), and *postdiction* (testing a prediction with additional data). Syntax can be implemented by using Finite State Networks, Augmented Transition Networks (ATNs), or simple heuristics.

2.3.2 Semantic Analysis

Semantic knowledge generally implies defining meaning by relating logical expressions to world models. Semantic knowledge in an ASR system can eliminate word sequence hypotheses that parse syntactically, but are not actually meaningful. There are empirical methods as well as formalized methods (such as Montague's system) of acquiring semantic knowledge. Semantics can be embedded in the syntax or can be implemented separately, using similar techniques (e.g. ATNs). If the syntax contains semantic notions, which is often the case in ATC applications, it is known as a *semantic grammar.*

2.3.3 Pragmatic Analysis

Pragmatic knowledge provides the capability for understanding a sentence in the context of the situation at hand. In ATC applications, the situation is described by the state of the airspace, that is, the state of each aircraft and the relative positions between aircraft. Acquiring pragmatic knowledge consists of storing context sensitive information from previous statements and deriving the speaker's beliefs and disbeliefs concerning the system. Pragmatics should be used on a rule-of-thumb basis, and should not be regarded as always correct. Possible implementations include knowledge based rules, Finite State Networks, and knowledge state databases.

2.4 Recognition Errors

A speech recognition system can yield one of the following results upon detecting a token[4] [35]:

- *Correct recognition:* A legal token is recognized as the correct legal word or sentence.
- *Correct rejection:* An illegal token is correctly rejected as not being a legal word or sentence.
- *Rejection error:* A legal token is not recognized as any legal word or sentence.
- *Substitution error:* A legal token is recognized as another legal word or sentence.
- *Insertion error:* An illegal token is recognized as a legal word or sentence (this includes recognition of non-speech tokens such as noise).

[4] Normally, a spoken instance of a word. For a continuous speech recognizer that is only capable of recognizing whole sentences, a token refers to a spoken instance of a sentence.

The first two of these possible results are desired (correct recognition and correct rejection), whereas the last three are errors. Since we can never guarantee the complete absence of these errors, their characteristics and implications to ATC applications must be understood.

While not one of the most frequent error types, the rejection error implies a loss of information. In the case of a speech recognition system that only recognizes full sentences, this is a particularly severe error as entire ATC commands may be lost. Substitution errors on the other hand are typically the most frequent errors, but are relatively easy to correct. If only one or two words in a sentence are substituted, the correct words can usually be determined through application of higher levels of knowledge such as syntax, semantics, or pragmatics. Insertion errors are rare, especially when noise cancelling equipment is used. When they do occur, they can usually be eliminated through post-processing. ASR systems that only recognize whole sentences are particularly unlikely to suffer from insertion errors.

2.5 Current State of ASR Technology

Due to the lack of standardized performance measures, it is difficult to evaluate the current state of the technology, except through empirical methods. Current performance assessment methods have shortcomings and tend to overemphasize baseline recognition accuracy [36]. Manufacturers invariably claim recognition rates of 95% and higher. Usually, these results are achievable, but only in ideal conditions. Laboratory benchmark tests are useful for comparing ASR equipment, but are not efficient for predicting actual performance in operational systems [37]. Nonetheless, some conclusions can be drawn regarding the state of the art of technology [38].

Word recognition rates as high as 99.9% have been reported, and can certainly be achieved under certain conditions. Systems used operationally however, can be expected to achieve word recognition accuracies of 90-95% under degraded conditions. Recognition delays are reasonable, usually on the order of 1 s or less. Most continuous speech recognition systems available today are speaker dependent, although a few speaker independent system do exist. Typical prices are $2,000 and below for a small vocabulary recognizer, $10,000-$15,000 for a medium size vocabulary recognizer, and $45,000 and above for large vocabulary systems. Many commercially available systems are delivered with either speech playback or speech synthesis capability. Noise cancelling microphones are usually used.

In summary, current ASR recognizers are useful for preliminary research and simple training and simulation applications in the ATC environment. Limitations include lack of robustness to speech variations, sensitivity to background noise, sensitivity to microphone placement, vocabulary size/cost ratio, lengthy training procedures for speaker dependent systems, lack of cost effective speaker independent systems, and black box software architecture. Speech output equipment suffer fewer limitations, although more natural quality of speech is still required for speech synthesizers.

CHAPTER THREE

THE ATC ENVIRONMENT

3.1 THE PHYSICAL ENVIRONMENT

The U.S. Air Traffic Control system features several different environments where ASR technology may be used. Each physical environment has unique implications on the use of speech technology. Hence, it is necessary to examine those facilities which may benefit from the ability of capturing the information transmitted to the aircraft. These include the entities that provide clearances and vectors, that is, the towers, the Terminal Radar Control facilities (TRACONs), and the Air Route Traffic Control Centers (ARTCCs). Variations in the physical characteristics include background noise, controller mobility, number of active controllers, equipment used, and frequency of manual input.

3.1.1 THE TOWER ENVIRONMENT

Towers exist at many, but not all, U.S. airports. The need for a tower is usually dictated by the amount of traffic the airport handles. At smaller airports, towers may not be open twenty-four hours. The purpose of the tower is to provide aircraft with clearance deliveries, ground control, and local control over departures and arrivals. Towers at small airports may only have a few controllers, usually at least three. In this minimum case, one controller is dedicated to clearance deliveries, one to ground control, and one to local control. At busier airports, the tower may be staffed with more people, including supervisors, a controller dedicated to helicopter control, etc.

The tower cab is typically located at the top of a structure, providing a birds eye view over most if not all of the airport. The controllers rely heavily on visual acquisition of the aircraft under their control, particularly in the case of ground control. Local control relies more on radar, combined with vision to determine if aircraft have landed, departed, etc. Despite the availability of radar information, there is little automation present in the tower environment, and controllers rarely make manual inputs to the system. Because of this dependence on viewing the outside world, mobility is required in the tower environment, as controllers move about to visually identify aircraft. Their freedom of travel is limited by the physical connection between the headset and the microphone inputs, but they seldom sit down.

Tower cabs are typically heavily insulated with absorbing materials to dampen both noise from the outside and noise generated within the cab itself. Nonetheless, background noise exists, both in the form of aircraft engines and controller voices. Rotating radar transmitters mounted on the tower cab roof also contribute to the background noise. Equipment noise generated by fans and electrical equipment poses less of a problem.

As tower controllers often provide secondary information regarding activities on the runway (such as the presence of animals, temporary holds for snow removal etc.), the syntax used is much less constrained than in the TRACONs and the ARTCCs. Unexpected events occur frequently at airports, and hence the controllers' work is very tactical in nature. Pre-defined grammars are not well suited for formulating tactical solutions, as they often represent unique cases. Hence many transmissions are in natural language form, or modifications of the prescribed syntax, making the use of speech recognition technology less feasible.

3.1.2 THE TRACON ENVIRONMENT

TRACONs provide radar services to departures and arrivals at airports. Usually the TRACON is located in the same building as the airport's tower facility, although this is not always the case. Controller functions include managers, supervisors, radar controllers, and other controllers that handle flight data strips and other functions. Each radar controller is responsible for a subset of the controlled airspace. Divisions are usually made geographically (i.e. north, west, east, south), or by type of traffic (i.e. arrivals or departures), or by both (e.g. west departures). Other controllers may be responsible for coordination with satellite airports, or provide final control.

TRACON rooms are noise insulated, just like tower cabs, but typically have no windows to the outside. Radar controllers sit in a row along a bank of equipment. Hence, the background noise in the TRACON environment consists mostly of neighboring controllers' voices. There is some movement in TRACONs, typically by managers, supervisors, and controllers that are beginning or ending shifts. In general however, mobility is not a requirement for the TRACON radar controller.

The TRACON controllers rely almost exclusively on primary and secondary radar information. In major TRACON facilities, this information is presented in processed form. The most functional system available is the Automated Radar Terminal System (ARTS), of which there are several variants. The most sophisticated of these is ARTS-IIIA which detects, tracks, and predicts primary as well as secondary radar derived aircraft targets. These are displayed in symbolic form with additional alphanumeric information, depicting flight identification, aircraft altitude, ground speed, and flight plan data [12]. The controller communicates with the ARTS computer through the use of a keyboard and a trackball located on the console.

TRACON controllers perform tactical as well as strategic work. As a result, the adherence to syntax is stricter than in the tower environment. The terminal areas are frequently congested and require improvements in safety and efficiency. This, combined with the relative frequency of manual inputs, make the TRACON environment a suitable candidate for voice input technology.

3.1.3 The ARTCC Environment

The primary purpose of the ARTCCs is to provide aircraft separation during the en route phase. ARTCC facilities are located at twenty centers in the continental U.S.[5] Each center is subdivided into a number of low and high altitude sectors. Each sector is typically controlled by two or three controllers providing radar control, handoff control, and other services such as flight data strip handling. Each sector team has its own work station complete with a radar console, a flight strip bay, and a flight strip printer. The radar console has a number of keyboards and a trackball that allow the controller to communicate with the system.

An ARTCC room consists of several banks of sector work stations. Often, adjacent work stations correspond to adjacent sectors in the airspace. The ambient conditions are similar to those found in TRACONs in that there are no windows, there is ample noise insulation, and most of the background noise consists of distinct voices. Except for managers and supervisors, ARTCC controllers remain seated.

[5] In addition there is an ARTCC in both Hawaii and Puerto Rico.

Compared to the TRACON and tower environments, much of the work done in the ARTCCs is strategic, especially that which is related to flow control. The syntax used is therefore more constrained than in other ATC environments. Manual inputs are made relatively frequently. This makes ARTCCs more amenable to ASR technology than towers or TRACONs. However, en route sectors tend to be less congested than terminal areas and hence the potential improvements in safety and efficiency due to the introduction of voice input equipment are less significant. This may change if sector sizes are increased in the future.

3.2 COGNITIVE WORKLOAD

As has been described above, physical factors that may have an influence on the performance of ASR equipment in the ATC environment include the level and nature of background noise, controller mobility, and the use of manual input modalities. However, as increases in air traffic are causing the terminal areas to become more and more congested, cognitive workload has become a critical issue. With the introduction of ARTS the controllers have been provided with more advanced detection, tracking, and prediction functions. At the same time however, as more information has been made available to the controller, information overload has become a relevant problem [39].

It should be noted that the design of airspace sectors is largely influenced by the cognitive workload levels a sector team is considered able to handle. Factors contributing to controller workload include [40]:

- *Traffic variables:* Average traffic volume, peak instantaneous aircraft count, traffic type and mix.
- *Geometric variables:* Sector size, airway geometry, sector flight time, altitudes involved.
- *Sector type:* High altitude en route, low altitude en route, transition, terminal.
- *Coordination and interaction considerations:* Nearby terminals, sector coordination, activity of adjoining sectors, presence of satellite airports.
- *Sector team control procedures:* Control position organization, number of members in the controller team.
- *Technology aspects:* Amount of information presented, frequency of interaction with system, type of input/output modality, allocation of resources.

Various measurements of workload include airspace and traffic parameters, frequency and type of controller actions, allocation of resources, subjective evaluations, physiological measurements, social factors, psychological tests, medical data, and incident accounts [41].

Workload is often described as having two components. The first, *taskload*, refers to all the demands that the system and man-machine interface place on the controller. The second, ***workload***, is the amount of effort the controller invests to achieve a desired level of performance with the given taskload. An experiment based on subjective evaluation techniques was conducted at the FAA Technical Center to investigate the effects of taskload on the workload perceived by the controllers [39]. In addition to self-reported measurements of workload, observers were used to provide user independent estimates. Airspace and system activity parameters were also sampled. Taskload, the independent variable, was controlled by changing the average aircraft count and by activating and deactivating restricted areas. All measures indicated that workload changed significantly with increased taskload. Furthermore, an inverse relationship between effectiveness and workload was found.

The introduction of ASR technology may have an influence on the taskload, if it requires frequent intervention by the controller to detect and correct errors. An increase in taskload would then result in an increase in perceived workload, and a decrease in effectiveness. Alternatively, if the technology is implemented in such a way that it requires little user action, it may reduce workload and increase safety by providing useful automation tools to the controller. It is not possible to determine with certainty what the effect of ASR technology on controller workload will be. Careful system design and extensive evaluation are required to ensure that information overload will not be the ultimate result of introducing speech recognition technology into the ATC system.

The implications of the controller workload issue on the introduction of speech recognition equipment are clear. Cognitive workload in the ATC environment has reached a critical level. If the introduction of ASR systems has the net effect of increasing the controllers' workload further, they will not be accepted. The technology must be sufficiently advanced that baseline recognition accuracy is nearly perfect, even when speech is degraded by background noise or speech variations. The technology should be as transparent as possible to the controllers. Feedback to the user must be implemented in such a way that it does not provide another task for the controller to monitor, but rather induces a sense of reduced levels of stress. The success of ASR technology depends on its ability to reduce overall cognitive workload, not increase it.

3.3 THE ATC LANGUAGE

The set of words and grammar rules that a controller uses when directing aircraft can be thought of as a language in its own right. Although it is essentially a subset of English[6], it has its own syntax and a specialized vocabulary. A native speaker of English that is not familiar with the ATC language would most likely have difficulties understanding a controller. The ATC language was designed for radio communications which typically implies degraded communication channels. Hence, it was designed to be unambiguous, clear, and simple. The language is defined in what is known as the "air traffic controller's handbook" or simply "the 7110"[7] [12]. In addition, the pilot's perspective of the language is given in the *Airman's Information Manual*, usually referred to as "the AIM" [13].

The linguistic characteristics of the ATC language are key to the potential success of introducing ASR technology into the ATC system. Hence, the vocabulary, syntax, and information content of the language must be examined. In addition, it is necessary to determine to what extent controllers adhere to the syntax prescribed in the 7110.

3.3.1 THE ATC VOCABULARY AND SYNTAX

The air traffic controller's handbook describes all phases of air traffic control, with definitions of the appropriate syntax. Examples are usually supplied to demonstrate how the syntax should be used in practice. Figure 3.1 shows a simplified excerpt from the handbook. Upper case letters are used to indicate fixed words, whereas lower case letters within parentheses indicate variables that the controller must supply. A slash (/) indicates that there is a set of alternate words. Note that the aircraft callsign is an implied variable preceding the entire phrase. Elsewhere in the handbook are definitions for the callsign syntax and the altitude syntax. Combined, these subsyntaxes form the entire syntax for the altitude assignment command. The ATC grammar is the union of the syntaxes for all the commands defined in the controller's handbook. The vocabulary then is simply the set of all words present in the syntax definitions.

[6] In the U.S. ATC system, English is used exclusively. In other countries, English is used as the international language of ATC, in conjunction with the local language.
[7] After its FAA document number. At the time of writing, the current version is 7110.65F.

Figure 3.1: The altitude assignment syntax.

```
Phraseology:
CLIMB/DESCEND AND MAINTAIN (altitude).
Example:
"United Four Seventeen, descend and maintain six thousand."
```

In order to use speech recognition technology, the vocabulary and syntax used in the application at hand must be represented in a formal notation that can be processed by the recognition software. Many different notations exist – the one chosen for this paper is the syntax notation developed by Verbex, Inc. for their family of speech recognizers. This choice is an arbitrary, yet convenient one, as a Verbex speech recognizer was used for this particular research effort. This notation provided a simple, yet flexible and powerful means of representing the ATC syntax. Figure 3.2 contains a possible representation for the altitude assignment command using the Verbex notation. Lower case words are fixed words, whereas upper case words prefixed with a period represent substructures. The definition for the climb/descend substructure is also displayed.

Figure 3.2: The altitude assignment syntax in formal notation.

```
.CALLSIGN .CLIMB-DESCEND and maintain .ALTITUDE
.CLIMB-DESCEND=
     climb
     descend
```

Due to the limitations of the speech recognition devices available today only a subset of the 7110 can be implemented. This constraint may have to be overcome by the time the technology is introduced operationally. However, since a set of few commands constitute the majority of the radio traffic between controllers and participating aircraft, it is feasible to use only a subset of the entire grammar. In particular, the commands of significance are those that imply a change in the airspace state. The prime objective of ATC, ensuring traffic separation, is dependent on the positions and velocities of the aircraft in the sector. Thus, our primary concern lies with the commands that change altitude, heading, and airspeed. The altitude assignment syntax was described in Figures 3.1 and 3.2. Figure 3.3 contains a description of the heading and airspeed assignment commands.

Figure 3.3: The heading and airspeed assignment syntax.

```
Phraseology:
TURN LEFT/RIGHT HEADING (degrees).
FLY HEADING (degrees).
TURN (number of degrees) DEGREES LEFT/RIGHT.

Phraseology:
INCREASE/REDUCE SPEED: TO (specified speed in knots).
INCREASE/REDUCE SPEED (number of knots) KNOTS.
```

The syntax actually used contained most of these commands, as well as a number of additional commands to increase the realism of the simulation (see Appendix A).

3.3.2 INFORMATION CONTENT OF THE ATC LANGUAGE

The level of difficulty associated with adopting a grammar and a vocabulary for use with a speech recognizer depends on the size of the vocabulary and the complexity (or branching factor) of the grammar. The complexity of a language is defined as $2^{\mathbf{H(w)}}$ where $\mathbf{H(w)}$ is the average *information content* of the set $\mathbf{W} = \{\mathbf{w}_1,\ldots,\mathbf{w}_n\}$. In our case $\mathbf{W}$ represents the ATC language, and $\mathbf{w}_i$ the individual words in the vocabulary. The average information content in turn is defined by $\mathbf{H(w)} = -\sum \mathbf{P(w_i)} \log_2 \mathbf{P(w_i)}$. The greater the complexity of the language is, the more difficult the recognition process becomes.

An intuitive understanding of these information theory concepts may be achieved by considering two extreme cases. First, consider a language consisting of the single phrase "the quick brown fox jumps over the lazy dog." In this case, the occurrence of each word is known with certainty, and hence $\mathbf{P(w_i)} = 1$ for all $\mathbf{i}$, resulting in an average information content given by $\mathbf{H(w)} = 0$ bits/word. The speech recognition process in this case is trivial. No matter what phrase is uttered, the phrase "the quick brown fox jumps over the lazy dog" should be recognized. Now, consider the natural English language. In this case, there is considerable uncertainty in determining which word will occur next in a sentence. Hence, $\mathbf{P(w_i)}$ is less than unity, and the information content is non-zero. As can be imagined, natural languages constitute one of the most difficult recognition tasks.

The ATC language has a branching factor somewhere in between that of our fictitious one phrase language and natural language. An empirical measure of the information content may be obtained with help of the *Shannon game*. In this game a sentence is chosen from the syntax of interest, and a player familiar with the language is asked to derive the sentence by guessing each word or letter. The player continues guessing until the correct word or letter has been found. The more constrained the language is, the fewer guesses the player will need. The information content is then approximated by $\mathbf{H(w)} = -\sum \mathbf{Q(j)} \log_2 \mathbf{Q(j)}$, where $\mathbf{Q(j)}$ is the fraction of words or letters for which $\mathbf{j}$ guesses where required. It is more practical to use letters instead of words since it gives the player a smaller set of possibilities to choose from.

A simple experiment was carried out where the Shannon game was used to obtain a comparative information content measure of the ATC language vs. natural language. Table 3.1 depicts the results of the game. The information content obtained from the natural language example, 1.44 bits/letter, compared favorably with a result obtained from the literature [42]. The results indicate that the ATC language is much less complex than natural language. Recall that the branching factor itself is defined as an exponential of the information content. Hence, a threefold difference in information content implies an eightfold difference in branching factor. The implication is that the ATC language is more constrained than natural language, and hence a more suitable candidate for speech recognition technology. If a subset of the ATC language is used, and pragmatic constraints added, the overall difficulty of the recognition task can be reduced to a manageable level. Speech recognition of natural language however, is currently not possible.

Table 3.1: Results from the Shannon game applied to the ATC language.

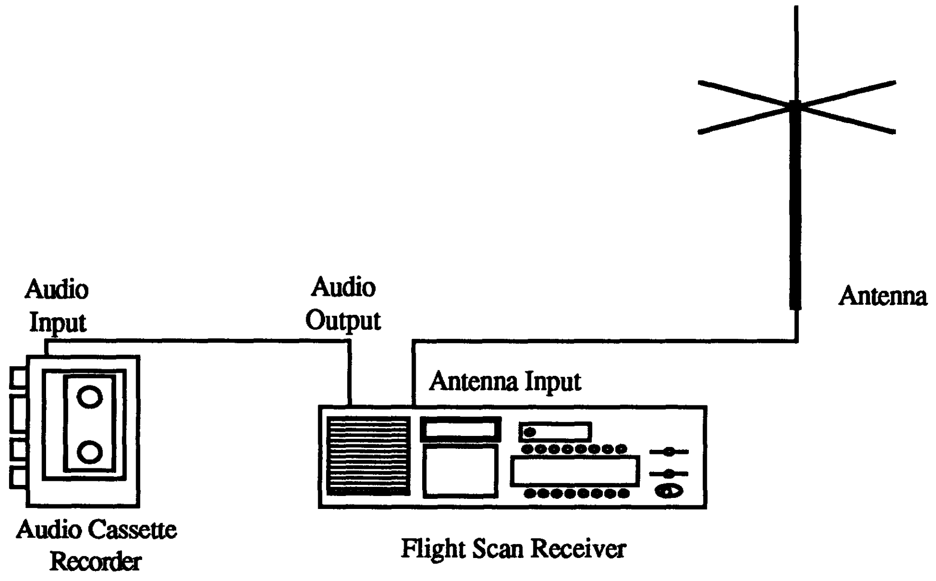
Language	Phrase	Information content
ATC	"Traffic alert ten o'clock altitude unknown advise you turn right."	0.580 bits/letter
ATC	"Descend at pilot's discretion maintain six thousand."	0.553 bits/letter
Natural	"I have not had a chance to discuss it with her yet."	1.442 bits/letter

3.3.3 The Syntax Deviation Problem

The feasibility of introducing speech recognition technology into ATC depends on the extent to which controllers adhere to the grammar specified by the FAA. Most implementations of speech recognition technology assume a rigid syntax. Recognition is greatly facilitated by the constraints imposed by a grammar. As has already been discussed, the less complex a grammar is, the more certainty there is about the occurrence of a word. The task of making a machine understand natural language is a difficult one which has not been solved with today's level of technology. As air traffic controllers deviate from the prescribed syntax and their speech begins to approximate natural language, the recognition process becomes more difficult. Hence there exists a need to examine to what extent today's controllers deviate from the ATC syntax and vocabulary.

To achieve this, a receiver station was set up in the Flight Transportation Laboratory. Figure 3.4 shows the basic elements of this station. Several hours of radio traffic between Boston Approach and Departure controllers and aircraft pilots were recorded. The recordings included different controllers as well as varying traffic and weather conditions. An attempt was made to transcribe the recordings, to monitor the extent of the syntax deviation problem.

Figure 3.4: The receiver station.



As VHF transmissions are line-of-sight, reception of Boston Approach and Departure was poor. The degraded quality of the recordings made it impossible to obtain meaningful statistics on syntax deviation. However, it was clear that the syntax deviation problem is highly dependent on the individual style of each controller. With some controllers, only 5% or less of all transmissions deviated from the ATC syntax. With other controllers however, 25% or more of all transmissions exhibited some form of syntax deviation. As only TRACON controllers were monitored, no experimental results were obtained from other ATC environments. Table 3.2 depicts some examples of transmissions that did not adhere to the prescribed syntax. Also included is an example of a hesitation, which would most likely cause a recognition error if voice input equipment was being used. As a sidenote, it was evident from monitoring the transmissions that air traffic controllers are much more careful about using correct vocabulary and syntax than the pilots involved.

Table 3.2: Examples of syntax deviations.

Actual transmission	Correct syntax
Two eighty three what's your current altitude?	Two eighty three say altitude.
One four three make that seven thousand five hundred.	One four three climb and maintain seven thousand five hundred.
Two sixty one, Boston Departure, contact.	Two sixty one, Boston Departure, radar contact.
Cherokee one bravo hotel contact Boston Approach on...uhhhhh...one two three point five.	Cherokee one bravo hotel contact Boston Approach on one two three point five.

There are several approaches to the solution of the syntax deviation problem. If ASR technology was introduced, the FAA could simply insist that the controllers adhere to the syntax. This is undesirable however, as it imposes an additional constraint on the controller, who may not even be able to change his or her individual style even if asked to. If speech recognition technology is introduced in controller training before it is being used operationally, as is likely to be the case, new controllers may begin to deviate less from the syntax. However, there would be no guarantee that they would always adhere to the syntax, and it ignores the generations of controllers trained without using ASR technology.

An analytic approach to dealing with the syntax deviation problem is the *case frame* representation of speech [35]. A traditional syntax is represented by a tree structure or a Finite State Machine. A sentence may fail to parse the syntax simply because a word is missing, or a word has been inserted, or parts of the sentence have been reversed. If case frames are used, the parser attempts to extract the meaningful information from the sentence, no matter where it is located, and all superfluous information is disregarded. A case frame consists of a *head concept* and a set of subsidiary concepts, *cases*. The header is a set of patterns which, if matched, indicate that a possible instance of that case frame has been entered. Then, the rest of the case frame is processed. Figure 3.5 depicts a possible case frame definition for the heading assignment command. This representation would parse all the various transmissions listed in the example, including those which do *not* adhere to the prescribed syntax.

Figure 3.5: A case frame representation of the heading assignment command.

Case frame definition:

```
[case frame:
     head concept: TURN
          [cases:
               callsign:
               direction:
               heading:
          ]
]
```

Example:

"four five two turn left heading one four zero"

```
[case frame:
     head concept: TURN
          [cases:
               callsign: 452
               direction: left
               heading: 140
          ]
]
```

Examples of syntax deviations:

"four five two make that a left turn heading one four zero"

"four five two could you turn left for me now, to heading one four zero"

"and a left turn heading one four zero for four five two"

"four five two...uhhhhh...turn left heading...uhhhhh...one four zero"

CHAPTER FOUR

THE HUMAN FACTORS PERSPECTIVE

4.1 THE SIGNIFICANCE OF HUMAN FACTORS

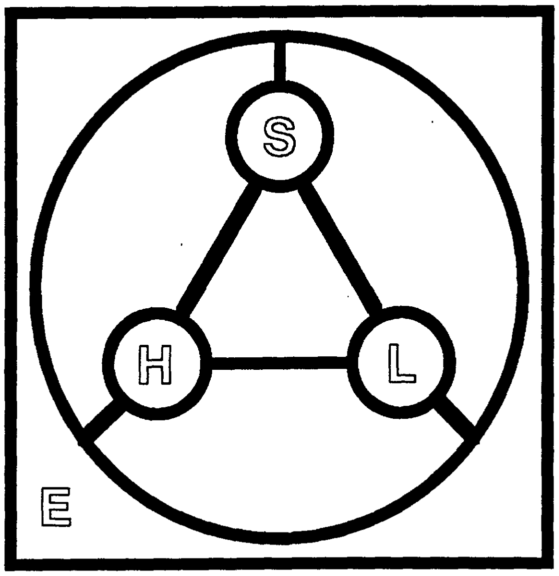
The introduction of ASR technology into the ATC environment has the potential to reduce human errors, resulting in increased system safety. However, the dilemma of ASR is that its purported advantages are not automatically realized by simply making the technology available. Careful human factors design is necessary to capitalize on its potential [43]. Automatic Speech Recognition is meant to provide a communication channel between the user and the system in question. Thus, by its very nature, speech recognition is inherently a part of the man-machine interface, and hence has an effect on both the operator and the system. At the very least, the effect of the technology on the operator's performance must be examined.

In the case of ATC, man-machine interface problems are exasperated by the fact that ATC is already plagued by human factors issues such as intense levels of workload during traffic peaks intermixed with controller boredom during low demand periods. The technology can have the effect of both worsening existing human factors problems and creating new ones. If these problems are not solved, it is unlikely that speech recognition technology will be accepted for operational applications. Furthermore, the possibility of loss of lives in the case of errors makes it imperative that the human factors problems created by introducing ASR into the Air Traffic Control system are properly addressed and solved.

4.2 MODELING HUMAN FACTORS

In order to approach human factors in an analytic way, a conceptual model of the system resources available can be used. The system resources include *software* (rules and regulations), *hardware*, *environment*, and *liveware* (users). The SHEL model, named by the initial letters of these resources, can be used to represent the components and their links [42]. Figure 4.1 contains a graphical representation of the SHEL model. The connecting lines between the system components represent the interfaces between the respective resources. It is at the interfaces to the *liveware* component that most human factors issues occur.

Figure 4.1: The SHEL model.



In terms of the SHEL model, examples of human factors problems include microphone placement and characteristics (hardware-liveware interface), speech variations due to background noise (environment-liveware interface), and design of error correction strategies (software-liveware interface). Note that not all human factors issues are strictly related to one single interface to the liveware component. Examples include fatigue, stress, boredom, and user acceptance of ASR technology. It should also be emphasized that ATC is a multi-user system. Thus, there are also liveware-liveware interfaces that must be considered.

4.3 Identifying Human Factors Issues

Identifying human factors issues related to ASR technology is a topic that has been covered adequately and extensively [43]. However, ATC is fundamentally different from other ASR applications in several ways:

- In ATC, voice is the primary communication channel, and microphones are already used.
- The ATC vocabulary and syntax are already defined and cannot be altered.
- Human errors in the ATC environment can lead to fatal results.
- The background noise consists of distinct voices, not random noise.

The SHEL model is useful for an initial analysis of the human factors issues, and for pinpointing specific problems. Three categories of human factors problems were found: *common issues* that are mutual to both ATC and other ASR applications, ***unique issues*** that are typically not encountered in other applications, and ***non-issues*** – problems that may be significant in other applications, but that do not play a major role in ATC.

4.3.1 COMMON ISSUES

Issues that are common to both Air Traffic Control and other applications include the problem of recognition errors due to variations in speech. This may be caused by stress, fatigue, colds, temporal drift, or inter-speaker variations. Speech variations can be long term or short term changes. Possible solutions include use of speaker independent systems, which by definition can cope with different speech patterns without prior training. However, commercially available speaker independent systems typically require some degree of speaker dependence to achieve an adequate level of performance. A more pragmatic approach is adaptive training, which is usually achieved by updating the user's speech model after each recognized phrase. The technology required for adaptive training is available today. A simple experiment using adaptive training indicated that recognition errors could be reduced from 10% to 1% in the best cases. However, it is still unclear whether this technique is capable of coping with short term changes such as stress induced speech variations.

Background noise is problematic since it can cause spurious recognitions and since it can also alter the speaker's voice. As background noise levels increase to mask out auditory feedback to the speaker, the voice changes. This phenomenon is known as the *Lombard* effect. For this reason, the microphone input is usually fed into an earphone mounted on the user's headset. Other solutions to handle the background noise problem include use of noise cancelling microphones and pre-processors. The technology is sufficiently advanced in this field that it no longer needs to be considered a serious problem in the ATC environment.

Other design issues of importance include user acceptance to the technology, proper presentation of feedback, and the design of error detection and correction strategies. These factors have an influence on performance and cognitive workload. If the users are not motivated to use the technology, it will not be accepted in the field. For this reason, feedback information describing recognition results should be as unintrusive as possible to the controllers. Syntactic, semantic, and pragmatic processing, as well as Confusion Matrices, should be used to provide automatic error detection and correction [45]. Several of these techniques were incorporated into the man-machine interface model developed in this research effort. These techniques will be discussed in more detail later.

User training and enrollment is especially a problem with speaker dependent systems. During the course of this project, cases were experienced where training took two to three hours to complete. Better training algorithms must be developed in these cases. Furthermore, as much training information as possible should be obtained during the actual operation of the ASR equipment, as the user's voice tends to be different during training than in operational situations.

There are human factors issues involved with the hardware selection procedure. For example, the better the baseline recognition accuracy of the system is, the lower the cognitive workload tends to be. Excessive numbers of recognition errors cannot be tolerated. Manufacturers of speech recognition equipment invariably claim recognition accuracies near 100%. For this reason, it is necessary to develop standardized tests specific to ATC applications. Equipment that is being considered for use operationally must be tested in the actual environment it is intended for, with the same users that will operate the system.

The use of mixed input modalities should be investigated. In earlier research completed at the Flight Transportation Laboratory, it was determined that voice alone is not suitable to correct recognition errors [9]. Therefore, the mixing of voice with mouse and keyboard input should be considered. The use of touch screens also has potential for ATC applications, and is being considered for the Advanced Automation System, but is not investigated in this paper.

Finally, as has been mentioned above, the problem of syntax deviation must be considered. This is especially true for recognizers where syntax checking is an inherent part of the recognition process. Other techniques such as the use of case frames should be considered.

4.3.2 Unique Issues

Now, let us consider some of the issues that are of special significance to Air Traffic Control. One of these is recognition errors due to stress induced speech variations. This is a critical issue in Air Traffic Control since the cases where speech recognition technology has the greatest potential to enhance safety are likely to be during stressful situations. It is exactly in the conditions where the technology is needed most, that it performs worst. Adaptive training has been suggested as a potential solution, but it is not clear that rapid changes induced by stress can be overcome using this technique. Furthermore, once the speech model has been adapted, recognition errors may occur if the speaker reverts to normal speech. Thus, adaptive training limits the ability to interpret both normal and stressed speech. Alternative methods include mapping techniques derived from speech synthesis research to normalize stressed voice [46]. As this issue is significant to military applications of voice input technology in the cockpit, considerable research has been directed towards solving this problem.

Another issue is that of cognitive workload. This is already a problem in Air Traffic Control, as controllers often face an information overload situation during traffic peaks. Thus, if the introduction of speech recognition technology has the effect of increasing workload instead of decreasing it, it will not be accepted by the users. Before voice input technology is introduced into the ATC environment, the impact on cognitive workload must be assessed. Furthermore, feedback and error correction strategies must be designed to minimize the impact on cognitive workload.

4.3.3 NON-ISSUES

For the sake of completeness, the issues that are of less significance to Air Traffic Control should be described. The choice of microphone type and mounting, and the human factors issues involved, are less significant in ATC since headset mounted microphones are already used. When speech recognition equipment is introduced, the same or similar headsets should be used if possible. This would minimize the impact of introducing new technology. Some care must be taken however, to ensure that the performance of the speech recognizer is not overly sensitive to microphone placement, as is often the case with noise cancelling microphones.

The design of the vocabulary and syntax is an important issue in most speech recognition applications. Care must be taken to determine the needs of the user, and design the grammar so that the voice dialog is unambiguous and instills a sense of confidence in the user [47]. In ATC however, the grammar has already been defined and for all practical purposes, cannot be altered. Thus, speech recognition must be adapted to suit the existing vocabulary and syntax.

Finally, a problem common to other applications is recognition errors caused by users talking to each other with live microphones. This issue was encountered during previous research conducted at the Flight Transportation Laboratory [9]. However, in that particular scenario no push-to-talk (PTT) switch was used. In today's ATC system, controllers already use PTT switches when issuing commands. This eliminates the problem of spurious recognitions caused by communicating with other users, and also simplifies endpoint detection during continuous speech recognition.

4.4 Applying Human Factors Expertise

Human factors differs from other engineering disciplines in that there is a lack of analytic procedures that can be applied to human factors problems. For some human factors issues, such as the choice of display colors, the design of controls, etc., there is extensive literature based on theoretical and empirical results [48, 49]. This information is of some use to this research effort, as it provides an analytic approach for designing the graphical representation of recognition results and user feedback. What is needed, in the absence of analytic methods, is a set of broad guidelines that address the man-machine interface issues involved in introducing ASR technology. Some solutions to specific human factors problems have been discussed above, but a more extensive set of guidelines is available from existing literature and is presented below.

4.4.1 Dialog Design

The design of the dialog – the interaction between the user and the machine – is less flexible in ATC applications of voice input technology, since the syntax and vocabulary are prescribed by the FAA. Nevertheless, the following guidelines should be maintained [50]:

- The user can become lost in a strictly defined complex syntax. More flexible representations such as case frames should be used.
- The average error rate may be unacceptable for critical commands.
- Insertions (legal recognition when no word was spoken) can confuse the user.
- On-the-job speech may not match voice patterns generated during off-line training conditions.

- Long host response times seem worse in voice input/output systems due to raised expectations from human-human like speech interaction.
- The user may assume that the system understands word equivalents (e.g. "nine" and "niner" in ATC).
- The user may think the system automatically knows when to recognize speech and when not to.
- The user may try to speak louder and slower when a recognition error occurs.

As can be seen, several of these potential pitfalls stem from the user's expectation of the performance of the technology. Because using voice input/output can appear more natural than typical input modalities such as using a keyboard or a mouse, the user expects the system to behave more intelligently.

It should be recognized that ASR systems are not always suitable as simple keyboard replacements [47]. Voice input technology is useful in ATC to capture information that is already transmitted verbally. However, it may also be considered for secondary functions such as controlling inter-facility communications. In these cases it is significant to ensure that the user perceives a need for ASR technology, or at least a deficiency with the current input modality. The man-machine interface should provide what the user wants, needs, and expects. Users should be in command and feel that the system is adapting to them. The user should feel confident with the transaction, and feedback should always be unambiguous. Furthermore, the transaction should make use of graphical representations and minimize the need for instruction manuals.

4.4.2 Design of Training Procedures

Training procedures should be designed in such a way that the user remains motivated. If the user becomes bored, the speech patterns will change, and recognition performance on the job will be poor. This puts a constraint on the time required for training as well as how repetitive a training session can be. Simple repetition of all words in the vocabulary is not acceptable, especially for large vocabulary applications [11, 51]. Instead, as much training information as possible should be obtained during operational use of the voice input equipment. If this is not possible, simulations should be used. Furthermore, the user should be provided with feedback information indicating the quality of the training data. Training should be based on continuous speech, to capture coarticulation features of the speaker's voice.

4.4.3 Background Noise

Most likely, any ASR system used operationally in ATC will be speaker dependent, or at least speaker adaptive. For speaker dependent systems, the noise levels during training can significantly influence recognition accuracy. Variable (versus constant) noise levels pose an addition problem [31]. Noise cancelling techniques may reduce part of the problem, but other guidelines exist to minimize training related problems with noise:

- Training should be done with the same noise levels as will be encountered in the application.
- If variable noise levels are present, more robust results will be achieved if training is done with high noise levels.
- Inclusion of one quiet training pass can improve performance for applications with variable levels of noise.

It should be noted that most manufacturers of speech recognition equipment provide their own guidelines for the proper training procedures in applications with significant background noise. However, the guidelines listed above are typical for the speaker dependent systems available today, including those used in this research project.

4.4.4 Task Analysis

Task analysis can be used to assess the feasibility of using ASR in specific ATC applications. As it includes considerations of the operator loads and requirements, it is also useful for identifying and evaluating human factors issues. Task analysis can be thought of as the process of describing a task by its atomic components. Commonly followed steps in task analysis include the following [52]:

- Specifications of system objectives and functions.
- Relation of these functions to operator inputs.
- Determination of information necessary for operator actions and feedback.
- Determination of constraints.
- Identification of operator loads.
- Identification of repetitious tasks.
- Determination of operator skill requirements.

Task analysis may also be useful in identifying user errors and needs not previously recognized. A task analysis based feasibility assessment may help in designing the user training program. The constraints imposed on the operator and system performance that are identified through task analysis must be acceptable if ASR technology is to be introduced.

4.4.5 Automatic Error Detection and Correction

The importance of automatic error detection and correction techniques has been emphasized several times. Recognition errors cause frustration and increased stress, which can induce a vicious circle that results in unacceptable recognition performance. Recognition error detection and correction can be partially handled by the user, but if voice means alone are used, correction commands may also be misunderstood. Hence, other input modalities should be used, preferably including some rapid pointing device. The trackball is an ideal candidate since it is already in use in the ATC environment.

Automatic error detection and correction techniques can be achieved using syntax, semantics, and pragmatics. Other techniques include the automatic insertion of alternate words, and the use of Confusion Matrices to aid correction in the case of substitution errors. There are also various Artificial Intelligence (AI) reasoning rules that can be applied to improve automatic error detection. In fact, the most concise way to implement an automatic error detection and correction system may be to employ an existing rule-based expert system shell, and provide a set of heuristics for the particular application. These heuristics should be dynamic and draw upon the syntactic, semantic, and pragmatic knowledge available. This would most likely require an expert system shell capable of temporal reasoning.

One algorithm, the SMART system, uses the following procedure to implement error detection and correction [45]:

- Check the recognized phrase for syntax.
- If a word is rejected, get the runner-up word. Compute the score difference between the first word and the runner-up word, and increment the evidence score ES.
- Refer to the appropriate Confusion Matrix and determine if the first word and the runner-up word are frequently confused. Increment ES accordingly.

- Check the user's speech pattern statistics, to see if the runner-up word is used frequently in that position. Increment **ES** accordingly.

At the end of the procedure, if **ES** is above a threshold value, the runner-up word should automatically be inserted. Otherwise, the word should be inserted but verified by the user, or the user should be prompted for a new word. The weights of the different components of the evidence score and the magnitude of the threshold must be chosen from experience.

An automatic error correction procedure as provided by the SMART system points out the significance of being able to obtain scores for runner-up words. This is critical in choosing an alternative automatically, when it can be determined with some certainty that a recognition error has occurred. However, many speech recognition systems commercially available today use Viterbi (or similar) algorithms to find the best match. These algorithms, by their nature, cannot return the scores of alternative hypotheses. Thus, important information is not made available to the automatic error correction algorithm.

Chapter Five

ASR Equipment Evaluation

5.1 Results from Previous Research

Prior to the current research effort conducted at the Flight Transportation Laboratory, preliminary work was completed to demonstrate the feasibility of using commercially available speech recognizers for ATC applications [9]. In May 1985, the LIS'NER 500, a speaker dependent, isolated speech recognition system with a 64 word vocabulary was purchased. The LIS'NER 500 was manufactured by Micro Mint, Inc., and was intended for the Apple II family of computers. Baseline recognition accuracies of 70-80% were achieved. Recognition delays were on the order of 2-5 s. These high delays were partially due to the LIS'NER 500's dependence on the Apple II 6052 processor. These delays, combined with poor recognition accuracy, made the system difficult to use, even for preliminary research purposes.

A second, more capable system was purchased for the preliminary research effort. This was the Votan VPC 2000, a speaker dependent, continuous speech recognizer with an active vocabulary of 64 words, to be used with the IBM PC series of personal computers. In contrast to the LIS'NER 500, the VPC 2000 contains its own processor, a Motorola 6809, as well as a customized signal processing chip. When operated as an isolated speech recognizer, the VPC 2000 achieved recognition rates of 97%, with recognition delays of 0.8 s. Thus, both in terms of accuracy and recognition delay, the VPC 2000 was found to be superior to the LIS'NER 500. When operated in continuous speech recognition mode however, recognition accuracy was degraded. In particular, it was found that the VPC 2000 could not successfully cope with speech variations due to coarticulation.

5.2 The Hardware Selection Process

The selection of the LIS'NER 500 and the VPC 2000 seems to have been motivated by market availability and price, rather than specifications relevant to the application at hand. In the course of the current research effort however, a set of criteria was defined to aid in the choice of new recognition hardware. Purchase of new equipment was motivated by recent advancements in the technology as well as deficiencies of the LIS'NER 500 and the VPC 2000. The required performance specifications desired for ATC applications have been discussed extensively in Chapter 2, but are summarized below:

- Continuous speech recognition.
- Vocabulary size greater than 100 words.
- High baseline recognition accuracy.
- Efficient training procedures.
- Accessibility to the recognition software.
- Noise robustness.
- Low sensitivity to variations in speech (including coarticulation).
- Recognition delays less than 0.5 s.
- Speech playback or speech synthesis capability.
- Availability of noise cancelling microphone with PTT switch.

Equipment were evaluated and ranked according to these criteria. The results of this evaluation process are published elsewhere [30].

The evaluation study concluded with the recommendation that the ITT Defense Communication Division's VRS1280/PC be purchased. Consequently, an order was placed for one unit. However, due to delivery problems, the order was cancelled, and in June 1989, a Verbex Series 5000 Voice Development System with the extended vocabulary option was purchased instead. The Verbex 5000 was not available during the initial market evaluation, but appeared equal or superior to the VRS1280/PC. Both the VRS1280/PC and the Verbex 5000 have established themselves as market leaders. The VRS1280 series have been used extensively for government and military applications, whereas the Verbex 5000 series have been applied primarily in commercial systems.

5.3 The Evaluation Process

The purpose of the evaluation process was primarily to obtain performance measures for the newly acquired Verbex 5000, to determine its suitability for ATC applications. In doing so, it was decided to obtain both absolute performance data as well as to perform a comparative evaluation between the Votan VPC 2000 and the Verbex 5000. The Votan VPC 2000 was a suitable candidate for testing, since it ranked highly in the equipment selection study[8]. The main performance criterion used was baseline recognition accuracy, measured in terms of the word recognition rate.

The evaluation tests were performed with scripts of 50 phrases generated randomly from a grammar used during the previous research completed by Trikas. Hence, the testing used ATC commands, although there was some deviation from the syntax specified by the 7110. These deviations were due to shortcomings in the original grammar design. Despite its flaws, it was decided to use the existing grammar to maintain consistency with previous results. Once the evaluation phase was over however, a new vocabulary and syntax were defined for the current research effort. Table 5.1 represents a sample of the phrases used for evaluation testing of the VPC 2000 and the Verbex 5000. Overall, the script contained 50 phrases with a total of 429 words.

Table 5.1: Sample phrases used for evaluating ASR hardware.

```
"Air Canada 558 climb and maintain five thousand five hundred"
"United 452 cleared for final"
"Trans World 120 reduce speed 210"
"Air Canada 341 turn left heading 280 over"
"United 780 descend and maintain three thousand"
```

5.4 Characteristics of the ASR Systems

The evaluation tests relied primarily on word recognition accuracy as a measure of performance. Recognition delays were also measured as a secondary parameter. However, as has been discussed previously, ASR systems possess many other characteristics that influence their suitability for ATC applications. These include robustness to noise and speech variations, vocabulary size, design of the training procedures, accessibility to the recognition software, the availability of speech playback or synthesis, and microphone issues. An overview of the characteristics of the Votan VPC 2000 and the Verbex 5000 is presented below.

[8] The ASR hardware selection study actually evaluated the Votan VPC 2100, the successor of the VPC 2000. However, according to a company spokesman, the products differ little in performance.

5.4.1 THE VOTAN VPC 2000

The Votan VPC 2000 consists of three components: the Voicecard, Voice Key, and the Voice Programming Language (VPL) software [53]. The VPC 2000 is manufactured by Votan, Inc. of Fremont, California. At the time of purchase in 1986, the VPC 2000 cost approximately $1,500.

The Voicecard is the actual hardware – an IBM PC compatible add-in board that performs the actual signal processing and feature extraction. It contains its own Motorola 6809 Central Processing Unit (CPU) to offload the host computer's CPU, as well as custom-made signal processing chips and 22 kilobytes of Random Access Memory (RAM). The onboard RAM is used to store training templates during recognition. The VPC 2000 relies on the host computer's hard disk or diskette drives for permanent storage of user voice templates. Communication with the board is achieved through the internal system bus. The board has provisions for external audio input and output. The audio input can be used for storing digitized recordings of speech. This feature was not used during the comparative evaluation against the Verbex 5000 however.

Voice Key is a software package that allows the user to define a vocabulary, to train it, and to define a keyboard equivalent for each entry in the vocabulary. The user can also store audio segments to be used for feedback purposes. In addition to adding new training templates, Voice Key also provides the ability to perform recognition to test the lexicon. Both isolated and continuous recognition are supported. The purpose of Voice Key is to design voice equivalents to standard keyboard entries for frequently used software applications such as spreadsheet programs. However, it can also be used to design application vocabularies, and to perform training and testing within one program environment. For example, Voice Key was the software used for the evaluation of the VPC 2000 conducted within this research effort.

VPL is a programming language in its own right, designed for the purpose of creating voice input/output applications using the VPC 2000. Source files are created with a regular text editor, and then compiled and executed. VPL was used in the early Flight Transportation Laboratory research effort to create an ATC simulation featuring voice input and output [9]. The recognition output was redirected to the IBM PC's serial port, which was connected to a Texas Instrument Explorer Lisp machine running an ATC simulator. VPL provides more flexibility than Voice Key, but requires some programming skills.

The VPC 2000 is capable of storing an active vocabulary of 64 utterances[9]. However, VPL provides a vocabulary switching feature, so that the effective vocabulary size can be much larger, if the application can be logically subdivided into sequential tasks. Training consists of simply uttering each word to store a reference template. It is possible to store multiple templates for each word, and it is recommended that at least two templates per word be used. However, the number of possible training tokens is limited by the onboard memory capacity. Voice Key contains an embedded training feature that allows the user to extract templates from phrases spoken at a natural rate. The extraction can be performed manually or automatically. Thus, it is possible to store one token uttered during isolated speech recognition and another from continuous speech recognition. The VPC 2000 does not possess any capabilities beyond this to deal with the coarticulation problem.

5.4.2 The Verbex Series 5000 Voice Development System

The Verbex Series 5000 Voice Development System consists of the Verbex Voice Input/Output System, voice cartridges, the Vupdate, Emulate, Convert, and Transfer utilities, and the Voice Developer software package [54, 55]. The Verbex 5000 is manufactured by Verbex Voice Systems, Inc. of Edison, New Jersey and Littleton, Massachusetts. At the time of purchase in 1989, the system (with the extended vocabulary option) cost $8,495.

Unlike the Votan VPC 2000, the Verbex Voice Input/Output System is housed in a separate casing with dimensions 17x4x12 inches. Communication with the host computer is achieved through an asynchronous serial interface (RS-232). The Verbex system can be used as an input/output device for any computer or terminal with an RS-232 interface. However, the application design process can only be conducted on those host computers for which Verbex provides the necessary software. In this particular case, an IBM XT was used. Transfer rates of up to 19,200 baud are supported. The system is capable of communicating with other Verbex units in a chain network. This feature was not used however. The Verbex system has provisions for audio input and output, as well as a slot for voice cartridges. The host computer's storage capabilities (e.g. hard drive or diskette drives) can also be used for permanent storage of grammar definitions and user speech patterns.

[9] An utterance is usually a single word, but it is sometimes more convenient to train logical combinations of words as one utterance (e.g. "Air Canada").

The main system components are mounted on a main circuit board and a daughter board horizontally mounted inside the casing. The main circuit board contains three processors: a Texas Instruments TMS320, an Intel 8088, and a Verbex proprietary processor known as the "Template Manager". The main circuit board has 512 kilobytes of RAM and provisions for up to 256 kilobytes of Read Only Memory (ROM). The daughter board provides the circuitry for the voice output facility. It has a Motorola 68000 processor, 16 kilobytes of RAM, and 192 kilobytes of ROM.

The voice cartridges are storage devices for maintaining individual copies of recognition files and user voice files. The concept is that each user has a voice cartridge to store training data for the particular application. Before a user starts a session, the voice cartridge is simply inserted to access the appropriate voice files. Each cartridge contains 64 kilobytes of battery-backed CMOS[10] RAM. Each voice cartridge can store one recognition file and one voice file, which together form a complete voice input/output application.

Several programs are supplied with the Verbex 5000 system to design voice applications and train the system. One of these programs, Vupdate, converts voice files from earlier versions, to the current version of the software. The Convert program takes an ASCII source file describing the vocabulary and grammar definition using Verbex' notation, and converts it into a recognition file. The Transfer utility is then used to download the recognition file to the Verbex 5000 system. This utility can also be used to transfer voice or recognition files between the voice cartridges, the Verbex 5000 internal memory, and the host computer. Finally, the Emulate program allows the user to access a menu based software package that includes facilities for user training to create voice files, recognition tests, manipulation of voice and recognition files, as well as diagnostics. The Voice Developer System is a menu driven program that provides a user friendly environment for the user to develop and test voice input/output applications.

The Verbex 5000 has a maximum vocabulary size of 80 words. However, there is an extended vocabulary option which allows for vocabulary switching, which can be used to increase the effective vocabulary size. Training consists of two phases. First, during the enrollment phase, the user simply repeats each word in the vocabulary twice. During the second phase, user training, a script is presented with phrases generated from the syntax.

[10] Complimentary Metal Oxide Semiconductor (CMOS) chips are useful for battery-backed permanent storage due to their relatively low power consumption.

Continuous speech recognition is then used to train these phrases, providing important coarticulation information. The training script is designed such that each word is repeated a certain number of times, according to a changeable parameter. It is also possible for the application designer to define the training script. The user's speech patterns are stored using HMM representations. Whenever a phrase is trained, the probability parameters in that user's particular HMM network are updated. Unlike the VPC 2000, adding more training tokens does not increase the memory required to store the speech patterns. This allows for adaptive training techniques to be applied without having to delete earlier training data. The size of the HMM representation is determined by the number of words in the vocabulary, and the branching factor of the syntax. These two characteristics are constrained by the computational power of the Verbex 5000 and by the memory available for storage.

5.5 Evaluation Results

The first test consisted of an evaluation of the Votan VPC 2000. Trikas reported 97% recognition accuracy with isolated speech, but did not specify to what extent the results were degraded with continuous speech. Since continuous speech capability is essential for ATC applications, the test was performed using a natural rate of speech. The resulting accuracy rate was found to be 83.4%. This represents a significant reduction in recognition accuracy from the isolated speech case. This degradation in recognition accuracy amounted to a more than fivefold increase in error rate, from 3% to 16.6%. The overwhelming majority of the errors were substitution errors, although rejection and insertion errors also occurred.

When the Verbex 5000 was tested, the initial results were remarkably poor. The word recognition rate achieved was only 70.2%, significantly lower than the baseline recognition accuracy measured for the VPC 2000. It was found that the recognition performance was sensitive to the placement of the noise cancelling microphone. By varying the position of the microphone element, the recognition rate was improved to 76.0%. However, this result was still much lower than the desired 98% or above.

When analyzing the type of errors that had occurred, it became clear that rejection errors constituted the prime cause for the low recognition accuracy. If the rejection errors were excluded from the performance statistics, the recognition accuracy was found to be in the range of 98.0% to 99.5%. This disproportionate impact caused by these errors was due to the method the Verbex 5000 uses to handle rejection errors. In the case of a rejection error, the

Verbex 5000 rejects the entire phrase, unlike the VPC 2000 which only rejects individual words. This phenomenon is due to the search algorithm employed by the Verbex 5000 to find the best match between the recognized input and the stored speech templates. The algorithm, while efficient, does not allow the Verbex 5000 to formulate alternative hypotheses once a word has been rejected. For the same reason, the Verbex 5000 cannot report scores for hypotheses other than the best match.

Given this finding, an attempt was made to reduce the frequency of rejection errors. This was particularly important due to the marked degradation in recognition accuracy caused by these errors, and because rejection errors imply a loss of information. Like most speech recognizers, the Verbex 5000 utilizes an arbitrary threshold score to determine whether a rejection error has occurred or not. The threshold can be altered by the application designer. For the Verbex 5000 this is not a documented feature, but it was pointed out by Verbex' technical support personnel. Changing the value of this threshold parameter provides a trade-off between the frequency of rejection errors and the frequency of substitution or insertion errors. In the case of the Verbex 5000, lowering the threshold increases the likelihood that a recognized token will be rejected. If the threshold is increased, tokens that were previously candidates for rejection will now be recognized. However, in this case there is an increased chance of the recognized phrase containing substituted or inserted words, but this is less problematic than the loss of an entire ATC command. Thus, the application designer must find the threshold value that yields an optimum balance between rejection errors and other types of recognition errors.

Essentially the only method to find an optimal threshold value is by trial and error. Several values were tried, before obtaining test results with virtually no rejection errors, and only a marginal increase in substitution and insertion errors. A new evaluation test was performed with the new threshold value, yielding only two substitution errors out of the total 429 utterances, or a 99.5% recognition accuracy. This level of performance, although achieved under near ideal noise conditions and using a prepared script, exceeds the 98% level required for operational applications.

A final experiment was completed several weeks after the initial series of evaluation tests to measure robustness to temporal drift in an individual user's speech patterns. A recognition rate of 89.7% was recorded, reflecting a considerable degradation in recognition accuracy. Once again, most of the errors were due to rejections of entire phrases. After the test was completed, a new training pass was performed to update the speech patterns. The recognition accuracy improved to 99.5%, the same value achieved previously. It would have been possible to once again increase the rejection threshold value, but that method cannot be continued indefinitely as it will ultimately result in an intolerable increase in substitution and insertion errors. Thus, these results emphasize the importance of adaptive training to maintain high levels of robustness to variations in speech.

The results from the evaluation tests are summarized in Table 5.2 below. The tests confirmed that the Verbex 5000 is superior in performance to the VPC 2000, and is achieving the levels of recognition accuracy required for operational applications. However, problems remain with sensitivity to microphone placement and to speech variations. The evaluation results indicate that adaptive training has the potential to improve recognition accuracy. The VPC 2000 does have some advantages over the Verbex 5000, including the ability to reject just words instead of whole phrases, and considerably lower purchasing costs. In terms of performance however, the VPC 2000 cannot be considered a true continuous speech recognizer, but rather a connected speech system as its recognition performance is significantly degraded when using a natural rate of speech. The Verbex 5000 was judged to be a true continuous speech recognizer, suitable for preliminary research applications.

Table 5.2: ASR hardware evaluation results.

Equipment	Case	Word recognition accuracy	Delay
LIS'NER 500	[isolated speech]	70-80%	2-5 s
Votan VPC 2000	[isolated speech]	97.0%	0.8 s
	[continuous speech]	83.4%	
Verbex 5000	[basecase]	70.2%	0.1-0.5 s
	[improved microphone placement]	76.0%	
	[increased rejection threshold]	99.5%	
	[temporal drift, before retraining]	89.7%	
	[temporal drift, after retraining]	99.5%	

5.6 Simulation of the ATC Environment

The evaluation tests were conducted using scripts of randomly generated phrases. Although the phrases represented valid ATC commands, they did not constitute a coherent ATC task. As a result, the cognitive workload during the evaluation tests was relatively low, and hence stress induced speech variations were practically absent. Furthermore, the voice patterns resulting from a reading task are significantly different from the patterns resulting from a tactical planning task such as ATC. Hence, there exists a need to provide an ATC simulation to be used in conjunction with the voice input/output equipment that is being evaluated. This simulation should exhibit sufficient realism to provide the user with a task, not just the capability of arbitrarily vectoring aircraft. The taskload should also be variable so that the issue of cognitive workload and its impact on speech recognition can be investigated.

Trikas, in his research effort, used the Votan VPC 2000 in conjunction with an ATC simulator developed at the Flight Transportation Laboratory for the TI Explorer Lisp Machine [9]. Using a VPL program, the output of the recognition process was transferred to the Lisp Machine via a serial connection. There, it was processed by the ATC simulator to control the simulated aircraft. The VPC 2000's voice storage capability was used to incorporate pilot responses to the controller. These responses consisted of the callsign followed by the acknowledgement "roger" or "say again" in the case of a recognition error.

Although the simulation used by Trikas possessed adequate functionality for his preliminary research, it does not provide an inherent task for the user. Aircraft are presented on a simulated radar display, and the user can vector the aircraft using voice recognition, but no flight plan information is available, nor are neighboring sectors simulated. Furthermore, the use of the TI Explorer Lisp Machine reduces the portability of the simulation. For these reasons, a different ATC simulation was selected for the research effort presented in this paper, the Wesson International TRACON simulator. TRACON is a PC-based simulator, and hence can be used on the same host computer that is communicating with the Verbex 5000. This greatly enhanced the level of portability, especially since IBM PC machines are much more common than TI Explorers. Furthermore, TRACON provides the user with a list of flight strips that define a task. The number of simulated aircraft per unit time can be varied to control the taskload [54]. Hence, TRACON provides a low cost ATC simulation that suits the requirements for the type of testing required for this research effort. It was also useful as a *technology demonstrator* of the capabilities of voice input/output technology.

In order to run TRACON in conjunction with the Verbex 5000 speech recognizer, a software utility named Softkey was used. Softkey provides a virtual pipeline from the IBM PC serial port to the keyboard input stream. The recognition output from the Verbex 5000 could be directed to the keyboard input for the TRACON program. Table 5.3 below lists the ATC commands supported by TRACON, and their keyboard equivalents. Hence, an off-the-shelf ATC simulator was used, without any special modifications to incorporate voice input.

Table 5.3: The TRACON command set.

ATC command	Keyboard input
"Turn right NN degrees"	Right arrow, then two digits.
"Turn left NN degrees"	Left arrow, then two digits.
"Turn right heading NNN"	Right arrow, then three digits.
"Turn left heading NNN"	Left arrow, then three digits.
"Climb and maintain N thousand N hundred"	Up arrow, then two digits.
"Descend and maintain N thousand N hundred"	Down arrow, then two digits.
"Reduce speed to NNN"	Insert key, then three digits.
"Increase speed to NNN"	Insert key, then three digits.
"Resume normal speed"	Delete key.
"Resume normal navigation"	Delete key.
"Cleared direct to **FIX**"	Home key, then three or five letters.
"Hold at **FIX**"	Page Down key, then three or five letters.
"Say heading and airspeed"	Page Up key.
"Cleared for ILS approach"	End key.
"Cleared for VOR approach"	End key.
"Contact center, good day"	End key.

Note: All commands are preceded by the aircraft callsign, entered explicitly or selected using the mouse or cursor keys. Also, note that some keyboard commands result in different ATC actions depending on the situation.

As has been described above, the Verbex 5000 contains a voice response feature. Hence, speech synthesis could be used with TRACON to simulate pilot acknowledgements. This was not necessary however, since pilot responses using voice synthesis are already featured in TRACON. This speech synthesis is achieved on a software level: speech data is stored on the hard disk and played back through the IBM PC's internal speaker, or an external speaker connected through the parallel port. Compared to the simulator used by Trikas, the pilot responses provided by the technology demonstrator based on TRACON are more extensive, and include simulated pilot errors and communication problems.

CHAPTER SIX

A MAN-MACHINE INTERFACE MODEL

6.1 MOTIVATION

The level of ASR technology available today allows for the design of technology demonstrators such as the one developed using the Verbex 5000 and Wesson International's TRACON simulator. Simulations of this type are useful for demonstrating the feasibility of using voice input/output in ATC applications. They can also be used for investigating potential problems such as background noise and stress induced speech variations. However, the simulator used for this research effort provides no error correction capability. For example, if the recognized callsign is not in the active aircraft set, an error message is displayed. The rest of the recognized command however, is processed. Since the aircraft callsign was not parsed successfully, the processing of the remainder of the command fails, and a second error message may be issued. The user must then repeat the sentence until it is recognized successfully. Clearly, this lack of automatic error correction strategies is unacceptable for operational applications of ASR technology.

The underlying reason behind the inadequacy of the simulator set-up is that it does not take advantage of higher levels of knowledge inherent in the system. In short, the voice input parser does not possess situational intelligence. For example, if a callsign is recognized that differs from a valid callsign by only one character, then one can safely substitute the recognized callsign by the best match from the set of active aircraft. This is an example of the use of pragmatic, or situational, knowledge: the use of knowledge about the system state. As has been described above, there are seven levels of knowledge of significance to the speech recognition process: acoustics, phonetics, prosodics, lexical analysis, syntax, semantics, and pragmatics. Of these, the last three represent the higher levels of knowledge. These can be used to create an intelligent voice input parser. By acquiring and processing higher levels of knowledge, it may be possible to achieve the high levels of recognition accuracy required for operational usage of the technology, and to add robustness to the recognition process. Hence, a model of a potential man-machine interface for operational ATC applications was developed, to investigate the effect of adding intelligence to the speech recognition process.

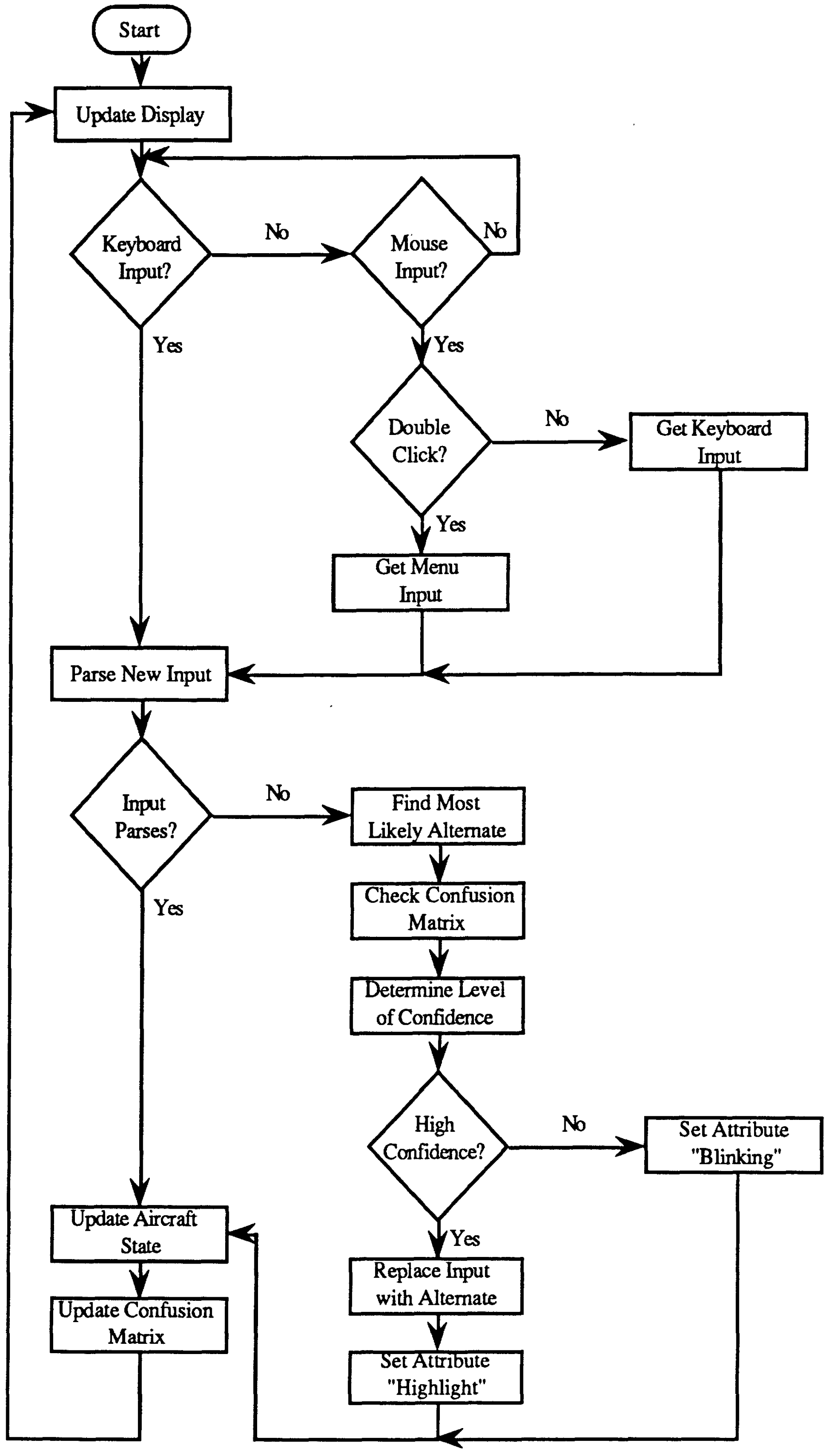
It should be noted that processing higher levels of knowledge is only part of the automatic error correction process. Just as significant is the presentation of feedback to the controller. For example, if an error has been detected, and an alternate hypothesis has been found with a high level of confidence, it may be possible to correct the error without requiring any intervention by the user. The introduction of ASR technology into the ATC environment must not result in increased levels of workload. Hence, the design of user feedback and the frequency of user intervention are critical problems. Another issue of interest is the use of mixed input modalities. These aspects were also investigated using the man-machine interface model.

6.2 ATCVIP - The Air Traffic Control Voice Input Parser

A model called the Air Traffic Control Voice Input Parser (ATCVIP) was developed and implemented in the form of a MicroSoft QuickBASIC computer program. Figure 6.1 contains a flowchart describing the essentials of the ATCVIP program. Notice that there is no explicit mention of voice input. This is because the Softkey utility was used to pipe the voice recognition output into the keyboard input stream. Hence, the voice input is represented by keyboard input. This not only simplified the input/output handling, but allowed for exact control during testing, as recognition errors could be simulated by entering errors via the keyboard. The ATCVIP program is essentially unable to distinguish between actual keyboard input and voice input pipelined to the keyboard input stream.

ATCVIP is intended to be used in conjunction with an ATC simulator. It relies extensively on the situational state of the airspace, and hence requires two-way communication with the simulator. However, the TRACON program is a black box in the sense that it cannot provide airspace state information to ATCVIP. Hence, it could not be used with the man-machine interface model. Furthermore, TRACON does not provide the capability to introduce new output features, as would be required by ATCVIP. As a result, ATCVIP was developed as a stand-alone product. It contains its own representation of the airspace, which is displayed to the user, although not in a graphical form. By leaving the airspace state representation as generic as possible, ATCVIP can in theory be connected to any ATC simulator. The actual ATC simulation is of secondary nature – the purpose of ATCVIP is to demonstrate the ability to increase recognition accuracy by making the man-machine interface intelligent, and to present possible solutions to the user feedback and mixed input modalities issues.

Figure 6.1: ATCVIP flowchart.



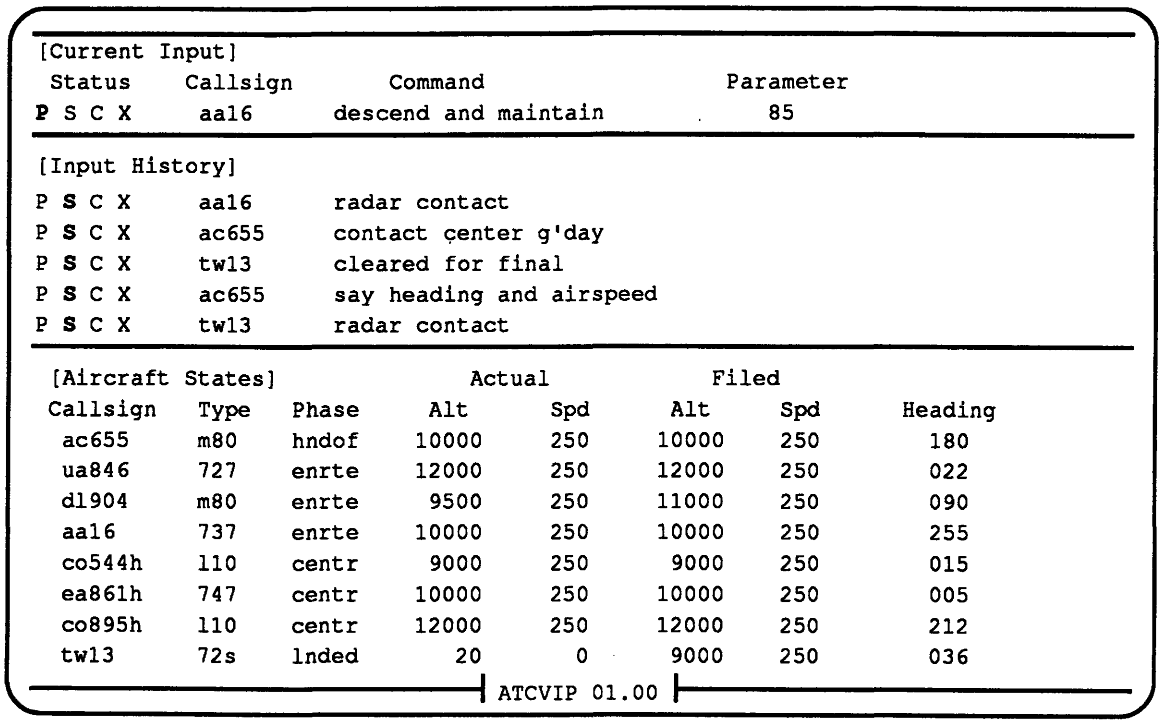
In order to maintain commonality with the technology demonstrator developed using TRACON, the same syntax definition (i.e. recognition file) was used for ATCVIP. Hence, the same voice files that were developed for the technology demonstrator could be used with ATCVIP. However, some modifications had to be made to the translation tables that define what keyboard characters should be placed in the keyboard input stream when a token has been recognized. These modifications were due to TRACON's use of the same keyboard equivalent for different ATC commands. For example, the commands "reduce speed to" and "increase speed to" both use the Insert key (see Table 5.3).

In the case of commands where a keyboard equivalent is shared, TRACON uses pragmatic information to determine which command is meant. For example, TRACON compares the recognized speed to the current speed of the aircraft in question to determine which command should be selected when the Insert key has been pressed. This implies that the recognized speed is correct however, and that no recognition errors have occurred. This assumption is feasible when using TRACON with keyboard input only, but cannot be applied when using voice input. Hence, some parameter must be introduced to distinguish these ambiguous inputs from each other. This was achieved by inserting a special character in the translation table to provide a unique identifier for each command in the syntax. The caret character ("^") was used, since it is transparent to the TRACON simulation. This allowed for the use of one recognition file for both the TRACON simulation and ATCVIP.

6.2.1 THE ATCVIP DISPLAY

As has been mentioned above, ATCVIP should ideally be used in conjunction with an existing ATC simulator. The output should be displayed on the simulated radar screen. Graphic symbols could be used to display recognized commands next to the corresponding aircraft radar return. However, since this was not possible with the TRACON simulator, and would require a customized version of ATCVIP for every simulator supported, a generic display was used instead, relying on text instead of graphics. ATCVIP is not meant to provide solutions to the knobs-and-dials type human factors issues involved with introducing ASR technology into the ATC environment. The graphical representation to the controller is an important issue, but it is a secondary problem to the functionality of the interface. The ATCVIP display contains the information that would be required in an operational interface, but in a simplified form. Figure 6.2 is a representation of the screen as it would appear to the ATCVIP user.

Figure 6.2: The ATCVIP display.



```
[Current Input]
 Status    Callsign       Command                      Parameter
P S C X     aa16      descend and maintain              85

[Input History]
P S C X     aa16      radar contact
P S C X     ac655     contact center g'day
P S C X     tw13      cleared for final
P S C X     ac655     say heading and airspeed
P S C X     tw13      radar contact

 [Aircraft States]                 Actual              Filed
 Callsign   Type   Phase      Alt       Spd      Alt      Spd       Heading
  ac655     m80    hndof     10000      250     10000     250         180
  ua846     727    enrte     12000      250     12000     250         022
  dl904     m80    enrte      9500      250     11000     250         090
  aa16      737    enrte     10000      250     10000     250         255
  co544h    110    centr      9000      250      9000     250         015
  ea861h    747    centr     10000      250     10000     250         005
  co895h    110    centr     12000      250     12000     250         212
  tw13      72s    lnded        20        0      9000     250         036
                              ATCVIP 01.00
```

The display is subdivided into three sections: *current input*, *input history*, and *aircraft states*. The current input section contains the most recently recognized command, after parsing has been completed. The command is broken down into three logical parts: the aircraft callsign, the ATC command being issued, and an optional parameter (e.g. altitude, airspeed, heading). If any of these three parts has been altered through the automatic correction mechanism, it is highlighted. If one of the parts has failed the parsing process, the original input is shown blinking. This notifies the user that the input has not been parsed successfully and must be edited manually.

In addition to the recognized command, a set of status flags is shown. Only one flag can be active at the time. The active flag is highlighted. The four possible flags are pending ("P"), sent ("S"), correction ("C"), and cancelled ("X"). A newly recognized command defaults to the pending status. When the next command is recognized the command status changes to "S", indicating it has been sent (i.e. transmitted) to the aircraft. Flags can be selected by moving the mouse cursor to the appropriate letter and clicking the mouse button. This provides the capability to send, correct, and cancel recognized phrases.

The input history contains a list of the most recently recognized commands. The most recent command is displayed at the top. The structure is identical to that of the current input section, except that a command in the input history cannot have the pending flag active. The user can select the "C" status flag by clicking on it with the mouse to correct a command. In this case, the command is inserted into the current input section and treated as a newly entered command. Editing is achieved by selecting the part of the command that needs to be changed, using the mouse. A single mouse click activates an input field where the user can enter a new value using the keyboard. A double click activates a drop-down menu from which the user can select a new value. The options in the menu are sorted according to the likelihood scores determined during the parsing process, so that the most likely replacement value occurs at the top of the menu.

The final section, the aircraft states table, lists the aircraft currently in the controlled airspace, or those about to enter it. Each aircraft is identified by its callsign, the aircraft type, the phase of the flight, the aircraft's actual and filed altitude and speed, and finally, its heading. The set of aircraft used was static. If ATCVIP were to be used in conjunction with an ATC simulator, the aircraft states would be provided by the simulator, and would change with time.

The ATCVIP parsing process makes use of information about the phase of the flight. Five phases are recognized: en route in a neighbor sector prior to handoff ("centr"), on the ground prior to take-off ("preto"), en route in the controlled sector ("enrte"), on the ground after landing ("lnded"), and en route in a neighbor sector after handoff ("hndof"). The aircraft states table represents the pragmatic knowledge available to ATCVIP. The states are updated as new commands are recognized. Most of the information that would be used in an operational interface is present, except for the aircrafts' positions relative to each other and temporal knowledge.

6.2.2 The ATCVIP Process

The ATCVIP display and its functionality have been described above, and the overall functionality of the ATCVIP process is outlined in Figure 6.1. However, the parsing process has only been briefly described. The parser represents the knowledge processing function of ATCVIP, where syntactic, semantic, and pragmatic knowledge are used to add situational intelligence to the man-machine interface. The full source code for the ATCVIP program is listed in Appendix B. The language used is MicroSoft's QuickBASIC, Version 4.5. Although it may be possible to discern the functionality of the ATCVIP interface by examining the source code, a description of the process is presented below to describe the concepts employed.

The first segment of the program, the initialization process, sets up basic arrays and parameters. It also contains a data segment containing the initial values for the aircraft states, the list of supported ATC commands, and the navigational fixes and airport identifiers. The display is initialized and cleared. Subroutines are then executed to initialize the mouse handler, dynamic variables, and the display.

After the initialization process has been completed, the main loop is entered. During each iteration of the loop, a subroutine is called to wait for user input. As the ATCVIP simulation is not real time, no processing is required between inputs, except for the processing of the inputs themselves. The input subroutine remains in an idle state until either a keyboard or a mouse action is recognized. If a keyboard action is recognized, it is assumed that an ATC command is being entered, and the keyboard input stream is monitored until a terminator character is detected. If a mouse action is recognized, the number of clicks is recorded, as well as the position of the mouse cursor. A mouse action consists of clicking the mouse button one or more times. Simply moving the mouse is not considered to be a mouse action. Once an input has been recognized and recorded, the program returns to the main loop. Depending on the type of input, either the keyboard or the mouse handling subroutine is executed. The main loop is repeated until an exit flag is detected in which case ATCVIP is terminated.

The keyboard handling subroutine is activated whenever voice input is recognized, or when the keyboard is used to simulate voice input. First, the subroutine checks to see that the last input is ready to be pushed onto the input history. If parsing the last input failed, and the user did not make a correction, the new input is stored in a buffer until the last input is modified. If the last input does not present a problem, a subroutine is called to move it to the top of the input history. Also, the aircraft states table is updated to reflect any changes implied by the last input. To simplify the simulation, these changes occur instantaneously – no temporal concepts are included. After the input history and aircraft states have been updated, the parser is invoked. The parser consists of the serial execution of three routines, one to parse the aircraft callsign, one for the command, and one for the optional parameter. These routines are described in more detail below. When parsing is completed, the recognized input is displayed in the current input section, with the flag "P" (i.e. pending). If a recognition error was detected during parsing, the segment involved is displayed highlighted or blinking.

The mouse handling routine first determines if the mouse cursor was in a valid field when the mouse button was clicked. Possible areas include the callsign, command, and parameter[11] fields of the current input section. Status flags can be selected in both the current input and the input history sections, by a single mouse click. If the "S" symbol is selected in the current input section, the status of the most recent command is changed from pending to sent, and the aircraft states and input history are updated. If the "X" symbol is selected, the status is changed from pending to deleted, and the input history is updated. The aircraft states are not modified in this case. Finally, if the "C" symbol is selected in the input history section, the corresponding input is copied into the current input section where it is treated as a new keyboard input. This provides the user with a facility to correct previously spoken commands.

If the mouse button is clicked while the mouse cursor is in the callsign field of the current input line, the mouse handler checks the number of clicks to determine what action to take. In the case of a single click, a subroutine is called that allows the user to edit the callsign field. The user simply types in a new callsign, using the Backspace key to make corrections. The new input is terminated by pressing the Enter key. If the Escape key is pressed at any time while entering a new callsign, the input procedure is terminated and the original callsign is restored. After the input has been completed, the parser is invoked, and the current input section is updated. In the case of a double click, a menu of all active callsigns is displayed. The options in the menu are ranked according to how closely they match the current entry in the callsign field. The rankings result from the pattern matching process, and do not contain any acoustic or phonetic information, as the Verbex 5000 does not provide scores for runner-up words. Thus, the entry the user wants is likely to be at the top of the menu, requiring little if any mouse movement to select it. The user moves the mouse to highlight the option of choice, and then clicks the mouse button to select it. If the mouse cursor is moved outside of the menu borders, the menu is hidden and no selection is made. Once a new callsign has been selected, the parser is invoked, and the current input line is updated.

Mouse input can occur in the ATC command field. Since the number of commands is static and relatively constrained, there is no option to single click to access a direct input field. Instead, a menu is displayed with all the commands, regardless of whether the mouse button was clicked once or twice. The user then simply moves the mouse cursor and clicks to select a new command. Once a command has been selected, the parser is invoked, and the current input section is updated.

[11] If the ATC command in question requires a parameter.

If mouse input is detected in the parameter field, the mouse handler first determines if the associated ATC command is one that requires a parameter. If no parameter is required, the mouse input is ignored. Else, the mouse handler determines if the mouse button was clicked once or twice. In the case of a single click, a subroutine is called to provide direct input of a new parameter value. In the case of a double click, a menu of suggested parameter choices is presented, ranked by order of likelihood. The process is similar to the aircraft callsign mouse handler. If a new parameter is selected, the parser subroutines are executed, and the current input section is modified to reflect the change.

The intelligence of ATCVIP is embedded in the three parser subroutines: the callsign parser, the command parser, and the parameter parser. Each subroutine contains a set of heuristics that access syntactic, semantic, and pragmatic knowledge to determine if a recognition error has occurred, and to determine what the most likely alternate hypothesis is. The confidence level is determined through heuristics, the degree of similarity between the alphanumeric representations of the alternate and the original input, and past input history. If the confidence level is high, the recognition error is automatically corrected. The alternate is inserted and displayed using the highlight display attribute to alert the user that a correction has been made. If at all possible, an attempt is made to automatically correct the error since user intervention should be kept at a minimum. If it is not possible to find a suitable alternate however, the original input is retained, but is displayed with the blinking attribute. This notifies the user that a manual correction must be made before the new input can be processed.

The callsign parser first checks to see if the recognized callsign matches one of the aircraft in the active set. Since short forms[12] are supported, it is possible that the recognized callsign may match two or more aircraft. For example, the callsign "four five two" could refer to both "United four five two" and "Delta six four five two". If the recognized callsign matches a unique aircraft, the parser terminates successfully. If not, a pattern matching algorithm is used to match all active callsigns to the recognition input. A customized algorithm is used that places extra weight on the airline identifier, if it is included. If an aircraft is found that scores higher than all other callsigns, an automatic substitution is made. Else, the parser fails, and a flag is set to notify the user. The callsign parser primarily makes use of pragmatic information, by determining if the callsign is in the active aircraft set and by finding a suitable alternate if it is not.

[12] The short form of an aircraft callsign normally consists of the last three digits or letters of the full callsign.

The command parser relies mostly on pragmatic knowledge to determine if the recognized command is meaningful. Each command is evaluated against a unique set of heuristics. The most useful information to the command parser is the flight phase. For example, a typical rule may be "if the aircraft is issued a left turn, then it must be en route". This rule makes use of the simple assumption that ATC will not issue vectors to aircraft that have not taken off yet, or that have landed. The command parser is the simplest of the three parser subroutines. ATCVIP makes the assumption that the command is recognized with a high level of confidence, so that it can be used as a reference point against which the callsign and the parameter are evaluated. This is acceptable however, as the commands always contain more phonetic and acoustic information than the callsigns and the parameters, and are less likely to be subject to recognition errors.

The parameter parser is the most complex of the three parser subroutines, as it makes use of syntactic, semantic, and pragmatic information. The parser determines which command has been recognized to select a set of heuristics that are executed to evaluate the meaningfulness of the parameter. For example, if the recognized command is "hold at" or "cleared direct to", then syntactic and semantic constraints necessitate that the parameter be a navigational fix or an airport identifier. Thus, the utterance must contain 3 or 5 alphanumeric characters. If the recognized utterance is not one of the fixes in the airspace under control, a best match is found as an alternate.

The parameter parser also makes use of Confusion Matrices. These are tables that contain information from past corrections. They are updated whenever an automatic correction is accepted by the user, or when the user makes a manual correction. Confusion Matrices are used to evaluate the level of confidence in the case where it cannot be determined by other means alone [45]. For example, assume that the command "united four five two turn left heading seven eight zero" is recognized, and it parses successfully, except for the heading. Semantic and pragmatic constraints may dictate that the heading should be either "zero eight zero", "one eight zero", or "two eight zero". That is, the word "seven" was misrecognized and should be replaced by either "zero", "one", or "two". If the Confusion Matrix entry for "seven" indicates that it has been substituted with "zero" five times in the past, but never with "one" or "two", then the parser can safely assume that the heading should be "zero eight zero". If the user accepts the automatic correction, the entry for "seven" being substituted by "zero" is incremented to six times. If the user manually corrects the error to "one", then an entry is made for "zero" being substituted by "one". Two Confusion Matrices are used: one for navigational fixes, and one for numerics.

6.2.3 ATCVIP Evaluation Results

In order to evaluate the potential increase in recognition accuracy due to ATCVIP, a script was generated with 52 ATC commands, for a total of 345 utterances. Unlike the script used for the ASR hardware evaluation, this script was not generated randomly. Instead, it was constructed to represent a typical dialog between the controller and the aircraft in the sector. Although using a script eliminates stress induced speech variations, this method was preferred as it provides precise control over the input. Furthermore, the purpose of this evaluation was to determine how ATCVIP deals with errors. The source of the error is irrelevant, as long as sufficient errors are present to be able to determine a change in the recognition accuracy.

Since the baseline recognition accuracy of the Verbex 5000 is relatively high, using the script just one time yielded error rates of 1-2%. Hence, errors were polled from several trials using the script. These were then deliberately concentrated into a single script with a total error rate of 8%. Evaluation tests were then conducted using the keyboard to reproduce this script, including the errors. Using the keyboard instead of voice ensured that the type of errors, their occurrence in time, and their frequency were identical for all evaluations. The results from the evaluation tests are summarized in Table 6.1 below.

Table 6.1: ATCVIP evaluation results.

Case	Word Recognition Error
Base case	7.8%
First pass	2.9%
Second pass	1.4%

The results show a decrease in the error rate from 7.8% to 2.9% during the first pass of using ATCVIP. Since ATCVIP is adaptive, and learns from the past using Confusion Matrices, a second pass was completed to determine if the error rate could be reduced further. In fact, the error rate decreased to 1.4%. Analysis of the type of errors that occurred showed that this was the lowest error rate that could be achieved with ATCVIP given the errors inherent to this particular script. Hence, the learning curve in this case levelled out after two passes, or approximately 100 commands. The learning rate is a function of the error rate however, and would change if a different script was used.

The evaluation tests also attempted to determine how much user interaction was required to correct the errors that occurred. The results for a typical first pass are described in Table 6.2 below.

Table 6.2: ATCVIP error correction history.

Error type		Frequency	Share
Automatic correction		10	45.5%
One mouse action required		11	50.0%
Breakdown:	No menu action	2	9.1%
	Option #1 on menu	5	22.7%
	Option #2 on menu	2	9.1%
	Option #3 on menu	1	4.6%
	Option #4 on menu	0	0.0%
	Option #5 on menu	1	4.6%
Two mouse actions required		1	4.6%

These results indicate that there was approximately a 50-50% distribution between automatic and manual error corrections. Of the manual corrections, all but 2.3% required only one mouse action. In 55.6% of the cases that required the use of menu selection, the desired alternate was the first option on the menu. In these cases only little mouse movement was required to select the alternate. If ATCVIP was tailored to include situational knowledge about the traffic flows in a specific sector, lower overall error rates could be expected.

CHAPTER SEVEN

CONCLUSION

7.1 FEASIBILITY ASSESSMENT

The current state of the ASR technology can be summarized as "barely adequate." Commercially available equipment such as the Verbex 5000 are achieving baseline recognition rates of 98% and higher. True continuous speech recognition is available, and the coarticulation problem that plagued the earlier Votan VPC 2000 appears to have been solved. Recognition delays are on the order of a few tenths of a second, which is acceptable. Background noise can be eliminated through the combined use of noise cancelling microphones, filters, and speech enhancing equipment. Speech synthesis and playback technology is sufficiently advanced, and is often combined with speech recognition hardware to form a complete voice input/output system.

However, the technology exhibits several deficiencies that prevent its introduction into the ATC environment. Vocabulary sizes for true continuous speech systems are usually limited to a few hundred words. The allowable branching factor of the syntax is sometimes constrained, in which case the full ATC grammar cannot be implemented. Speaker independent systems are excessively costly and inefficient, and often require some level of speaker dependence to achieve acceptable recognition rates. Training procedures and recognition software remain inadequate for operational applications. Another critical issue is robustness to speech variations, which can degrade recognition accuracy to unacceptable levels. These issues must all be addressed before the technology can be used operationally.

The near future holds some promise for improvements in the technology. Continuous speech recognizers are currently being developed that will be capable of vocabularies with several thousand words. Clearly, the training routines must be improved by then, in order to avoid a 5-10 hour requirement for user training. The introduction of newer processors, such as the 286 and 386 families, will enhance the computational power of the recognition hardware. This will allow for the introduction of more powerful pattern matchers, with an overall improvement in performance and efficiency. Ongoing research on speech understanding may result in better language models and the capability to include prosodic analysis. The cost/performance ratio can also be expected to improve with time.

Today's ASR technology is not sufficiently mature for operational applications in the ATC system. However, systems such as the Verbex 5000 are suitable for preliminary research such as the work presented within this paper. ATC training suites and simulators can be developed using continuous speech recognition in conjunction with speech synthesis or playback. This would allow for the creation of a fully automated ATC simulation possessing a high degree of fidelity. In fact, such systems are currently in development, both in the United States and abroad. These simulators can then be used for further preliminary research prior to the introduction of ASR equipment into the operational environment. They may also have the beneficial effect of making new controllers familiar with the technology.

7.2 THE ATC ENVIRONMENT

Several operational aspects of the ATC environment influence the potential success of speech input technology, including background noise, controller mobility requirements, number of active controllers, equipment used, and the frequency and nature of manual input. Three distinct environments can be identified: the tower, TRACON, and ARTCC. Of these, ARTCC appears to be most amenable to the introduction of ASR technology, followed by TRACON, and then the tower environment. There is an inverse relation between feasibility and potential benefit, however. Safety and efficiency improvements are likely to be greater in the ATC environments that experience the highest levels of traffic congestions – the tower and the TRACON.

Cognitive workload in the ATC environment is a critical issue for the introduction of ASR technology, as it presents a constraint on the information processing capacity of the controllers. Factors contributing to controller workload include traffic variables, geometric variables, sector type, controller interaction considerations, sector team control procedures, and technology aspects. Ideally, the introduction of speech recognition technology should result in a reduction of controller taskload, and hence cognitive workload. However, if the man-machine interface is poorly designed, and requires frequent user intervention, it may result in an increase in workload. If this is the case, the introduction of voice input/output technology will not be accepted.

The set of ATC commands can be considered as a language in its own right. The vocabulary and syntax are specified in the FAA's air traffic controller's handbook. The language was designed to be unambiguous, to reduce the possibility of recognition errors. As a result, the syntax is constrained, and possesses a branching factor well below that of the natural English language. This is inherent in the relatively low information content of the ATC language. These constraints on the ATC syntax and vocabulary facilitate the speech recognition task. However, the relative ease of applying speech recognition to ATC applications cannot alone justify the introduction of the technology. Furthermore, with the vocabulary sizes of current speech recognizers only a subset of the entire ATC grammar can be recognized.

The limited information content of the ATC language can only be capitalized upon if the controllers adhere to the prescribed syntax. Monitoring ATC radio frequencies in the Boston area indicated that the syntax deviation problem may be significant, occurring in 5-25% of all transmissions. A related problem is the frequency of hesitations and in-phrase corrections, which may confuse a speech recognizer. A promising solution to the syntax deviation problem appears to be the use of case frames, where the desired information is extracted from the recognized phrase regardless of syntax.

7.3 HUMAN FACTORS ISSUES

Although the ASR technology is not sufficiently mature for operational applications, the state of the technology itself is not likely to be the inhibiting factor to the introduction of voice input equipment. Instead, the critical issue is likely to be the human factors aspects. The relevant human factors problems can be identified by applying a model of the system and its resources, such as the SHEL (Software-Hardware-Environment-Liveware) model. Then the specific problems can be subdivided into categories of issues common to ATC and other ASR applications, issues of special importance to ATC, and issues of no or little significance.

Common issues include speech variations, background noise, user acceptance, user training, baseline recognition accuracy, use of mixed input modalities, and syntax deviation. Speech variations, especially short term changes such as those induced by stress, can rapidly degrade recognition performance, resulting in frustration and reduced safety. Adaptive training and speech normalization techniques under investigation for fighter cockpit applications hold some promise for improving robustness to speech variations. Background noise may also cause speech variations due to the Lombard effect, or may result in spurious insertion errors. Possible solutions include noise cancelling microphones and pre-processors, as well as speech enhancing equipment. User acceptance must be evaluated operationally, but can be improved by automating the error correction process as much as possible. User training must be implemented in such a way that it does not consume excessive amounts of time. If user motivation drops, recognition performance is likely to suffer. High baseline recognition performance will reduce the amount of user intervention required, and is necessary to instill confidence in the technology. When recognition errors do occur, it is not feasible to use voice alone to correct them. Instead, the use of keyboard and mouse input, combined with intelligent parsing, should be implemented to provide an efficient correction mechanism.

Human factors issues that are of special significance to ATC include stress induced speech variations and cognitive workload. The potential benefits of ASR technology are greatest in scenarios with high levels of traffic. However, these situations are likely to be stressful for the controller, which can result in degradation of recognition performance. Workload related issues must be solved in order for ASR technology to be of benefit in the situations where it is needed most.

Issues of less importance in the ATC environment include the choice of microphone type, vocabulary and syntax design, and inter-user communications with live microphones. In ATC, headset mounted microphones are already used, and should be retained if ASR technology is introduced. The design of the vocabulary and the syntax is a key issue for most ASR applications, since it has an impact on performance and user acceptance. The ATC syntax is already prescribed by the FAA. However, it may be useful to consider redesigning this syntax. Finally, the problem of spurious recognitions due to controllers talking to each other with live microphones is not likely to occur since PTT switches are already in use.

7.4 ASR Equipment Evaluation

Evaluation tests were performed to compare the Votan VPC 2000 with the Verbex Series 5000 Voice Development System. The evaluation results indicate that the Verbex 5000 is superior in terms of recognition accuracy, delay, and vocabulary size. This is not surprising however, as the Verbex 5000 represents a new generation beyond the technology level of the Votan VPC 2000. The baseline recognition accuracy of the Verbex 5000 matches or exceeds that required for operational applications. The evaluation tests were conducted under near ideal conditions, however. The Verbex 5000 lacks the necessary robustness to background noise and speech variations. Retraining the speech patterns before testing resulted in improved recognition performance, demonstrating the potential benefit of adaptive training. The most limiting aspect of the Verbex 5000 for this project was the constraint on the syntax complexity.

There exists a need to provide a realistic environment for evaluating ASR equipment intended for ATC applications. This was provided by incorporating voice input from the Verbex 5000 with Wesson International's TRACON simulator. This set-up included both speech recognition and speech synthesis to simulate pilot responses. Although this provided a low cost, high fidelity demonstrator, it was difficult to use it for controlled experiments. There exists a trade-off between the amount of control over the dialog and the realism of the simulation. If a script is provided, complete control over the voice input is retained, but the speech patterns are likely to differ from those that would be generated in an operational environment. If a general ATC task is provided instead, with no prescribed script, more realistic voice patterns can be obtained, including stress induced speech variations. However, in this case, the utterances and the resulting recognition errors will differ from test to test.

7.5 THE AIR TRAFFIC CONTROL VOICE INPUT PARSER

An intelligent voice input parser was developed to investigate some of the techniques that may be used in an operational man-machine interface in the ATC environment. Syntactic, semantic, and pragmatic information was applied to increase the effective recognition accuracy. This was achieved through a set of heuristics that represented situational knowledge acquired from the state of the airspace, as well as syntactic and semantic constraints defined in the controller's handbook. Confusion Matrices were used to codify past correction patterns. The voice input parser was thus able to reduce the recognition error, both instantaneously and over a longer period of experience. A short term improvement was achieved through automatically correcting callsigns and parameters that failed parsing, and for which suitable alternates could be found. A long term learning effect was achieved by using Confusion Matrices, to learn from past corrections.

Other techniques under investigation included the use of need-to-know type of feedback. In the cases where an automatic correction could be made with a high level of confidence, no user intervention was required. The user was simply notified that some uncertainty existed, by highlighting the corrected utterance. If the confidence score was relatively low, or no suitable alternate could be found, the utterance was displayed blinking on the screen. The user then had the option to correct the error using the mouse and keyboard. Menu actions were provided for quick corrections, with the options sorted in order of likelihood. Hence, the few errors that did require user intervention could usually be corrected with just one quick mouse action. Although no quantitative evaluation was performed of this correction mechanism, it proved to be a considerably more convenient method to correct errors than by having to repeat the misrecognized command verbally. In an operational environment, a man-machine interface using combinations of input modalities, such as the one designed for this research effort, is likely to be preferred over using voice only for corrections. In addition to mouse and keyboard input, touch screen technology may be considered.

7.6 Recommendations for Future Work

The research presented within this paper has demonstrated the feasibility of using ASR technology in the ATC system. The ATC environment and linguistic issues have been described. Technological limitations and human factors problems have been identified. Commercially available ASR equipment has been evaluated. The acquisition and processing of higher levels of knowledge have been shown to improve recognition accuracy and provide a man-machine interface with reduced levels of user interaction. These achievements provide some solutions for the problems facing the introduction of speech recognition technology, but also point out a need for future research.

7.6.1 Operational Testing of ASR Equipment

The evaluation tests conducted within this research effort were all performed under laboratory conditions with a small sample of subjects. Although the subjects were familiar with ATC, none of them were controllers. In order to investigate potential problems such as user acceptance, background noise, syntax deviation, and speech variations, the technology must be tested under operational conditions. Such testing can be conducted in two phases. First, controllers can be invited to the laboratory to perform evaluation tests with the ATC simulator and with the voice input parser. This will allow for an initial judgement of controller reaction to the technology. Problems with syntax deviations can be monitored. The efficiency and user acceptance of the man-machine interface contained in ATCVIP can be investigated.

The second phase would include evaluation tests of commercially available ASR technology in ATC facilities with a higher degree of operational realism. This would allow for the investigation of environmental issues such as background noise and controller mobility. User acceptance, stress induced speech variations, and cognitive workload issues could also be addressed. An attempt should also be made to evaluate the technology with actual controllers and ATC operations. Most likely however, such testing would only be permitted using ATC simulators.

7.6.2 Introduction of Adaptive Training

The promise of adaptive training was demonstrated during the evaluation tests of ASR hardware. Retraining the speech patterns immediately before an evaluation run resulted in a reduction of recognition error from 10.3% to 0.5%. Adaptive training has the potential to improve robustness to gradual speech variations due to temporal drift, fatigue, etc. Verbex has developed preliminary software for the 5000 recognizer that allows for host controlled training. Adaptive recognition could be implemented in the following manner: when a phrase is recognized, the speech input data is stored temporarily. The input is then parsed, and automatic and manual error corrections are performed. Once an hypothesis has been formulated with a high level of likelihood, it could be submitted to the Verbex 5000 together with the recorded speech patterns. The HMM representation of the user's speech patterns could then be updated with this information.

Adaptive training would thus result in the continuous updating of the user's training patterns. This would in turn reflect any changes in the speaker's voice, and would maintain high levels of recognition accuracy. Several topics of investigation remain outstanding, however: to what extent should the most recent patterns be weighted to achieve rapid updating of the speech patterns? Will adaptive training be able to cope with short term speech variations? If a user's voice changes, and then reverts to normal, how can the changed voice patterns be erased from the training data? Future research should address these questions and obtain a quantitative measure of the possible performance improvements due to adaptive training.

7.6.3 Improving ATCVIP

The current version of ATCVIP uses a textual interface and relies on a highly simplified simulation of the airspace. In order to better evaluate the techniques included in ATCVIP, it should be implemented in conjunction with an ATC simulation such as TRACON or the Flight Transportation Laboratory's simulator. The recognition results can be displayed in graphical or symbolic form, adjacent to the aircraft icons on the radar display. The aircraft states should be updated using realistic flight dynamics. Speech synthesis should be added to simulate pilot responses. These responses should include simulated communication errors and incorrect readbacks. Wesson International's TRACON simulator would be a useful basis for such an implementation of ATCVIP. Furthermore, ATCVIP should be improved to allow voice to be used for making corrections, to complement the mouse and keyboard modalities.

Currently, ATCVIP supports the same set of ATC commands as Wesson International's TRACON. Although this set is considerably more extensive than the grammar used by Trikas in his work, it still contains some limitations and non-standard phraseology. Due to its constraint on the branching factor of the syntax, the Verbex 5000 cannot support a more extensive command set than the one currently implemented. If however, more capable equipment is acquired, the grammar should be extended to include a more representative set of a controller's vocabulary and syntax. Furthermore, due to the syntax complexity limit, the current set-up only supports a limited number of airline callsigns, excluding many airlines as well as all general aviation, military, and special purpose callsigns. Once again, with new ASR equipment the grammar should be extended to include a wider variety of callsigns.

7.6.4 The Callsign Recognition Problem

Improving ATCVIP to support a more varied set of callsigns is relatively easy to achieve. However, the problem of recognizing all possible callsigns that can be encountered operationally represents a challenging problem. Callsigns include airlines, general aviation aircraft, military flights, and special purpose identifiers for search and rescue missions, time critical medical transportations, navigation aid calibration flights, etc. Military callsigns alone provide practically an infinite number of possibilities. Furthermore, callsigns occur in both long and short forms, and the numerical part of a call sign can occur as just a list of digits (e.g. four five two) or in group form (e.g. four fifty-two). Hence, the size and branching factor of a syntax for recognizing callsigns alone can become prohibitively large. Other methods to recognize aircraft callsigns must be investigated. One possibility may be to have the controller train each callsign when the aircraft first enters the sector. This would be an extension of the adaptive training feature. As the aircraft leaves the ATC sector, situational knowledge can be used to remove the voice patterns for that particular callsign.

7.6.5 Evaluation of Case Frame Representations

Using rigid syntactic constraints provides valuable knowledge for the parsing of recognition input. The Verbex 5000 takes advantage of this by requiring the use of a syntax for all applications. The syntax is then used to implement a HMM representation of the user's speech patterns. This approach enables the Verbex 5000 to achieve good performance, without suffering from coarticulation problems, while maintaining short recognition delays. However, the use of a syntax can cause problems when a phrase contains even a minor deviation. This is manifested by the Verbex 5000's tendency to reject entire phrases.

An alternate to using a strict syntax implementation may be the use of case frames, where word order is less significant than semantic content. Another technique, word spotting, attempts to recognize specific words in a sentence that may contain them in any order, interspersed with other words which are not present in the recognition vocabulary. These methods should be investigated to overcome the syntax deviation problem. However, some acquisition and processing of syntactic knowledge may have to be retained in order to achieve a sufficiently high recognition accuracy.

7.6.6 Scoring Alternate Hypotheses

ATCVIP relies on syntactic, semantic, and pragmatic processing, as well as Confusion Matrices to implement an automatic error detection and correction feature. If an error is detected, an alternate hypothesis is formulated, and its likelihood is assessed based on the similarity to the textual representation of the recognized phrase, and past input patterns stored in the Confusion Matrices. However, no comparison is done of the phonetic and acoustic characteristics of the alternate and the recognized input. The Verbex 5000 does not make available recognition scores for alternate hypotheses. This limitation is due to the nature of the HMM representation and the search algorithm. The ability to evaluate alternate hypotheses against the user's speech patterns would provide a valuable tool in an operational man-machine interface using voice input. It should be a characteristic of ASR systems acquired for future research on ATC applications.

7.6.7 Evaluating Technology Advancements

Future speech recognizers are likely to provide reduced recognition delays, increased vocabulary sizes, more flexible syntax constraints, and better performing and more robust recognition algorithms. The cost for a given level of performance can also be expected to decline. These technology advancements are fundamental to the success of applying ASR technology to the operational ATC environment. Hence, the development of the technology should be monitored, and new ASR equipment should be acquired and evaluated. In the course of this research project it was found that many manufacturers of speech recognition equipment are eager to accommodate user needs. The performance requirements for ATC applications should be continuously refined, and passed on to the equipment suppliers.

7.6.8 Additional Applications in the ATC System

The emphasis within this paper has been on operational applications within the tower, TRACON, and ARTCC environments. The goal has been to capture the information transmitted by the controllers to the participating aircraft. The potential of using ASR technology in ATC training and simulation systems has also been described. However, there may be other application areas within the ATC system that may benefit from the introduction of speech recognition. Potential candidates include Flight Service Stations, military Precision Approach Radar control, and inter-sector and facility communications. These possible applications should be investigated, and simulation set-ups should be used to determine the feasibility of introducing speech recognition technology.

7.6.9 Continued Human Factors Research

Throughout this research effort, a human factors perspective has been adopted. The critical issue has been identified as the man-machine interface, not the level of the technology. Yet, this research is only an initial step in a cyclic human factors design process. Future work should continue investigation of the more challenging man-machine issues such as speech variations, cognitive workload, automatic error correction, and feedback design. Several of the problems inherent in the ATC environment are also found in other applications where the use of ASR technology is being investigated. Two areas that possess some degree of similarity are fighter crew stations and Airborne Warning and Control Systems (AWACS). Solutions found for these application areas should be examined for their suitability to the ATC system.

Several significant human factors questions remain unanswered. Will there be sufficient motivation for the air traffic controllers to justify the introduction of the technology? If a recognition error requires user intervention, how will the controller's workload and cognitive process be affected? What happens if a controller neglects to correct a recognition error? The technology will develop continuously and achieve better and better recognition performance. However, we will never be able to assume that a perfect recognition capability has been achieved. Therefore, these questions will continue to pose challenges to the human factors engineer. They must be answered before the users of the ATC system can capitalize on the benefits of speech recognition.

BIBLIOGRAPHY

[1] Thomas R. Martin, and John R. Welch, "Practical Speech Recognizers and Some Performance Effective Parameters", *Trends in Speech Recognition*, Prentice Hall, Inc., Englewood Cliffs, N.J. (1980), pp. 24-38.

[2] J. B. Peckham, "Speech Recognition – What is it Worth?", *Proceedings of the 1st International Conference on Speech Technology*, IFS Publications Ltd., Bedford, U.K. (1984), pp. 39-47.

[3] National Research Council, *Automatic Speech Recognition in Severe Environments*, National Academy Press, Washington, D.C. (1984).

[4] Donald W. Connolly, *Voice Data Entry in Air Traffic Control*, National Aviation Facilities Experimental Center, Department of Transportation, Atlantic City, N.J. (August 1979).

[5] Robert F. Hall, *Voice Recognition and Artificial Intelligence in an Air Traffic Control Environment*, Arizona State University, Tempe, AZ (May 1988).

[6] John A. Harrison, G. R. Hobbs, J. R. Howes, and N. Cope, "The Use of Speech Technology in Air Traffic Control Simulators", *Second International Conference on Simulators*, Institution of Electrical Engineers, London, U.K. (1986), pp. 15-19.

[7] G. R. Hobbs, "The Application of Speech Input/Output to Training Simulators", *Proceedings of the 1st International Conference on Speech Technology*, IFS Publications Ltd., Bedford, U.K. (1984), pp. 121-131.

[8] James D. Reierson, and Karol Kerns, *Use of Voice Interactive Technology to Study Effects of Display and Response Modes*, MITRE Corporation, McLean, VA (June 1988).

[9] Thanassis Trikas, *Automated Speech Recognition in Air Traffic Control*, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA (January 1987).

[10] Martin Cooper, "Human Factors Aspects of Voice Input/Output", *Speech Technology*, Vol. 3, No. 4 (March/April 1987), pp. 82-86.

[11] Howard C. Nusbaum, and David B. Pisoni, "Human Factors Issues for the Next Generation of Speech Recognition", *The Official Proceedings of Speech Tech '86*, Media Dimensions, Inc., New York, N.Y. (1986), pp. 140-144.

[12] *Air Traffic Control*, Air Traffic Operations Service, Federal Aviation Administration, Department of Transportation, Washington, D.C. (September 21, 1989).

[13] *Airman's Information Manual: Official Guide to Basic Flight Information and ATC Procedures*, Federal Aviation Administration, Department of Transportation, Washington, D.C. (June 30, 1988).

[14] John M. Fabry, *Research Summary: Federal Aviation Administration Sponsored Small Business Innovation Research Five Year Project Summaries*, Federal Aviation Administration Technical Center, Department of Transportation, Atlantic City, N.J. (November 1987), pp. 28, 33, 54, 63.

[15] Arthur Gerstenfeld, "Speech Recognition Integrated with ATC Simulation", *ICAO Bulletin*, Vol. 45, No. 5 (May 1988), pp. 22-23.

[16] Philip Shinn, *Computer Voice and Speech Entry and Recognition – FAA Phase I SBIR Proposal and Final Report*, Speech Systems, Inc., Tarzana, CA (October 1988).

[17] Philip Shinn, "Making Speech Work – A Summary of Federally Sponsored Research at SSI", *Official Proceedings of Military and Government Speech Tech '89*, Media Dimensions, Inc., New York, N.Y. (1989), pp. 15-19.

[18] Wendy Wylegala, ed., "French Team Designing Voice System for Air Traffic Controller Training", *International Voice Systems Review*, Vol. 1, No. 1 (January/February 1989), p. 18.

[19] Michel M. Sadoune, *Terminal Area Flight Path Generation Using Parallel Constraint Propagation*, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA (May 1989).

[20] Wayne A. Lea, "Speech Recognition: Past, Present, and Future", *Trends in Speech Recognition*, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1980), pp. 39-98.

[21] M. W. Grady, and M. Hicklin, *Use of Computer Speech Understanding in Training: A Demonstration Training System for the Ground Controlled Approach Trainer*, Naval Training Equipment Center, Naval Air Development Center, U.S. Navy, Moffet Field, CA (December 1976).

[22] James D. Reierson, *Software for Demonstrating and Testing the Application of Voice Input/Output to Advanced Air Traffic Control*, MITRE Corporation, McLean, VA (April 1988).

[23] Steven Cushing, *Language and Communication – Related Problems of Aviation Safety*, paper presented at the Annual Meetings of the American Association for Applied Linguistics, San Francisco, CA (December 1987) and the 24th International Congress of Psychology, Sydney, Australia (August 1988).

[24] R. K. Moore, "Overview of Speech Input", *Proceedings of the 1st International Conference on Speech Technology*, IFS Publications Ltd., Bedford, U.K. (1984), pp. 39-98.

[25] Zue, Victor W., "Homomorphic Analysis", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (March 1989).

[26] Zue, Victor W., "Linear Predictive Analysis", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (March 1989).

[27] Victor W. Zue, "Vector Quantization and Dynamic Time Warping", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (March 1989).

[28] Victor W. Zue, "Pattern Recognition (I)", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (April 1989).

[29] Victor W. Zue, "Hidden Markov Modelling (I)", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (April 1989).

[30] Victor W. Zue, "Hidden Markov Modelling (II)", lecture handout for *Automatic Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA (April 1989).

[31] Ann Rollins, and Jennifer Wiesen, "Speech Recognition and Noise", *International Conference on Acoustics, Speech & Signal Processing*, Vol. 2, Institute of Electrical and Electronics Engineers, New York, N.Y. (1983), pp. 523-526.

[32] Martin Cooper, "Human Factors Aspects of Voice Input/Output", *Speech Technology*, Vol. 3, No. 4 (March/April 1987), pp. 82-86.

[33] Harry F. Waller, "Choosing the Right Microphone for Speech Applications", *The Official Proceedings of Speech Tech '85*, Media Dimensions, Inc., New York, N.Y (1985), pp. 45-47.

[34] Frederick Hayes-Roth, "Syntax, Semantics, and Pragmatics in Speech Understanding Systems", *Trends in Speech Recognition*, Prentice-Hall, Inc., Englewood Cliffs, N.J. (1980), pp. 206-333.

[35] Kalyan Ganesan, "A Unified Framework for Representation and Recognition of Speech Using Case Frames", *Speech Technology*, Vol. 3, No. 2 (September/October 1986), pp. 68-71.

[36] Carol A. Simpson, and John C. Ruth, "The Phonetic Discrimination Test for Speech Recognizers: Part I", *Speech Technology*, Vol. 3, No. 4 (March/April 1987), pp. 48-53.

[37] David S. Pallett, "Performance Assessment for Speech Recognizers", *The Official Proceedings of Speech Tech '85*, Media Dimensions, Inc., New York, N.Y. (1985), pp. 162-164.

[38] Joakim Karlsson, *Review of Commercially Available Automatic Speech Recognition Systems for Air Traffic Control Applications*, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA (March 1989).

[39] Earl S. Stein, "Controller Workload: Past – Present – Future", *29th Annual Air Traffic Control Association Fall Conference Proceedings*, Air Traffic Control Association, Arlington, VA (1984), pp. 291-293.

[40] David K. Schmidt, "On Modeling ATC Work Load and Sector Capacity", *Journal of Aircraft*, Vol. 13, No. 7 (July 1976), pp. 531-539.

[41] V. David Hopkin, "Air Traffic Control", *Human Factors in Aviation*, eds. Wiener, Earl L., and Nagel, David C., Academic Press, Inc., San Diego, CA (1988), pp. 639-663.

[42] Frederick Jelinek, *Language Modeling for Speech Recognition*, IBM T. J. Watson Research Center, Yorktown Heights, N.Y. (1989).

[43] J. V. F. Berman, "Speech Technology in a High Workload Environment", *Proceedings of the 1st International Conference on Speech Technology*, IFS Publications Ltd., Bedford, U.K. (1984), pp. 69-76.

[44] Elwyn Edwards, "Introductory Overview", *Human Factors in Aviation*, eds. Wiener, Earl L. and Nagel, David C., Academic Press, Inc., San Diego, CA (1988), pp. 3-25.

[45] K. H. Loken-Kim, "An Investigation of Automatic Error Detection and Correction in Speech Recognition Systems", *The Official Proceedings of Speech Tech '85*, Media Dimensions, Inc., New York, N.Y. (1985), pp. 72-74.

[46] Laurie Fenstermacher, "Voice Synthesis Techniques Applied to Speaker Normalization", paper presented at *Military and Government Speech Tech '89*, Arlington, VA (November 1989).

[47] Robert S. Van Peursem, "Do's and Don'ts of Interactive Voice Dialog Design", *The Official Proceedings of Speech Tech '85*, Media Dimensions, Inc., New York, N.Y. (1985), pp. 48-56.

[48] H. P. Van Colt, and R. G. Kinkade, eds., *Human Engineering Guide to Equipment Design*, U.S. Government Printing Office, Washington, D.C., 1972.

[49] *Flight Deck Panels, Controls, and Displays*, Society of Automotive Engineers, Inc., Warrendale, PA (July 1988).

[50] Betsy J. Constantine, "Human Factors Considerations in the Design of Voice Input/Output Applications", *The Official Proceedings of Speech Tech '84*, Media Dimensions, Inc., New York, N.Y. (1984), pp. 219-224.

[51] Paul A. Mangione, "What About the User?", *The Official Proceedings of Speech Tech '86*, Media Dimensions, Inc., New York, N.Y. (1986), pp. 154-156.

[52] F. Gail James, "Task Analysis as a Means of Determining the Feasibility of Voice Input/Output Applications", *The Official Proceedings of Speech Tech '85*, Media Dimensions, Inc., New York, N.Y. (1985), pp. 45-78.

[53] *VPC 2000 User's Guide*, Votan, Inc., Fremont, CA (1985).

[54] *Verbex Series 5000 Voice Input/Output System User's Guide*, Verbex Voice Systems, Inc., Edison, N.J. (August 1988).

[55] *Verbex Series 5000 Voice I/O System Voice Developer User's Guide*, Verbex Voice Systems, Inc., Edison, N.J. (October 1987).

[56] Robert B. Wesson, and Dale Young, *TRACON Air Traffic Control Simulator*, Wesson International, Austin, TX (1988).

APPENDIX A

THE TRACON/ATCVIP SYNTAX

```
;
; This is an experimental grammar for operation with the Wesson International
; TRACON V1.5 simulation.  Due to the constraints on the complexity of the
; grammar, some sentences can be generated that are not allowed in reality.
; More important is that most allowed forms are supported.
;
; In order to work properly, the utility SOFTKEY must be used to pipe the
; input from the serial port to the keyboard buffer.
;
; Addendum: Due to the constraints on the complexity of the grammar, only
; commercial airline jets are supported.  Callsigns can be shortened, but must
; always include the airline identifier.  Furthermore, compound commands (i.e.
; using "then") are not supported.
;

; Set-up block.

#setup
      custom_training_enable = on
      custom_training_cont_count = 10
      custom_training_screen_limit = off
      host_data_bits = 8
      host_parity = none
      host_stop_bits = 1
      host_baud = 9600
      verbex_wildcard_offset = 100


; Preliminary part of grammar:

!tracon_grammar=
#recognition
#grammar


; Main part of grammar:

.CALLSIGN .COMMAND


; Definitions of grammar structures:

.CALLSIGN=
      .TYPE @0,1 .DIGIT @2,4 heavy @0,1

.TYPE=
      air-canada
;     air-france
      american
;     british
;     caledonian
      clipper
      continental
      delta
      eastern
```

```
;     egypt-air
;     el-al
;     iberia
;     klm
;     kuwait
;     lufthansa
;     midway
;     piedmont
;     sabena
;     saudi-air
;     scandinavian
;     swissair
      trans-world
      united

.FIX=
      arcer
      bosox
      burdy
      celts
      cinky
      drunk
      exalt
      inndy
      lobby
      milis
      rever
      tonni
      waxen
      wimpy
      witch
      woons
      bedford
      beverly
      logan
      norwood

.DIGIT=
      zero
      .NONZERO

.NONZERO=
      one
      two
      three
      four
      five
      six
      seven
      eight
      nine

.COMMAND=
      radar-contact
      released
      turn-left-heading .DIGIT @3
      turn-right-heading .DIGIT @3
      turn-left .DIGIT @2 degrees
      turn-right .DIGIT @2 degrees
      climb-and-maintain .ALTITUDE
      descend-and-maintain .ALTITUDE
      increase-speed-to .NONZERO .DIGIT @2
      reduce-speed-to .NONZERO .DIGIT @2
;     cleared-for-vor-approach contact-tower-at-faf
;      cleared-for-ils-approach  contact-tower-at-faf
       cleared-for-final
       cleared-direct-to  .FIX
       hold-at  .FIX
       say-heading-and-airspeed
       contact-center-g'day
       resume-normal-speed
       resume-normal-navigation
       show-flight-plan
       show-information

.ALTITUDE=
       .THOUSANDS  thousand  .HUNDREDS  @0,1

.THOUSANDS=
       one  zero
       one  one
       one  two
       .NONZERO

.HUNDREDS=
       .NONZERO  hundred


;  Translation  tables  to  simulate  keyboard.

#translations

<  [                              ;  To  distinguish  voice  from  keyboard  input.
|  ||                             ;  No  separator  string.
>  ]                              ;  To  mark  the  end  of  voice  input.

air-canada                        ac
;air-france                       af
american                          aa
;british                          ba
;caledonian                       br
clipper                           pa
continental                       co
delta                             dl
eastern                           ea
;el-al                            ly
;egypt-air                        ms
;iberia                           ib
;klm                              kl
;kuwait                           ku
;lufthansa                        lh
;midway                           ml
;piedmont                         pi
;sabena                           sn
;saudi-air                        su
;scandinavin                      sk
;swissair                         sr
trans-world                       tw
united                            ua
arcer                             arcer            |015]
bosox                             bosox            |015]
burdy                             burdy            |015]
celts                             celts            |015]
cinky                             cinky            |015]
drunk                             drunk            |015]
exalt                             exalt            |015]
inndy                             inndy            |015]
lobby                             lobby            |015]
```

```
milis                          milis          |015]
rever                          rever          |015]
tonni                          tonni          |015]
waxen                          waxen          |015]
wimpy                          wimpy          |015]
witch                          witch          |015]
woons                          woons          |015]
bedford                        bed            |015]
beverly                        bvy            |015]
logan                          bos            |015]
norwood                        owd            |015]
zero                           0              |015]
one                            1              |015]
two                            2              |015]
three                          3              |015]
four                           4              |015]
five                           5              |015]
six                            6              |015]
seven                          7              |015]
eight                          8              |015]
nine                           9              |015]
thousand                       +              0|015] ; Trick for bug.
hundred                        |015
heavy                          !
radar-contact                  |015           |015]
released                       |015|136       |015]
turn-left-heading              |036|064|136
turn-right-heading             |036|066|136
turn-left                      |036|064
turn-right                     |036|066
degrees                        |015
climb-and-maintain             |036|070
descend-and-maintain           |036|062
increase-speed-to              |036|060|136
reduce-speed-to                |036|060
;cleared-for-vor-approach      |036|061
;cleared-for-ils-approach      |036|061
cleared-for-final              |036|061|136
;contact-tower-at-faf          ||
cleared-direct-to              |036|067
hold-at                        |036|063
say-heading-and-airspeed       |036|071
contact-center-g'day           |036|061
resume-normal-speed            |036|056|136
resume-normal-navigation       |036|056
show-flight-plan               |036|106
show-information               |036|111
```

APPENDIX B

THE ATCVIP SOURCE CODE

```
'                                                                  V01.00
' Copyright (C) 1989 by the Massachusetts Institute of Technology.
'
' Air Traffic Control Voice Input Parser (ATCVIP).
'
' Permission to use, copy, and modify this software for internal purposes
' only and without fee is hereby granted provided that the above copyright
' notice and this permission appear on all copies of the code.  For any
' other use of this software, in original or modified form, including but
' not limited to, adaptation as the basis of a commercial software or hardware
' product, or distribution in whole or in part, specific prior permission
' and/or the appropriate license must be obtained from MIT.  This software
' is provided "as is" without any warranties whatsoever, either expressed
' or implied, including but not limited to the implied warranties of
' merchantibility and fitness for a particular purpose.  This software is
' a research program, and MIT does not represent that it is free of errors
' or bugs or suitable for any particular task.
'
'
' Purpose:
'
' Parses voice input arriving through the keyboard buffer.  Format for ATC
' commands is based on Wesson International's TRACON V1.52.  This program
' represents a model of a possible interface between the controller and
' Automatic Speech Recognition (ASR) technology.  It includes the following
' features:
'               * Automatic error detection and correction.
'               * Use of syntactic, semantic, and pragmatic knowledge.
'               * Mixed input modalities (voice, keyboard, mouse).
'
' 1989-10-17    V00.00  Initial development version by Joakim Karlsson.
' 1989-10-25    V01.00  First completed version.  Does not include adaptive
'                         training.  No automatic error processing for
'                         altitudes above 9,900 feet (three digit altitudes).
'                         Also, buffered voice input is not processed.
'
' Editing:      qb atcvip.bas /l qbmouse.qlb
' Compiling:    bc atcvip.bas ;
' Linking:       link atcvip.obj,,,brun45.lib+mouse.lib /e
'


' Initialization:

DECLARE SUB MouseS (m1%, m2%, m3%, m4%)
DECLARE SUB SetState (row%, state%)

TYPE AircraftState
      Callsign AS STRING * 7
      AircraftType AS STRING * 3
      Phase AS STRING * 5
      ActAlt AS INTEGER
      ActSpd AS INTEGER
      FilAlt AS INTEGER
      FilSpd AS INTEGER
      Heading AS INTEGER
END TYPE

CONST TRUE = -1, FALSE = 0, NULL = -1
CONST ACN = 10, ACX = 13, ACLENGTH = 7
CONST CMDN = 18, CMDX = 23, CMDLENGTH = 30
CONST PARMX = 57, PARMLENGTH = 5, PARMMENUN = 25
CONST LOW = 7, HIGH = 8, BLINK = 16
CONST HISTORY = 5, BUFN = 5, MENUN = 21, FIXN = 20
CONST BS = 8, CR = 13, ESC = 27, SPACE = 32, VOICEON = 91, VOICEOFF = 93
CONST HOME = -71, UP = -72, PGUP = -73, LFT = -75, RGT = -77, ENDKEY = -79
CONST DOWN = -80, PGDN = -81, INS = -82, DEL = -83
CONST AL$ = "abcdefghijklmnopqrstuvwxyz!", NUM$ = "1234567890"
CONST STAT$ = "P S C X     "

DIM inpbuffer$(BUFN), scroll$(HISTORY), menu$(MENUN), sort%(MENUN)
DIM parmmenu$(PARMMENUN), parmsort%(PARMMENUN)
DIM states%(HISTORY), acscr%(ACN), cmds$(CMDN)
DIM fixconfus%(FIXN, FIXN), numconfus%(10, 10)
DIM newstate AS AircraftState, acstates(ACN) AS AircraftState

DataStatements:

DATA "ac655  ", "m80", "centr", 10000, 250, 10000, 250, 135
DATA "ua846  ", "727", "preto",   133,   0, 12000, 250, 022
DATA "dl904  ", "m80", "centr", 11000, 250, 11000, 250, 090
DATA "aa16   ", "737", "centr", 10000, 250, 10000, 250, 255
DATA "co544h ", "110", "centr",  9000, 250,  9000, 250, 015
DATA "ea861h ", "747", "centr", 10000, 250, 10000, 250, 005
DATA "co895h ", "110", "centr", 12000, 250, 12000, 250, 212
DATA "co474h ", "ab3", "centr", 11000, 250, 11000, 250, 330
DATA "ea894h ", "768", "centr", 11000, 250, 11000, 250, 010
DATA "tw13   ", "72s", "preto",    20,   0,  9000, 250, 036

DATA "released", "radar contact", "turn left", "turn left heading"
DATA "turn right", "turn right heading", "climb and maintain"
DATA "descend and maintain", "increase speed to", "reduce speed to"
DATA "cleared for final", "contact center g'day", "cleared direct to"
DATA "say heading and airspeed", "hold at", "resume normal speed"
DATA "resume normal navigation", "*** unknown command ***"

DATA "arcer", "bosox", "burdy", "celts", "cinky", "drunk", "exalt", "inndy"
DATA "lobby", "milis", "rever", "tonni", "waxen", "wimpy", "witch", "woons"
DATA "bed  ", "bos  ", "bvy  ", "owd  "

SCREEN 0, , 0, 0: CLS

GOSUB InitMouse
GOSUB InitVariables
GOSUB BuildScreen


' Main loop:

DO
      GOSUB WaitForInput
      IF newinp$ <> "" THEN
            GOSUB HandleKeyboard
      ELSEIF mousepresses% > 0 THEN
            GOSUB HandleMouse
      END IF
LOOP UNTIL exitflag%
```

```
FinishUp:

END


' Mouse initialization:

InitMouse:

     DEF SEG = 0
     mseg = 256 * PEEK(51 * 4 + 3) + PEEK(51 * 4 + 2)
     mousel = 256 * PEEK(51 * 4 + 1) + PEEK(51 * 4) + 2
     IF mseg OR (mousel - 2) THEN
          DEF SEG = mseg
          IF PEEK(mousel - 2) = 207 THEN
               PRINT "Mose Driver Not Found!"
               BEEP
               GOTO FinishUp
          ELSE
               m1% = 0: CALL MouseS(m1%, m2%, m3%, m4%)
          END IF
     ELSE
          PRINT "Mouse Driver Not Found!"
          BEEP
          GOTO FinishUp
     END IF

RETURN


' Initialization of main and utility variables:

InitVariables:

     exitflag% = FALSE: acrec% = FALSE: cmdrec% = FALSE: parmreq% = FALSE
     accol% = LOW: cmdcol% = LOW: parmcol% = LOW
     EMPTY$ = SPACE$(64): current$ = EMPTY$: buffer% = 0
     allac$ = "": allfix$ = ""
     newstate.Phase = SPACE$(5): newstate.ActAlt = NULL
     newstate.ActSpd = NULL: newstate.Heading = NULL
     acind% = 0: state% = 0

     FOR i% = 1 TO ACN
            READ acstates(i%).Callsign, acstates(i%).AircraftType
            READ acstates(i%).Phase, acstates(i%).ActAlt, acstates(i%).ActSpd
            READ acstates(i%).FilAlt, acstates(i%).FilSpd, acstates(i%).Heading
            allac$ = allac$ + acstates(i%).Callsign + ","
     NEXT i%

     FOR i% = 1 TO CMDN
          READ cmds$(i%)
          cmds$(i%) = LEFT$(cmds$(i%) + EMPTY$, CMDLENGTH)
     NEXT i%

     FOR i% = 1 TO FIXN
          READ fix$
          allfix$ = allfix$ + fix$ + ","
     NEXT i%

     FOR i% = 1 TO 5
          scroll$(5) = EMPTY$
     NEXT i%

RETURN
```

```
' Displaying static screen elements:

BuildScreen:

     CLS

     PRINT STRING$(80, "-")
     PRINT "[Current Input]"
     PRINT " Status    Callsign        Command                         Parameter"
     PRINT
     PRINT STRING$(80, "-")
     PRINT "[Input History]"
     LOCATE 12, 1, 0
     PRINT STRING$(80, "-")
     PRINT "[Aircraft States]               Actual                Filed"
     PRINT "Callsign    Type    Phase     Alt       Spd     ";
     PRINT "  Alt       Spd        Heading"
     LOCATE 25, 1, 0
     PRINT STRING$(32, "-"); "| ATCVIP 01.00 |"; STRING$(32, "-");

     GOSUB UpdateStates

RETURN


' Update the display of the aircraft state table:

UpdateStates:

     FOR i% = 1 TO ACN
          LOCATE 14 + i%, 1, 0
          PRINT " "; acstates(i%).Callsign; "    ";
           PRINT acstates(i%).AircraftType; "     "; acstates(i%).Phase;
           PRINT USING "#########"; acstates(i%).ActAlt;
           PRINT USING "########"; acstates(i%).ActSpd;
           PRINT USING "##########"; acstates(i%).FilAlt;
           PRINT USING "########"; acstates(i%).FilSpd;
          PRINT "          ";
           PRINT RIGHT$("00" + LTRIM$(STR$(acstates(i%).Heading)), 3);
          IF i% < ACN THEN PRINT
     NEXT i%

RETURN


' Wait for voice, mouse, or keyboard input:

WaitForInput:

     newinp$ = "": mousepresses% = 0
     m1% = 1: CALL MouseS(m1%, m2%, m3%, m4%)
     m1% = 5: m2% = 0: CALL MouseS(m1%, m2%, m3%, m4%)

     DO
          inp$ = INKEY$
          m1% = 5: m2% = 0: CALL MouseS(m1%, m2%, m3%, m4%)
     LOOP WHILE inp$ = "" AND m2% = 0

     IF m2% > 0 THEN
          mousepresses% = m2%
          IF mousepresses% = 1 THEN
               t1 = TIMER
               DO UNTIL TIMER - t1 > .5
               LOOP
```

```
                m1% = 5: m2% = 0: CALL MouseS(m1%, m2%, m3%, m4%)
                mousepresses% = mousepresses% + m2%
            END IF
            mousex% = 1 + m3% / 8: mousey% = 1 + m4% / 8
        END IF

        m1% = 2: CALL MouseS(m1%, m2%, m3%, m4%)

        IF inp$ <> "" THEN
            cod% = ASC(LEFT$(inp$, 1))
            IF cod% = 0 THEN cod% = -ASC(RIGHT$(inp$, 1))
            'PRINT cod%,
            'IF cod% > 32 THEN PRINT inp$ ELSE PRINT
            SELECT CASE cod%
                CASE VOICEON
                    DO
                        DO
                            inp$ = INKEY$
                        LOOP WHILE inp$ = ""
                        cod% = ASC(LEFT$(inp$, 1))
                        IF cod% = 0 THEN cod% = -ASC(RIGHT$(inp$, 1))
                        'PRINT cod%,
                        'IF cod% > 32 THEN PRINT inp$ ELSE PRINT
                        newinp$ = newinp$ + inp$
                    LOOP UNTIL cod% = VOICEOFF
                CASE SPACE
                    exitflag% = TRUE
                CASE ESC
                    RESTORE DataStatements
                    GOSUB InitVariables
                    GOSUB BuildScreen
                CASE ELSE
            END SELECT
        END IF

RETURN


' Update the aircraft state table:

PushState:

        IF acind% <> 0 AND state% <> 4 THEN
            IF newstate.Phase <> SPACE$(5) THEN
                acstates(acind%).Phase = newstate.Phase
            END IF
            IF newstate.ActAlt <> NULL THEN
                acstates(acind%).ActAlt = newstate.ActAlt
            END IF
            IF newstate.ActSpd <> NULL THEN
                acstates(acind%).ActSpd = newstate.ActSpd
            END IF
            IF newstate.Heading <> NULL THEN
                acstates(acind%).Heading = newstate.Heading
            END IF
        END IF

        GOSUB UpdateStates

RETURN


' Update the command history:
```

```
UpdateHistory:

      SELECT CASE state%

           CASE 1, 2, 4
                 IF current$ <> EMPTY$ THEN
                      FOR i% = HISTORY TO 2 STEP -1
                            scroll$(i%) = scroll$(i% - 1)
                            states%(i%) = states%(i% - 1)
                      NEXT i%
                      scroll$(1) = current$
                      IF state% = 4 THEN
                            states%(1) = 4
                      ELSE
                            states%(1) = 2
                      END IF
                      current$ = EMPTY$
                 END IF

      END SELECT

      GOSUB DisplayHistory

RETURN


' Display the command history:

DisplayHistory:

      FOR i% = 1 TO HISTORY
            LOCATE 6 + i%, 1, 0: PRINT scroll$(i%)
            CALL SetState(6 + i%, states%(i%))
      NEXT i%

RETURN


' Keyboard input handler:

HandleKeyboard:

      IF acreq% OR cmdreq% OR parmreq% THEN
            GOSUB PushOnBuffer
      ELSE
            GOSUB PushState
            state% = 1
            GOSUB UpdateHistory
            GOSUB GetBestAc
            GOSUB GetBestCmd
            GOSUB GetBestParm
            GOSUB UpdateCurrent
      END IF

RETURN


' Mouse input handler:

HandleMouse:

      'LOCATE 1, 1: PRINT SPACE$(80);
      'LOCATE 1, 1: PRINT mousepresses%, mousex%, mousey%
```

```
IF mousey% = 4 AND current$ <> EMPTY$ THEN

    SELECT CASE mousex%

        CASE 3
            GOSUB PushState
            state% = 2
            GOSUB UpdateHistory
            LOCATE 4, 1: PRINT current$;

        CASE 7
            state% = 4
            GOSUB UpdateHistory
            LOCATE 4, 1: PRINT current$;

        CASE ACX TO ACX + ACLENGTH - 1
            IF mousepresses% = 1 THEN
                GOSUB EditAircraft
            ELSEIF mousepresses% = 2 THEN
                menulength% = ACN + 1
                menuwidth% = ACLENGTH
                menux% = ACX
                menuy% = 4
                orgac$ = LEFT$(orgac$ + SPACE$(ACLENGTH), ACLENGTH)
                FOR i% = 1 TO ACN
                    menu$(i%) = acstates(i%).Callsign
                    sort%(i%) = acscr%(i%)
                    IF menu$(i%) = orgac$ THEN menulength% = ACN
                NEXT i%
                IF accol% <> LOW THEN
                    DO
                        swaps% = FALSE
                        FOR i% = 1 TO ACN - 1
                            IF sort%(i%) < sort%(i% + 1) THEN
                                SWAP sort%(i%), sort%(i% + 1)
                                SWAP menu$(i%), menu$(i% + 1)
                                swaps% = TRUE
                            END IF
                        NEXT i%
                    LOOP WHILE swaps%
                END IF
                IF menulength% > ACN THEN
                    menu$(menulength%) = orgac$
                END IF
                GOSUB HandleMenu
                IF menuchoice% > 0 THEN
                    ac$ = RTRIM$(menu$(menuchoice%))
                    acind% = 0: accol% = LOW: acreq% = FALSE
                    COLOR accol%, 0
                    acpos% = INSTR(allac$, ac$)
                    rest$ = RIGHT$(allac$, LEN(allac$) - acpos%)
                    IF acpos% <> 0 THEN
                        IF INSTR(rest$, ac$) = 0 THEN
                            acind% = 1 + INT(acpos% / 8)
                        END IF
                    END IF
                    cmdstart% = oldcmdstart%
                    newinp$ = oldnewinp$
                    state% = 1
                    GOSUB GetBestCmd
                    GOSUB GetBestParm
                    GOSUB UpdateCurrent
                END IF
            END IF
```

```
CASE CMDX TO CMDX + CMDLENGTH - 1
     IF mousepresses% > 0 THEN
          FOR i% = 1 TO CMDN
               menu$(i%) = cmds$(i%)
          NEXT i%
          menulength% = CMDN
          menuwidth% = CMDLENGTH
          menux% = CMDX
          menuy% = 4
          GOSUB HandleMenu
          cmdi$ = ""
          SELECT CASE menuchoice%
               CASE 1
                    cmdi$ = CHR$(CR) + CHR$(CR) + "^"
               CASE 2
                    cmdi$ = CHR$(CR) + CHR$(CR)
               CASE 3
                    cmdi$ = CHR$(0) + CHR$(ABS(LFT))
               CASE 4
                    cmdi$ = CHR$(0) + CHR$(ABS(LFT)) + "^"
               CASE 5
                    cmdi$ = CHR$(0) + CHR$(ABS(RGT))
               CASE 6
                    cmdi$ = CHR$(0) + CHR$(ABS(RGT)) + "^"
               CASE 7
                    cmdi$ = CHR$(0) + CHR$(ABS(UP))
               CASE 8
                    cmdi$ = CHR$(0) + CHR$(ABS(DOWN))
               CASE 9
                    cmdi$ = CHR$(0) + CHR$(ABS(INS)) + "^"
               CASE 10
                    cmdi$ = CHR$(0) + CHR$(ABS(INS))
               CASE 11
                    cmdi$ = CHR$(0) + CHR$(ABS(ENDKEY)) + "^"
               CASE 12
                    cmdi$ = CHR$(0) + CHR$(ABS(ENDKEY))
               CASE 13
                    cmdi$ = CHR$(0) + CHR$(ABS(HOME))
               CASE 14
                    cmdi$ = CHR$(0) + CHR$(ABS(PGUP))
               CASE 15
                    cmdi$ = CHR$(0) + CHR$(ABS(PGDN))
               CASE 16
                    cmdi$ = CHR$(0) + CHR$(ABS(DEL)) + "^"
               CASE 17
                    cmdi$ = CHR$(0) + CHR$(ABS(DEL))
               CASE 18
                    cmdi$ = "*"
          END SELECT
          IF cmdi$ <> "" THEN
               newinp$ = oldnewinp$
               nil$ = LEFT$(newinp$, oldcmdstart% - 1)
               nir$ = MID$(newinp$, cmdstart% + 1, 100)
               newinp$ = nil$ + cmdi$ + nir$
               cmdstart% = oldcmdstart%
               state% = 1
               GOSUB GetBestCmd
               cmdreq% = FALSE: cmdcol% = LOW
               GOSUB ParmCheck
               GOSUB UpdateCurrent
               IF parmcheckflag% THEN
                    GOSUB EditParameter
               ELSE
                    GOSUB GetBestParm
                    END IF
                    GOSUB UpdateCurrent
                END IF
            END IF

        CASE PARMX TO PARMX + PARMLENGTH - 1
            IF RTRIM$(parm$) <> "" THEN
                IF mousepresses% = 1 THEN
                    m1% = INSTR(allfix$, RTRIM$(parm$))
                    oldparm$ = parm$
                    l1% = LEN(RTRIM$(parm$))
                    GOSUB EditParameter
                    IF oldparm$ <> parm$ THEN
                        parmcol% = LOW: parmreq% = FALSE
                        GOSUB EvalParm
                        GOSUB UpdateCurrent
                    END IF
                    m2% = INSTR(allfix$, RTRIM$(parm$))
                    IF m1% <> 0 THEN m1% = INT(m1% / 6) + 1
                    IF m2% <> 0 THEN m2% = INT(m2% / 6) + 1
                    IF m1% <> m2% AND m1% > 0 AND m2% > 0 THEN
                        p% = fixconfus%(m1%, m2%)
                        p% = p% + 1
                        fixconfus%(m1%, m2%) = p%
                    END IF
                    l2% = LEN(RTRIM$(parm$))
                    IF l1% = l2% THEN
                        FOR i% = 1 TO l1%
                            c$ = MID$(oldparm$, i%, 1)
                            i1% = INSTR(NUM$, c$)
                            c$ = MID$(parm$, i%, 1)
                            j% = INSTR(NUM$, c$)
                            IF i1% = 0 OR j% = 0 THEN
                                i1% = 0: j% = 0
                            END IF
                            IF i1% <> j% THEN
                                p% = numconfus%(i1%, j%)
                                p% = p% + 1
                                numconfus%(i1%, j%) = p%
                            END IF
                        NEXT i%
                    END IF
                ELSEIF mousepresses% = 2 AND parmmenul% > 0 THEN
                    FOR i% = 1 TO parmmenul%
                        menu$(i%) = parmmenu$(i%)
                    NEXT i%
                    menulength% = parmmenul%
                    menuwidth% = LEN(menu$(1))
                    menux% = PARMX
                    menuy% = 4
                    GOSUB HandleMenu
                    IF menuchoice% > 0 THEN
                        orgparm$ = RTRIM$(parm$)
                        parm$ = menu$(menuchoice%)
                        IF orgparm$ <> parm$ THEN
                            parmcol% = LOW: parmreq% = FALSE
                        END IF
                        m1% = INSTR(allfix$, orgparm$)
                        m2% = INSTR(allfix$, parm$)
                        IF m1% <> 0 THEN m1% = INT(m1% / 6) + 1
                        IF m2% <> 0 THEN m2% = INT(m2% / 6) + 1
                        IF m1% <> m2% AND m1% > 0 AND m2% > 0 THEN
                            p% = fixconfus%(m1%, m2%)
                            p% = p% + 1
```

```
                                        fixconfus%(m1%, m2%) = p%
                                   END IF
                                   l1% = LEN(orgparm$)
                                   l2% = LEN(parm$)
                                   IF l1% = l2% THEN
                                        FOR i% = 1 TO l1%
                                             c$ = MID$(orgparm$, i%, 1)
                                             i1% = INSTR(NUM$, c$)
                                             c$ = MID$(parm$, i%, 1)
                                             j% = INSTR(NUM$, c$)
                                             IF i1% = 0 OR j% = 0 THEN
                                                  i1% = 0: j% = 0
                                             END IF
                                             IF i1% <> j% THEN
                                                  p% = numconfus%(i1%, j%)
                                                  p% = p% + 1
                                                  numconfus%(i1%, j%) = p%
                                             END IF
                                        NEXT i%
                                   END IF
                                   GOSUB EvalParm
                                   GOSUB UpdateCurrent
                              END IF
                         END IF
                    END IF

               CASE ELSE

          END SELECT

     END IF

RETURN


' Menu input handler:

HandleMenu:

     SCREEN 0, , 1, 0

     GOSUB BuildScreen
     GOSUB DisplayHistory
     GOSUB UpdateCurrent

     COLOR 0, LOW

     LOCATE menuy% - 1, menux% - 2
     PRINT "/" + STRING$(menuwidth% + 2, "-") + "\";
     FOR i% = 1 TO menulength%
          LOCATE menuy% + i% - 1, menux% - 2
          PRINT "| " + menu$(i%) + " |";
     NEXT i%
     LOCATE menuy% + menulength%, menux% - 2
     PRINT "\" + STRING$(menuwidth% + 2, "-") + "/";

     m1% = 5: m2% = 0: CALL MouseS(m1%, m2%, m3%, m4%)

     SCREEN 0, , 1, 1

     menuind% = 0: menuchoice% = 0
     menuexit% = FALSE
     m1% = 1: CALL MouseS(m1%, m2%, m3%, m4%)
     DO
          m1% = 3: CALL MouseS(m1%, m2%, m3%, m4%)
          mousex% = 1 + m3% / 8: mousey% = 1 + m4% / 8
          m1% = 5: m2% = 0: CALL MouseS(m1%, m2%, m3%, m4%)
          mousepresses% = m2%
          IF mousepresses% > 0 THEN
               mousex% = 1 + m3% / 8: mousey% = 1 + m4% / 8
          END IF
          IF mousey% < menuy% OR mousey% > menuy% + menulength% - 1 THEN
               menuexit% = TRUE
          ELSEIF mousex% < menux% - 1 OR mousex% > menux% + menuwidth% THEN
               menuexit% = TRUE
          ELSEIF mousey% <> menuy% + menuind% - 1 THEN
               m1% = 2: CALL MouseS(m1%, m2%, m3%, m4%)
               IF menuind% <> 0 THEN
                    LOCATE menuy% + menuind% - 1, menux% - 1, 0
                    COLOR 0, LOW: PRINT " " + menu$(menuind%) + " ";
               END IF
               menuind% = mousey% - menuy% + 1
               LOCATE mousey%, menux% - 1, 0
               COLOR LOW, 0: PRINT " " + menu$(menuind%) + " ";
               m1% = 1: CALL MouseS(m1%, m2%, m3%, m4%)
          END IF
          IF mousepresses% > 0 AND NOT menuexit% THEN
               menuchoice% = menuind%
               menuexit% = TRUE
          END IF
     LOOP UNTIL menuexit% OR INKEY$ = CHR$(ESC)

     m1% = 2: CALL MouseS(m1%, m2%, m3%, m4%)

     SCREEN 0, , 0, 0

RETURN


' Direct entry of aircraft callsign using keyboard:

EditAircraft:

     LOCATE 4, ACX: COLOR 0, LOW
     editac$ = SPACE$(ACLENGTH): PRINT editac$;
     curs% = 0: exitedac% = FALSE

     DO
          LOCATE 4, ACX + curs%, 1
          DO
               inp$ = INKEY$
          LOOP UNTIL LEN(inp$) = 1
          IF INSTR(AL$ + NUM$, inp$) <> 0 THEN
               MID$(editac$, curs% + 1, 1) = inp$
               IF curs% + 1 < ACLENGTH THEN curs% = curs% + 1
               LOCATE 4, ACX, 0: PRINT editac$;
          ELSEIF ASC(inp$) = BS THEN
               IF curs% > 0 THEN
                    IF MID$(editac$, ACLENGTH, 1) <> " " THEN
                         MID$(editac$, ACLENGTH, 1) = " "
                    ELSE
                         MID$(editac$, curs%, 1) = " "
                         curs% = curs% - 1
                    END IF
                    LOCATE 4, ACX, 0: PRINT editac$;
               END IF
          ELSEIF ASC(inp$) = CR OR ASC(inp$) = ESC THEN
```

```
                exitedac% = TRUE
            END IF
        LOOP UNTIL exitedac%

        IF ASC(inp$) = ESC THEN
            GOSUB UpdateCurrent
        ELSEIF ASC(inp$) = CR THEN
            ac$ = RTRIM$(editac$)
            orgac$ = ac$
            acind% = 0: accol% = LOW: acreq% = FALSE
            COLOR accol%, 0
            acpos% = INSTR(allac$, ac$)
            IF acpos% <> 0 THEN
                rest$ = RIGHT$(allac$, LEN(allac$) - acpos%)
                IF INSTR(rest$, ac$) = 0 THEN
                    acind% = 1 + INT(acpos% / 8)
                END IF
            END IF
            cmdstart% = oldcmdstart%
            newinp$ = oldnewinp$
            state% = 1
            GOSUB GetBestCmd
            GOSUB GetBestParm
            GOSUB UpdateCurrent
        END IF

RETURN


' Direct entry of parameter value using keyboard:

EditParameter:

        LOCATE 4, PARMX: COLOR LOW, 0: PRINT SPACE$(5)
        LOCATE 4, PARMX: COLOR 0, LOW

        SELECT CASE cmdind%
            CASE 3, 5: parml% = 2: allow$ = NUM$
            CASE 4, 6, 7, 8, 9, 10: parml% = 3: allow$ = NUM$
            CASE 13, 15: parml% = 5: allow$ = AL$
        END SELECT
        editparm$ = SPACE$(parml%): PRINT editparm$;
        curs% = 0: exitedparm% = FALSE

        DO
            LOCATE 4, PARMX + curs%, 1
            DO
                inp$ = INKEY$
            LOOP UNTIL LEN(inp$) = 1
            IF INSTR(allow$, inp$) <> 0 THEN
                MID$(editparm$, curs% + 1, 1) = inp$
                IF curs% + 1 < parml% THEN curs% = curs% + 1
                LOCATE 4, PARMX, 0: PRINT editparm$;
            ELSEIF ASC(inp$) = BS THEN
                IF curs% > 0 THEN
                    IF MID$(editparm$, parml%, 1) <> " " THEN
                        MID$(editparm$, parml%, 1) = " "
                    ELSE
                        MID$(editparm$, curs%, 1) = " "
                        curs% = curs% - 1
                    END IF
                    LOCATE 4, PARMX, 0: PRINT editparm$;
                END IF
            ELSEIF ASC(inp$) = CR OR ASC(inp$) = ESC THEN
```

```
                exitedparm% = TRUE
            END IF
        LOOP UNTIL exitedparm%

        IF ASC(inp$) = ESC THEN
            GOSUB UpdateCurrent
        ELSEIF ASC(inp$) = CR THEN
            parm$ = RTRIM$(editparm$)
            orgparm$ = parm$
            GOSUB EvalParm
            parmcol% = LOW: parmreq% = FALSE
            COLOR parmcol%, 0
            state% = 1
            GOSUB UpdateCurrent
        END IF

RETURN


' Find the best aircraft callsign match:

GetBestAc:

        ac$ = "": acreq% = FALSE
        FOR i% = 1 TO LEN(newinp$)
            IF INSTR(AL$ + NUM$, MID$(newinp$, i%, 1)) <> 0 THEN
                IF MID$(newinp$, i%, 1) <> "!" THEN
                    ac$ = ac$ + MID$(newinp$, i%, 1)
                ELSE
                    ac$ = ac$ + "h"
                END IF
            ELSE
                EXIT FOR
            END IF
        NEXT i%

        orgac$ = ac$

        cmdstart% = i%

        acind% = 0: accol% = LOW
        acpos% = INSTR(allac$, ac$)
        IF acpos% = 0 THEN
            GOSUB FindBestAcMatch
        ELSE
            rest$ = RIGHT$(allac$, LEN(allac$) - acpos%)
            IF INSTR(rest$, ac$) = 0 THEN
                acind% = 1 + INT(acpos% / 8)
            ELSE
                GOSUB FindBestAcMatch
            END IF
        END IF

RETURN


' Aircraft callsign pattern matcher:

FindBestAcMatch:

        acl% = LEN(ac$)

        IF RIGHT$(ac$, 1) = "h" THEN
            heavy% = TRUE
```

```
      acl% = acl% - 1
      ac$ = LEFT$(ac$, acl%)
ELSE
      heavy% = FALSE
END IF

IF VAL(ac$) = 0 THEN
      airline$ = LEFT$(ac$, 2)
      acl% = acl% - 2
      ac$ = RIGHT$(ac$, acl%)
ELSE
      airline$ = ""
END IF

IF heavy% THEN
      acl% = acl% + 1
      ac$ = ac$ + "h"
END IF

tie% = FALSE: maxscore% = 0

FOR c% = 1 TO ACN

       test1$ = LEFT$(acstates(c%).Callsign, 2)
       test2$ = RTRIM$(RIGHT$(acstates(c%).Callsign, 5))

      acscr%(c%) = 0

       IF airline$ = test1$ THEN acscr%(c%) = acscr%(c%) + 9

       IF INSTR(test2$, ac$) <> 0 THEN
             acscr%(c%) = acscr%(c%) + 4 * acl%
      ELSE
            FOR i% = 1 TO acl%
                  FOR j% = 1 TO LEN(test2$)
                        IF MID$(test2$, j%, 1) = MID$(ac$, i%, 1) THEN
                              dist% = 2 * ABS(i% - j%)
                              IF dist% = 0 THEN dist% = 1
                              acscr%(c%) = acscr%(c%) + 4 / dist%
                        END IF
                  NEXT j%
            NEXT i%
      END IF

       IF acscr%(c%) > maxscore% THEN
            maxscore% = acscr%(c%)
            acind% = c%
            tie% = FALSE
       ELSEIF acscr%(c%) = maxscore% THEN
            tie% = TRUE
      END IF

NEXT c%

IF NOT tie% AND acind% <> 0 THEN
       ac$ = RTRIM$(acstates(acind%).Callsign)
      accol% = LOW + HIGH
ELSE
      ac$ = airline$ + ac$
      accol% = LOW + HIGH + BLINK
      acind% = 0
      acreq% = TRUE
END IF
```

```
RETURN


' Find the best command match:

GetBestCmd:

      cmdreq% = FALSE: cmdcol% = LOW

      cmdcod% = ASC(MID$(newinp$, cmdstart%, 1))
      oldcmdstart% = cmdstart%: oldnewinp$ = newinp$

      IF cmdcod% = 0 THEN
            cmdstart% = cmdstart% + 1
            cmdcod% = -ASC(MID$(newinp$, cmdstart%, 1))
      END IF

      option$ = MID$(newinp$, cmdstart% + 1, 1)
      IF option$ = "^" THEN cmdstart% = cmdstart% + 1
      IF option$ = CHR$(CR) AND cmdcod% = CR THEN
            option$ = MID$(newinp$, cmdstart% + 2, 1)
            IF option$ = "^" THEN cmdstart% = cmdstart% + 2
      END IF


      newstate.Phase = SPACE$(5)

      cmdind% = 18

      SELECT CASE cmdcod%

           CASE CR
                 IF option$ = "^" THEN
                       cmdind% = 1
                 ELSE
                       cmdind% = 2
                 END IF
                 IF acind% <> 0 THEN
                       SELECT CASE acstates(acind%).Phase
                             CASE "preto", "centr"
                                   newstate.Phase = "enrte"
                             CASE ELSE
                                   cmdreq% = TRUE
                       END SELECT
                 END IF

           CASE LFT
                 cmdind% = 3
                 IF option$ = "^" THEN
                       cmdind% = 4
                 END IF
                 IF acind% <> 0 THEN
                       SELECT CASE acstates(acind%).Phase
                             CASE "preto", "lnded", "hndof"
                                   cmdreq% = TRUE
                             CASE ELSE
                       END SELECT
                 END IF

           CASE RGT
                 cmdind% = 5
                 IF option$ = "^" THEN
                       cmdind% = 6
                 END IF
      IF acind% <> 0 THEN
            SELECT CASE acstates(acind%).Phase
                 CASE "preto", "lnded", "hndof"
                      cmdreq% = TRUE
                CASE ELSE
           END SELECT
      END IF

 CASE UP
      cmdind% = 7
      IF acind% <> 0 THEN
            SELECT CASE acstates(acind%).Phase
                 CASE "preto", "lnded", "hndof"
                      cmdreq% = TRUE
                CASE ELSE
           END SELECT
      END IF

 CASE DOWN
      cmdind% = 8
      IF acind% <> 0 THEN
            SELECT CASE acstates(acind%).Phase
                 CASE "preto", "lnded", "hndof"
                      cmdreq% = TRUE
                CASE ELSE
           END SELECT
      END IF

 CASE INS
      IF option$ = "^" THEN
           cmdind% = 9
      ELSE
           cmdind% = 10
      END IF
      IF acind% <> 0 THEN
            SELECT CASE acstates(acind%).Phase
                 CASE "preto", "lnded", "hndof"
                      cmdreq% = TRUE
                CASE ELSE
           END SELECT
      END IF

 CASE ENDKEY
      IF option$ = "^" THEN
           cmdind% = 11
      ELSE
           cmdind% = 12
      END IF
      IF acind% <> 0 THEN
            SELECT CASE acstates(acind%).Phase
                 CASE "enrte"
                      IF option$ = "^" THEN
                            newstate.Phase = "lnded"
                      ELSE
                            newstate.Phase = "hndof"
                      END IF
                 CASE ELSE
                      cmdreq% = TRUE
           END SELECT
      END IF

 CASE HOME
      cmdind% = 13
      IF acind% <> 0 THEN
```

```
                    SELECT CASE acstates(acind%).Phase
                         CASE "preto", "lnded", "hndof"
                              cmdreq% = TRUE
                         CASE ELSE
                    END SELECT
               END IF

          CASE PGUP
               cmdind% = 14
               IF acind% <> 0 THEN
                    SELECT CASE acstates(acind%).Phase
                         CASE "hndof"
                              cmdreq% = TRUE
                         CASE ELSE
                    END SELECT
               END IF

          CASE PGDN
               cmdind% = 15
               IF acind% <> 0 THEN
                    SELECT CASE acstates(acind%).Phase
                         CASE "preto", "lnded", "hndof"
                              cmdreq% = TRUE
                         CASE ELSE
                    END SELECT
               END IF

          CASE DEL
               IF option$ = "^" THEN
                    cmdind% = 16
               ELSE
                    cmdind% = 17
               END IF
               IF acind% <> 0 THEN
                    SELECT CASE acstates(acind%).Phase
                         CASE "enrte"
                         CASE ELSE
                              cmdreq% = TRUE
                    END SELECT
               END IF

          CASE ELSE
               cmdreq% = TRUE

     END SELECT

     cmd$ = RTRIM$(cmds$(cmdind%))

     IF cmdreq% THEN cmdcol% = LOW + HIGH + BLINK

RETURN


' Find the best parameter match:

GetBestParm:

     ERASE parmsort%

     parm$ = "": allm$ = "": parmreq% = FALSE: parmcol% = LOW: parmmenul% = 0

     IF cmdstart% < LEN(newinp$) THEN
          FOR j% = cmdstart% + 1 TO LEN(newinp$)
               char$ = MID$(newinp$, j%, 1)
```

```
                IF INSTR(AL$ + NUM$, char$) <> 0 AND char$ <> "+" THEN
                    parm$ = parm$ + char$
                END IF
            NEXT j%
        END IF

        orgparm$ = parm$

        numflag% = TRUE
        FOR i% = 1 TO LEN(parm$)
            IF INSTR(NUM$, MID$(parm$, i%, 1)) = 0 THEN
                numflag% = FALSE
                EXIT FOR
            END IF
        NEXT i%

        IF LEN(parm$) = 0 THEN numflag% = FALSE

        IF numflag% THEN
            first$ = LEFT$(parm$, 1)
            middle$ = ""
            IF LEN(parm$) > 2 THEN middle$ = MID$(parm$, 2, 1)
            last$ = RIGHT$(parm$, 1)
        END IF

        IF LEN(parm$) <> 0 AND NOT numflag% THEN
            fixind% = INSTR(allfix$, parm$)
            IF fixind% <> 0 THEN fixind% = 1 + INT(fixind% / 6)
        ELSE
            fixind% = 0
        END IF

        SELECT CASE cmdind%

            CASE 1, 2, 11, 12, 14, 16, 17, 18
                parm$ = SPACE$(5)

            CASE 13, 15
                IF fixind% <> 0 THEN
                    parmmenul% = FIXN
                    FOR i% = 1 TO FIXN
                        parmmenu$(i%) = MID$(allfix$, i% * 6 - 5, 5)
                        allm$ = allm$ + parmmenu$(i%) + ","
                        parmsort%(i%) = fixconfus%(fixind%, i%)
                    NEXT i%
                    DO
                        swaps% = FALSE
                        FOR i% = 1 TO FIXN - 1
                            IF parmsort%(i%) < parmsort%(i% + 1) THEN
                                SWAP parmsort%(i%), parmsort%(i% + 1)
                                SWAP parmmenu$(i%), parmmenu$(i% + 1)
                                swaps% = TRUE
                            END IF
                        NEXT i%
                    LOOP WHILE swaps%
                ELSE
                    parmmenul% = FIXN + 1
                    FOR i% = 1 TO FIXN
                        parmmenu$(i%) = MID$(allfix$, i% * 6 - 5, 5)
                        allm$ = allm$ + parmmenu$(i%) + ","
                    NEXT i%
                    parmmenu$(i%) = LEFT$(parm$ + SPACE$(5), 5)
                    allm$ = allm$ + parmmenu$(i%) + ","
                END IF
```

```
CASE 3, 5
      IF numflag% AND LEN(parm$) = 2 THEN
           IF last$ <> "5" AND last$ <> "0" THEN
                parmmenu$(1) = first$ + "0"
                parmmenu$(2) = first$ + "5"
                allm$ = parmmenu$(1) + "," + parmmenu$(2) + ","
                parmmenul% = 2
                c5% = numconfus%(INSTR(NUM$, last$), 5)
                c0% = numconfus%(INSTR(NUM$, last$), 10)
                IF c5% <> c0% THEN
                     IF c5% > c0% THEN
                           SWAP parmmenu$(1), parmmenu$(2)
                           parm$ = first$ + "5"
                     ELSE
                           parm$ = first$ + "0"
                     END IF
                     parmcol% = LOW + HIGH
                ELSEIF c5% > 0 AND c0% > 0 THEN
                     parmcol% = LOW + HIGH + BLINK
                     parmreq% = TRUE
                END IF
           END IF
           FOR i% = 1 TO 10
                j% = i% + parmmenul%
                parmmenu$(j%) = MID$(NUM$, i%, 1) + last$
                allm$ = allm$ + parmmenu$(j%) + ","
                parmsort%(i%) = numconfus%(INSTR(NUM$, first$), i%)
           NEXT i%
           DO
                swaps% = FALSE
                FOR i% = 1 TO 9
                     j% = i% + parmmenul%
                     IF parmsort%(i%) < parmsort%(i% + 1) THEN
                           SWAP parmsort%(i%), parmsort%(i% + 1)
                           SWAP parmmenu$(j%), parmmenu$(j% + 1)
                           swaps% = TRUE
                     END IF
                NEXT i%
           LOOP WHILE swaps%
           parmmenul% = parmmenul% + 10
      END IF

CASE 4, 6
      IF numflag% AND LEN(parm$) = 3 THEN
           IF last$ <> "5" AND last$ <> "0" THEN
                parmmenu$(1) = first$ + middle$ + "0"
                parmmenu$(2) = first$ + middle$ + "5"
                allm$ = parmmenu$(1) + "," + parmmenu$(2) + ","
                parmmenul% = 2
                c5% = numconfus%(INSTR(NUM$, last$), 5)
                c0% = numconfus%(INSTR(NUM$, last$), 10)
                IF c5% <> c0% THEN
                     IF c5% > c0% THEN
                           SWAP parmmenu$(1), parmmenu$(2)
                           parm$ = first$ + middle$ + "5"
                     ELSE
                           parm$ = first$ + middle$ + "0"
                     END IF
                     parmcol% = LOW + HIGH
                ELSEIF c5% > 0 AND c0% > 0 THEN
                     parmcol% = LOW + HIGH + BLINK
                     parmreq% = TRUE
                END IF
           END IF
```

```
 find% = INSTR(NUM$, first$)
 parmsort%(1) = numconfus%(find%, 0)
 parmmenu$(1 + parmmenul%) = "0" + middle$ + last$
 parmsort%(2) = numconfus%(find%, 1)
 parmmenu$(2 + parmmenul%) = "1" + middle$ + last$
 parmsort%(3) = numconfus%(find%, 2)
 parmmenu$(3 + parmmenul%) = "2" + middle$ + last$
addp% = 3
 IF VAL(middle$ + last$) <= 60 THEN
        parmsort%(4) = numconfus%(find%, 3)
        parmmenu$(4 + parmmenul%) = "3" + middle$ + last$
       addp% = 4
END IF
FOR i% = 1 TO addp%
       allm$ = allm$ + parmmenu$(i% + parmmenul%) + ","
NEXT i%
DO
      swaps% = FALSE
      FOR i% = 1 TO addp% - 1
            j% = i% + parmmenul%
             IF parmsort%(i%) < parmsort%(i% + 1) THEN
                   SWAP parmsort%(i%), parmsort%(i% + 1)
                   SWAP parmmenu$(j%), parmmenu$(j% + 1)
                  swaps% = TRUE
            END IF
      NEXT i%
LOOP WHILE swaps%
parmmenul% = parmmenul% + addp%
IF VAL(first$) > 3 THEN
       IF parmsort%(1) > parmsort%(2) THEN
             parm$ = parmmenu$(parmmenul% - addp% + 1)
            parmcol% = LOW + HIGH
      ELSE
            parmcol% = LOW + HIGH + BLINK
      END IF
ELSE
      addp% = 0
       mind% = INSTR(NUM$, middle$)
      FOR i% = 1 TO 10
            p$ = first$ + MID$(NUM$, i%, 1) + last$
            IF VAL(p$) <= 360 THEN
                  allm$ = allm$ + p$ + ","
                  addp% = addp% + 1
                   parmmenu$(parmmenul% + addp%) = p$
                   parmsort%(i%) = numconfus%(mind%, i%)
            END IF
      NEXT i%
      DO
            swaps% = FALSE
            FOR i% = 1 TO addp% - 1
                  j% = i% + parmmenul%
                   IF parmsort%(i%) < parmsort%(i% + 1) THEN
                         SWAP parmsort%(i%), parmsort%(i% + 1)
                         SWAP parmmenu$(j%), parmmenu$(j% + 1)
                        swaps% = TRUE
                  END IF
            NEXT i%
      LOOP WHILE swaps%
      parmmenul% = parmmenul% + addp%
      IF addp% > 0 AND VAL(parm$) > 360 THEN
            IF parmsort%(1) > parmsort%(2) THEN
                  parm$ = parmmenu$(parmmenul% - addp% + 1)
                 parmcol% = LOW + HIGH
           ELSE
```

```
                        parmcol% = LOW + HIGH + BLINK
                    END IF
                END IF
            END IF

        END IF

    CASE 7, 8
        IF numflag% AND LEN(parm$) = 2 THEN
            IF last$ <> "5" AND last$ <> "0" THEN
                parmmenu$(1) = first$ + "0"
                parmmenu$(2) = first$ + "5"
                allm$ = parmmenu$(1) + "," + parmmenu$(2) + ","
                parmmenul% = 2
                c5% = numconfus%(INSTR(NUM$, last$), 5)
                c0% = numconfus%(INSTR(NUM$, last$), 10)
                IF c5% <> c0% THEN
                    IF c5% > c0% THEN
                        SWAP parmmenu$(1), parmmenu$(2)
                        parm$ = first$ + "5"
                    ELSE
                        parm$ = first$ + "0"
                    END IF
                    parmcol% = LOW + HIGH
                ELSEIF c5% > 0 AND c0% > 0 THEN
                    parmcol% = LOW + HIGH + BLINK
                    parmreq% = TRUE
                END IF
            END IF
            find% = INSTR(NUM$, first$)
            FOR i% = 1 TO 10
                p$ = MID$(NUM$, i%, 1) + last$
                allm$ = allm$ + p$ + ","
                parmmenu$(parmmenul% + i%) = p$
                parmsort%(i%) = numconfus%(find%, i%)
            NEXT i%
            DO
                swaps% = FALSE
                FOR i% = 1 TO 9
                    j% = i% + parmmenul%
                    IF parmsort%(i%) < parmsort%(i% + 1) THEN
                        SWAP parmsort%(i%), parmsort%(i% + 1)
                        SWAP parmmenu$(j%), parmmenu$(j% + 1)
                        swaps% = TRUE
                    END IF
                NEXT i%
            LOOP WHILE swaps%
            parmmenul% = parmmenul% + 10
            IF acind% <> 0 THEN
                i% = 1
                alt% = acstates(acind%).ActAlt / 100
                DO WHILE i% <= parmmenul%
                    compressflag% = FALSE
                    IF cmdind% = 7 THEN
                        IF VAL(parmmenu$(i%)) < alt% THEN
                            compressflag% = TRUE
                        ELSE
                            i% = i% + 1
                        END IF
                    ELSEIF cmdind% = 8 THEN
                        IF VAL(parmmenu$(i%)) > alt% THEN
                            compressflag% = TRUE
                        ELSE
                            i% = i% + 1
                        END IF
                    END IF
                    IF compressflag% THEN
                        parmmenul% = parmmenul% - 1
                        FOR j% = i% TO parmmenul%
                            parmmenu$(j%) = parmmenu$(j% + 1)
                        NEXT j%
                    END IF
                LOOP
                IF cmdind% = 7 AND VAL(parm$) < alt% THEN
                    parmcol% = LOW + HIGH + BLINK
                    parmreq% = TRUE
                    FOR i% = 1 TO parmmenul%
                        p$ = parmmenu$(i%)
                        IF VAL(p$) >= alt% THEN
                            parmmenul% = parmmenul% + 1
                            parmmenu$(parmmenul%) = parm$
                            parm$ = p$
                            parmcol% = LOW + HIGH
                            parmreq% = FALSE
                            EXIT FOR
                        END IF
                    NEXT i%
                ELSEIF cmdind% = 8 AND VAL(parm$) > alt% THEN
                    parmcol% = LOW + HIGH + BLINK
                    parmreq% = TRUE
                    FOR i% = 1 TO parmmenul%
                        p$ = parmmenu$(i%)
                        IF VAL(p$) <= alt% THEN
                            parmmenul% = parmmenul% + 1
                            parmmenu$(parmmenul%) = parm$
                            parm$ = p$
                            parmcol% = LOW + HIGH
                            parmreq% = FALSE
                            EXIT FOR
                        END IF
                    NEXT i%
                END IF
            END IF
        END IF

    CASE 9, 10
        IF numflag% AND LEN(parm$) = 3 THEN
            IF last$ <> "5" AND last$ <> "0" THEN
                parmmenu$(1) = first$ + middle$ + "0"
                parmmenu$(2) = first$ + middle$ + "5"
                allm$ = parmmenu$(1) + "," + parmmenu$(2) + ","
                parmmenul% = 2
                c5% = numconfus%(INSTR(NUM$, last$), 5)
                c0% = numconfus%(INSTR(NUM$, last$), 10)
                IF c5% <> c0% THEN
                    IF c5% > c0% THEN
                        SWAP parmmenu$(1), parmmenu$(2)
                        parm$ = first$ + middle$ + "5"
                    ELSE
                        parm$ = first$ + middle$ + "0"
                    END IF
                    parmcol% = LOW + HIGH
                ELSEIF c5% > 0 AND c0% > 0 THEN
                    parmcol% = LOW + HIGH + BLINK
                    parmreq% = TRUE
                END IF
            END IF
            find% = INSTR(NUM$, first$)
```

```
 parmsort%(1) = numconfus%(find%, 1)
 parmsort%(2) = numconfus%(find%, 2)
 p1$ = "1" + middle$ + last$
 p2$ = "2" + middle$ + last$
 IF parmsort%(1) < parmsort%(2) THEN
      parmmenu$(1 + parmmenul%) = p2$
      parmmenu$(2 + parmmenul%) = p1$
      allm$ = allm$ + p2$ + "," + p1$ + ","
ELSE
      parmmenu$(1 + parmmenul%) = p1$
      parmmenu$(2 + parmmenul%) = p2$
      allm$ = allm$ + p2$ + "," + p1$ + ","
END IF
 parmmenul% = parmmenul% + 2
 mind% = INSTR(NUM$, middle$)
FOR i% = 1 TO 10
      p$ = first$ + MID$(NUM$, i%, 1) + last$
      allm$ = allm$ + p$ + ","
      parmmenu$(parmmenul% + i%) = p$
      parmsort%(i%) = numconfus%(mind%, i%)
NEXT i%
DO
      swaps% = FALSE
      FOR i% = 1 TO 9
            j% = i% + parmmenul%
            IF parmsort%(i%) < parmsort%(i% + 1) THEN
                  SWAP parmsort%(i%), parmsort%(i% + 1)
                  SWAP parmmenu$(j%), parmmenu$(j% + 1)
                  swaps% = TRUE
            END IF
      NEXT i%
LOOP WHILE swaps%
parmmenul% = parmmenul% + 10
IF acind% <> 0 THEN
      i% = 1
      speed% = acstates(acind%).ActSpd
      DO WHILE i% <= parmmenul%
            compressflag% = FALSE
            IF cmdind% = 9 THEN
                  IF VAL(parmmenu$(i%)) < speed% THEN
                        compressflag% = TRUE
                  ELSE
                        i% = i% + 1
                  END IF
            ELSEIF cmdind% = 10 THEN
                  IF VAL(parmmenu$(i%)) > speed% THEN
                        compressflag% = TRUE
                  ELSE
                        i% = i% + 1
                  END IF
            END IF
            IF compressflag% THEN
                  parmmenul% = parmmenul% - 1
                  FOR j% = i% TO parmmenul%
                        parmmenu$(j%) = parmmenu$(j% + 1)
                  NEXT j%
            END IF
      LOOP
      IF cmdind% = 9 AND VAL(parm$) < speed% THEN
            parmcol% = LOW + HIGH + BLINK
            parmreq% = TRUE
            FOR i% = 1 TO parmmenul%
                  p$ = parmmenu$(i%)
                  IF VAL(p$) >= speed% THEN
```

```
                                          parmmenul% = parmmenul% + 1
                                          parmmenu$(parmmenul%) = parm$
                                          parm$ = p$
                                          parmcol% = LOW + HIGH
                                          parmreq% = FALSE
                                          EXIT FOR
                                     END IF
                                NEXT i%
                           ELSEIF cmdind% = 10 AND VAL(parm$) > speed% THEN
                                parmcol% = LOW + HIGH + BLINK
                                parmreq% = TRUE
                                FOR i% = 1 TO parmmenul%
                                     p$ = parmmenu$(i%)
                                     IF VAL(p$) <= speed% THEN
                                          parmmenul% = parmmenul% + 1
                                          parmmenu$(parmmenul%) = parm$
                                          parm$ = p$
                                          parmcol% = LOW + HIGH
                                          parmreq% = FALSE
                                          EXIT FOR
                                     END IF
                                NEXT i%
                           END IF
                      END IF
                 END IF

            CASE ELSE

       END SELECT

'      GOSUB RemoveExtraParm

       GOSUB EvalParm

RETURN


' Remove duplicate parameters from the menu list:

RemoveExtraParm:

       IF LEN(allm$) > 0 THEN

            i% = 1: div% = INSTR(allm$, ",") - 1

            DO WHILE LEN(allm$) > div%
                 ppos% = INSTR(allm$, parmmenu$(i%))
                 allm$ = RIGHT$(allm$, LEN(allm$) - ppos% - div%)
                 IF INSTR(allm$, parmmenu$(i%)) <> 0 THEN
                      compressflag% = FALSE
                      FOR j% = i% + 1 TO parmmenul%
                           IF parmmenu$(j%) = parmmenu$(i%) THEN
                                compressflag% = TRUE
                           END IF
                           IF compressflag% THEN
                                parmmenu$(j%) = parmmenu$(j% + 1)
                           END IF
                      NEXT j%
                      parmmenul% = parmmenul% - 1
                 ELSE
                      i% = i% + 1
                 END IF
            LOOP
```

```
     END IF

RETURN


' Evaluate the recognized parameter:

EvalParm:

      newstate.ActAlt = NULL: newstate.ActSpd = NULL: newstate.Heading = NULL

      SELECT CASE cmdind%
           CASE 1, 2, 11, 12, 13, 14, 15, 16, 17, 18
                numflag% = FALSE
           CASE 3, 4, 5, 6, 7, 8, 9, 10
                numflag% = TRUE
      END SELECT

      IF LEN(RTRIM$(parm$)) = 0 THEN numflag% = FALSE

      IF numflag% AND acind% <> 0 THEN

           SELECT CASE cmdind%

                CASE 3
                       newstate.Heading = acstates(acind%).Heading - VAL(parm$)
                      IF newstate.Heading <= 0 THEN
                           newstate.Heading = newstate.Heading + 360
                     END IF

                CASE 4, 6
                      newstate.Heading = VAL(parm$)

                CASE 5
                       newstate.Heading = acstates(acind%).Heading + VAL(parm$)
                      IF newstate.Heading > 360 THEN
                           newstate.Heading = newstate.Heading - 360
                     END IF

                CASE 7, 8
                      newstate.ActAlt = 100 * VAL(parm$)

                CASE 9, 10
                      newstate.ActSpd = VAL(parm$)

                CASE ELSE

           END SELECT

      ELSEIF acind% <> 0 THEN

           SELECT CASE cmdind%

                CASE 16
                       newstate.ActSpd = acstates(acind%).FilSpd

                CASE 1, 17
                       newstate.ActSpd = acstates(acind%).FilSpd
                       newstate.ActAlt = acstates(acind%).FilAlt

                CASE 11
                      newstate.ActAlt = 20
                      newstate.ActSpd = 0
                      newstate.Heading = 36
                CASE ELSE

            END SELECT

        END IF

    RETURN


    ' Check the recognized parameter value:

    ParmCheck:

        parm$ = RTRIM$(parm$)
        parml% = LEN(parm$)

        numflag% = TRUE
        FOR i% = 1 TO parml%
            IF INSTR(NUM$, MID$(parm$, i%, 1)) = 0 THEN
                numflag% = FALSE
                EXIT FOR
            END IF
        NEXT i%

        parmcheckflag% = FALSE

        SELECT CASE cmdind%

            CASE 1, 2, 11, 12, 14, 16, 16
                IF parml% <> 0 THEN
                    parm$ = SPACE$(5)
                END IF

            CASE 3, 5, 7, 8
                IF parml% <> 2 OR NOT numflag% THEN
                    parmcheckflag% = TRUE
                END IF

            CASE 4, 6, 9, 10
                IF parml% <> 3 OR NOT numflag% THEN
                    parmcheckflag% = TRUE
                END IF

            CASE 13, 15
                IF (parml% <> 3 AND parml% <> 5) OR numflag% THEN
                    parmcheckflag% = TRUE
                END IF

        END SELECT

        IF parmchekflag% THEN parm$ = SPACE$(5)

    RETURN


    ' Update the current command line:

    UpdateCurrent:

        ac$ = LEFT$(ac$ + SPACE$(10), 10)
        cmd$ = LEFT$(cmd$ + SPACE$(34), 34)
        parm$ = LEFT$(parm$ + SPACE$(10), 10)
        current$ = STAT$ + ac$ + cmd$ + parm$
        COLOR LOW, 0

        LOCATE 4, 1, 0: PRINT STAT$;
        COLOR accol%: PRINT ac$; : COLOR LOW
        COLOR cmdcol%: PRINT cmd$; : COLOR LOW
        COLOR parmcol%: PRINT parm$; : COLOR LOW
        CALL SetState(4, state%)

RETURN


' Get a recognized command from the buffer:

GetFromBuffer:

        newinp$ = inpbuffer$(buffer%)
        buffer% = buffer% - 1

RETURN


' Push a recognized command onto the buffer:

PushOnBuffer:

        IF BUFN > buffer% THEN

              buffer% = buffer% + 1

              IF buffer% > 1 THEN
                    FOR i% = buffer% TO 2 STEP -2
                          inpbuffer$(i%) = inpbuffer$(i% - 1)
                    NEXT i%
              END IF

              inpbuffer$(1) = newinp$

        END IF

RETURN


SUB SetState (row%, state%)

        IF state% > 0 THEN
              COLOR LOW + HIGH
              LOCATE row%, 2 * state% - 1, 0
              PRINT MID$(STAT$, 2 * state% - 1, 1);
              COLOR LOW
        END IF

END SUB
```