Technical Report 691

AD-A170 954

Human and Computer Task Allocation in Air Defense Systems

James O. Chinnis, Jr., Marvin S. Cohen, and Terry A. Bresnick
Decision Science Consortium, Incorporated

Battlefield Information Systems Technical Area
Systems Research Laboratory

U. S. Army

Research Institute for the Behavioral and Social Sciences

September 1985

U. S. ARMY RESEARCH INSTITUTE
FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the
Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

WM. DARRYL HENDERSON
COL, IN
Commanding

Research accomplished under contract
for the Department of the Army

Decision Science Consortium, Inc.

Technical review by

John K. Hawley
Dorothy L. Finley

NOTICES

AD-A170 925

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT For public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report 84-2	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARI Technical Report 691
6a. NAME OF PERFORMING ORGANIZATION Decision Science Consortium, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences
6c. ADDRESS (City, State, and ZIP Code) 7700 Leesburg Pike, Suite 421 Falls Church, VA 22043	7b. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Research Institute 8b. OFFICE SYMBOL (If applicable) PERI-SF	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-84-C-0032
8c. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 6.55.02A; PROJECT NO. 2Q162717A790; TASK NO.; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Human and Computer Task Allocation in Air Defense Systems (U)

12. PERSONAL AUTHOR(S)
Chinnis, James O., Jr., Cohen, Marvin S., and Bresnick, Terry A.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 12/15/83 TO 7/15/84	September 1985	96

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Decision Aids; Man Computer Interface; Decision Making; Man Machine Systems

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The advent of computer systems that offer direct support for high-level cognitive tasks has called basic assumptions about the proper respective roles of computers and humans into question. Reallocation of cognitive tasks from human to computer has sometimes resulted in user rejection of resulting systems. In cases where automation has been complete or nearly so, system performance may suffer from the lack of contributions best made by human operators. In either case, systems become overly complex and costly, and perform less than optimally.

Within a general air defense background, a small experimental study has been conducted to examine basic issues related to the allocation of cognitive tasks in human-computer systems. Research questions include examination of (1) variables which determine the relative superiority of humans or computers, (2) the impact of information load on the optimality of human decision rules, (3) flexible versus fixed allocation schemes, and (4) issues related to the optimal locus of control of the task allocation process.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Irving Alderman	22b. TELEPHONE (Include Area Code) (202) 274-9046 22c. OFFICE SYMBOL PERI-SF

18. Subject Terms (continued)

Human Factors Engineering
Air Defense
Antiaircraft Defense Systems
Task Allocation

19. Abstract (continued)

A prototype research system was developed to simulate relevant aspects of the air defense environment. Experimental subjects acted as operators of the prototype system and made decisions which required them to identify each approaching simulated aircraft as friend or hostile. Four conditions were compared, ranging from a manual condition in which subjects must make all ID decisions to a completely automated condition in which the computer made all ID decisions.

Results indicated that under low loads and where humans had access to information not in the computer model, any conditions involving human participation yielded performance superior to that of the automated system alone. At high loads, however, the full benefit from human participation could only be maintained if the computer controlled the task allocation process itself, directing human attention to subproblems where help was required. Implications of these preliminary results for system design are discussed.

Technical Report 691

Human and Computer Task Allocation in Air Defense Systems

James O. Chinnis, Jr., Marvin S. Cohen, and Terry A. Bresnick
Decision Science Consortium, Incorporated

for

Contracting Officer's Representative
Irving N. Alderman

Battlefield Information Systems Technical Area
Dorothy L. Finley, Acting Chief

Systems Research Laboratory
Franklin L. Moses, Acting Director

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES
5001 Eisenhower Avenue, Alexandria, Virginia 22333-5600

Office, Deputy Chief of Staff for Personnel
Department of the Army

September 1985

Army Project Number
2Q162717A790

Human Performance Effectiveness and Simulation

ARI Research Reports and Technical Reports are intended for sponsors of R&D tasks and for other research and military agencies. Any findings ready for implementation at the time of publication are presented in the last part of the Brief. Upon completion of a major phase of the task, formal recommendations for official action normally are conveyed to appropriate military agencies by briefing or Disposition Form.

FOREWORD

The critical role of human decision making has long been recognized as a significant factor in the development of command and control (C^2) systems. Current projections of the air-land battle with greatly increased density of highly mobile and lethal threats require C^2 systems with increased and enhanced capabilities. In order to achieve these capabilities, the emerging technologies in computer science and in the higher order human cognitive processes are being explored to provide enhanced processing of enriched information from the battle environment. The allocation of system functions to computer and human according to the unique capabilities of each offers a system approach to satisfy operational requirements.

The present research focuses on development and evaluation of an experimental paradigm to assess the relative contribution of interactive computer and human functions in satisfying operational requirements under conditions of alternative workload and allocation schemes. The long-term goal of this effort is to develop design guidelines and aids for the development of integrated human-computer interfaces.

EDGAR M. JOHNSON
Technical Director

ACKNOWLEDGMENTS

The work reported here has been sponsored by the Army Research Institute, under the Small Business Innovation Research Program. We are grateful to ARI's project monitor, Dr. Irving Alderman, for his advice and attention to our needs, and to Dr. Charles Howard (then of ARI) for his interest and advice during early phases of the work. In addition, we received much constructive advice and feedback from members of the ARI Field Office, Fort Bliss, Texas, including Dr. Michael Strub, Dr. John Hawley, and Ed Dawdy, and from Don Harris of the Directorate of Combat Development, Fort Bliss, Texas.

Appreciation is due as well to Drs. John Payne of Duke University and Scott Randall of Image Technology Company for serving as consultants to assist in the development of the experimental approach and software, respectively.

Other members of DSC's staff also contributed to this work. In particular, Dr. Stephen Watson contributed much to the analysis of data. Special thanks also go to Ms. Lynn Merchant-Geuder, who succeeded both in securing and managing our experimental subjects and in production of the report.

HUMAN AND COMPUTER TASK ALLOCATION IN AIR DEFENSE SYSTEMS

EXECUTIVE SUMMARY

Requirement:

The advent of increasingly sophisticated and expensive human-machine systems has called into question basic assumptions about the proper respective roles of computers and humans. In particular, the reallocation of cognitive tasks from human to computer has sometimes resulted in user rejection of resulting systems or in systems which may not take full advantage of human contributions to the overall task. The preliminary research described here investigated some of the issues related to task allocation between human and computer in such systems. In addition, an experiment was performed to test and refine resulting hypotheses about the design of human-computer systems.

Procedure:

Within a general Army air defense context a small experimental study examined variables which determine the relative superiority of humans or computers, the impact of information load on the optimality of human decision rules, flexible versus fixed task allocation schemes, and issues related to the optimal locus of control of the allocation process itself. A simplified simulation of a generalized air defense system was developed on a small computer and naive experimental subjects operated the system, making decisions regarding the identity of approaching aircraft as friendly or hostile, based on a number of identification cues. Five partially diagnostic cues were available, four of which were directly utilized by the computer aid, and one of which had to be inferred and learned by the operator. Four conditions were compared: (1) a manual condition in which subjects made all identification decisions, (2) a screening condition in which the computer directed the operator's attention selectively to those aircraft where cue conflict was greatest, (3) an override condition in which the computer made its decisions and the operator could reverse them at will, and (4) an automated condition in which the computer made all decisions.

Findings:

Under low loads, all conditions in which human participation was permitted yielded performance superior to that of the automated system alone. At high loads, however, the full improvement due to human participation could be maintained only if the computer controlled the task allocation process itself, directing human attention to identification problems where cues available to the computer were particularly undiagnostic.

Utilization of Findings:

Preliminary experimental results support the hypothesis that performance of human-computer systems can depend significantly upon the allocation of tasks between the human and computer components. In addition, there is evidence that systems which incorporate a collaborative mode of interaction under the control of the computer are superior to designs which have been used to date. These preliminary findings are suggestive of basic principles of fundamental importance to the design of air defense and other systems capable of delivering the maximum performance for the lowest cost.

HUMAN AND COMPUTER TASK ALLOCATION IN AIR DEFENSE SYSTEMS

CONTENTS

1.0 INTRODUCTION AND BACKGROUND

1.1 The Problem

Computer systems that offer direct support for high-level cognitive tasks--such as classification, diagnosis, prediction, and choice--have called into question basic assumptions about the respective roles of computers and humans. Opinions clash as to whether such systems are best regarded as "tools," "advisors," "surrogates," "replacements," "supervisors," or "subordinates." Nevertheless, with the expanding operational role of computers, increasing attention is focused on the manner in which computer-implemented processing of information can be expected to support, complement, or stand in for human cognitive capabilities.

In general, there is little doubt that certain tasks related to cognition should be allocated to the machine: for example, storage and recall of information in large data bases and automatic control in well-defined and repetitive situations. As machines have become capable of increasingly intelligent activity, however, the appropriate allocation has tended to become less clear. This trend has been felt keenly in a variety of contexts where systems have been introduced to support high-level cognitive tasks: e.g., in command and control, medical diagnosis, and business management. Intended users have frequently been reluctant to consult computer-based systems; in some cases there has been severe organizational and institutional resistance; and reallocation of cognitive tasks from the human to the machine has sometimes resulted in sharply decreased system performance (e.g., Miller, 1980; Beard, 1977; Swanson, 1974). Perhaps the chief cause of these problems is the mismatch between a user's cognitive abilities, habits, and preferences and the requirements imposed on him by the computer as it assumes a larger and larger role in tasks previously reserved for the human.

In these instances, the presuppositions and methods traditionally governing task allocation between humans and machines have fallen short. Human factors guidelines for the person-computer interface (such as Ramsey and Atwood, 1979; Engel and Granda, 1975) have contributed to an understanding of the design of displays, input devices, and many aspects of user-computer dialogue, but have barely touched on methods for optimally and acceptably interweaving human and machine thought (cf., Cohen, 1983). One result has been that interactive properties of cognitive aids have, of necessity, evolved by trial and error. Another, very recent, result has been the appearance of sophisticated systems which entrust their duties almost wholly to the computer and leave little or no opportunity for human contributions.

What is required, both to encourage user acceptance and to enhance overall system performance is (1) a repertoire of techniques for blending the knowledge and skills of the user and the computer, and (2) guidelines describing the appropriate circumstances under which each technique should be used. The research reported here is a step toward filling these needs. We propose, and test, two principal theses:

- that the allocation of cognitive tasks or task elements between machine and human must take into account human patterns of analysis and thought; and
- that the best allocation of cognitive tasks will often be a dynamic and flexible one, which adapts to such factors as task complexity,

processing load, the training or ability of the human relative to the computer, and the problem-solving strategies favored by the human user.

1.2 Current Issues in the Allocation of Cognitive Tasks

Traditionally, task allocation in human-machine systems has been according to the purported strengths of each (e.g., Fitts, 1951). Such methods have for the most part aimed at a fixed allocation of broadly defined activities (such as numerical computation or long-term data storage), and are based on static generalizations regarding the relative superiority of persons and computers. It is becoming increasingly clear that such methods fail to acknowledge or adequately handle a variety of problems that arise uniquely in computer-assisted cognitive processing. For example:

- In tactical C^2 contexts computers may support decision makers in a rapid succession of tasks that are quite diverse in their demands on knowledge, time, and attention.
- No two occurrences of the "same" task or task element will be exactly alike in their demands on knowledge, time, and attention.
- Decision makers themselves differ significantly in preferred problem-solving style, in knowledge, and in their ability to handle varying degrees and kinds of mental workload.
- Task assignments may be meaningless or inappropriate in terms of user preferences and problem-solving styles.
- Novel approaches or problem solutions that are made possible by human-computer combination may be overlooked.
- The human may be left unprepared in event of computer breakdown.

These and other issues have been discussed by Rouse (1977), Ramsey and Atwood (1979), Singleton (1974), and Cohen et al. (1982).

There is, in general, a complementarity of expertise between aids and their users. Attempts to draw fixed boundaries, however, are likely to fail as the technology for computer representation and manipulation of knowledge advances. Buchanan (1981) and McCarthy (1977), for example, list a variety of concepts for which current artificial intelligence methods are at least in part inadequate, including causal reasoning, propositional attitudes, conflicting plans or representations, analogical reasoning, and reasoning about dynamic three-dimensional relationships.

Nevertheless, a critical characteristic of virtually all higher-level systems or models (including decision-analytic aids and knowledge-based expert systems) is that they incorporate the assumptions and modes of reasoning of human specialists. As a result, in complex problem domains, there is never a guarantee that all potentially relevant factors or principles of reasoning will have been incorporated into a computerized aid. Thus, computer systems which altogether eliminate user input during the solution process risk the loss of a valuable cognitive resource. For the same reasons, however, the precise opportunities for collaborative problem solving cannot always be foreseen. Systems which

confine the human's contribution to a prespecified subtask, without regard to shifting advantages of user and machine, may well fail, on any given occasion, to exploit the human's full potential contribution.

Traditional methods of task allocation are neither fine-grained nor flexible enough for many important applications (as noted, for example, by Singleton, 1974; Rouse, 1977; Cohen et al., 1982). The development of an alternative approach, however, is itself not without difficulties. It will be helpful to survey briefly some of the methodological challenges that must be overcome before a fully adequate technology for cognitive task allocation can be achieved:

Task identification. Methods for representing cognitive tasks in terms of psychologically meaningful subtasks or elements must be developed. A variety of current approaches are employed in the segmentation and classification of cognitive performance, ranging from those based on observation of problem-solving behavior, to analysis of verbal protocols obtained during think-aloud problem solving, to direct descriptions by problem solvers of rules, procedures, or principles. These vary in the degree to which they rely on the self-knowledge of the problem solver versus the preconceptions or theoretical assumptions of the cognitive scientist. It is as yet unclear which methods or combinations of methods yield the most valid representations of cognitive tasks for different problem types or kinds of skill.

Task invention. The most effective design, however, may not result from a simple allocation of pre-existing cognitive tasks. As noted by Ramsey and Atwood (1979), the combination of human and machine may allow problems to be approached in a way not possible at all for the human (or machine) alone. Simply analyzing how people perform a complex job and designing a human-machine system by allocating certain of the identified task elements to the machine may well end up in the perpetuation of an existing inferior approach.

Performance evaluation. How can we tell whether the man alone, the computer alone, or some combination of the two is more successful in a given cognitive task? Although speed is often a relevant evaluative criterion, there sometimes is no ready at hand method for assessing the quality of the cognitive performance. Higher-level tasks often involve significant judgmental components. Different experts may disagree in their methods, in their conclusions, or in both. Where possible, of course, some measure of ground truth or ultimate outcomes (e.g., number of hostile aircraft destroyed) should be used. Such measures, however, must take into account the "difficulty" of the scenario in which performance is assessed (Hawley, Howard, and Martellaro, 1982). Moreover, in decision making under uncertainty, a good outcome is not a guarantee that the best decision was made, and a bad outcome does not entail that a bad decision was made. Normative models can be employed as standards for comparison, but care must be taken that the assumptions of the model are in fact satisfied in the judgment of competent problem solvers in the domain, and that there are not other experts or models which lead to divergent conclusions.

Task coherence. Users have their own preferences and styles of problem solving that may not correspond to otherwise optimal patterns of allocating cognitive tasks. Assignment of tasks purely in terms of the relative competence of the computer and the man could lead to disaster, if it disrupts the person's perception of the continuity and meaningfulness of his own performance.

Organizational factors. Similarly, an otherwise optimal allocation pattern will incur serious resistance if it violates the human's concept of his own role or responsibilities. Military officers are often reluctant to relinquish higher-level decisions involving the safety of the platform or the ultimate success of the mission.

Coordination overhead. The effort to share cognitive processing between human and machine (or indeed between two humans or two machines) exacts a toll in the need to transfer information efficiently and accurately between the participants. In the case of human-computer interaction, this implies that outputs provided by the computer and inputs demanded of the user be natural and meaningful to the human. In the context of cognitive collaboration, this requirement goes beyond a simple human factors analysis of display and input devices or formats. It requires that human and computer methods for attacking a problem be commensurable, to a degree determined by the nature and extent of the collaboration. Computer and human problem-solving methods need not be the same. But it is necessary that the human understand, to some degree, how conclusions arrived at by the computer were achieved, in order to make most effective use of them. He must have a mental model of the computer's method of problem solving and of his own role in the overall task.

Cognitive load leveling. A principal objective of dynamic task allocation is to equalize levels of effort, relative to capacity, of diverse personnel and computational devices. Unlike the measurement of physical work, however, the determination of cognitive effort is, in many contexts, highly conjectural and ill-defined. A variety of methods exist: analytic (e.g., based on information theory), direct subjective judgment, or the measurement of concurrent performance in a secondary task. None of these approaches is fully successful in dealing with all the factors that must be considered: including the likely role of multiple specialized cognitive capacities or subsystems instead of, or in addition to, a single central resource; decrements in capacity over time; and improvements in capacity with practice or with minor changes in the task.

1.3 Research Questions

Against this background, the present study isolates several more specific questions for experimental investigation:

(1) What variables determine the relative superiority of users and computers? In particular, we investigate the role of (a) workload and (b) the completeness with which a computerized algorithm covers relevant evidence. Within the space defined by these two variables, a region may exist in which the human outperforms the computer--e.g., under conditions of moderate workload, where there are relevant cues not utilized by the computer but recognizable by the human.

(2) What is the impact of workload on use by humans of suboptimal decision rules? Performance may suffer under low workload, since the human's ability to participate where required may be compromised by decreased vigilance. At the other extreme, also, as the number or complexity of decisions requiring human input increases, human performance is likely to degrade. In particular, there is evidence that humans shift from reliance on more nearly optimal decision rules to use of suboptimal simplifying strategies or heuristics. For example, as workload increases, decisions may be made by comparing options to cut-off points on one or a small number of relevant dimensions, rather than evaluating each option with respect to all available cues (Payne, 1978). Under these

conditions, the relative advantage of humans over computers, even when the computer's model is incomplete, may be lost.

(3) How do flexible allocation schemes compare with fixed ones? Fixed schemes involve decision making either by the human only or by the computer only, while flexible schemes may enlist computer and/or human contributions in any decision, depending on situational variables such as workload and the relative expertise of the user and computer. For example, in a relatively low-workload situation, an attack planning or targeting C^2 system might allocate data collection and display functions to the computer and leave inferences and predictions regarding critical events (e.g., identity and intentions of hostile and friendly contacts), target selection, and decisions to engage or not to engage to the human operator. Under a high-workload multi-threat situation, however, the system "executive" might reallocate more of the integrative (Phelps, Halpin, and Johnson, 1981) tasks to the computer. As stress and load increased, for example, the computer might begin to display recommended attack plans, target priorities, and weapon-target assignments. Under still higher stress and load, the computer might assume control of the actual firing of weapons or configuration of combat equipment.

(4) To what degree should the human or the computer control task allocation? An important variable in the design of flexible systems is the assignment of the "executive." Executive tasks represent in essence a special category of tasks which then determine the allocation of remaining tasks to human and machine. Thus, some of the same considerations enter into the assignment of these tasks as in the assignment of the original, lower-order tasks. Can humans make effective use of the situational variables that determine optimal allocation? In particular, can they adequately assess the strength of the conclusion drawn by the computer in a given decision, the completeness of the computer algorithm, and the availability of cues to the user which are not incorporated in the computer? Asking humans to perform the allocation task themselves, without computer assistance, may defeat the objective of using the computer to reduce human workload. Computer assistance in the allocation function, if consistent with user conceptions of how tasks ought to be assigned, might result in a more cost-effective utilization of human cognitive effort. The question, then, is: can computer aids be designed which are sensitive to their own shortcomings and which alert users regarding decisions where user judgment might prove of value?

1.4 The Air Defense Setting

The Air Defense environment has been selected as a specific testbed in which these questions can be addressed. This environment is characterized by:

- high stakes, with expectations of heavy losses in short periods of time;
- potentially heavy peak loads, where the skies can be saturated with enemy aircraft;
- sophisticated, high-performance threat weapon systems, resulting in minimum reaction time;
- highly critical vital assets requiring friendly air defense protection.

The specific problem selected as the focus of the present research is identification of aircraft as friend or foe. Air defense systems exist in the field that run the gamut from virtually all identification (ID) decisions left in the hands of the user (e.g., IHAWK) to virtually all ID decisions made by the computer (e.g., PATRIOT). Current plans call for modifying the IHAWK system to improve its capabilities in the direction of PATRIOT, and there is a broad spectrum of intermediate capability options that must be considered.

During centralized operations, higher headquarters and adjacent units provide the dominant ID cues. At the fire unit level, targets are designated for engagement from battalion command centers, and the fire unit focuses on engagement rather than ID decisions. However, in wartime it is fully expected that a majority of combat engagements will be made under decentralized authority due to the inability to higher echelons to detect aircraft attacking at low altitudes. Additionally, it is expected that communications will be interrupted frequently, thus requiring fire units to operate autonomously.

In the IHAWK system, particularly during autonomous operations, the Tactical Control Officer (TCO) and Tactical Control Assistant (TCA) are responsible for identification decisions. Sources of information are varied:

- Identification Friend or Foe (IFF) equipment at the Platoon Command Post (PCP) (manually initiated by the TCA). This is done using a manual interrogation switch which is coded to receive specific responses from transponders in friendly aircraft. These codes are changed frequently to avoid exploitation by the enemy;
- correlation with flight plans and safe passage corridors--for example, the commander might establish a schedule which prohibits any firing at aircraft on certain headings during specified time periods;
- aircraft actions (dropping chaff, use of other Electronic Counter Measures, or ECM, attacking friendly troops);
- information passed from higher or adjacent units (if available);
- pop-up criteria, which are designated parameters such as speed, altitude, and bearing that must be observed by friendly aircraft.

Often these identification cues are missing or can be conflicting, and the friend or foe decision is a difficult one. Following prescribed rules of engagement, the TCO/TCA typically will use electronic means to make a hostile identification based upon the above criteria. For example, if an aircraft is not responding to the prescribed IFF, is outside of a safe passage corridor, and is closing at a speed in excess of a prescribed rate, it might be declared hostile. However, if it is not responding to IFF, is in the safe passage corridor, and exceeds the speed criterion, the identification is less clearcut. Subjective judgment, based on experience, will be used to combine the ID cues and reach a decision.

Similarly, in the PATRIOT system, input data come from the sources above, except the AN/TPX-46(V)7 IFF interrogator is used. Currently, cue conflict resolution depends upon the level of automation selected, the relative importance and reliability of the various data sources, and the decision-making style of the officer running the engagement. At the highest capability level, the automated

system combines cues using a pre-determined weighting algorithm and makes the ID determination. For example, IFF response, speed, altitude, and passage over restricted areas can be assigned weights that reflect their relative importance. Each aircraft is "scored" on each factor by the automated system, and based upon the mathematical combination of scores and weights, the aircraft is designated as friendly, hostile, or unknown. The TCO can change weights in the algorithm or can override automated decisions, but is not likely to do so in most cases. In this mode of operation, challenges using the IFF are initiated without TCO/TCA intervention.

The next scheduled improvement to the IHAWK system is planned to bring the capability level for ID closer to that of PATRIOT. More will be done by the automated portion of the system as far as challenging, analyzing data, and determining friend or foe identification. Yet, experience with the fully automated PATRIOT system, as suggested in recent conversations with personnel, has shown that this may not be the optimal configuration for the system. Some problems include excessive challenging to aircraft causing transponder damage, boredom of operators when in the fully automated mode, and lack of confidence in the fully automated system. Some have hypothesized that under combat conditions, TCO/TCAs will not even use electronic IFF challenges due to increased vulnerability to enemy exploitation of the signal. For these and other reasons, it may be the case that an intermediate level of automation will produce better results.

2.0 APPROACH

Within the context of Army air defense systems, an approach has been developed for the exploration of design principles. Our goal in Phase I has been to design a method of inquiry capable of shedding light on the proper allocation of tasks between human and machine in this very general class of systems, and to test in a preliminary way the usefulness of the research method. A particular emphasis is to uncover innovative methods for overcoming the related tendencies of users or operators to reject or fail to make proper use of a computer aid and of system designers to reject or fail to make proper use of a human operator. Thus our approach emphasizes the analysis of how the allocation of tasks might be made flexible, or, what is often the same thing, how the human and machine components of the system can effectively collaborate with one another.

3.0 A PROTOTYPE RESEARCH SYSTEM AND PILOT EXPERIMENT

To provide a basis for conducting a program of research, a prototype research system or testbed was developed and a pilot experiment conducted. The research system is designed to permit a wide range of studies to be done with moving visual stimuli presented in real time on a video screen, text displayed on either or both of two simultaneously driven video screens, and data input by keyboard, mouse, or other input device. The system has been developed along modular and very general lines to accommodate most tests of interest in an information-processing and decision-making situation where time pressure and information load are significant.

The pilot experiment was intended both to test the software and general approach and to generate some guidance for design principles and the design of experiments in Phase II.

Although the system has been designed to accommodate a wide range of studies, the method for a pilot experiment will be described in the following paragraphs prior to discussion of the research system in order to better motivate the discussion of system design.

3.1 Method

3.1.1 Objectives. The objects of the pilot experiment were to (1) demonstrate the feasibility and usefulness of the research system and approach, (2) determine whether the system as constructed was capable of displaying air defense data sufficiently rapidly and smoothly to permit testing of Ss' performance in high-load conditions, and (3) tentatively explore the effectiveness of different partitions of the overall air defense task between human and machine. For the third, task allocation, objective, sub-objectives were to obtain tentative answers to the following questions:

- Can Ss make use of cues relevant to discrimination of friendly from hostile that must be learned and are not part of the computer-based system?
- Do different task allocations lead to different performance by the system in discriminating friendly from hostile aircraft? In particular, what are the relative merits of different fixed (human-only or computer-only) and flexible allocation schemes? What is the effect of different degrees of computer or human control over flexible allocation?
- Do such differences in system performance depend upon the information load? E.g., do some task allocation schemes improve discrimination at high loads but not at low loads?

3.1.2 Subjects. Subjects were recruited from schools and universities in the Northern Virginia area.

3.1.3 Design. The task for the combined subject-computer system was to observe approaching aircraft and various data regarding the aircraft and to make decisions whether to shoot or not shoot each aircraft. Two computer-driven displays were used to simulate an air defense console and to present information to the Ss.

Aircraft symbols appeared at the top of one display (the "radar" display) moving at constant and identical speed toward the bottom center of the display, where the S and the air defense system were "located." Traversal of the screen from the top of the display to an "in-range" line, where missiles were fired, required approximately one minute. Four cues were available to both the computer and S. One of these was available from the graphics ("radar") screen, the other three from the companion text screen. In addition, a fifth cue--referred to as the "extra" cue--was available only to S; it was available from the graphics screen, but was not utilized in the computer's ID algorithm. Information provided in the instructions (regarding the overall location of friendly and hostile air bases) was intended to alert Ss to the potential significance of this cue.

The experiment used a within Ss design with two primary independent variables or treatments. The first is the task allocation, consisting of three conditions:

- A manual condition, in which all aircraft are initially shown as having "unknown" ID on the display screen and in which Ss must make all ID decisions. This is a fixed, human-only allocation scheme.

- An override condition, in which the computer applies an algorithm utilizing four cues to determine target ID and labels all targets as either hostile or friendly, but in which Ss may override the computer in those cases which they believe are wrong. This is a flexible scheme in which the human has ultimate control over task assignments.

- A screening condition, in which the computer processes the four cues available to it for each aircraft, but produces an ID decision only when 3 out of 4 of these cues agree. The computer requests S assistance (by means of an "unknown" ("?") symbol) for cases of extreme cue conflict. This is a flexible allocation scheme in which the computer assists the human in determining appropriate task assignment.

A fourth condition, useful for comparison purposes, is the fixed, computer-alone scheme. In this condition, system performance measures can be calculated analytically, without resort to experimental data.

The second independent variable is information load. This was used at two levels. Low load resulted in six aircraft simultaneously on the screen (a new aircraft appearing every 11 seconds), whereas high load resulted in fifteen simultaneous aircraft (a new aircraft appearing every 4 seconds). These aircraft counts refer to aircraft positioned above the "in-range" line, i.e., those far enough away from the S that decisions (shoot or don't shoot) can still be made. A few aircraft are usually on the screen beyond this point.

A full description of the displays, the discrimination cues provided, the Ss' task, feedback methods, and other aspects of the conduct of the experiment is provided in Appendix A via copies of the instructions to Ss. This detail is necessary for a complete understanding of the experimental procedure, but is not repeated here.

In all cases the S's mode of response was to type the number associated (on the display screen) with of an aircraft. The result of this action varied among the different task allocation conditions. In the manual condition, all aircraft were shown as unknowns; typing its target number caused an aircraft to be

designated hostile and to be destroyed. In the override condition, all aircraft were identified by the computer as either friendly or hostile; typing a target's number caused this designation to reverse. As before, hostile aircraft were destroyed. In the screening condition, aircraft associated by the computer with "unknown" symbols were treated as in the manual condition: i.e., targets whose numbers were typed were designated hostile and destroyed. Other aircraft, identified by the computer as either friendly or hostile, were treated as in the override condition: i.e., typing its number reversed the designation. Whenever an aircraft number was entered, the aircraft symbol shown on the radar screen was modified by enclosing it in a hexagon.

Feedback was provided to the Ss in three simultaneous ways:

- by the use of a flashing aircraft symbol in the lower part of the radar screen to indicate an error (friend destroyed, or hostile not destroyed);
- by the use of a "right" or "wrong" message displayed next to the appropriate aircraft data line on the text screen;
- by a running score displayed on the text screen in the form of "number of correct decisions out of number attempted."

Ss were paid six cents per aircraft correctly classified, leading to average earnings of approximately $9.00 per hour.

A counter-balanced approach was taken to the two independent variables in the within Ss design. All Ss participated in three two-hour sessions, with each session devoted to one of the task allocation conditions. In each session, each S participated first in a training session followed by a high-load condition and low-load condition in either order. Twenty-four Ss participated. Ss were recruited from the local area using bulletin board notices and other means.

A summary of the allocation of Ss to treatments is shown below:

Placement of subjects in treatment cells

		Order of Task Allocation Conditions					
		M-S-O	M-O-S	S-M-O	S-O-M	O-M-S	O-S-M
Order of Load Condition	H-L	1, 13	3, 15	5, 17	7, 19	9, 21	11, 23
	L-H	2, 14	4, 16	6, 18	8, 20	10, 22	12, 24

Note: Entries are subject identification numbers.

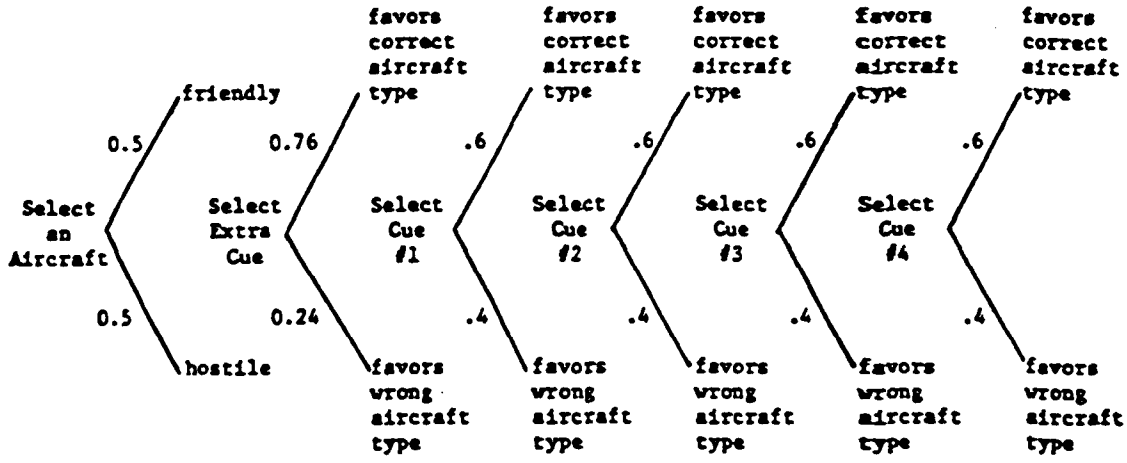
In the table, M-S-O refers to a task allocation treatment order of manual followed by screening followed by override. L-H refers to presentation of the load conditions in low load first, followed by high load.

For each S, and each of the six combinations of task allocation condition and load condition, 200 responses (aircraft classifications) were obtained and used as data. The first and last 25 responses out of each 250-response cell were discarded due to transient effects related to starting and terminating each data-collection segment. Thus a total of 28,800 responses were obtained.

3.1.4 Simulation and cue diagnosticity. Emphasis was placed on presenting Ss with a discrimination problem that was as representative of the air defense environment as possible. Therefore, although primary interest resided in the handling of discriminations for which substantial conflict of ID cues was present, a simulation was developed which provided an abstract but comprehensive situation from which Ss could learn about cue diagnosticities. No direct information was provided to Ss regarding the relative usefulness of cues; instead, they were provided with training prior to the experimental trials intended to enable them to extract the required information for themselves. This training consisted of a complete practice block of 250 trials at the start of each session, with a load intermediate between the low- and high-load conditions.

The computer software utilized a simulation module which generated a sequence of aircraft and ID cues according to the probability diagram shown below:

Diagram of simulation process



The simulation generates friendly and hostile aircraft with equal probability. Following the selection of the true aircraft type, each of the four cues utilized by the computer aid are independently generated so as to be appropriate to the type of aircraft with probability 0.6. Thus, for example, a friendly aircraft will be assigned to fly within a safe passage corridor with probability 0.6, whereas a hostile aircraft will be assigned to a safe passage corridor with

probability 0.4. The "extra" cue--not utilized by the computer aid, but available to Ss who learn it--is generated similarly, but is somewhat more reliable, in that it is selected to favor the true aircraft type with probability 0.76. This means, for instance, that a hostile aircraft will be positioned on the left side of the screen (where the hostile base is located) with probability 0.76.

The simulation process utilizing 5 binary cues results, of course, in 32 possible cue combinations for any aircraft. Since the four cues available to the computer aid are equivalent in diagnosticity, this can be represented more simply as a combination of (1) the number of the four computer cues that favor hostile, and (2) the extra cue. This representation will be used throughout this report. Schematically this is shown in the following table:

Diagnosticity of cue patterns involving extra cue

Number of Computer Cues Favoring Hostile	Extra Cue Favors	Probability of Cue Pattern If Aircraft Is		Likelihood Ratio of Cue Pattern
		Friend	Hostile	
0	f	0.0985	0.0061	0.062
1	f	0.2627	0.0369	0.140
2	f	0.2627	0.0829	0.316
0	h	0.0311	0.0195	0.627
3	f	0.1167	0.0829	0.710
1	h	0.0829	0.1167	1.408
4	f	0.0195	0.0311	1.595
2	h	0.0829	0.2627	3.169
3	h	0.0369	0.2627	7.126
4	h	0.0061	0.0985	16.147
		1.0	1.0	

The optimal strategy in this context depends, of course, on the scoring system. In this study, there was an equal penalty for either type of error--shooting down a friend or failing to shoot a hostile. Therefore, the optimal decision rule is to respond "hostile" in all cue patterns where the likelihood ratio exceeds 1--i.e., for all patterns that are more likely under the assumption of a hostile aircraft than under the assumption of a friendly aircraft.

In the table, notice that the extra cue--available only to Ss (who must infer it)--is sufficiently diagnostic to overwhelm all of the other four cues except when they all agree. In other words, optimal performance by an S would correspond to always responding according to the extra cue, unless all four of the other cues favor the other aircraft classification.

A similar table has been calculated for the computer aid alone--i.e., for use of the four cues without the extra cue:

Diagnosticity of cue patterns without extra cue

Number of Computer Cues Favoring Hostile	Probability of Cue Pattern If Aircraft Is Friend	Hostile	Likelihood Ratio of Cue Pattern
0	.130	.023	.77
1	.346	.154	.445
2	.346	.346	1.000
3	.154	.346	2.247
4	.023	.130	5.652
	1.0	1.0	

The optimal computer performance based on these four cues alone is to choose the aircraft identification suggested by the majority of cues: if more than 2 cues favor hostile, shoot; if fewer than 2 cues favor hostile, do not shoot; if 2 cues favor hostile, either response is equally likely to be correct.

Use of 5 cues rather than 4 will be an advantage in this task when the four computer cues do not agree. Note, however, that since no cue or cue pattern is perfectly associated with hostile or friendly, ID "errors" would still be expected even by an optimally performing subject. Thus, we need to distinguish between the theoretically appropriate or optimal response (the "best decision") and the response which happens to be correct on a given occasion (a "good outcome"). An optimal response rule will produce fewer mistaken IDs on the average (hence, a higher overall score) but will not be right every time.

The following table shows the optimal response for each pattern of cues and the percentage of correct IDs that would result. These figures are given both for the computer (which has access only to four cues) and for the total human-computer system (which has access to four cues plus the extra cue). Asterisks indicate conditions under which utilization of four cues plus the extra cue may lead to a different ID decision from use of four cues alone.

Impact of extra cue on optimal response and percent correct IDs

Number of Computer Cues Favoring Hostile	Extra Cue Favors	4 Cues Only: Optimal Response	4 Cues Only: % Correct IDs	4 Cues Plus Extra Cue: Optimal Response	4 Cues Plus Extra Cue: % Correct IDs
0	f	f	94.13	f	94.13
1	f	f	87.69	f	87.69
2	f	-	50.00	f*	76.00
0	h	f	61.52	f	61.52
3	f	h	41.54	f*	58.46
1	h	f	41.54	h*	58.46
4	f	h	61.52	h	61.52
2	h	-	50.00	h*	76.00
3	h	h	87.69	h	87.69
4	h	h	94.13	h	94.13

Note also that while use of the extra cue will help in cases of 1 cue pointing one way and 3 cues the other, it helps more when 2 cues point each way (the case of maximum ID conflict for the automated system).

3.2 The Prototype Research System--Software and Hardware

The prototype research system developed for this research program has been designed for an IBM-PC microcomputer and programmed in C. This arrangement was chosen as the least expensive means of supplying the needed computational power. By using machines equipped with Intel 8087 coprocessors and the C language, the software performs smoothly and with reserve capacity. Software development was performed on an IBM PC/XT with 640KB of random-access memory and 8088/8087 dual processors. Dual display cards were used to enable simultaneous control of both a text display and a 640 by 200 pixel graphics display. By using the dual video displays, the usual limitations of a low-resolution display were avoided, and a form of viewporting or windowing was achieved.

The minimum configuration for using the current research program is as follows:

- An IBM-PC or compatible microcomputer
- 128KB of RAM
- One diskette drive
- An IBM monochrome display card and monitor
- An IBM color-graphics display card and compatible monochrome or color monitor

An Intel 8087 coprocessor (which sells for less than $200) is highly recommended in addition. Without the coprocessor chip, the code executes more slowly, and although the same overall timing characteristics are preserved by the software design, the frequency with which the displays are redrawn is reduced; under very

high-load situations, a discontinuous motion of the aircraft begins to be noticeable and annoying.

The software used for the pilot experiment is listed in Appendix B. It is written in modular fashion, making good use of the principles of structured programming. In particular, the graphics routines--those most likely to be machine specific, or even display-card specific--utilize calls to assembly language subroutines based on the European Graphic Kernal System (GKS). (See, for example, Association for Computing Machinery, Inc., 1982.) By using this approach, major future changes can be made with a minimum of revision to the program. The graphics approach used, for example, permits all images to be rescaled by changing a few program constants; this means, for example, that a higher-resolution adapter and display can be driven by the program for a few hours of reprogramming effort.

3.3 Results

Most of the trials in the experiment were necessary to produce an ecologically valid set of stimuli which could be regarded as reasonably representative of the air defense environment and from which subjects could learn the diagnostic values of cues. The primary result of interest, however, has to do with the handling by the human-computer system of those cases involving a high level of ID cue conflict. Since we are interested to see which task allocation conditions, if any, enable better integration and utilization of both computer and human contributions to the problem, attention will focus below on the case of two of the cues available to the computer pointing toward hostile and two pointing toward friend.

3.3.1 *Comparison of task allocation conditions for cases of high ID cue conflict.* The dependent variable is the appropriateness of the human-computer system response to each aircraft--in this case, shooting when the extra cue favors hostile and not otherwise. The data available consist of six scores per subject: a score for both high and low loading in each of three task allocation conditions (manual, screening, override). Each score is the number of appropriate responses out of 68 observations. Since there are 24 subjects, we have almost 10,000 total observations.

The following table summarizes these data. It shows the percentage of responses that were appropriate in each of the six conditions, averaged across subjects. For each cell in the table, the highest achievable score is 100%. (A score of 100% would mean that all responses had conformed to the optimal decision rule for the 2 vs. 2 cue conflict condition, i.e., to respond in the direction of the extra cue. It does not mean that all responses would in fact have been correct identifications.) Data for the computer conditions were derived analytically, and reflect the 50% optimal response rate expected when there is no knowledge of the extra cue. This rate is equivalent to chance performance. 50% reflects the level of performance expected by chance alone in all cells of the table.

Effect of task allocation and workload on human-computer system performance

	Manual	Screening	Override	Computer
Low Load	74.6	72.2	71.7	50.0
High Load	61.3	70.9	64.5	50.0

Note: Entries are the mean percentage of system responses that were optimal, across all subjects.

Note that the percentage correct responses is a linear function of the percentage optimal responses. Thus, although the analyses to be presented here are in terms of optimal responses, the results would be essentially unchanged in an analysis based on correct responses.

The next table (from which the first was derived) shows the actual number of appropriate responses for each subject. The bottom rows of the table show the mean number of optimal responses under each condition averaging across subjects; the standard deviation of the number of optimal responses; and the results of a statistical comparison of performance under the six experimental conditions with the computer-only condition.

Performance by individual subjects under conditions involving human participation

Subject Number	Low Load			High Load		
	Manual	Screening	Override	Manual	Screening	Override
1	55	61	50	35	42	46
2	40	35	39	41	36	36
3	36	52	54	41	46	46
4	50	49	45	43	48	39
5	60	46	61	43	40	55
6	57	40	33	37	36	43
7	43	49	55	42	44	51
8	52	27	51	41	58	44
9	62	66	39	38	62	33
10	54	46	44	46	59	43
11	67	63	62	51	52	39
12	48	40	46	33	40	44
13	34	36	48	34	38	40
14	41	57	36	37	66	58
15	42	59	42	43	47	42
16	44	57	62	50	61	59
17	64	63	66	41	62	62
18	63	61	64	64	54	66
19	55	47	43	48	46	36
20	55	44	43	43	41	41
21	55	51	48	34	54	25
22	48	37	44	41	44	32
23	47	48	51	36	38	35
24	45	45	44	39	43	37
Mean	50.71	49.13	48.75	41.71	48.21	43.83
SD	9.04	10.28	9.12	6.78	9.23	10.15
t^a	9.056	7.221	7.924	5.571	7.540	4.744
p^a	<.001	<.001	<.001	<.001	<.001	<.001

Note: Entries are the number of system responses that were optimal out of a maximum of 68 responses.

[a] *t* and *p* refer to comparison, in terms of mean number of optimal responses, between conditions involving humans and computer-only condition. n=24 for all conditions.

The most salient observation regarding these data is the superiority of all the conditions in which human participation occurred in comparison to the computer-only condition. Statistical comparisons (by means of Student's *t*) were performed in each of the six experimental conditions comparing the mean number of appropriate responses with the 50% optimal response rate expected of the com-

puter algorithm (i.e., 34 optimal responses out of a maximum of 68). As shown in the above table, all tests were significant at a level well over .001. Since computer performance is optimal with respect to the four cues available to the computer, the explanation for superior human performance (leaving aside the very remote possibility of chance) must involve use by humans of the extra cue. Subjects were successful in learning the value of the extra cue and in employing that knowledge to improve overall user-computer performance.

This is confirmed by a closer examination of individual subject performance. The next table shows the total number of optimal responses for each subject, summing across the six workload and task allocation conditions. The number of optimal responses expected by chance, i.e., without knowledge of the extra cue, is 34 x 6 = 204. The z-score represents a normal approximation to the binomial, and is used to test the hypothesis that the obtained totals were generated by a chance process (i.e., without knowledge of the extra cue). For 22 out of 24 subjects that hypothesis can be rejected at a confidence level exceeding .001 (one-tailed). The lowest level of significance achieved is .012, for subject 2.

Comparison of performance by individual subjects with chance

Subject Number	Total Optimal Responses[a]	z-score	p
1	289	8.416254	<.001
2	227	2.27734	.012
3	275	7.030048	<.001
4	274	6.931033	<.001
5	305	10.00049	<.001
6	246	4.15862	<.001
7	284	7.921181	<.001
8	273	6.832019	<.001
9	300	9.505417	<.001
10	292	8.713299	<.001
11	334	12.87192	<.001
12	251	4.653694	<.001
13	230	2.574384	.005
14	295	9.010342	<.001
15	275	7.030048	<.001
16	333	12.7729	<.001
17	358	15.24827	<.001
18	372	16.63448	<.001
19	275	7.030048	<.001
20	267	6.23793	<.001
21	267	6.23793	<.001
22	246	4.15862	<.001
23	255	5.049753	<.001
24	253	4.851723	<.001

[a]Entries are the number of system responses that were optimal out of a maximum of 408 responses.

The next step is to compare the six experimental conditions among themselves. A two-factors repeated measures analysis of variance was performed on the data, the results of which are summarized in the following table. The effect of task allocation was not significant. However, the impact of workload was highly significant ($F(1,23) = 15.61$) at a level exceeding .001, and the task allocation by workload interaction was also highly significant ($F(2,46) = 5.29$) at a level of .006.

Analysis of variance for major factors

Source	df	Sum of Squares	Mean Square	F-ratio	p
Allocation	2	187.0625	93.53125	1.310146	>.25
Workload	1	880.125	880.125	15.60896	<.001**
Allocation x Workload	2	392.0313	196.0156	5.294706	.006**
Allocation x Subjects	46	3283.938	71.38995		
Workload x Subjects	23	1296.875	56.38587		
Allocation x Workload x Subjects	46	1702.969	37.02106		

To explore further the interaction between task allocation and workload, a subsidiary analysis was performed. Separate ANOVAs were used to test the impact of task allocation at each of the two levels of workload. The results are shown in the following table. Task allocation had no effect at low workload, but had a highly significant effect ($F(2,46) = 5.23$) at the .006 level under high workload.

Analysis of variance for low workload condition

Source	df	Sum of Squares	Mean Square	F-ratio	p
Allocation	2	51.85938	25.92969	.44704	>.50
Allocation x Subjects	46	2668.141	58.00306		

Analysis of variance for high workload condition

Source	df	Sum of Squares	Mean Square	F-ratio	p
Allocation	2	527.25	263.625	5.229865	.006**
Allocation x Subjects	46	2318.75	50.40761		

In order to obtain a more detailed understanding of these data, a series of contrasts involving paired comparison *t*-tests were carried out, as outlined below.

- Question 1: Does high or low workload affect performance?

  - Manual Condition. The difference in scores between the high- and low-workload conditions for each subject in the manual condition were computed, and these 24 data elements were used to test the null hypothesis that the mean of these numbers was zero. The *t*-statistic was -4.87, with a (two-tailed) significance level of over .001. Thus loading clearly affects performance under human-only task allocation: performance is worse under high loading.

  - Override Condition. For the override condition a similar test was carried out. Here the *t*-statistic was -2.41, with a significance level of .026. It appears that, under the override condition as with the manual condition, high loading reduces performance.

  - Screening Condition. A similar test was performed on the data for the subjects under screening. In this case the *t*-statistic was exactly zero. There was, therefore, no evidence to reject the hypothesis that under this condition performance was not affected by loading level.

- Question 2: Do the different task allocation conditions affect performance?

  - Low Loading. Differences for each subject were computed between the scores for each pair of conditions. The *t*-statistics and accompanying (two-tailed) significance levels were as follows:

	screening - manual	override - screening	override - manual
t:	-0.71	-0.71	-1.29
p(two-tailed):	.486	.486	.211

None of these tests achieves a significance level high enough to reject the hypothesis that the mean is zero in each case. This confirms our previous conclusion that under low loading performance is not affected by the task allocation conditions tested.

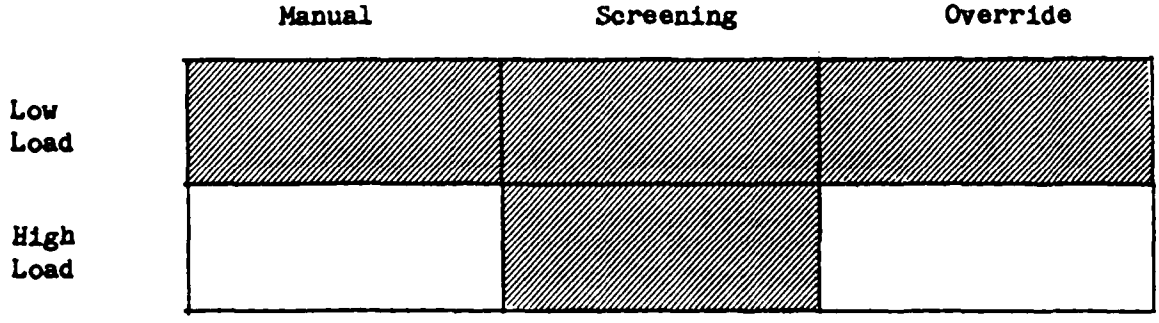
  - High Loading. Similar tests for the data from the high loading cases gave the following results:

	screening - manual	override - screening	override - manual
t:	3.20	-1.898	1.102
p(two-tailed):	.005	.074	.284

Under high loads system performance is worse under the manual condition than under the screening condition. The evidence is also strongly suggestive (although the significance level is only .074) that performance under the override condition is worse than under screening. Finally, however, we cannot reject the hypothesis that performance under manual and override conditions is the same.

These results fit a definite pattern: (1) at low levels of workload, the three allocation conditions are equivalent; (2) high workload causes decrements in the manual and override conditions, but has no effect on performance in the screening condition. Thus, performance under the conditions (manual/low-load), (screening/low-load), (override/low-load), and (screening/high-load) are all the same; but marked worsening in performance occurs under the two treatment combinations (manual/high-load) and (override/high-load). The following table graphically illustrates this pattern.

Decrements in performance due to workload under different task allocation conditions



Note: There were no significant differences among shaded cells.

The presence of this pattern was, finally, tested directly. We computed the average performance under the four shaded conditions for each individual, and subtracted the average performance under the other two conditions. For this data the t-statistic was 5.13--more significant than any other paired comparison we have considered above, at far above the .001 level. We conclude that under manual and override conditions a real worsening of performance occurs under high load, but that for the screening condition load does not affect performance within the range tested.

Note on statistical significance: In all, 12 contrasts have been performed, including the two subsidiary ANOVAs. The primary purpose of these tests has been to explore patterns in the data, as a basis for further research; thus, the results have been reported in terms of the ordinary criteria of statistical significance. It should be noted, however, that a somewhat higher standard of significance than usual would need to be applied to maintain a low overall error rate. If the probability of obtaining one or more significant results by chance across 12 orthogonal, planned comparisons is to be kept at 10% or less, the criterion of significance for individual comparisons would have to be set at α=.009 (cf., Myers, 1972; p.359). In fact, that standard would not affect the main thrust of our conclusions: only the impact of workload in the override condition (at a significance level of "merely" .026) would be called into question. However, since the present set of comparisons is far from orthogonal, a .009 criterion would be highly conservative. For example, if we estimate our 12 non-orthogonal comparisons as roughly equivalent to four orthogonal comparisons, the required criterion of significance drops to .026, and all comparisons which we regarded as significant under the conventional criterion remain so.

3.3.2 Other analyses. Analyses have been performed on issues associated with the stability of Ss behavior over the course of the six hours of data collection per S. There are no significant differences between early and late trials in each session or between Ss encountering the experimental conditions in different orders.

3.4 Conclusions

The results of the pilot experiment support the following conclusions:

- The ability of Ss to learn in a short time the usefulness of an ID cue not utilized by the computer has been confirmed.
- The ability of Ss to utilize information not processed by the computer to improve system performance has been demonstrated.
- Of the three interactive task allocation schemes compared in the experiment--human operator making all ID decisions, human override of computer decisions, and computer screening of decisions--the screening approach led to a significant improvement in performance when the processing load was high.

Finally, and perhaps most important, the usefulness of this research paradigm for the experimental testing of hypotheses regarding human-computer task allocation and for the evaluation of alternative task allocation systems has been demonstrated.

4.0 IMPLICATIONS FOR HUMAN-COMPUTER SYSTEM DESIGN

4.1 Present Results

There has been a tendency in the design of increasingly sophisticated human-computer systems to assign all possible tasks to the computer, leaving the human operator with a smaller and smaller role. There are many reasons for this trend, but a major one may be our lack of knowledge about how to design a system which makes maximum simultaneous use of both the human and the computer. The research reported here has amounted to a pilot demonstration of one way to generate the knowledge needed to identify task allocation design principles for complex human-computer systems.

Within the air defense context used for demonstration, it is clear that major human contributions center around human abilities to perceive patterns, to learn, and to adapt over relatively short periods of time. Although the prototype representation of the air defense system employed here is much simpler than any real system, the results suggest that some strategies for task allocation will work better than others and that in some settings under high processing loads neither fully manual nor fully automated systems work as well as a scheme permitting some collaboration between the system's human and computer elements.

Under low-workload conditions, and where the computer model in the ID task was significantly incomplete, any of the three conditions involving human participation was superior to computer-only performance. However, at high workload this advantage could only be maintained if the computer assisted the human in the function of "allocating tasks," i.e., by directing the user's attention to subproblems where his contribution was most needed. In short, a complementarity has been observed, in which the human helps the computer by learning and adapting to novel situations, and in which the computer helps the human by reducing the size of the problem.

Although the results of this project must be regarded as tentative, they suggest that provision of a capability for the operator to override computer decisions is suboptimal when compared with methods which are more collaborative in nature. Of the four very different task allocation systems tested, the only one which enabled both the computer and the human operator to make maximum use of their respective contributions was one which utilized the high-speed capability of the computer to process all aircraft according to its available (and programmed) information resources and then to signal the operator to attend to those aircraft it was unable to classify reliably.

While the importance of the role of a particular human operator in a complex and highly automated distributed or hierarchical system is not always easy to determine, in the case of more localized systems or in the case of breakdowns in the networking of distributed systems the ability for the human operator to supply information to the system and affect system behavior may become highly desirable or even essential. The present pilot research has attempted to shed some light on how that capability for human-machine interaction might best be achieved and to suggest the general direction of further research.

4.2 Future Directions

The present research is a first step toward (1) a more comprehensive set of guidelines for the design of person-computer systems, and (2) the development of

specific useful products and recommendations for Army air defense requirements. An experimental testbed has been developed within which further progress can take place on both of these fronts.

Successful design guidelines will eventually match properties of the task, user, and computer-based model to appropriate schemes of collaborative problem solving. We have addressed only a small subset of the relevant issues in this research, but many others seem capable of elucidation by much the same approach. Important questions that remain to be addressed include:

- Learning. What are the limits of human ability to acquire and utilize knowledge not in the computer, and in particular, to adapt rapidly in dynamic environments? Experiments need to be conducted in which human reliance on different cues is quantified in contexts where a variety of rates of change in the relevance and diagnosticity of "extra" cues are simulated. Categorizations of cues and information processing conditions are needed to enable us, eventually, to predict potential human contributions to human-computer collaborative performance.

- Human and computer knowledge representations. Closer examination is required of what is in fact learned when people learn about an "extra" cue, i.e., how humans internally represent knowledge they acquire, the compatibility of those representations with computer-driven displays, and how those representations affect the human's ability to use what they learn in human-computer collaboration. Experiments need to be conducted which compare the "mental models" of successful and unsuccessful users, and which vary the degree of match with computer-presented displays.

- Cognitive task analysis. How should computer and human tasks be broken down as a prelude to task allocation? Can a methodology based on hypotheses about component cognitive tasks (such as the "elementary information processes" of Newell and Simon, 1972) lead to an analysis of performance that is more fine-grained and flexible than traditional task analyses? What "elementary information processes" are critical in human cognitive performance of the type that occurs in human-computer problem solving?

- Suboptimal processes in inference and choice. What are the implications of biases and fallacies in human reasoning for person-computer system design? For example, in the present research although the "extra" cue was clearly learned, performance still fell far short of the 100% optimal response rate. Should this be attributed to shortcomings in comprehending the full relevance or importance of the extra cue, and/or to problems in combining it with other information to draw conclusions? Experiments are needed to clarify the circumstances under which the human should serve merely as an additional "sensor" for the computer system, and when the human should perform higher-level integrative processes in parallel with or in place of the computer.

- Workload and attention. Does low workload lead to suboptimal performance that is comparable to decrements observed under high workloads? Very little research has been done to analyze the degradation of cognitive processes that takes place under conditions of diminished

"vigilance." Other important questions center on human strategies for combining concurrent tasks and switching attention.

- "Blending" human-computer expertise. To what extent is the optimal task allocation not a crisp delegation of tasks to either the human or computer, but rather a more integrated blending of roles? For example, the human (or computer) might take primary responsibility for a task while the other monitors his (its) performance, intervening only when certain conditions are fulfilled. The screening condition in the present study is a first step toward exploring this concept.

Experimental work to address these and other questions can be conducted in a framework similar to that developed in the present research. In addition, the similarity of the present experimental system to actual air defense environments might be increased along several dimensions: for example,

- by restricting the availability of weapons, thus making ID and engagement separate decisions;
- by reducing feedback, thus more nearly approximating the current "fire and forget" practice;
- by simulating adaptive enemy responses to air defense engagements; and
- by simulating breakdowns in the networks supplying information and commands to the air defense unit.

At the same time, concrete design alternatives pertinent to Army systems currently under development can be evaluated. The present research, as noted in Section 1.4, was inspired largely by design dilemmas arising in the development of the IHAWK air defense system toward a more automated configuration. The question of what to automate and what to leave in the hands of the operator is a pointed one in the light of expressed dissatisfaction with aspects of the more fully-automated PATRIOT system, in conjunction with the inadequacy of largely manual air defense systems.

The development and testing of design guidelines can and should be motivated by real-world problems. The result is, we expect, both a more valid set of general guidelines and some very real, immediate benefits for those who must make design choices in today's system development environment.

REFERENCES

Association for Computing Machinery, Inc. Computer graphics: A quarterly report of SIGGRAPH. SIGGRAPH '82 Conference Proceedings, 16(3). Baltimore, 1982.

Beard, L.H. Planning a management information system: Some caveats and contemplations. Financial Executive, May 1977, pp. 34-39.

Buchanan, B.G. Research on expert systems (Report No. HPP-8-1). Stanford, CA: Computer Science Department, School of Humanities and Sciences, February 1981.

Cohen, M.S. Research on cognitive collaboration between persons and computers. Proceedings of the 6th MIT/ONR Workshop on C^3 Systems. Cambridge, MA: Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, December 1983.

Cohen, M.S., Brown, R.V., Seaver, D.A., and Ulvila, J.W. Operability in attack submarine combat systems: An exploratory overview (Technical Report 82-2). Falls Church, VA: Decision Science Consortium, Inc., April 1982.

Engel, S.E., and Granda, R.E. Guidelines for man/display interfaces (Technical Report TR00.27200). Poughkeepsie, NY: IBM Poughkeepsie Laboratory, December 1975.

Fitts, P.M. Engineering psychology and equipment design. In Stevens, S.S. (Ed.), Handbook of experimental psychology. New York: John Wiley & Sons, Inc., 1951.

Hawley, J.K., Howard, C.W., and Martellaro, A.J. Optimizing operator performance on advanced training simulators: Preliminary development of a performance assessment and modeling capability (Technical Report 573). Ft. Bliss, TX: ARI Field Unit, U.S. Army Research Institute for the Behavioral and Social Sciences, February 1982.

McCarthy, J. Epistomological problems of artificial intelligence. Proceedings of the Fifth International Joint Conference on Artificial Intelligence. Cambridge, MA: Massachusetts Institute of Technology, 1977.

Miller, R.S. A framework for establishing the need for and value of C^2 programs. Monterey, CA: U.S. Naval Postgraduate School, June 1980.

Myers, J.L. Fundamentals of experimental design. Boston, MA: Allyn and Bacon, Inc., 1972.

Newell, A., and Simon, H.A. Human problem solving. Englewood Cliffs, NJ: Prentice-Hall, 1972.

Payne, J.W. Task complexity and contingent processing in decision making: An information search and protocol analysis. Organizational Behavior and Human Performance, 1978, 16, 366-387.

Phelps, R.H., Halpin, S.M., and Johnson, E.M. A decision support framework for decision aid designers (Technical Report 504). Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences, January 1981.

Ramsey, H.R., and Atwood, M.E. Human factors in computer systems: A review of the literature (Technical Report SAI-79-111-DEN). Englewood, CO: Science Applications, Inc., September 1979.

Rouse, W.B. Human-computer interaction in multitask situations. IEEE Transactions on Systems, Man, and Cybernetics, May 1977, 7(5), 384-392.

Singleton, W.T. Man-machine systems. Baltimore, MD: Penguin Books, Inc., 1974.

Swanson, E.B. Management and information systems: Appreciation and involvement. Management Science, 1974, 21, 178-188.

APPENDIX A: INSTRUCTIONS TO SUBJECTS

The first part of each of the three experimental sessions for each S was devoted to instruction and practice. Each S was first given a set of written instructions to read. Each set of instructions consisted of two booklets, one containing written instructions and the other containing three pairs of printouts of the "radar" and text screens made at three different times. Other than combining the text and display printout booklets and reorienting the pages here for the different binding, the instructions for the three sessions (corresponding to the three task allocation conditions) are included as presented to Ss.

Note that "Condition 1.0" corresponds to the manual condition, "Condition 2.0" to the screening condition, and "Condition 3.0" to the override condition. The printouts of screen displays in the booklets correspond most closely to the practice session of 50 aircraft with a load set intermediate to the low- and high-load conditions of the experimental sessions.

For a quicker reading of this material, it is possible to read only the Condition 2.0 (screening) booklet, remembering that the other two conditions differ either by showing all aircraft as unknowns (manual condition) or by responding randomly as hostile or friend when 2 cues favor each (override condition).

All Ss were told not to rely on any of the illustrative aircraft classifications made in the instructional material, that the illustrations were used only to show how the displays and keyboard were to be used.

1.0 INSTRUCTIONS TO SUBJECTS

Purpose of Experiment

The experiment you are about to take part in is one which may help us to understand how people handle various kinds of information in making decisions.

Your Basic Job

You will be the operator of a fictional air defense system and must decide which of the aircraft approaching you are friendly and which are enemies. You will see a "radar" picture of the territory you are responsible for covering, and--on a second display screen--you will find a variety of information that, in addition to the radar display, will help you make decisions. Your job is to make sure that as many enemy aircraft as possible are shot down and that as few friendly aircraft as possible are shot down. You will be paid according to how well you do this.

The first pair of displays in the display booklet includes an example on the left of what the radar screen looks like. You are located at the bottom, where the two straight lines come together. The whole pie-slice-shaped area is the area your radar can see. An enemy airbase is located out of range of your radar, off the top left of the screen; a friendly airbase is located out of range of your radar, off the top right of the screen. At the bottom where you are located, is a friendly town which you are trying to protect.

The "U"-shaped figures on the screen represent the locations of UNKNOWN aircraft. These aircraft may be either friends or enemies. All aircraft will appear at the top of the screen and move slowly toward you. All have the same speed.

You have an unlimited supply of missiles which you can use to shoot down aircraft. When an aircraft is selected for shooting before it reaches the second dotted circle from you (as explained later), a missile will automatically be fired at it when it reaches the second dotted circle, and it will be destroyed and vanish from the screen when it reaches the dotted circle nearest you.

When aircraft reach the second dotted circle, enemies always begin to attack the town, and friends, of course, do not. Although you will not "see" this with your radar, the computer will then know which aircraft are friends and which are enemies, and any mistakes you make (not shooting down an enemy, or shooting down a friend) will be pointed out to you by the computer. This will be done by having the aircraft symbol begin to flash on and off after it reaches the second dotted circle. A flashing symbol means you have made an error.

How to Tell Friendly from Hostile Aircraft

In today's experiment, there are a number of clues to help you tell a friend from an enemy. On the radar screen itself you can see the locations of the aircraft. From the radar screen you can see where the aircraft appear to be coming from. In addition, the two straight dotted lines are known as "safe passage corridors." These are the paths friendly pilots have been told to use if possible when travelling through your area. Therefore, all other things being equal, friendly aircraft are more likely to be travelling down these dotted lines than are enemy aircraft.

At the same time you watch the radar screen you must also watch a second screen which will display additional information to help you make decisions. An example of the second screen, recorded at the same time as the radar screen you have been looking at, is shown to the right of the radar screen in the display booklet. Most of the display is a table showing several kinds of information for each aircraft.

On the left side are the numbers for the aircraft shown on the screen at this moment. The next three columns show information about the aircraft that will be useful to you in making your decisions. These are called the ALTITUDE ("ALT"), the "IFF," and the "FREQUENCY" of each aircraft. What these really are makes no difference here. Friends have been asked to fly through your zone at high altitude, while enemies--preparing for attacks on the town--will usually be flying lower. In the table, for example, aircraft number 1 is flying low ("lo"), while aircraft number 5 is flying high ("hi"). On the basis of this one cue only, #1 looks like an enemy and #5 looks like a friend.

"IFF" is a signal generated by the aircraft in response to your signal to it. The computer will show you whether this signal appears to be from a friendly ("f") or hostile ("h") aircraft. In the table, for example, aircraft number 1, based on the IFF cue only, looks like a friend, while #2 looks hostile.

Finally, the "FRQ" column shows the frequency being used by radio transmissions from the different aircraft. Friendlies usually use uhf frequencies and hostiles usually use vhf frequencies.

To summarize, you have the geographic information provided by the radar screen plus four other cues:

	IN DOTTED CORRIDOR	ALT	IFF	FRQ
friend	yes	hi	f	uhf
hostile	no	lo	h	vhf

The above table shows the most likely observations you will make for a friend or enemy on several cues. NONE of these cues are completely reliable. Friends sometimes forget to follow the dotted lines and enemies sometimes happen to fly down them; sometimes friends fly too low and enemies too high, and so on. Nevertheless, if you use as many of the cues as you can, along with information provided by the radar screen you can do a good, but not perfect, job of telling friends from enemies.

How to Shoot at Aircraft

Up until an aircraft reaches the second dotted circle from you, you can tell the system that you believe an aircraft is an enemy by typing an aircraft number followed by a carriage return. On the radar screen shown before, the operator (you) has typed the numbers 3, 4, and 8, each followed by a carriage return. The computer indicates this by placing a six-sided figure around the aircraft symbol. When these aircraft reach the lowest dotted circle they will be destroyed. Other aircraft will not be shot at.

If you make a mistake or change your mind, simply type the number again and the screen symbol will change back. Incorrect numbers (numbers of aircraft not on the screen) will be ignored.

You can type aircraft numbers until just before they reach the second dotted circle.

The second screen will indicate in the OPERATOR column, your decision for each aircraft on the screen. In the previous chart, the operator has asked to shoot aircraft 3, 4, and 8.

The column labeled SYSTEM is not used today.

Results and Your Score

When an aircraft reaches the second dotted circle, as shown in the second pair of displays [3 & 4] in the display booket, its true identity becomes known to the computer, and it will indicate on the second screen in the RESULT column if your decision was right or wrong. At the same time, any aircraft for which you have done the wrong thing will begin to flash on and off. On the radar screen shown, aircraft #2 is momentarily "off" in its flashing pattern. For each aircraft you identify correctly, you receive one point, shown as your score in the upper right-hand corner of the second screen. Just below your score will be shown the total number of aircraft that have reached the second dotted circle so far.

Payment

You will be paid six cents for each aircraft you correctly identify before the experiment ends. No one will be paid less than four dollars per hour. A few aircraft at the very end of the experiment will not count.

One More Pair of Screens

As aircraft are shot down or reach the bottom of the radar screen, they vanish. On the second (tabular) screen, information about new targets appearing at the top is written over information about the oldest targets. In other words, the oldest lines on the second screen are replaced with information on the newest aircraft. In this way a particular aircraft remains in the table only long enough for you to check the result before it is replaced with a new line of information about a new aircraft. The third pair of displays [5, 6] shows how the screens might look after aircraft 1 through 4 have gone on by or been destroyed. Aircraft 4 has been destroyed, so is missing from the radar screen. It is to be replaced in the next second or so on the second screen by information on aircraft number 20.

Final Instructions

Please reread these instructions if anything is not clear and ask any questions you have before beginning. There will be a short practice session before we begin the actual experiment.

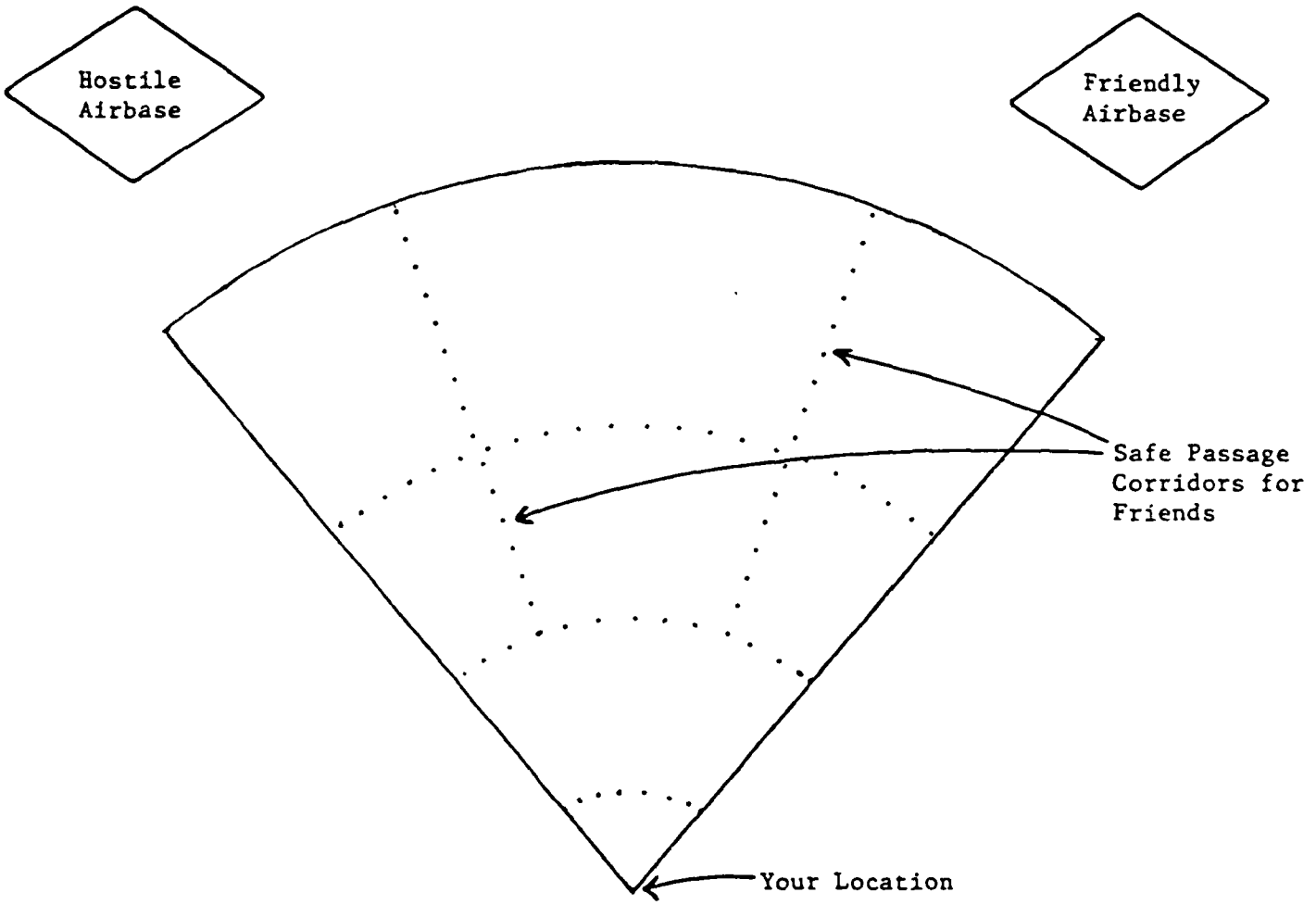
In order to do well you will need to pay attention to anything and everything that might help you make decisions. Remember the general setting: the enemy is

off screen to the left, and the friendly base is off-screen to the right. Remember to notice who is in the "safe passage corridors." Remember to check the altitude, IFF, and frequency of each target. By using all possible information, you can do very well. A summary of information that can help you is given on the next page for use now and during the experiment.

Remember too, some tests and days will be easier or harder than others.

SUMMARY OF INFORMATION

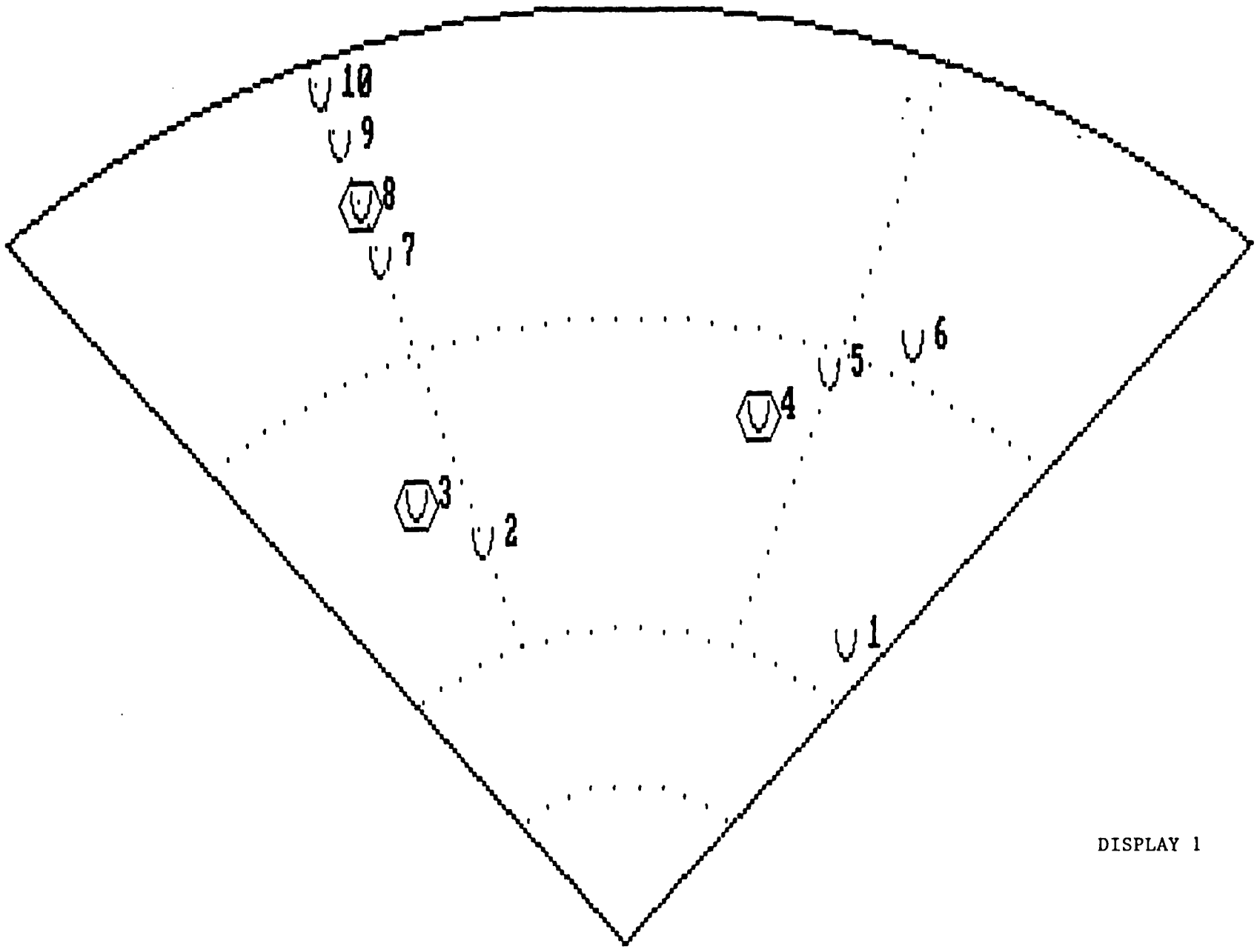
Radar Screen



Tabular Screen

Cues tend to be as follows:

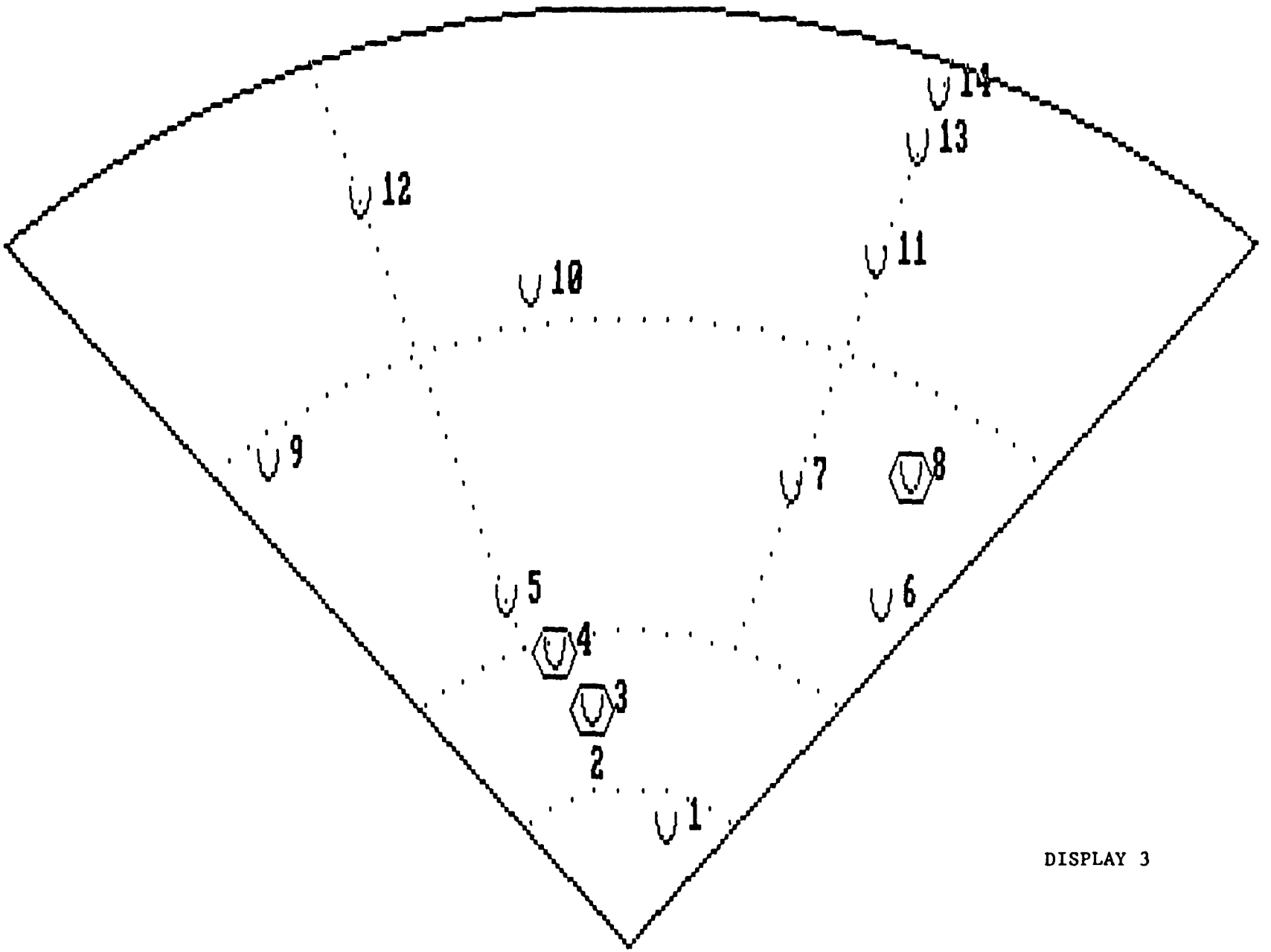
	ALT	IFF	FRQ
Friend	hi	f	uhf
Hostile	lo	h	vhf



DISPLAY 1

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
1	lo	f	uhf	?			0
2	lo	h	uhf	?			
3	lo	h	uhf	?	shoot		OUT OF
4	hi	f	vhf	?	shoot		0
5	hi	f	uhf	?			
6	lo	h	vhf	?			
7	lo	f	uhf	?			
8	hi	f	uhf	?	shoot		
9	hi	h	vhf	?			
10	hi	h	uhf	?			

DISPLAY 2



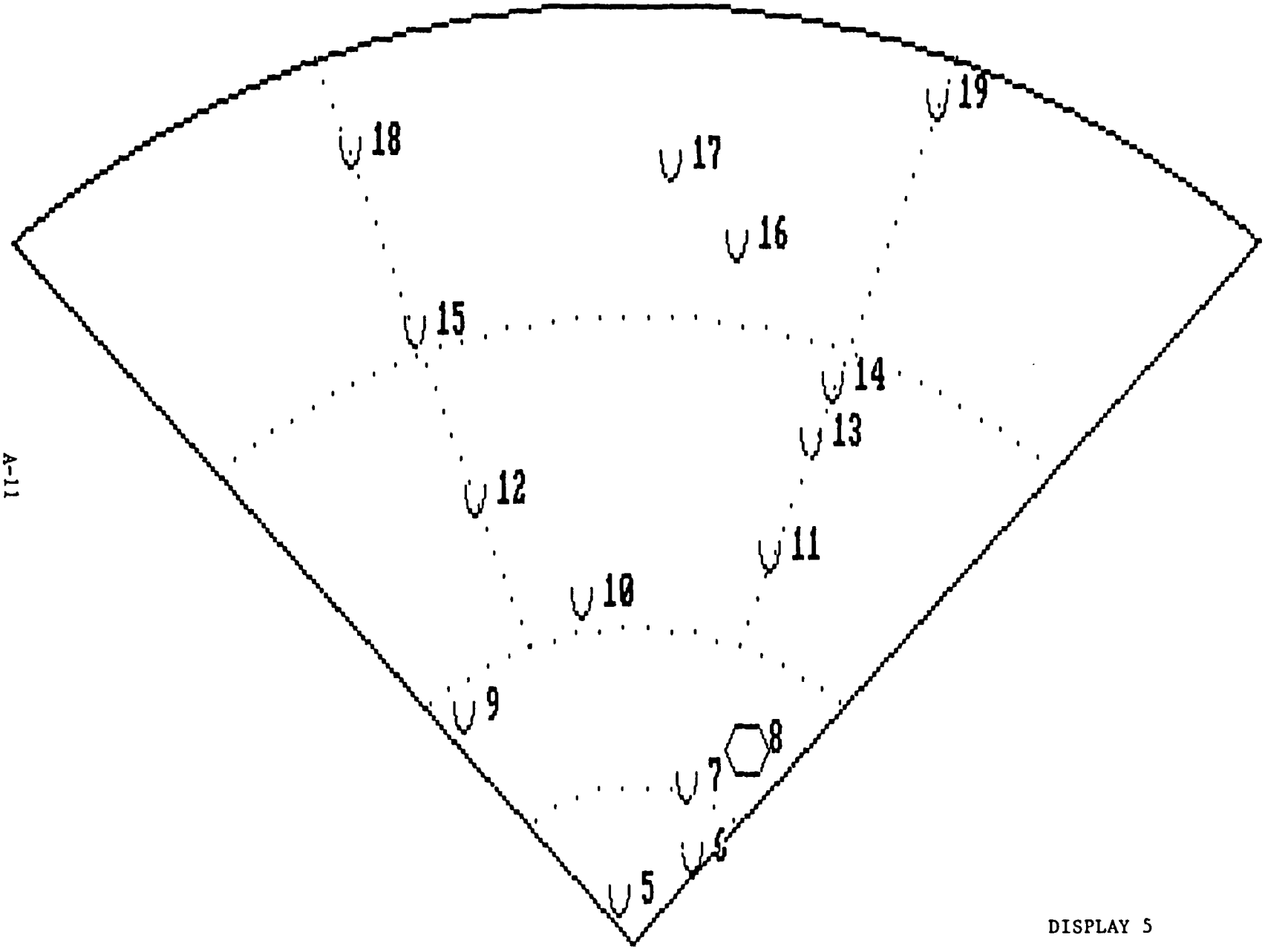
DISPLAY 3

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT
1	hi	h	vhf	?		right
2	hi	h	uhf	?		wrong
3	hi	h	uhf	?	shoot	right
4	hi	f	vhf	?	shoot	right
5	lo	f	vhf	?		
6	lo	h	vhf	?		
7	hi	f	uhf	?		
8	hi	f	vhf	?	shoot	
9	hi	h	vhf	?		
10	lo	h	vhf	?		
11	lo	f	vhf	?		
12	lo	h	vhf	?		
13	hi	f	vhf	?		
14	lo	f	uhf	?		

SCORE
3

OUT OF
4

DISPLAY 4



DISPLAY 5

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
17	lo	h	uhf	?			5
18	hi	f	vhf	?			
19	hi	h	vhf	?			OUT OF
4	hi	f	vhf	?	shoot	right	9
5	lo	f	vhf	?		wrong	
6	lo	h	vhf	?		right	
7	hi	f	uhf	?		right	
8	hi	f	vhf	?	shoot	wrong	
9	hi	h	vhf	?		wrong	
10	lo	h	vhf	?			
11	lo	f	vhf	?			
12	lo	h	vhf	?			
13	hi	f	vhf	?			
14	lo	f	uhf	?			
15	lo	h	uhf	?			
16	lo	f	vhf	?			

DISPLAY 6

2.0 INSTRUCTIONS TO SUBJECTS

Purpose of Experiment

The experiment you are about to take part in is one which may help us to understand how people handle various kinds of information in making decisions.

Your Basic Job

You will be the operator of a fictional air defense system and must decide which of the aircraft approaching you are friendly and which are enemies. You will see a "radar" picture of the territory you are responsible for covering, and--on a second display screen--you will find a variety of information that, in addition to the radar display, will help you make decisions. Your job is to make sure that as many enemy aircraft as possible are shot down and that as few friendly aircraft as possible are shot down. You will be paid according to how well you do this.

The first pair of displays in the display booklet includes an example on the left of what the radar screen looks like. You are located at the bottom, where the two straight lines come together. The whole pie-slice-shaped area is the area your radar can see. An enemy airbase is located out of range of your radar, off the top left of the screen; a friendly airbase is located out of range of your radar, off the top right of the screen. At the bottom where you are located, is a friendly town which you are trying to protect.

The "U"-shaped figures on the screen represent the locations of UNKNOWN aircraft. These aircraft may be either friends or enemies. The "+"-shaped figures on the screen represent aircraft the computer aid has decided are probably friends. The "◊"-shaped figures represent aircraft the aid has decided are probably hostile. The aid has made these decisions on the basis of how many of four cues available to it (corridor, altitude, iff, and frequency) agree on friend or hostile. Where it is unable to decide on the basis of these cues it marks the aircraft with a "U." All aircraft will appear at the top of the screen and move slowly toward you. All have the same speed.

You have an unlimited supply of missiles which you can use to shoot down aircraft. When an aircraft is selected for shooting before it reaches the second dotted circle from you (as explained later), a missile will automatically be fired at it when it reaches the second dotted circle, and it will be destroyed and vanish from the screen when it reaches the dotted circle nearest you.

When aircraft reach the second dotted circle, enemies always begin to attack the town, and friends, of course, do not. Although you will not "see" this with your radar, the computer will then know which aircraft are friends and which are enemies, and any mistakes you make (not shooting down an enemy, or shooting down a friend) will be pointed out to you by the computer. This will be done by having the aircraft symbol begin to flash on and off after it reaches the second dotted circle. A flashing symbol means you have made an error.

How to Tell Friendly from Hostile Aircraft

In today's experiment, there are a number of clues to help you tell a friend from an enemy. On the radar screen itself you can see the locations of the aircraft. From the radar screen you can see where the aircraft appear to be

coming from. In addition, the two straight dotted lines are known as "safe passage corridors." These are the paths friendly pilots have been told to use if possible when travelling through your area. Therefore, all other things being equal, friendly aircraft are more likely to be travelling down these dotted lines than are enemy aircraft.

At the same time you watch the radar screen you must also watch a second screen which will display additional information to help you make decisions. An example of the second screen, recorded at the same time as the radar screen you have been looking at, is shown to the right of the radar screen in the display booklet. Most of the display is a table showing several kinds of information for each aircraft.

On the left side are the numbers for the aircraft shown on the screen at this moment. The next three columns show information about the aircraft that will be useful to you in making your decisions. These are called the ALTITUDE ("ALT"), the "IFF," and the "FREQUENCY" of each aircraft. What these really are makes no difference here. Friends have been asked to fly through your zone at high altitude, while enemies--preparing for attacks on the town--will usually be flying lower. In the table, for example, aircraft number 1 is flying low ("lo"), while aircraft number 3 is flying high ("hi"). On the basis of this one cue only, #1 looks like an enemy and #3 looks like a friend.

"IFF" is a signal generated by the aircraft in response to your signal to it. The computer will show you whether this signal appears to be from a friendly ("f") or hostile ("h") aircraft. In the table, for example, aircraft number 1, based on the IFF cue only, looks like a friend, while #2 looks hostile.

Finally, the "FRQ" column shows the frequency being used by radio transmissions from the different aircraft. Friendlies usually use uhf frequencies and hostiles usually use vhf frequencies.

To summarize, you have the geographic information provided by the radar screen plus four other cues:

	IN DOTTED CORRIDOR	ALT	IFF	FRQ
friend	yes	hi	f	uhf
hostile	no	lo	h	vhf

The above table shows the most likely observations you will make for a friend or enemy on several cues. NONE of these cues are completely reliable. Friends sometimes forget to follow the dotted lines and enemies sometimes happen to fly down them; sometimes friends fly too low and enemies too high, and so on. Nevertheless, if you use as many of the cues as you can, along with information provided by the radar screen, you can do a good, but not perfect, job of telling friends from enemies.

How to Shoot at Aircraft

Up until an aircraft reaches the second dotted circle from you, you can tell the system if you disagree with its classifications (as unknown, friend, or hostile)

by typing an aircraft number followed by a carriage return. On the radar screen shown before, the operator (you) has typed the numbers 1, 3, and 6, each followed by a carriage return. The computer indicates this by placing a six-sided figure around the aircraft symbol. Unknowns or friends will be reclassified as hostile, and hostiles will be reclassified as friends this way. Aircraft identified as hostile will be destroyed when they reach the lowest dotted circle. Other aircraft will not be shot at.

If you make a mistake or change your mind, simply type the number again and the screen symbol will change back. Incorrect numbers (numbers of aircraft not on the screen) will be ignored.

You can type aircraft numbers until just before they reach the second dotted circle.

The second screen will indicate in the OPERATOR column, your decision for each aircraft on the screen. In the previous chart, the operator has reversed the computer's identifications for aircraft 1, 3, and 6.

The column labeled SYSTEM shows the computer aid's recommended action--the action that will be taken unless you change it.

Results and Your Score

When an aircraft reaches the second dotted circle, as shown in the second pair of displays [3 & 4] in the display booklet, its true identity becomes known to the computer, and it will indicate on the second screen in the RESULT column if your decision was right or wrong. At the same time, any aircraft for which you have done the wrong thing will begin to flash on and off. For each aircraft you identify correctly, you receive one point, shown as your score in the upper right-hand corner of the second screen. Just below your score will be shown the total number of aircraft that have reached the second dotted circle so far.

Payment

You will be paid six cents for each aircraft you correctly identify before the experiment ends. No one will be paid less than four dollars per hour. A few aircraft at the very end of the experiment will not count.

One More Pair of Screens

As aircraft are shot down or reach the bottom of the radar screen, they vanish. On the second (tabular) screen, information about new targets appearing at the top is written over information about the oldest targets. In other words, the oldest lines on the second screen are replaced with information on the newest aircraft. In this way a particular aircraft remains in the table only long enough for you to check the result before it is replaced with a new line of information about a new aircraft. The third pair of displays [5, 6] shows how the screens might look after aircraft 1 through 4 have gone on by or been destroyed. Aircraft 6 has been destroyed, so is missing from the radar screen.

Final Instructions

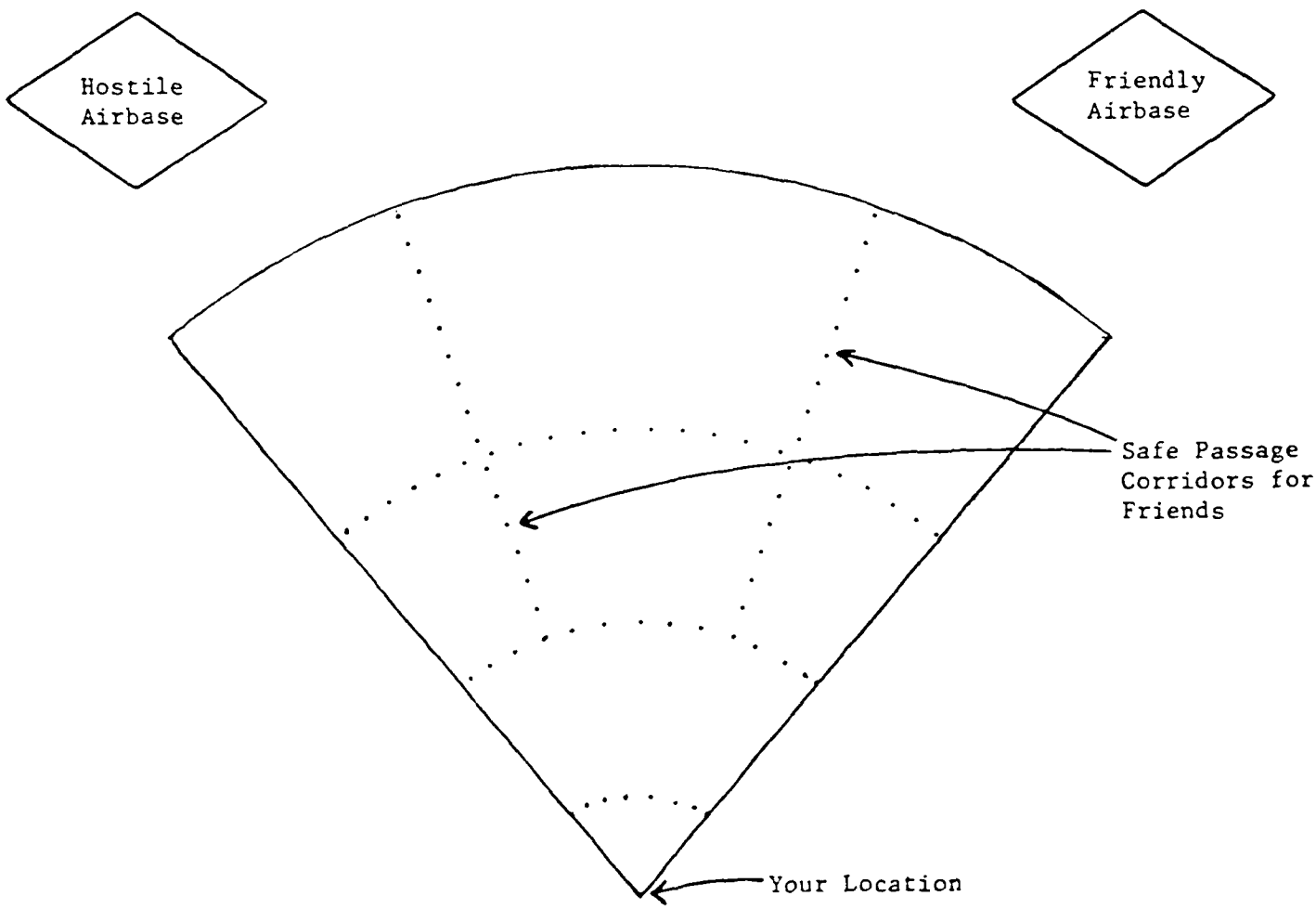
Please reread these instructions if anything is not clear and ask any questions you have before beginning. There will be a short practice session before we begin the actual experiment.

In order to do well you will need to pay attention to anything and everything that might help you make decisions. Remember the general setting: the enemy is off screen to the left, and the friendly base is off-screen to the right. Remember to notice who is in the "safe passage corridors." Remember to check the altitude, IFF, and frequency of each target. By using all possible information, you can do very well. A summary of information that can help you is given on the next page for use now and during the experiment.

Remember too, some tests and days will be easier or harder than others.

SUMMARY OF INFORMATION

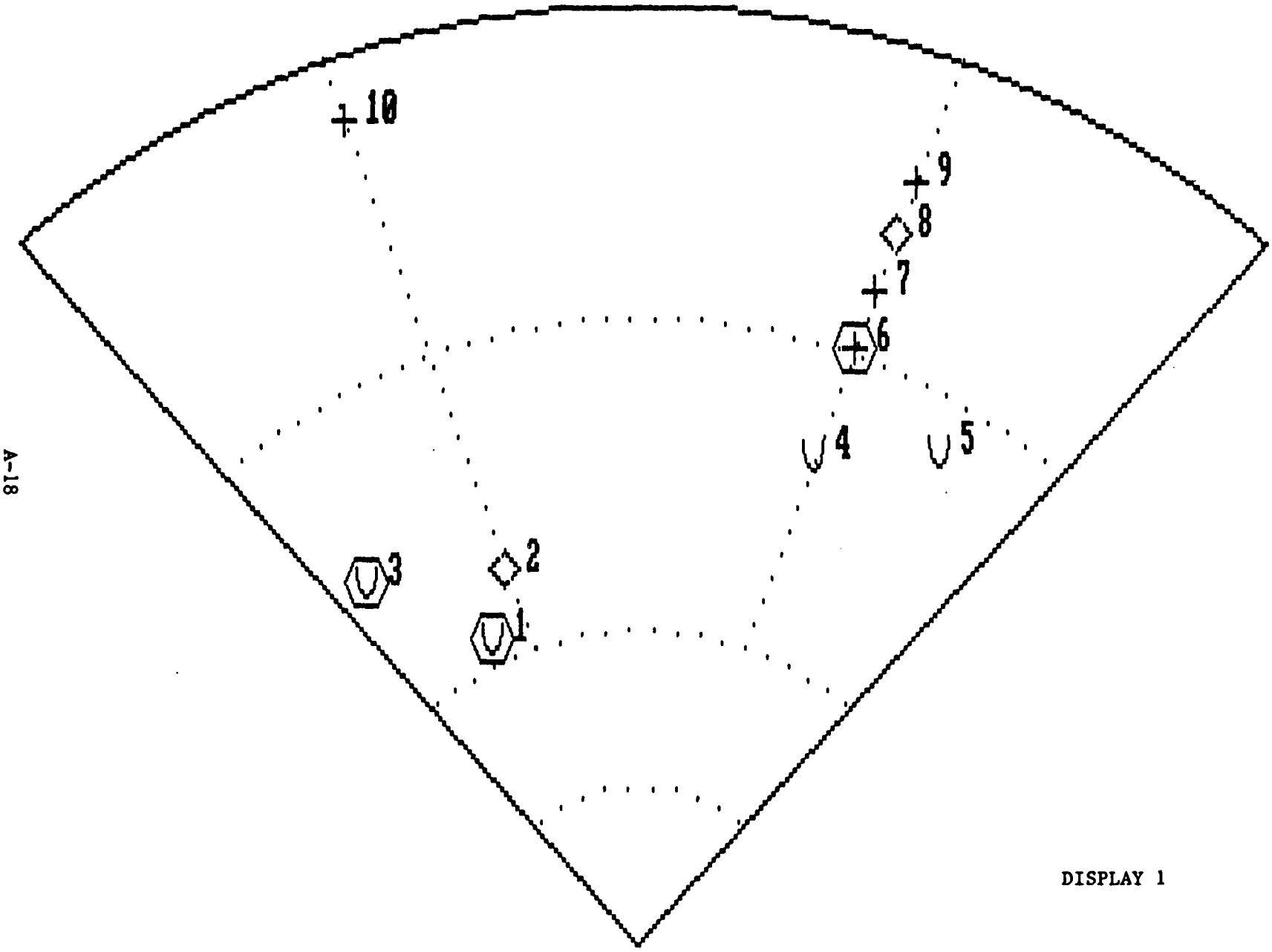
Radar Screen



Tabular Screen

Cues tend to be as follows:

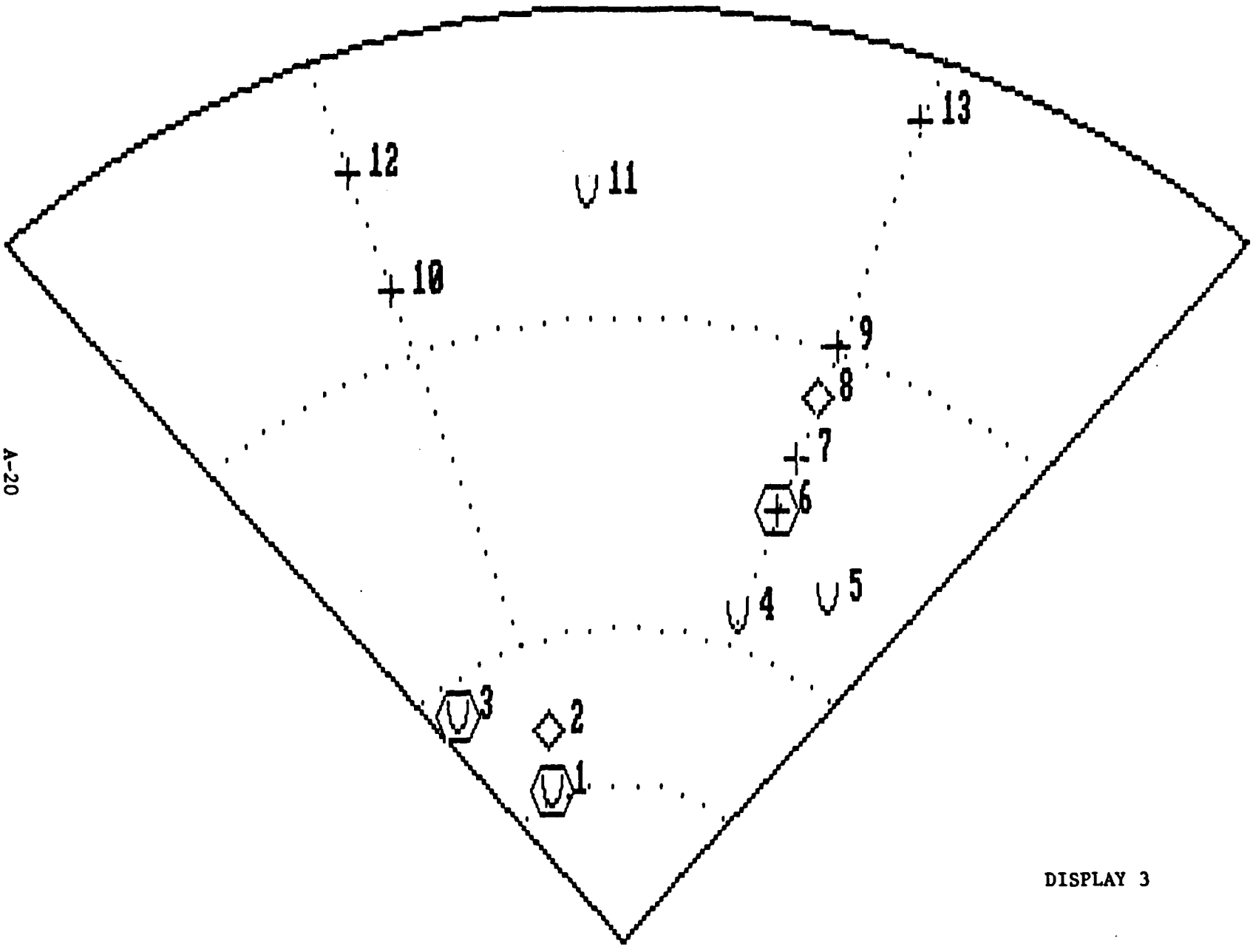
	ALT	IFF	FRQ
Friend	hi	f	uhf
Hostile	lo	h	vhf



DISPLAY 1

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
1	lo	f	uhf	?	shoot		0
2	lo	h	vhf	shoot			
3	hi	h	uhf	?	shoot		OUT OF
4	lo	f	vhf	?			0
5	lo	f	uhf	?			
6	hi	h	uhf	hold	shoot		
7	hi	f	vhf	hold			
8	lo	h	vhf	shoot			
9	hi	f	uhf	hold			
10	hi	f	vhf	hold			

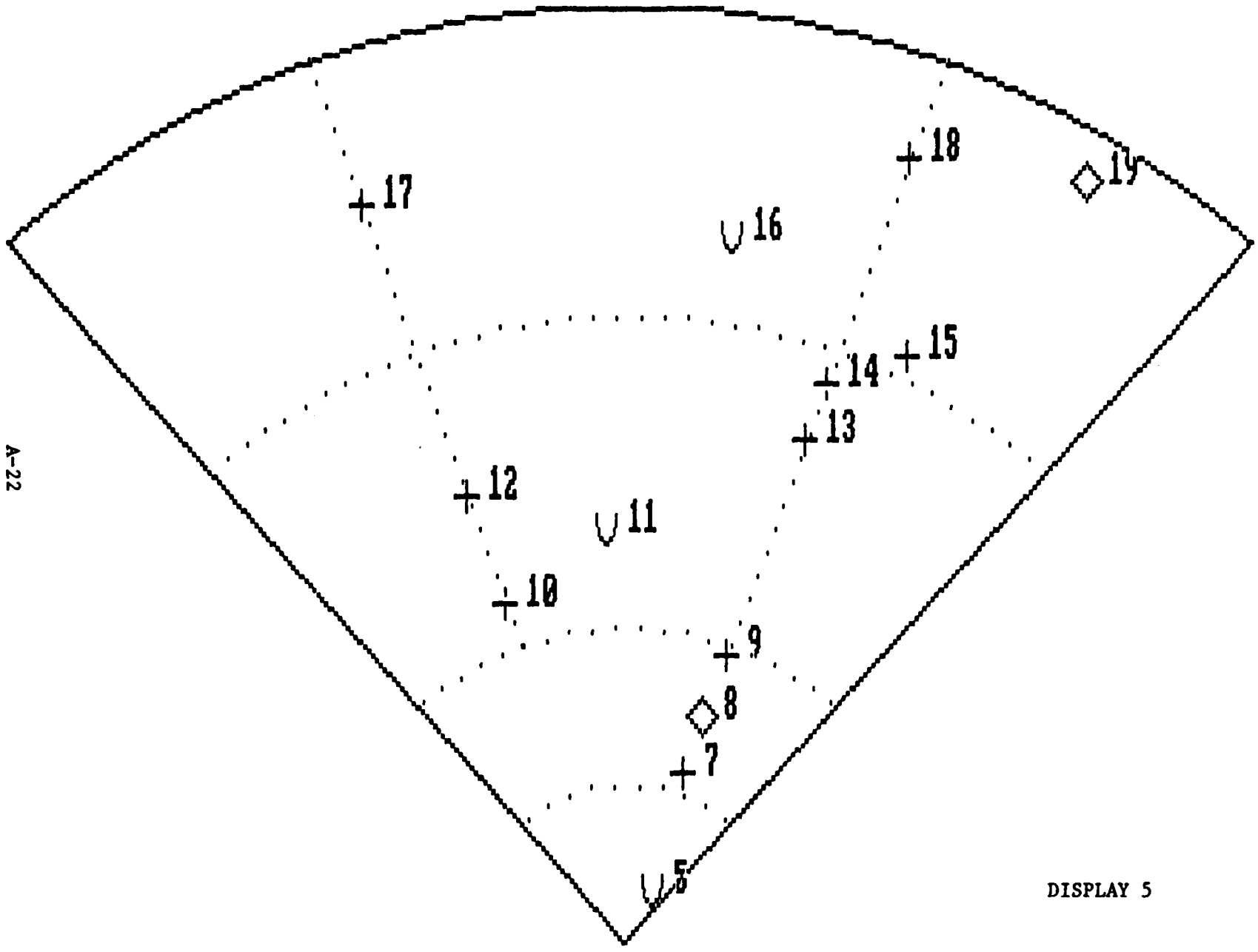
DISPLAY 2



DISPLAY 3

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
1	lo	f	uhf	?	shoot	right	3
2	lo	h	vhf	shoot		right	
3	hi	h	uhf	?	shoot	right	OUT OF
4	lo	f	vhf	?			3
5	lo	f	uhf	?			
6	hi	h	uhf	hold	shoot		
7	hi	f	vhf	hold			
8	lo	h	vhf	shoot			
9	hi	f	uhf	hold			
10	hi	f	vhf	hold			
11	lo	f	uhf	?			
12	hi	f	uhf	hold			
13	hi	h	uhf	hold			

DISPLAY 4



DISPLAY 5

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
17	hi	h	uhf	hold			6
18	hi	h	uhf	hold			
19	lo	h	vhf	shoot			OUT OF
4	lo	f	vhf	?		right	9
5	lo	f	uhf	?		wrong	
6	hi	h	uhf	hold	shoot	wrong	
7	hi	f	vhf	hold		right	
8	lo	h	vhf	shoot		wrong	
9	hi	f	uhf	hold		right	
10	hi	f	vhf	hold			
11	lo	f	uhf	?			
12	hi	f	uhf	hold			
13	hi	h	uhf	hold			
14	hi	f	vhf	hold			
15	hi	f	uhf	hold			
16	hi	f	vhf	?			

DISPLAY 6

3.0 INSTRUCTIONS TO SUBJECTS

Purpose of Experiment

The experiment you are about to take part in is one which may help us to understand how people handle various kinds of information in making decisions.

Your Basic Job

You will be the operator of a fictional air defense system and must decide which of the aircraft approaching you are friendly and which are enemies. You will see a "radar" picture of the territory you are responsible for covering, and--on a second display screen--you will find a variety of information that, in addition to the radar display, will help you make decisions. Your job is to make sure that as many enemy aircraft as possible are shot down and that as few friendly aircraft as possible are shot down. You will be paid according to how well you do this.

The first pair of displays in the display booklet includes an example on the left of what the radar screen looks like. You are located at the bottom, where the two straight lines come together. The whole pie-slice-shaped area is the area your radar can see. An enemy airbase is located out of range of your radar, off the top left of the screen; a friendly airbase is located out of range of your radar, off the top right of the screen. At the bottom where you are located, is a friendly town which you are trying to protect.

These aircraft may be either friends or enemies. The "+"-shaped figures on the screen represent aircraft the computer aid has decided are probably friends. The "◇"-shaped figures represent aircraft the aid has decided are probably hostile. The aid has made these decisions on the basis of how many of four cues available to it (corridor, altitude, iff, and frequency) agree on friend or hostile. Where it is unable to decide on the basis of these cues it makes a guess ("flips a coin"). All aircraft will appear at the top of the screen and move slowly toward you. All have the same speed.

You have an unlimited supply of missiles which you can use to shoot down aircraft. When an aircraft is selected for shooting before it reaches the second dotted circle from you (as explained later), a missile will automatically be fired at it when it reaches the second dotted circle, and it will be destroyed and vanish from the screen when it reaches the dotted circle nearest you.

When aircraft reach the second dotted circle, enemies always begin to attack the town, and friends, of course, do not. Although you will not "see" this with your radar, the computer will then know which aircraft are friends and which are enemies, and any mistakes you make (not shooting down an enemy, or shooting down a friend) will be pointed out to you by the computer. This will be done by having the aircraft symbol begin to flash on and off after it reaches the second dotted circle. A flashing symbol means you have made an error.

How to Tell Friendly from Hostile Aircraft

In today's experiment, there are a number of clues to help you tell a friend from an enemy. On the radar screen itself you can see the locations of the aircraft. From the radar screen you can see where the aircraft appear to be coming from. In addition, the two straight dotted lines are known as "safe

passage corridors." These are the paths friendly pilots have been told to use if possible when travelling through your area. Therefore, all other things being equal, friendly aircraft are more likely to be travelling down these dotted lines than are enemy aircraft.

At the same time you watch the radar screen you must also watch a second screen which will display additional information to help you make decisions. An example of the second screen, recorded at the same time as the radar screen you have been looking at, is shown to the right of the radar screen in the display booklet. Most of the display is a table showing several kinds of information for each aircraft.

On the left side are the numbers for the aircraft shown on the screen at this moment. The next three columns show information about the aircraft that will be useful to you in making your decisions. These are called the ALTITUDE ("ALT"), the "IFF," and the "FREQUENCY" of each aircraft. What these really are makes no difference here. Friends have been asked to fly through your zone at high altitude, while enemies--preparing for attacks on the town--will usually be flying lower. In the table, for example, aircraft number 1 is flying low ("lo"), while aircraft number 6 is flying high ("hi"). On the basis of this one cue only, #1 looks like an enemy and #6 looks like a friend.

"IFF" is a signal generated by the aircraft in response to your signal to it. The computer will show you whether this signal appears to be from a friendly ("f") or hostile ("h") aircraft. In the table, for example, aircraft number 1, based on the IFF cue only, looks like a hostile, while #2 looks friendly.

Finally, the "FRQ" column shows the frequency being used by radio transmissions from the different aircraft. Friendlies usually use uhf frequencies and hostiles usually use vhf frequencies.

To summarize, you have the geographic information provided by the radar screen plus four other cues:

	IN DOTTED CORRIDOR	ALT	IFF	FRQ
friend	yes	hi	f	uhf
hostile	no	lo	h	vhf

The above table shows the most likely observations you will make for a friend or enemy on several cues. NONE of these cues are completely reliable. Friends sometimes forget to follow the dotted lines and enemies sometimes happen to fly down them; sometimes friends fly too low and enemies too high, and so on. Nevertheless, if you use as many of the cues as you can, along with information provided by the radar screen, you can do a good, but not perfect, job of telling friends from enemies.

How to Shoot at Aircraft

Up until an aircraft reaches the second dotted circle from you, you can tell the system if you disagree with its classifications (as friend or hostile) by typing an aircraft number followed by a carriage return. On the radar screen shown

before, the operator (you) has typed the numbers 1, 4, and 5, each followed by a carriage return. The computer indicates this by placing a six-sided figure around the aircraft symbol. Friends will be reclassified as hostile, and hostiles will be reclassified as friends this way. Aircraft identified as hostile will be destroyed when they reach the lowest dotted circle. Other aircraft will not be shot at.

If you make a mistake or change your mind, simply type the number again and the screen symbol will change back. Incorrect numbers (numbers of aircraft not on the screen) will be ignored.

You can type aircraft numbers until just before they reach the second dotted circle.

The second screen will indicate in the OPERATOR column, your decision for each aircraft on the screen. In the previous chart, the operator has reversed the computer's identifications for aircraft 1, 4, and 5.

The column labeled SYSTEM shows the computer aid's recommended action--the action that will be taken unless you change it.

Results and Your Score

When an aircraft reaches the second dotted circle, as shown in the second pair of displays [3 & 4] in the display booklet, its true identity becomes known to the computer, and it will indicate on the second screen in the RESULT column if your decision was right or wrong. At the same time, any aircraft for which you have done the wrong thing will begin to flash on and off. For each aircraft you identify correctly, you receive one point, shown as your score in the upper right-hand corner of the second screen. Just below your score will be shown the total number of aircraft that have reached the second dotted circle so far.

Payment

You will be paid six cents for each aircraft you correctly identify before the experiment ends. No one will be paid less than four dollars per hour. A few aircraft at the very end of the experiment will not count.

One More Pair of Screens

As aircraft are shot down or reach the bottom of the radar screen, they vanish. On the second (tabular) screen, information about new targets appearing at the top is written over information about the oldest targets. In other words, the oldest lines on the second screen are replaced with information on the newest aircraft. In this way a particular aircraft remains in the table only long enough for you to check the result before it is replaced with a new line of information about a new aircraft. The third pair of displays [5, 6] shows how the screens might look after aircraft 1 through 4 have gone on by or been destroyed. Aircraft 9 is flashing, indicating an error. Aircraft 4 has been destroyed, so is missing from the radar screen. It is to be replaced in the next second or so on the second screen by information on aircraft number 20.

Final Instructions

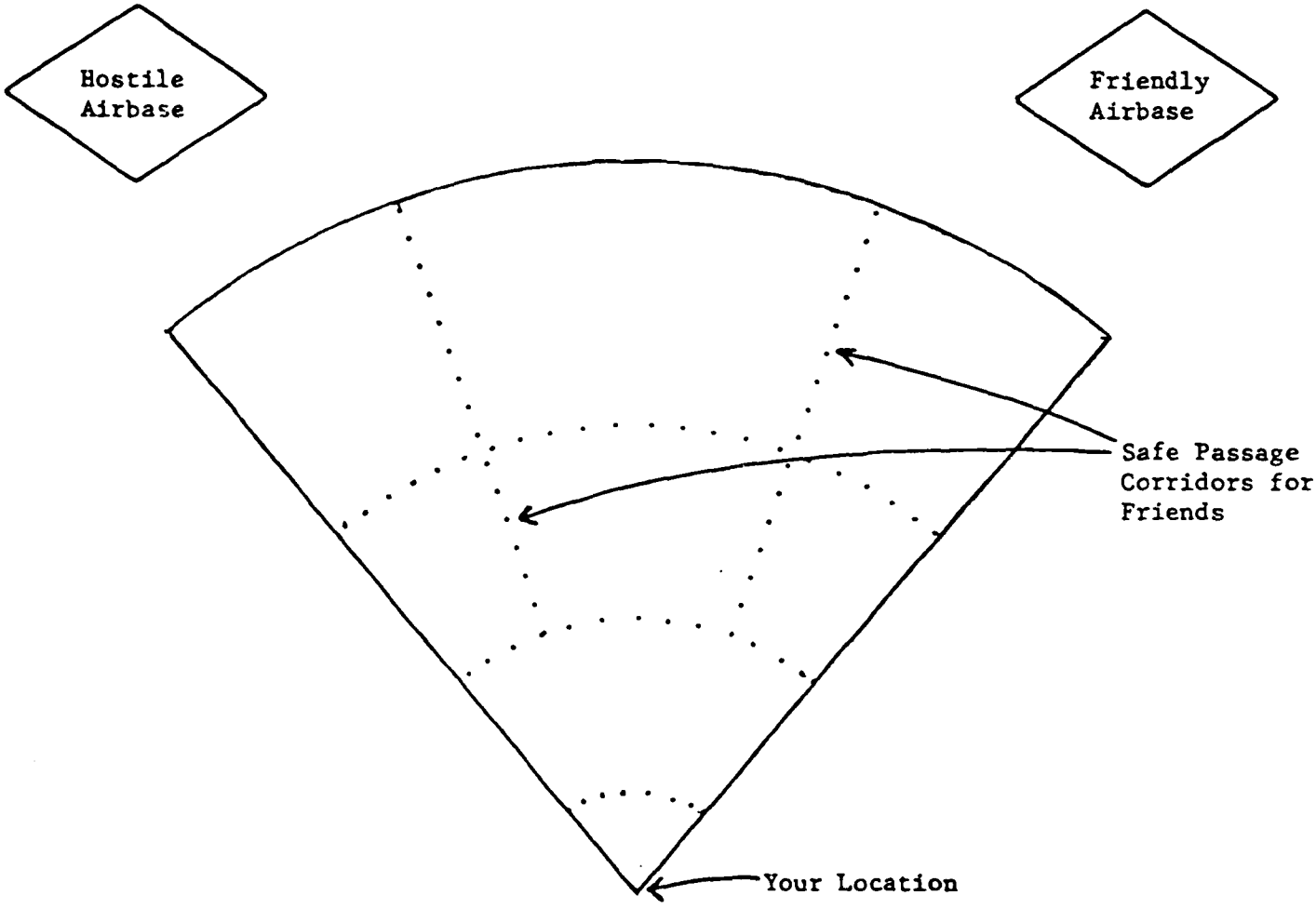
Please reread these instructions if anything is not clear and ask any questions you have before beginning. There will be a short practice session before we begin the actual experiment.

In order to do well you will need to pay attention to anything and everything that might help you make decisions. Remember the general setting: the enemy is off screen to the left, and the friendly base is off-screen to the right. Remember to notice who is in the "safe passage corridors." Remember to check the altitude, IFF, and frequency of each target. By using all possible information, you can do very well. A summary of information that can help you is given on the next page for use now and during the experiment.

Remember too, some tests and days will be easier or harder than others.

SUMMARY OF INFORMATION

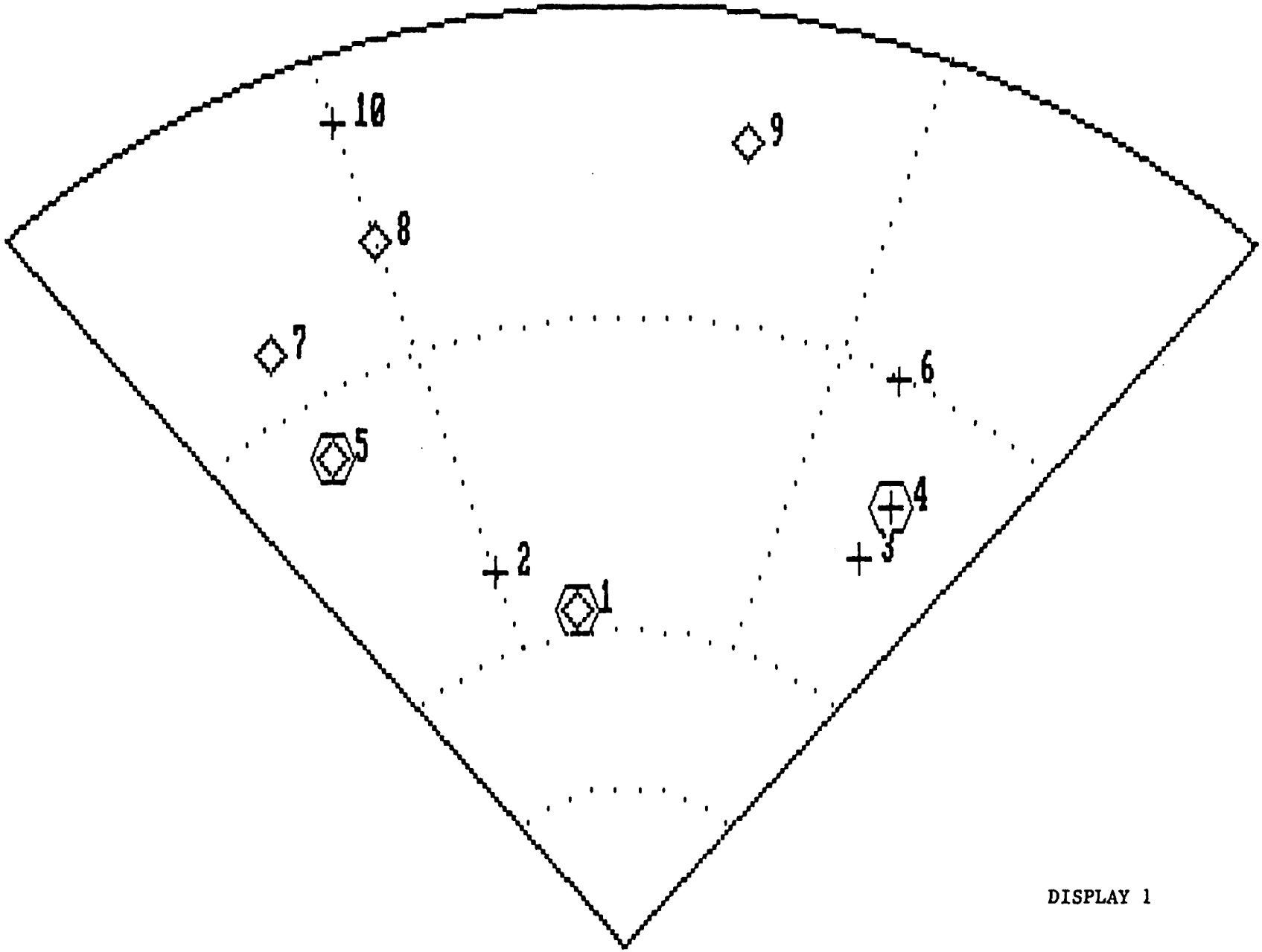
Radar Screen



Tabular Screen

Cues tend to be as follows:

	ALT	IFF	FRQ
Friend	hi	f	uhf
Hostile	lo	h	vhf



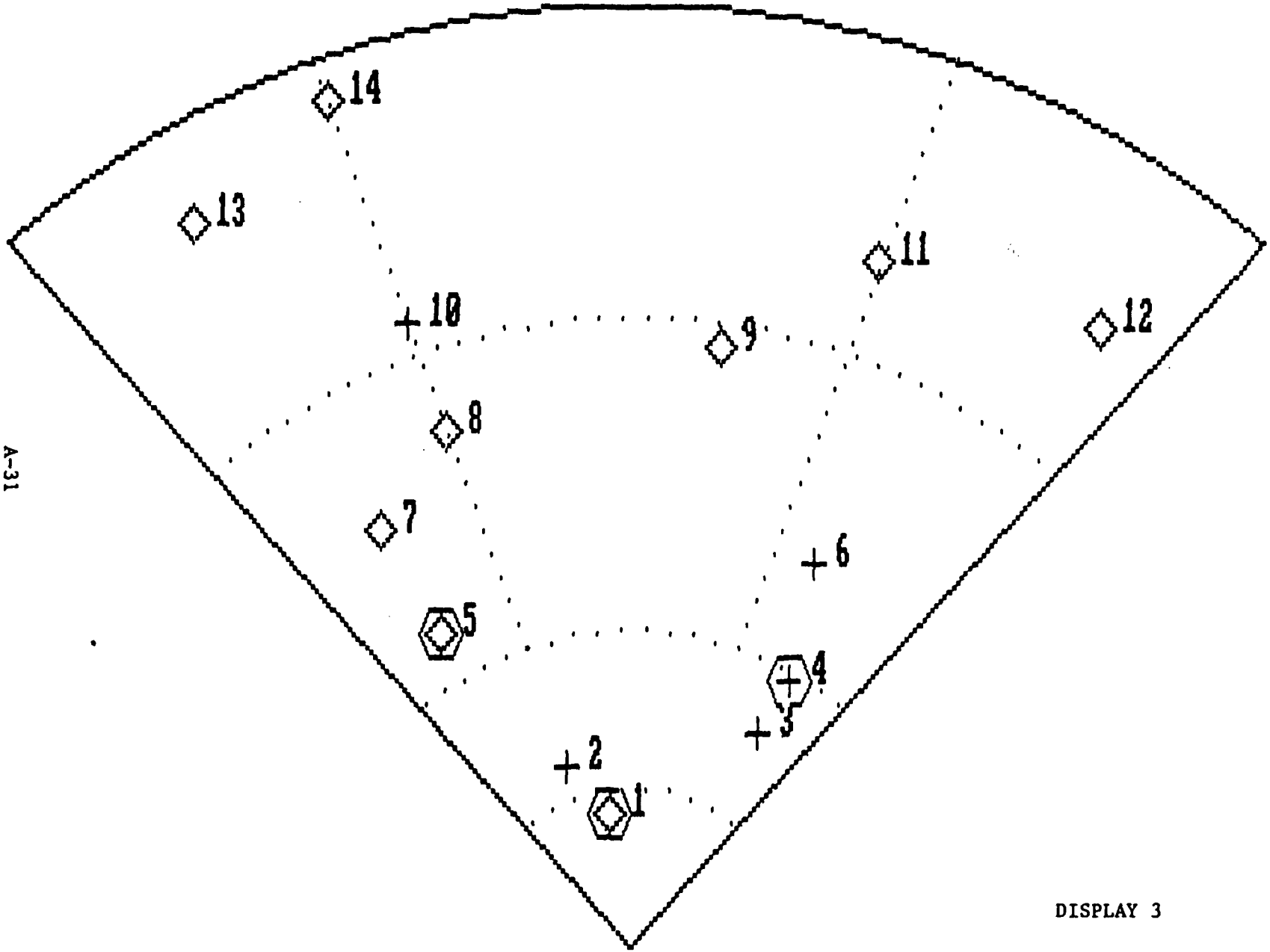
DISPLAY 1

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT
1	lo	h	vhf	shoot	hold	
2	lo	f	uhf	hold		
3	lo	f	uhf	hold		
4	lo	f	uhf	hold	shoot	
5	lo	f	vhf	shoot	hold	
6	hi	f	uhf	hold		
7	lo	h	uhf	shoot		
8	lo	h	vhf	shoot		
9	hi	h	uhf	shoot		
10	lo	f	uhf	hold		

SCORE
0

OUT OF
0

DISPLAY 2

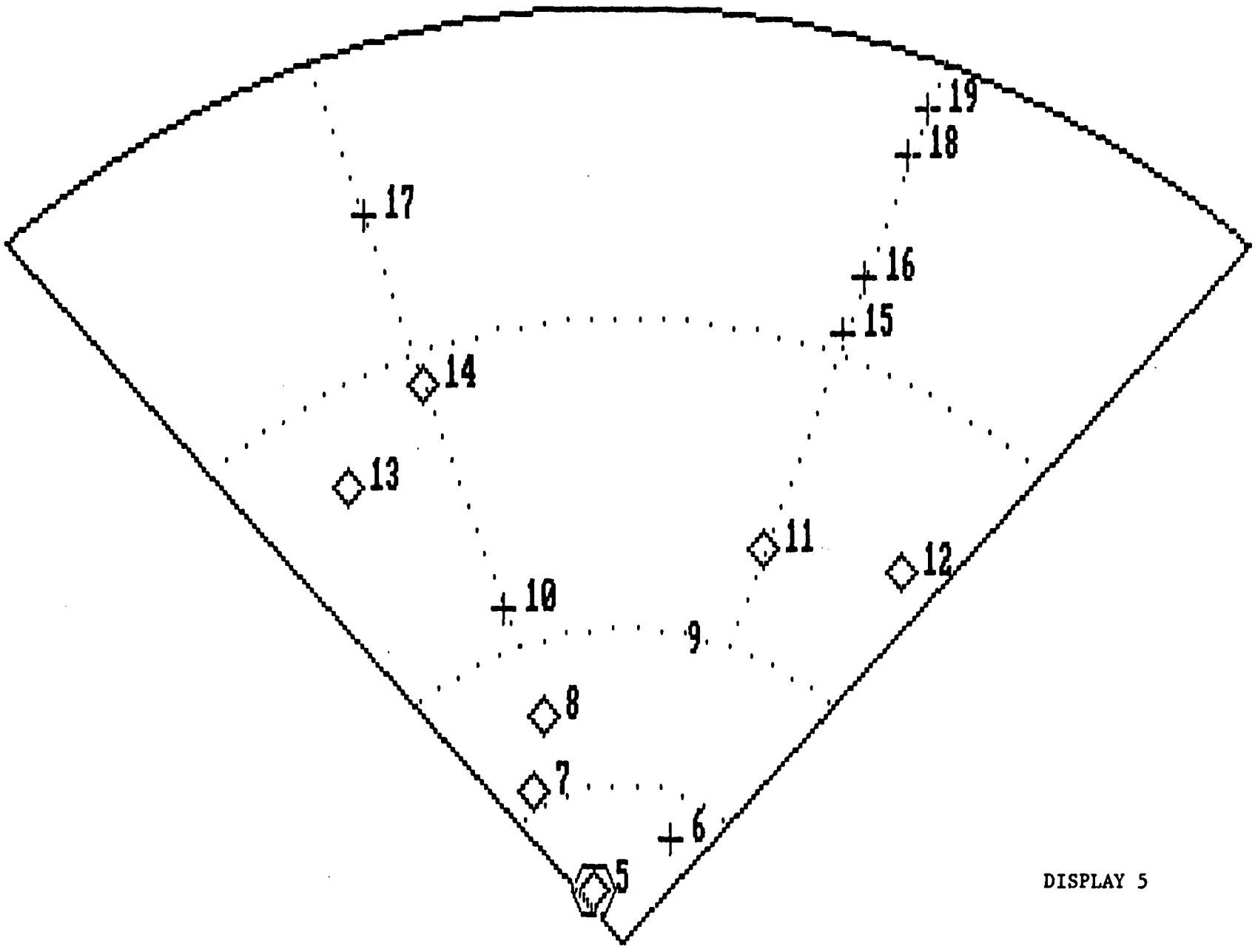


A-31

DISPLAY 3

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
1	lo	h	vhf	shoot	hold	wrong	1
2	lo	f	uhf	hold		wrong	
3	lo	f	uhf	hold		right	OUT OF
4	lo	f	uhf	hold	shoot	wrong	4
5	lo	f	vhf	shoot	hold		
6	hi	f	uhf	hold			
7	lo	h	uhf	shoot			
8	lo	h	vhf	shoot			
9	hi	h	uhf	shoot			
10	lo	f	uhf	hold			
11	lo	h	vhf	shoot			
12	lo	h	uhf	shoot			
13	lo	h	uhf	shoot			
14	lo	h	vhf	shoot			

DISPLAY 4



DISPLAY 5

NO.	ALT	IFF	FRQ	SYSTEM	OPERATOR	RESULT	SCORE
17	hi	f	vhf	hold			4
18	hi	h	uhf	hold			
19	hi	h	uhf	hold			OUT OF
4	lo	f	uhf	hold	shoot	wrong	9
5	lo	f	vhf	shoot	hold	wrong	
6	hi	f	uhf	hold		right	
7	lo	h	uhf	shoot		right	
8	lo	h	vhf	shoot		right	
9	hi	h	uhf	shoot		wrong	
10	lo	f	uhf	hold			
11	lo	h	vhf	shoot			
12	lo	h	uhf	shoot			
13	lo	h	uhf	shoot			
14	lo	h	vhf	shoot			
15	hi	f	uhf	hold			
16	hi	h	uhf	hold			

DISPLAY 6

APPENDIX B: SOFTWARE LISTING

Both the prototype research system software and the data analysis software are written in C for the Lattice C-compiler for the Intel 8086 and MS-DOS operating system (Version 2.0 or higher). They should compile, however, under most C-compilers with only minor changes, if any. The primitive graphics subroutine calls are to Intel 8086 assembly language routines developed by Media Cybernetics, Inc. and supplied under their "Halo" trademark.

The prototype experiment program listing follows.

```
/* ari */
#include "stdio.h"
#define NTGTS 25        /* # of simultaneously displayed tgts allowed */
#define NTRIALS 550     /* # of tgts generated */
#define MAXH 639        /* max horiz coordinate */
#define MAXV 199        /* max vert coordinate */
#define CENH 639/2      /* x-coord of center */
#define CENV 199/2      /* y-coord of center */
#define ASP (5.0/12.0)  /* ((MAXV+1)/(MAXH+1))*(240/180) aspect of ellipse */
#define T 1
#define F 0
int score = 0;          /* cumulative score = #right */
int outof = 0;          /* cumulative count of tgts having reached r=160 */
double preast, prcue;   /* probability east & other cues correct */
int expcond; /* # of experimental condition:1-no aid, 2-aid screens, 3-no user*/
int subject;            /* subject # */
int gaptime;            /* seconds between tgt appearances */
int nblocks;            /* # blocks of 100 trials in experiment */
double bspeed;          /* base speed factor */
double xspeed[14], yspeed[14];  /* component speeds for track 0, 1, ... */
int xapp[14], yapp[14]; /* coords of appearance for ea track */
double hdg[14];         /* heading in radians for ea track */
int ntgts;              /* max # of tgts on screen at once */
int scentime = 0;       /* seconds into scenario */
int nseq = 0;           /* next target to be added to screen */
int nerased = 0;        /* #tgts erased from screen so far */
struct stype {          /* scenario and experimental data structure */
        int trueid, id, corr, iff, alt, cue4, east, track, engage, tengage;
} type[NTRIALS];
struct stgt {
        int num;        /* number used as tag */
        int *tag;       /* pointer to image of 2-digit target tag */
        int id;         /* id: friend (0), unknown, hostile */
        int *image;     /* pointer to image array */
        int time;       /* time last displayed in sec/100 */
        int x, y;       /* last coordinates */
        double xd, yd;  /* last calculated coordinates */
        double xspeed, yspeed;
        int trueid, corr, iff, alt, cue4, east;
        int inzone;     /* flag true if in bottom ring */
        int erased;     /* flag true if within 80 of bottom */
        int show;       /* flag: tgt still onscreen */
        int flash;      /* flash tgt */
        int vis;        /* tgt is visible */
        int engage;     /* flag: tgt is engaged */
        int tengage;    /* time subject entered tgt # in scentime (secs) */
        int track;      /* track # (0-13) */
} tgt[NTGTS];
struct stag {
        int tag[12];
} bigtag[100];
int imageh[22], imageu[22], imagef[22], imagee[22], imagei[22];
```

```
main(argc, argv)
int argc;
char *argv[];
{
        int y, err, zero = 0, one = 1, r, wide = 8, high = 12;
        float asp = ASP;
        static char fname[] = "b:s00c0t00.ari";
        FILE *fp, *fopen();

        if (argc != 7) {
           puts("\n Error: usage requires 6 arguments.");
           exit(1);
        }
        for (err = T; err == T;) {
                puts("\nEnter the subject number.");
                if (err = 1 != scanf("%d", &subject))
                        puts("\nUnrecognizable subject number...");
                else
                        if (err = subject < 0 || subject > 99)
                                puts("\nNumber must be between 0 & 99.");
        }
        stcd_i(argv[1], &expcond);
        stcd_i(argv[2], &r);
        bspeed = ((double) r) / 10000.0;
        stcd_i(argv[3], &gaptime);
        stcd_i(argv[4], &r);
        prcue = ((double) r) / 100.0;
        stcd_i(argv[5], &r);
        preast = ((double) r) / 100.0;
        stcd_i(argv[6], &nblocks);
        if (nblocks) {
                sprintf(fname + 3,"%2d",subject);
                if (subject < 10)
                        fname[3] = '0';
                fname[5] = 'c';
                fname[6] = *argv[1];
                sprintf(fname + 8, "%2d", gaptime);
                if (gaptime < 10)
                        fname[8] = '0';
                fname[10] = '.';
                if ((fp = fopen(fname, "r")) != NULL) {
                        printf("Data file %s already exists.\n", fname);
                        exit(1);
                }
                if ((fp = fopen(fname, "w")) == NULL) {
                        printf("Data file %s cannot be created.\n", fname);
                        exit(1);
                }
        }
        initgraphics(&one);
        setasp(&asp);
        setcolor(&one);
        settext(&one,&one,&zero,&zero);
        settextcir(&one,&zero);
        inittcur(&one,&one,&zero);
        srand48(subject+100*gaptime+10000*expcond);/* initialize rn generator */
```

```
        simulate();                         /* store simulated tgt sequence */
        gentgts();                          /* display & store targets */
        dstat();                            /* display static image */
        ntgts = 1+(480 - 10)/(bspeed*gaptime*100); /* #tgts on screen at once */
        if (ntgts > NTGTS) {
                printf("MAIN: TOO MANY TARGETS");
                exit(1);
        }
        run();
        initgraphics(&zero);
        inittcur(&one, &one, &zero);
        settext(&high, &wide, &zero, &zero);
        settextclr(&one, &zero);
        movtcurabs(&zero, &(y=99));
        text("Thank");
        movtcurabs(&(wide=32), &(r=199));
        text("you!");
        if (nblocks)
                savedata(argc, argv, fp);
        while (getch() != '=')
                ;
        closegraphics();
}
```

```
simulate()                          /* init type structure */
{
        int block,k,start,end,i,t,sum, need[2] [5] [2], count[2] [5] [2];
        double p, a, sin(), cos(), drand48();
        int *pc, *pn, class;

        for (sum = i = 0; i < 2; i++)
          for (k = 0; k < 2; k++) {
            sum += (need[i] [0] [k] = .5+100*.5*((i==k)?preast:(1-preast))*
                prcue*prcue*prcue*prcue);
            sum += (need[i] [1] [k] = .5+100*.5*((i==k)?preast:(1-preast))*
                4*prcue*prcue*prcue*(1-prcue));
            sum += (need[i] [2] [k] = .5+100*.5*((i==k)?preast:(1-preast))*
                6*prcue*prcue*(1-prcue)*(1-prcue));
            sum += (need[i] [3] [k] = .5+100*.5*((i==k)?preast:(1-preast))*
                4*prcue*(1-prcue)*(1-prcue)*(1-prcue));
            sum += (need[i] [4] [k] = .5+100*.5*((i==k)?preast:(1-preast))*
                (1-prcue)*(1-prcue)*(1-prcue)*(1-prcue));
          }
        if (sum != 100) {
                printf("Error: input probabilities do not round to integers.");
                exit(1);
        }
        for (block = 0; block <= nblocks+1; block++) {
                start = block ? 25 + (block-1) * 100 : 0;
                end = block ? start + 99 : 24;
                end = (block == nblocks+1) ? start + 24 : end;
                if (end >= NTRIALS) {
                        printf("Too many trials requested.");
                        exit(1);
                }
                for (pc = &count[0][0][0], pn = &need[0][0][0], i = 0; i < 20;
                        pc++, pn++, i++)
                        *pc = *pn;
                for (i = start; i <= end;) {
                        type[i].trueid = drand48() < 0.5;
                        p = (type[i].trueid) ? prcue : 1.0 - prcue;
                        type[i].corr = drand48() < p;
                        type[i].iff  = drand48() < p;
                        type[i].alt  = drand48() < p;
                        type[i].cue4 = drand48() < p;
                        p = (type[i].trueid) ? preast : 1.0 - preast;
                        type[i].east = drand48() < p;
                        class = (type[i].corr != type[i].trueid) +
                                (type[i].iff  != type[i].trueid) +
                                (type[i].alt  != type[i].trueid) +
                                (type[i].cue4 != type[i].trueid);
                        if (count[type[i].trueid] [class] [type[i].east]) {
                                --count[type[i].trueid] [class] [type[i].east];
                                if (expcond == 1)
                                        type[i].id = 1;
                                else {
                                        if (class < 2)
                                                type[i].id = 2 * type[i].trueid;
                                        else if (class > 2)
                                                type[i].id = 2 * !type[i].trueid;
```

```
                                        else
                                                type[i].id = 1;
                                        if (expcond == 3 && class == 2)
                                                type[i].id = 2 * (drand48()<0.5);
                                }
                                t = drand48() * 6.0;
                                t += t > 2;
                                type[i].track = type[i].corr ? (type[i].east ?
                                        t+7 : t) : (type[i].east ? 10 : 3);
                                if (type[i].track > 13 || type[i].track < 0) {
                                  printf("Error in simulate: %d",t);
                                  exit(1);
                                }
                                i++;
                        }
                }
        }
        for (i=0, a=0.920796326; i < sizeof(hdg)/sizeof(hdg[0]); i++, a+=0.1) {
                xapp[i] = 319 + 460 * cos(a);
                yapp[i] = 199 - 460 * ASP * sin(a);
                hdg[i] = a + 3.14159265;
                xspeed[i] = bspeed * cos(hdg[i]);
                yspeed[i] = bspeed * sin(hdg[i]-3.14159265) * ASP;
        }
}
```

```
gentgts()                                          /* display & store tgts */
{
        int i, x, y, mode = 3;
        int homex = CENH + 1, homey = CENV + 1, p0 = 0, m1 = -1, p1 = 1;
        int tgtx = homex + 23, tgty = homey + 11;
        char string[5];

        movabs( &(x=7+homex), &(y=2+homey) );           /* unknown */
        d(), d(), d(), rd(), d(), rd(), r(), rd(), r(), r();
        ru(), r(), ru(), u(), ru(), u(), u();
        movefrom( &homex, &homey, &tgtx, &tgty, imageu );
        moveto( &homex, &homey, imageu, &mode );
        movabs( &(x=12+homex), &(y=homey+2) );          /* hostile */
        rd(), r(), rd(), r(), rd(), r(), rd(), r();
        ld(), ld(), ld(), ld(), ld(), ld(), ld(), ld();
        lu(), l(), lu(), l(), lu(), l(), lu(), l();
        r(), ru(), r(), ru(), r(), ru(), r(), ru();
        movefrom( &homex, &homey, &tgtx, &tgty, imageh );
        moveto( &homex, &homey, imageh, &mode );
        movabs( &(x=12+homex), &(y=3+homey) );          /* friendly */
        lnrel( &p0, &(y=6) );
        movabs( &(x=6+homex), &(y=6+homey) );
        lnrel( &(x=12), &p0 );
        movefrom( &homex, &homey, &tgtx, &tgty, imagef );
        moveto( &homex, &homey, imagef, &mode );
        movabs( &(x=6+homex), &(y=1+homey) );           /* engaged modifier */
        lnrel( &(x=12), &p0 );
        lnrel( &(x=5), &x );
        lnrel( &(x=-5), &(y=5) );
        lnrel( &(x=-12), &p0 );
        lnrel( &y, &(y=-5) );
        lnrel( &(x=5), &y );
        movefrom( &homex, &homey, &tgtx, &tgty, imagee );
        moveto( &homex, &homey, imagee, &mode );        /* erase */
        movefrom( &homex, &homey, &tgtx, &tgty, imagei );/* invisible */
        movtcurabs( &homex, &(y=homey+7));      /* generate numeric tags */
        x = homex + 15, y = homey + 7;
        movefrom(&homex,&homey,&x,&y,bigtag[0].tag);    /* invisible tag */
        for (i=1; i < 100; i++) {
                stci_d(string, i, 4);
                movtcurabs( &homex, &y);
                text(string);
                x = homex + 15, y = homey + 7;
                movefrom(&homex,&homey,&x,&y,bigtag[i].tag);
                moveto( &homex, &homey, bigtag[i].tag, &mode);
        }
        deltcur();
}
```

```
dstat()
{
        int color = 1, cenx = CENH, ceny = CENV, zero = 0, maxx=639, maxy=199;
        int r, x, y;
        float ang1, ang2;

        spass();                        /* display safe passage corridor */
        movabs(&zero, &(y=50));         /* display screen border */
        lnabs(&cenx, &maxy);
        lnabs(&maxx, &y);
        movabs(&cenx, &maxy);           /* display range rings */
        ang1 = 0.8407;
        ang2 = 2.3005;
        r = 80;
        arcdotted((double) r, ang1, ang2);
        r = 160;
        arcdotted((double) r, ang1, ang2);
        r = 320;
        arcdotted((double) r, ang1, ang2);
        r = 480;
        movabs(&cenx,&maxy);
        arc(&r, &ang1, &ang2);
}
arcdotted(r, ang1, ang2)        /* display a dotted arc between ang1 & ang2 */
double r, ang1, ang2;
{
        int x, y;
        double ang, a, sin(), c, cos(), atan();

        ang = atan((5.0/ASP)/r);/* ang which changes y by 5 dots at horizont */
        for (a=1.570796+ang/2.0; a<ang2; a +=ang) {
                ptabs( &(x=0.5+cos(a)*r+319.0), &(y=0.5+MAXV-sin(a)*r*ASP) );
                ptabs( &(x=639-x), &y);
        }
}
```

```
spass()                                 /* display safe passage corridors */
{
        int x1l, x1r, y1r, x2l, x2r, y2r, mode = 3;
        double a, sin(), cos();

        setlnstyle(&mode);      /* dashed line mode */
        a = 1.220796;           /* angle defining center of right corridor */
        x1l = 638 - (x1r = cos(a) * 160 + 319);
        y1r = 199 - sin(a) * 160 * ASP;
        movabs(&x1r,&y1r);
        x2l = 638 - (x2r = 319 + cos(a) * 480);
        y2r = 199 - sin(a) * 480 * ASP;
        lnabs(&x2r,&y2r);       /* draw right-hand corridor */
        movabs(&x1l,&y1r);
        lnabs(&x2l,&y2r);       /* draw left-hand corridor */
        setlnstyle(&(mode=1));
}
```

```
run()                                   /* run a scenario */
{
        int i, tnew;
        int t;          /* most recent clock time in secs */

        for (i=0; i<ntgts; i++)
                tgt[i].show=F;
        clearsc();              /* clear text screen */
        puts("NO.| ALT  IFF  FRQ | SYSTEM OPERATOR | RESULT");
        pcvscp(0,75);
        puts("SCORE");
        pcvscp(1,79);
        puts("0");
        pcvscp(3,74);
        puts("OUT OF");
        pcvscp(4,79);
        puts("0");
        nseq = 0;               /* global counter */
        t = getsecs();
        while (nseq<50 + 100 * nblocks) {
                if (nseq * gaptime <= scentime)
                        addtgt();
                if (pckchk())
                        chkengage();
                scentime += ((tnew = getsecs()) < t) ? tnew + 60 - t : tnew - t;
                t = tnew;
                for (i=0; i<ntgts; i++) {
                        if (tgt[i].show) {
                                showtgt(i);
                                chkzone(i);
                        }
                }
        }
}
/* */
clearsc()               /* clear text screen */
{
        pcvscp(0, 0);
        pcvwc(80*25, ' ');
        pcvscp(0, 0);
}
```

```
addtgt()                                /* update tgt structure */
{
        int i, t, a, b;
        static int mode = 3;

        i = nseq % ntgts;
        tgt[i].tag = bigtag[ tgt[i].num = 1 + (nseq % 99) ].tag;
        tgt[i].trueid = type[nseq].trueid;
        tgt[i].corr = type[nseq].corr;
        tgt[i].iff = type[nseq].iff;
        tgt[i].alt = type[nseq].alt;
        tgt[i].cue4 = type[nseq].cue4;
        tgt[i].east = type[nseq].east;
        tgt[i].id = type[nseq].id;
        if (tgt[i].id)
                tgt[i].image = (tgt[i].id == 1) ? imageu : imageh;
        else
                tgt[i].image = imagef;
        t = tgt[i].track = type[nseq].track;
        tgt[i].xspeed = xspeed[t];
        tgt[i].yspeed = yspeed[t];
        tgt[i].xd = tgt[i].x = xapp[t];
        tgt[i].yd = tgt[i].y = yapp[t];
        tgt[i].show = tgt[i].vis = T;
        tgt[i].time = gettime();
        tgt[i].flash = tgt[i].inzone = tgt[i].erased = tgt[i].engage = F;
        tgt[i].tengage = -1;
        a = xapp[t] - 12;
        b = yapp[t] - 6;
        moveto(&a, &b, tgt[i].image, &mode);
        moveto(&(a += 24), &b, tgt[i].tag, &mode);
        addtext(i);                     /* add tgt to text screen */
        nseq++;
}
```

```
addtext(t)                              /* add tgt[i] to text screen */
int t;
{
        static char col = 0;
        static char ctl[] = "%2d | %s    %s   %s | %s          |       ";
        char row = t + 1;
        static struct slbl {
                char *alt;
                char *iff;
                char *cue4;
                char *id;
        } lbl[] ={
                { "hi", "f", "uhf", " hold" },
                { "lo", "h", "vhf", "  ?  " },
                { "  ", " ", "   ", "shoot" }
        };

        pcvscp(row, col);
        printf(ctl, tgt[t].num, lbl[tgt[t].alt].alt, lbl[tgt[t].iff].iff
                ,lbl[tgt[t].cue4].cue4, lbl[tgt[t].id].id);
}
/* */
double distance(i) /* return double precision dist of tgt #i from subject */
int i;
{
        double sqrt(), x, y;

        x = 319-tgt[i].x;
        y = ((double) 199-tgt[i].y)/ASP;
        return(sqrt (x*x + y*y) );
}
```

```
chkzone(i)              /* check whether tgt has crossed bottom ring */
int i;                  /* tgt # */
{
        double distance(), d;
        int a, b, s;
        static int mode = 3;

        if ( (d = distance(i)) <= 160 ) {
                if (!tgt[i].inzone) {
                        tgt[i].inzone = T;
                        type[nerased].engage = tgt[i].engage; /* save data */
                        type[nerased++].tengage = tgt[i].tengage;
                        if (tgt[i].trueid)
                                s = tgt[i].engage != (tgt[i].id == 2);
                        else
                                s = tgt[i].engage == (tgt[i].id == 2);
                        pcvscp(i + 1, 40);
                        puts(s ? "right" : "wrong");
                        score += s;
                        pcvscp(1, 77);
                        printf("%3d", score);
                        pcvscp(4, 77);
                        printf("%3d",++outof);
                        tgt[i].flash = !s;
                }
                else if (d <= 80 && !tgt[i].erased) {
                        tgt[i].erased = T;      /* in innermost zone */
                        if ((tgt[i].id == 2) != tgt[i].engage) {
                                a = tgt[i].x - 12;
                                b = tgt[i].y - 6;
                                if (tgt[i].vis)
                                        moveto(&a, &b, tgt[i].image, &mode);
                                if (tgt[i].engage)
                                        moveto(&a, &b, imagee, &mode);
                                moveto(&(a+=24), &b, tgt[i].tag, &mode);
                                tgt[i].show = F;
                        }
                }
                else if (d <= 15) {
                        tgt[i].show = F;
                        a = tgt[i].x - 12;
                        b = tgt[i].y - 6;
                        if (tgt[i].vis)
                                moveto(&a, &b, tgt[i].image, &mode);
                        if (tgt[i].engage)
                                moveto(&a, &b, imagee, &mode);
                        moveto(&(a+=24), &b, tgt[i].tag, &mode);
                }
        }
}
```

```
showtgt(i)
int i;          /* tgt # */
{
        int mode;
        int x, y, t, a, b, c;
        double interval;

        mode = 3;
        t = gettime();
        interval = t < tgt[i].time ? t+6000-tgt[i].time : t-tgt[i].time;
        tgt[i].time = t;
        c = 24 + (a = -12 + tgt[i].x);
        b = -6 + tgt[i].y;
        x = tgt[i].xd += tgt[i].xspeed * interval;
        y = tgt[i].yd += tgt[i].yspeed * interval;
        if (tgt[i].vis)
                moveto(&a, &b, tgt[i].image, &mode); /* erase vis tgt */
        moveto(&c, &b, tgt[i].tag, &mode);
        if (tgt[i].engage)
                moveto(&a, &b, imagee, &mode);
        c = 24 + (a = -12 + (tgt[i].x = x));
        b = -6 + (tgt[i].y = y);
        if (!tgt[i].flash || (tgt[i].vis = !tgt[i].vis))
                moveto(&a, &b, tgt[i].image, &mode);
        moveto(&c, &b, tgt[i].tag,   &mode);
        if (tgt[i].engage)
                moveto(&a, &b, imagee, &mode);
}
/* */
chkengage()                     /* check for keyboard input of tgt number */
{
        static int mode = 3;
        int t, i, a, b;

        while ((t = gettgt()) != -1) {
                i = 0;
                while(tgt[i].num!=t && i<ntgts)
                        i++;
                if  (i < ntgts)
                        if (!tgt[i].inzone && tgt[i].show) {
                                tgt[i].engage = !tgt[i].engage;
                                tgt[i].tengage = scentime;
                                a = tgt[i].x - 12;
                                b = tgt[i].y - 6;
                                moveto(&a, &b, imagee, &mode);
                                pcvscp(i + 1, 30);
                                puts(((tgt[i].id==2) == tgt[i].engage) ?
                                        " hold" : "shoot");
                    }
        }
}
```

```
gettime()                       /* return the time in secs/100 */
{                               /* recycles every minute! */
        int g, *pg;
        double gtime(), gg;

        gg = gtime();
        pg = (int *) &gg;
        g = *pg + 3;
        return(100 * (g >> 8) + (g & 0177));
}
getsecs()                       /* return the time in seconds */
{
        int g, *pg;
        double gtime(), gg;

        gg = gtime();
        pg = (int *) &gg;
        g = *pg + 3;
        return(g >> 8);
}
gettgt()                        /* checks for keypress & returns numeric */
{                               /* value of last digit sequence when CR  */
        static int tgtnum, old;
        int ascii;

        if (pckchk()) { /* from smorgasbord--forces level 0 io */
                ascii = pckrc() & 0377;
/*              if (ascii == '*') {
                setgprint(&(ascii = 1));
                gprint();
                getch();        /* pause so mono screen can be dumped */
                }
*/
                if (ascii != 13) {
                        if (ascii < '0' || ascii > '9')
                                tgtnum = 0;
                        else
                                tgtnum = tgtnum * 10 + ascii - '0';
                        return(-1);
                }
                old = tgtnum;
                tgtnum = 0;
                return(old);
        }
        return(-1);
}
```

```
savedata(argc, argv, fp)        /* write experimental data to .ari file */
int argc;
char *argv[];
FILE *fp;
{
        int i;

        fprintf(fp, "%2d", subject);
        for (i = 1; i < argc; i++)
                fprintf(fp, " %s", argv[i]);
        fprintf(fp, " %3d %3d\n", score, outof);
        for (i = 25; i < nblocks*100 + 25; i++)
                fprintf(fp, "%3d %d %d %d %d %d %d %d %2d %d %3d\n", i-24,
                 type[i].trueid, type[i].id, type[i].corr, type[i].iff,
                 type[i].alt, type[i].cue4, type[i].east, type[i].track,
                 type[i].engage, type[i].engage? type[i].tengage-i*gaptime : 0);
        fclose(fp);
}
```