IN-04
198101
184 P

NASA Contractor Report 191559

TECHNIQUES USED FOR THE ANALYSIS OF OCULOMETER EYE-SCANNING DATA OBTAINED FROM AN AIR TRAFFIC CONTROL DISPLAY

Daniel J. Crawford
Daniel W. Burdette
William R. Capron

LOCKHEED ENGINEERING & SCIENCES COMPANY
Hampton, VA

Contract NAS1-19000
December 1993

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

TECHNIQUES USED FOR THE ANALYSIS OF OCULOMETER EYE-SCANNING DATA OBTAINED FROM AN AIR TRAFFIC CONTROL DISPLAY

Table of Contents

Symbols, Abbreviations, and Definitions ii
Summary 1
1.0 Introduction 1
2.0 System Overview 2
2.1 Equipment, Environment, and Procedures 7
2.1.1 Measuring eye scanning behavior in glass display (CRT) environments 7
2.1.2 PPI Alignment Template 8
2.1.3 Video Alignment 10
2.1.4 Subject eye calibration 10
2.1.5 Visual Area of Interest 11
2.2 Quick Look Capability 13
2.3 Recording, Synchronizing, and Filtering the Data 16
2.4 Target Identification 19
2.5 Cross Check Scanning 20
2.6 Display Zones 29
3.0 Description of Statistics 29
4.0 Major Results and Concluding Remarks 44
References 45
Appendix A Data File Record Descriptors
Appendix B Program Block Diagrams
Appendix C Data Reduction and Analysis Source Code

Symbols, Abbreviations, and Definitions

ATC	air traffic control
ANOVA	analysis of variance
CCS	cross check scans
CSM	centerline slot marker display format
DICE	direct course error (time) countdown display format
FAA	Federal Aviation Administration
FAF	final approach fix
FASA	final approach spacing aid
FPL	full performance level
GM	graphic marker display format
kts	knots
MAN	manual/ARTS III display format (no FASA)
MOTAS	Mission Oriented Terminal Area Simulation
NASA	National Aeronautics and Space Administration
nmi	nautical mile(s)
P	level of significance for treatment effect in ANOVA
PLSD	protected least significant difference
PPI	ATC station plan position indicator
TIMER	traffic intelligence for the management of efficient runway scheduling
TRACON	terminal radar approach control
VOR	VHF omnidirectional radio range
170	170 knot pattern speed procedure
210	210 knot pattern speed procedure

Summary

A dynamic real-time simulation study was conducted at NASA-Langley to gather comparative performance data among three candidate final-approach spacing aid (FASA) display formats. That study, formally documented in references 1 and 2, included an analysis of subject-controller eye scan data recorded from an oculometer system. The FASA display study was different from earlier applications of the oculometer system because the gaze objects (e.g., aircraft) were moving. Most past NASA oculometer applications involved a fixed-position display such as cockpit instruments. In the FASA study, individual objects on the screen were associated with a lookpoint. This paper describes some of the methodology used in the eye scan portion of the FASA study. Synchronization of oculometer data with simulation data is discussed as is data filtering. Algorithms for identifying lookpoint targets and for identifying cross check scans are described. Flow charts, block diagrams, file record descriptors, and extensive source code are included. Three general categories of statistics are examined: total time spent looking at screen objects, average length of lookpoint fixations for a given controller and display format, and cross check scans. The latter is a back and forth eye scanning sequence between two display objects. Three sets of tables are provided for each of the three categories of statistics. In addition to overall numbers, breakdowns by type of object and screen zone are also reported. The tables indicate the lookpoint measures defined for the FASA study and are presented for possible oculometer use in future ATC display studies or other displays employing an oculometer. Some of the methodology reported here may be applied to other studies with individual moving elements.

1.0 Introduction

The oculometer facility at the NASA Langley Research Center was recently used in support of an air traffic control final approach spacing aid (FASA) display evaluation study. The study, documented in references 1 and 2, compared the relative merit of three proposed automation-aid supplements to the final approach radar display in addition to the baseline manual ARTS III format (MAN). The FASA display formats evaluated were extended-runway-centerline slot markers (CSM), direct course time error countdown (DICE), and graphic marker (GM). Several methods such as analysis of aircraft separation and delivery precision, controller response time to automation suggested vectors, and controller workload were used to assess the effectiveness of the proposed display changes. An additional evaluation technique used was an analysis of the eye scanning behavior of the subject-controllers. The results of the eye scanning behavior analysis were consistent with results from the other methods. This paper is concerned with documenting the techniques used in the FASA study to do eye scanning behavior analysis and with the programs that were used to reduce and analyze the data.

There are several reasons for documenting the work. First, it was unusual in oculometer applications for the gaze object (e.g., an aircraft on the display) to be moving. Normally the display is fixed for a test involving an oculometer. Thus, software had to be written to accommodate moving objects. References 1 and 2, because of space limitations and lack of general interest, does not include the details of the work. Readers interested in eye scanning analysis may wish, however, to see a more in-depth discussion. Second, an experiment using complex simulation equipment and full performance level FAA controllers will be difficult and expensive to reproduce, but it was possible that the FASA eye scan data would be examined again someday for related or unrelated studies. If so, this document could be very helpful. Third, there was latent interest in using the oculometer in planned air traffic control studies. Since new oculometers were being installed at both NASA Langley and the FAA Technical Center, it was expected that some of the lookpoint measures and methodology described in this paper will be carried over to future studies using the new equipment.

The programs written for the FASA study were in compiler basic (Microsoft Quick Basic) and were executed on a desktop computer. The file naming convention used throughout this paper was as follows: The data file name consisted of an eight character run identifier followed by a period and a three character file type designator. For example, the file DB16DC21.DAT was a .DAT (data) file, run number 16 for subject DB. The run was a DICE display format, 210-knot approach-pattern-speed test. The .DAT file is written by the oculometer data collection computer during the test and contains the lookpoint coordinates and other data. Throughout this paper files are referred to by their suffixes such as .SCN, .DAT, .ACP, .MRG, and .CCS. The data record descriptors for those files are discussed and illustrated in appendix A. The description, when appropriate, includes references to the source code that was used to read or write the file. The first file discussed in appendix A is the .CCS file, i.e., cross check scan. Usually, there are 96 individual files of any type. That was so because there were 12 controller subjects, each testing four display formats at two different pattern speeds. Understanding this convention is important to understanding this paper. The deceptively simple block diagrams presented in appendix B rely heavily on this convention. Those block diagrams, which show the relationships between the files and the processors (or programs), were used often during the study and proved themselves to be quite valuable.

Another convention that should be explained for clarity is that most of the programs are list driven. That is, they run from a list of files (a file itself) and don't stop until all the files on the list are processed. In appendix B, this list file is usually shown above the processor. Normally the list contained all the runs for one test condition. At times as many as three computers, each with its own list, were used in parallel to process the data more quickly.

The intent of this paper is that it will aid a data analyst faced with a task similar to that encountered in analyzing the FASA oculometer data. A description of the oculometer equipment, environment, and procedures are provided. The logic of certain algorithms, which are considered important but complex, are discussed in the body of the paper. Statistics from the FASA study omitted from reference 1 and 2 are also included here. Those are given, not to support the conclusions of the FASA study, but to illustrate the techniques used. Source listings for the programs included in appendix C and the block diagrams in appendix B make it easier to understand how the data was processed and where to look in the source code for a given function. The record descriptors in appendix A will be especially helpful for anyone re-examining the FASA data.

2.0 System Overview

A detailed description of the development of the Langley oculometer system, along with its installation and operating procedures can be found in Appendix A of reference 3. The oculometer facility computes and stores a time history of eye-scanning events. The block diagram (figure 1) shows the components of the system. The oculometer (blocks 1-7 of figure 1) projects a collimated near-infrared beam of light into the test subject's eye. The system depends on algorithms that can compute a lookpoint, if given the relative position of two eye reflections. The computer compares the large backlighted pupil reflection to the much smaller and more intense corneal reflection. Using split image techniques the system directs the illuminating beam through the same tracking mirror system that collects the reflected images. It uses the angles of the two automatic tracking mirrors and the manually controlled focus of the eye-camera optics to correct the lookpoint calculation for subject head position. The oculometer electro-optic head (blocks 4 and 5) is located directly in front of the subject (block 8) and just below the simulated radar display as shown in the photograph in figure 2. This location is well outside the final controller's normal scan area, which is concentrated around the center of the display. The subject can detect only a dull red light in the head's mirror system. For the purposes of calibration and monitoring real-time performance, the system mixes (block 9) the computed lookpoint position with the plan position indicator (PPI) video signal that is nonobtrusively recorded (blocks 10 and 11) from a repeater display shown in figure 3. The resulting combined display (block 12) contains a small circle of light representing the lookpoint as it moves among the display symbols. An observer can monitor in real time both system and subject performance by viewing the combined signal, and a video recorder (block 13) stores the signal on tape for post run analysis.

The oculometer control and data processing system is located in the Human Engineering Methods (HEM) laboratory one floor below the mission oriented terminal area simulation (MOTAS) facility used to represent a TRACON facility in the FASA study. The MOTAS facility is documented in reference 4. Figure 4 is a photograph taken in the HEM laboratory. In the background corner sits a display monitor that has the mixed video with the controller PPI display and the controller's lookpoint superimposed. The three 5-inch video monitors in front of the main system operator (in the foreground) are used for system monitoring and control. Details of their displays do not show well in the photograph. The left monitor is a duplicate of the mixed PPI/lookpoint display. The center monitor shows the bright corneal reflection on the much larger and darker pupil reflection in the background. The monitor on the right displays an image of the

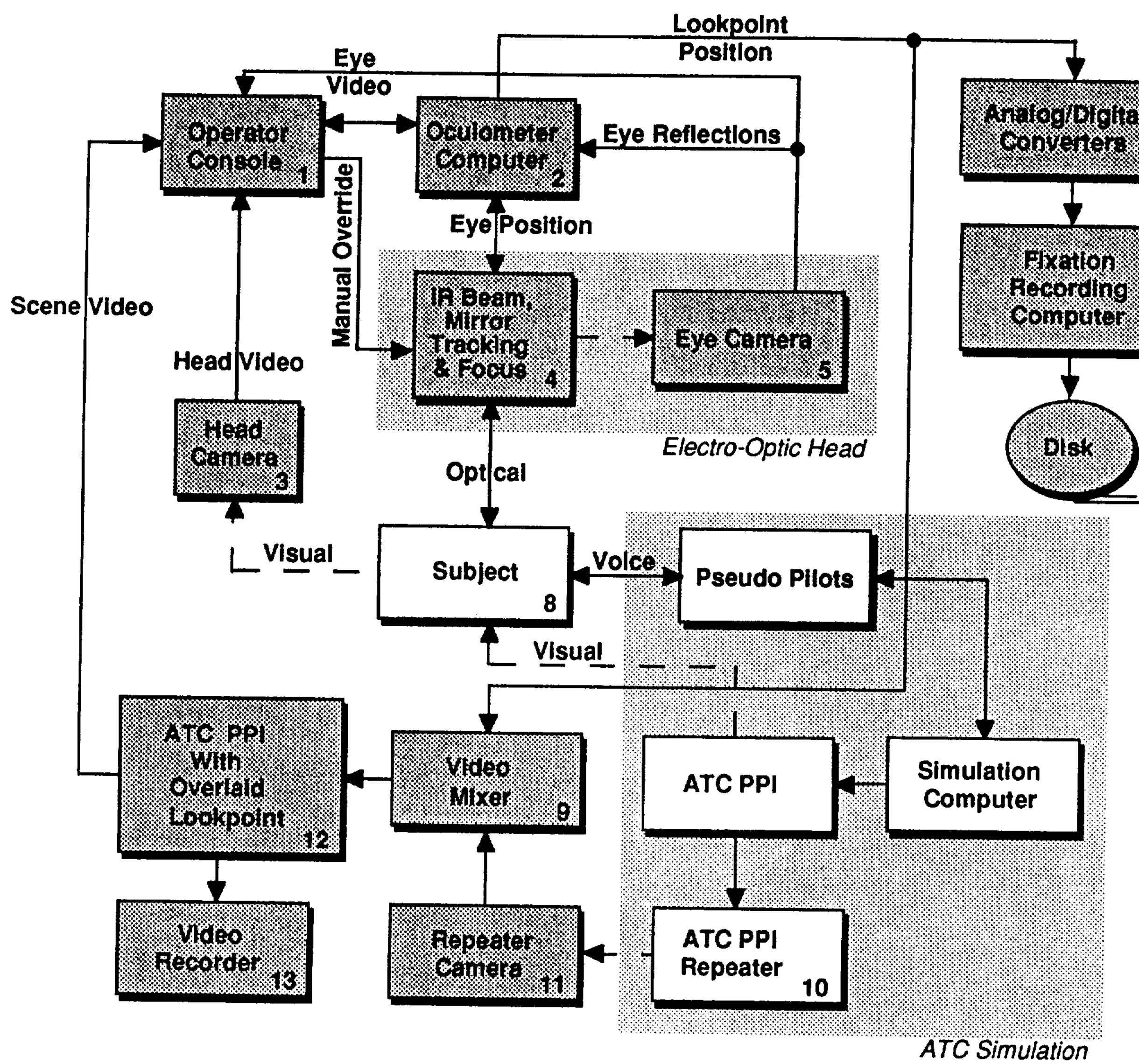


Figure 1. Operational block diagram of the NASA Langley oculometer facility interaction with the TIMER ATC simulation.

Figure 2. Photograph taken in the MOTAS facility showing a simulated ATC station with the oculometer electro-optic mechanism and video camera for monitoring subject head position (blocks 3, 4, and 5 in figure 1).

NASA

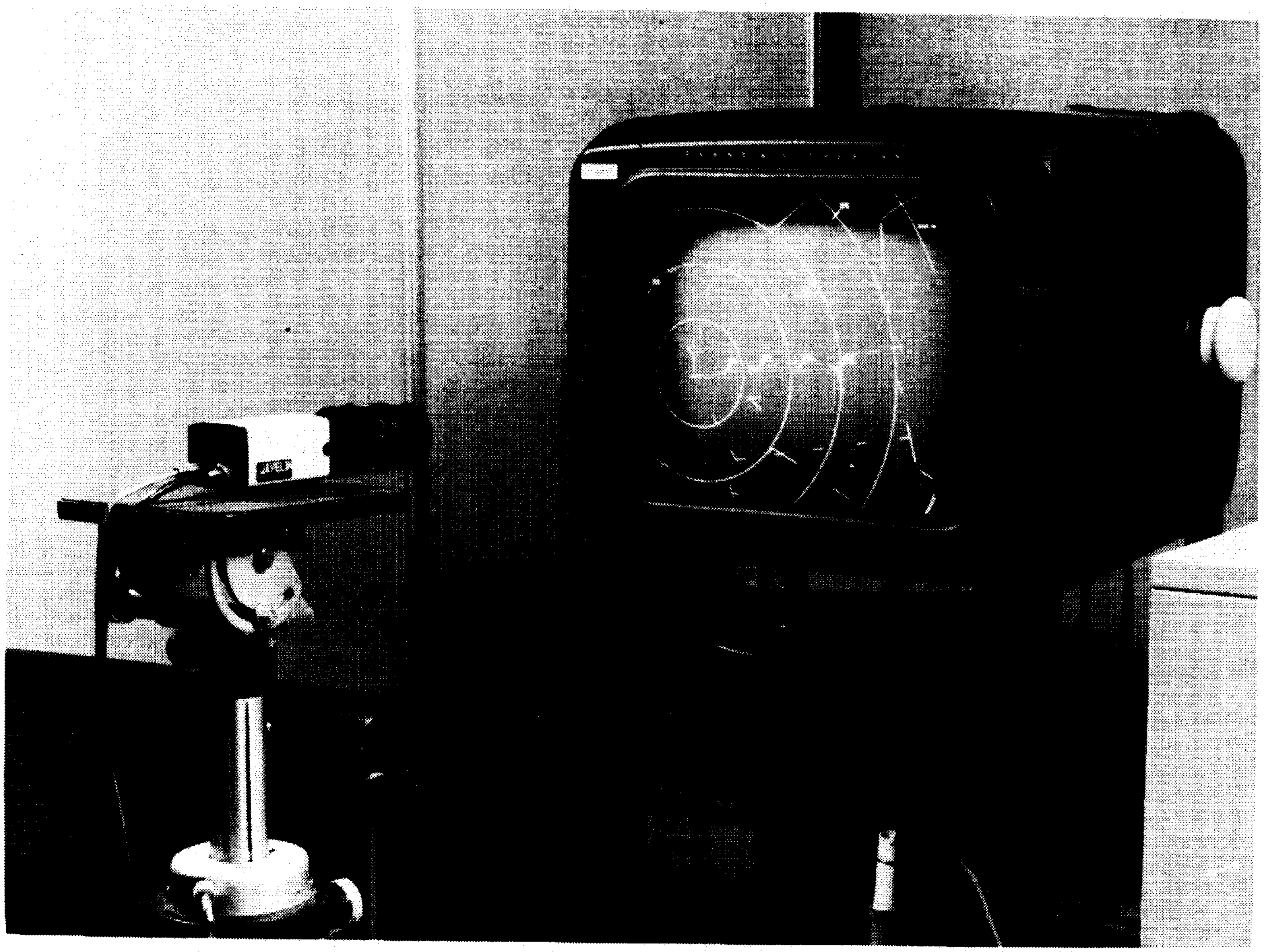


Figure 3. Photograph of PPI repeater display and video camera for generating video signal of the ATC display (blocks 10 and 11 in figure 1) to be video mixed with lookpoints.

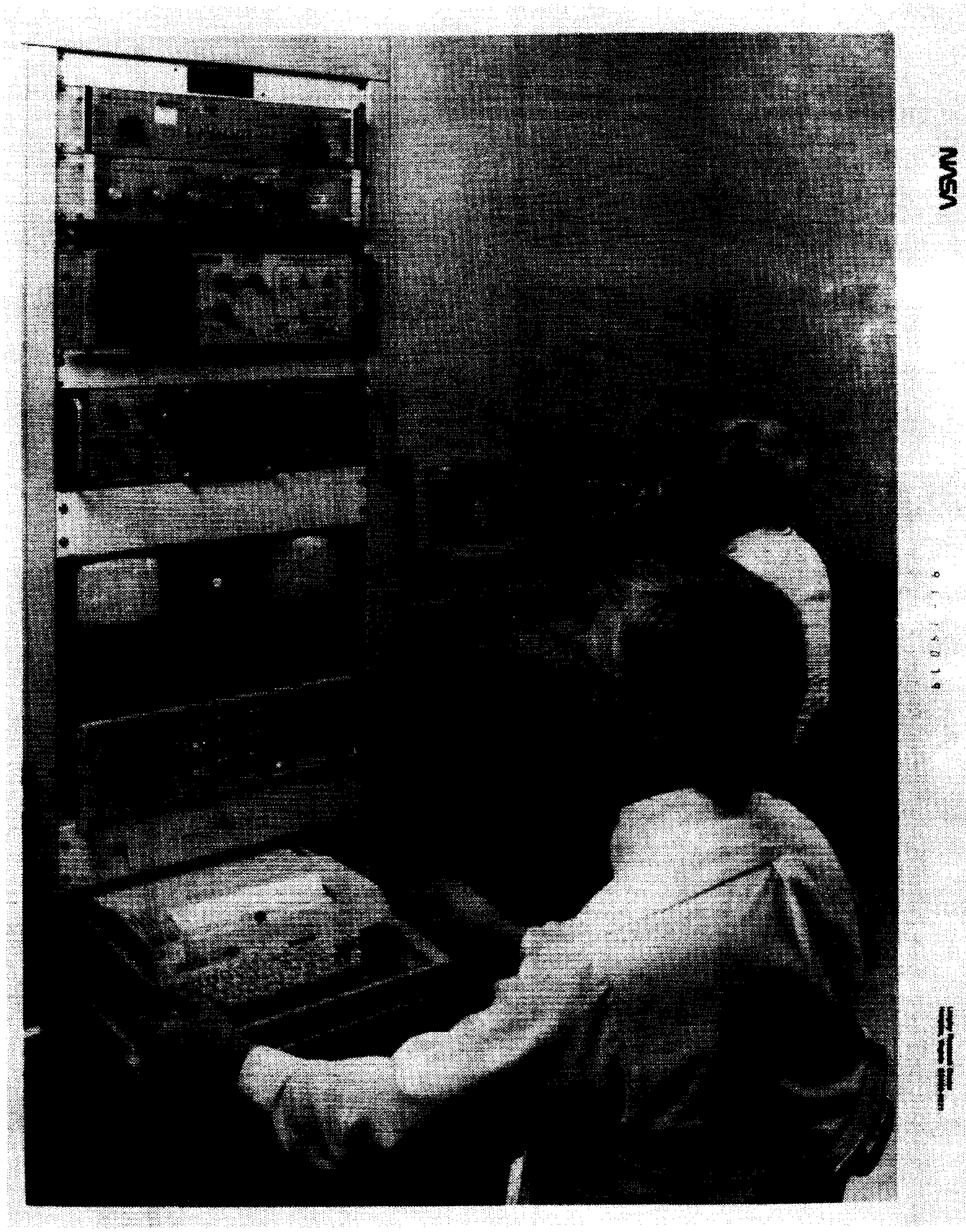


Figure 4. Photograph taken in the HEM laboratory showing the oculometer control and data processing system stations.

subject's head captured by the head position camera shown in figure 2. The operator uses this camera to observe the subject and, also, as an aid to recapture the subject's eye after losing track. The signal on the television waveform oscilloscope above the monitors is the sweep from the eye-camera used to determine the relative position of the two reflections. The central narrow peak indicates the corneal reflection. The broader peak at about half-voltage represents the pupil reflection, and the low voltage baseline represents the rest of the eye, including lids and lashes, all of which are adjusted to the video black reference level. By keeping track of video sweep count and timing when voltages cross specified levels, the system determines the center of each of the two reflections and, thus, their relative position on the camera vidicon.

Figure 4 does not include the oculometer computer. However, it does show several of the digital readouts and control inputs for the oculometer computer. Those include potentiometers, a standard typewriter keyboard, a joystick (under the operator's right hand), and a pushbutton (in the operator's left hand). Their principal use is for pre-run calibration, but the operator also uses them to dynamically compensate for the subject's posture adjustments. The mirror tracking is automatic and works well. The joy stick is a manual augmentation for the mirror tracking. The operator uses it to override the search algorithm when the eye is out-of-track and the system is trying to reacquire. He does not use the joy stick often but when used it speeds up reacquisition considerably. The fastest reacquisition involves the use of the pushbutton shown in his left hand which instantly returns the mirrors to a predefined nominal eye position. That technique is always tried first in reacquiring the eye, because most individuals performing a visual monitoring task often return to the same position after briefly rotating their heads to talk, read, or type on a keyboard. The operator controls the mirrors, the eye-camera focus, and collimated infrared beam intensity. He also determines and enters system parameters during calibration including parameters to adjust for inter-subject differences in corneal curvature.

The micro computer in the background of figure 4 collects and stores the visual events in real time in its random access memory (RAM). The computer contains a 16-channel analog-to-digital conversion circuit board that acquires eye scanning signals from the oculometer system and timing signals from the mainframe simulation computers. It should be noted that the mainframe computers are not used for eye scanning data collection during FASA because of the processing load required to operate the traffic intelligence for the management of efficient runway scheduling (TIMER) simulation program. At the end of each run the operator copies the records stored in RAM to a disk file for long term storage. Each record spans a variable time duration that is an integer multiple of the oculometer sample period (30 samples per second). There are four data fields per record containing lookpoint coordinates (2 dimensions), pupil diameter, and duration of the event. For an out-of-track event, the system records lookpoint coordinates as zero and stores a status code in the pupil diameter field. Once per simulation update, i.e., every four seconds, the system stores one other field on a second file. That field contains the sequence number of the visual event last completed as the ATC simulation interval started. The data reduction algorithms use this information later for synchronizing the recorded simulation data with the oculometer data.

2.1 Equipment, Environment, and Procedures

2.1.1 Measuring eye scanning behavior in glass display (CRT) environments

One of the greatest challenges presented by the FASA study for the eye scanning data analyst was that of answering the synchronization question, "What target was presented at location x,y at elapsed time 37:15 in the run?" (The '37:15' was chosen for illustration, only.) However, for the oculometer operator there was a more fundamental challenge underlying concerns for data quality, e.g., "What display feature was actually located 3 inches to the left and 2 inches below the center of the PPI?" In this type of visual environment all targets were behind glass. The oculometer system was simply measuring, in effect, where on the glass the subject was looking. If the runway threshold of 26L (the expected target) was actually being displayed at (-3, -2) inches then the data output was correct. If the PPI content had been shifted or rescaled, however, by a software offset or an electronic bias, the oculometer output would have been compromised. In this example, the subject actually may have been looking at the point (-3, -2) inches from the center, but some other display feature could have been displayed at that location because of a display error.

The oculometer was designed to report the lookpoint of a subject on a fixation plane defined relative to the face of the electro-optic head mirror box, or oculometer port. If the mirror box were to move to another location, the output data would be in error because the eye rotations would be viewed from a different observation point. Similar errors would occur if the features on the fixation plane moved to locations different from those previously described in the oculometer program geometry. In earlier studies at LaRC involving aircraft cockpits with conventional fixed-position instrumentation, the fixation plane was clearly defined by the stationary position of the visual targets. Now, with the advent of glass cockpits, the fixation plane can contain many possible display configurations, each with its own features and relative locations.

The air traffic control environment has been a glass environment for several decades, since the first use of radar displays for monitoring aircraft locations. Because the PPI made up the entire visual field of the final approach controllers in this study, the locations of all visual targets, including both fixed features and moving air traffic symbology, depended on the software driving the display as well as such hardware factors as control knobs and electronic circuit stability. In the Langley MOTAS facility, for example, the Evans & Sutherland CRT provided control knobs for adjusting display height and width and horizontal and vertical gains. An access cover was placed over those knobs during the FASA study to prevent rescaling and repositioning of the display features by the controllers, but other means of display alignment was necessary in order to insure the accuracy of PPI geometry for eye scanning data collection and analysis.

2.1.2 PPI Alignment Template

Figure 5 illustrates an alignment template which was designed for the FASA experiment and used at the beginning of each test session in order to insure that the fixation plane description of the PPI was accurate and consistent throughout the study. Each day the template was placed directly on the face of the final controller's PPI to check for proper positioning of the fixed features on the display. Exact placement of those features on the template was accomplished by direct measurement of the PPI during static display of stationary elements, including the Denver Stapleton runways, the final approach to runway 26L out to 20 nmi, the final approach fix (FAF), the final controller's airspace (referred to at Denver as the dump region), the Denver VORTAC (DEN), the airspace intersections FLOTS and WIFES, five nmi range-rings, and the outlines of the four approach corridors. The template also showed the positions of 25 aircraft placed at intervals of 5 nmi and arranged in a grid surrounding the final approach course. Each of those aircraft appeared on the template as an F connected by a leader to a representative ARTS III data block. This display of aircraft provided an array of visual targets for the subject controllers during oculometer calibration. An additional feature of the template was a reference axis system, shown as dotted lines, to allow rotational alignment with the Evans & Sutherland CRT enclosure.

After measurements were completed the template was plotted on clear acetate using computer-aided design and drawing (CADD) software. An axis transformation program was written to quickly convert FASA display coordinates to oculometer output coordinates. The conversion process, which included axis translation, rotation, and scaling, confirmed the physical PPI display location of each fixed feature or aircraft target generated by the simulation computer. In addition, the oculometer output voltages for each target were also confirmed, since they were scaled such that 1 volt = 1 inch. The simulation geometry was based on an axis system centered at the airport surveillance radar (ASR) site and represented aircraft locations in nautical miles north and east of that point. The origin appeared on the template as a small dot just north of the middle of runway 26L. The oculometer reference origin was located along the final approach course at a range of 10 miles, and the data output coordinates were represented in inches to the right and above that point (See figure 6 for the display area covered by the oculometer.). Relative to the oculometer axis system the FASA origin was translated 5.04 inches to the left and 0.9 inches below the oculometer origin. The 20 nmi final approach course on the PPI measured 9.625 inches, which resulted in a scale factor of 1 nmi = 0.48125 inches. While the oculometer axes were physically aligned with horizontal and vertical surfaces in the MOTAS facility, the FASA display axes were rotated counterclockwise 12 degrees. This rotation was the sum of the Denver local magnetic variation of 10 degrees east and a slight bias in the display CRT of 2 degrees in the same direction. In order to align properly with the PPI, the completed template contained the 12 degree rotation and the measured scale factor and axis translation. Thus

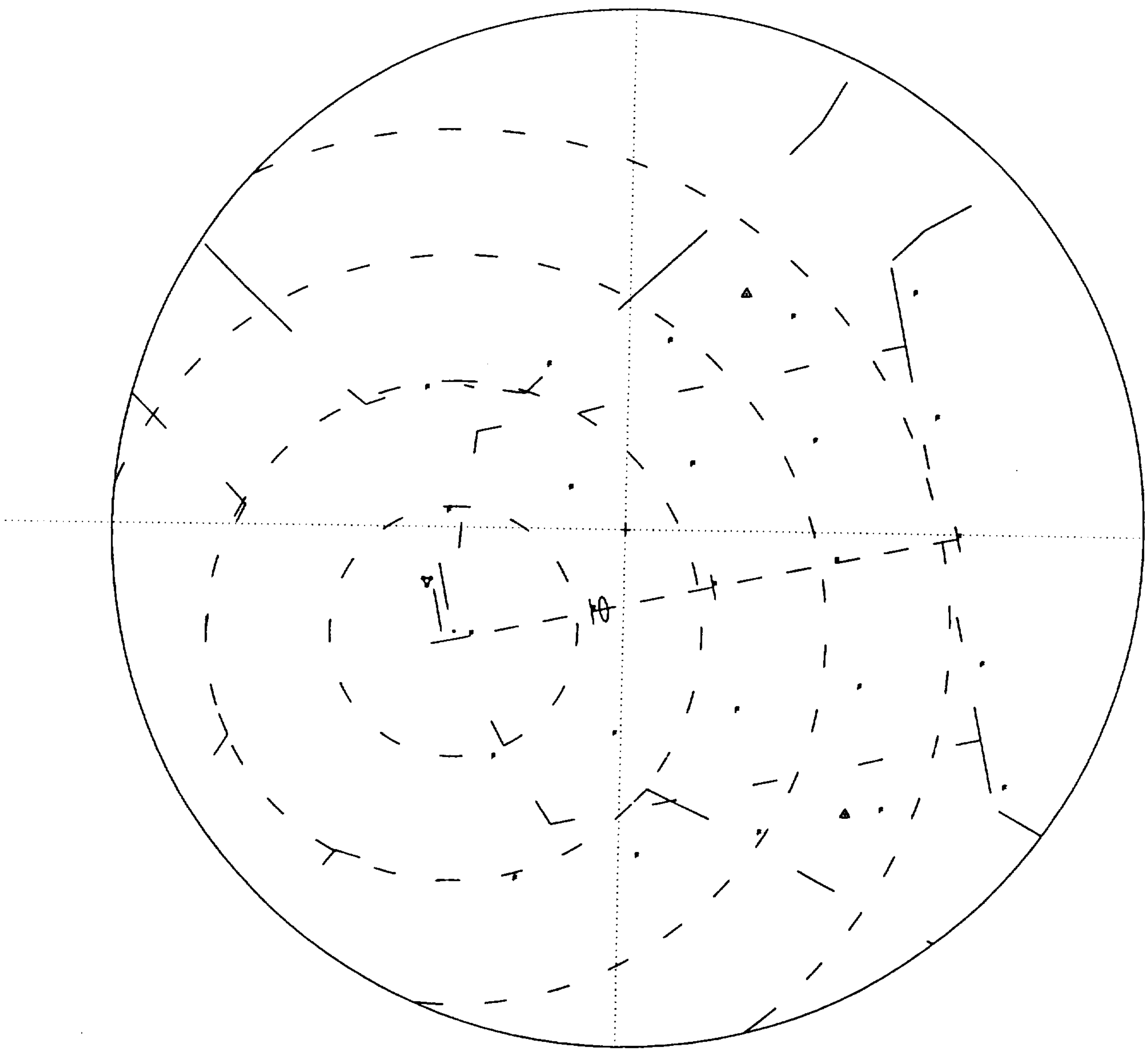


Figure 5. ATC PPI alignment template for the FASA study.

the relationship between the two axis systems can be represented by the equations:

XOCULO = SF*(X*SIN(A) + Y*COS(A)) - XOFF

YOCULO = SF*(X*COS(A) - Y*SIN(A)) - YOFF

where the variables XOCULO and YOCULO represent the horizontal and vertical oculometer output in inches (volts); the variables X and Y represent FASA coordinates north and east of the ASR in nautical miles; the constants XOFF and YOFF are the coordinates of the FASA axis system origin, in inches, as measured from the oculometer system origin; the constant SF represents the scale factor, 0.48125 nautical miles per inch; and the constant A represents the angle of rotation, -12 degrees (negative because the angular rotation is clockwise when going from the FASA system to the oculometer system).

After the display features were checked each day for proper positioning (scaling, offset, and rotation), the geometric relationship between the PPI visual targets and the oculometer port was confirmed. Then, scene video was aligned.

2.1.3 Video Alignment

Figure 3 shows the remote scene camera and PPI repeater display used for the FASA study. This use of a scene camera which was remotely located outside of the main test area was a first at LaRC. In previous Langley oculometer installations the scene camera was typically mounted over the subject's shoulder which caused geometric distortion in the view, as well as occasional problems with scene obstruction or camera disturbance by the subject. However, by referencing the oculometer data output to video from a repeater PPI rather than the subject-viewed PPI, several benefits were realized. First, the scene camera view remained physically undisturbed and unobstructed during the 70-minute test runs, because the camera was located outside of the active test area. Second, a true perspective of the PPI was made possible by positioning the camera squarely in front of the display (see figure 3). Third, the potential for wire clutter in the controller workspace was reduced by routing associated video cables into a separate part of the simulator room.

During video alignment the PPI features viewed by the remote scene camera were matched to their geometrical counterparts in the oculometer computer. As described in reference 3, the computer generated a series of points corresponding to measured features on the static display, including but not limited to the Denver VORTAC, the threshold of runway 26L, the FAF, and the four corners of the dump region. These points appeared as white dots on the oculometer control panel scene monitor (figure 4) and were fitted to the video scene by adjusting electronic potentiometers to correct for distortions in vertical and horizontal gain, bias, and cross talk. This procedure allowed the scene monitor, with its superimposed dots, to represent the actual data output of the oculometer system. Since the video information captured by the remote scene camera was aligned with corresponding points stored in the computer, any geometric distortions were canceled out. Data accuracy, therefore, was not compromised by using the PPI repeater monitor and remote scene camera.

2.1.4 Subject eye calibration

Prior to the first practice run by each subject controller, eye calibration was accomplished by directing the subject's gaze to each of 17 of the aircraft arranged in a grid around the final approach course. (Only 17 of the 25 aircraft generated for calibration were displayed within the 10-inch square visual area of interest.) After the targets were scanned, the oculometer operator manually adjusted linearization constants in the program to correct for errors in gain, offset, or pattern alignment, including distortions in vertical slant, horizontal tilt, pincushion, and curvatures along vertical or horizontal arcs. The test subject was allowed to look around freely and relax during the brief time adjustments were being made. The calibration targets were then re-scanned, and the process repeated until a reasonable result was achieved. Because of the potential for subject boredom leading to a poor calibration, the total time spent on the process was usually less than 3 minutes. Nearly two decades of eye scanning research at LaRC have demonstrated that this style of calibration procedure, although necessary, should be performed quickly and efficiently to permit the subject to move on to the task of interest. Controller interest and task engagement were high during the practice and data runs. The result was a more consistent lookpoint output and better calibration than were possible during the brief, but somewhat tedious manual calibration. Excellent final calibration results were achieved by fine tuning the distortion adjustments on the fly during the first practice run. The resulting calibration constants were saved for each subject and used for later test sessions.

2.1.5 Visual Area of Interest

Figure 6 shows the area of the PPI display covered by the oculometer used in the FASA study. Several tradeoffs were considered prior to the decision to restrict coverage to a 10-inch square. Sources of errors in eye scanning measurements can be grouped into three categories: 1) system errors, 2) operator errors, and 3) test subject eye physiology. They will be discussed individually, but the combined effects of the errors can reduce the value of a measurement technique to the point that questions relevant to the experiment cannot be adequately answered. The key to maximizing the utility of any measurement lies in examining the working hypotheses for a particular experiment. In the FASA study, it was necessary to decide what questions about eye scanning behavior were most relevant to the task of controlling aircraft in the final approach area. The three lookpoint measures selected for analysis included track time, average dwell time by object type, and number of cross checks (reference 1). Most of these measures required high-resolution data quality within the dump region even at the expense of lack of coverage at the edges of the 20-inch PPI. The task being observed depended heavily upon aircraft control within the dump region, and aircraft delivery-time accuracy was the primary performance criterion. The working hypotheses frequently demanded differentiation between the aircraft symbol and its data block, thereby requiring a lookpoint resolution of less than 0.5 inches which approached the operational limit of the Langley oculometer.

Each of the following three sources of measurement errors imposed data collection and analysis tradeoffs for the FASA project:

1) System errors. Each output channel of the oculometer has a range of 10 volts, as the output digital-to-analog converters (DACs) are capable of -5VDC to +5VDC. Because the equipment is a combination of electronic and optical subsystems, noise and distortion are inevitable. By covering a larger visual area, those errors create uncertainty over larger portions of the data field. For example, if the electrical noise were on the order of 0.5 VDC, then lookpoint jitters would be 1/20 the size of each linear dimension measured, which would be 0.5 inch in the 10-inch square covered during FASA and 1 full inch for the 20-inch square required to track the entire PPI. Fortunately, the electrical noise experienced during this study was much smaller and, together with the chosen scale factor, resulted in very slight jitters of the lookpoint. Similarly, optical and video limitations prevented the distinction between an aircraft symbol and its data block when the entire 20-inch PPI was viewed, but it was easy to distinguish such details when zoomed in to the 10-inch square shown in figure 6.

2) Operator errors. Although many of the features of the Langley oculometer allow for hands off operation, the system is not totally automated. Data quality depends on the ability of a human operator to accurately calibrate each test subject and to monitor system performance throughout the entire test. Both of those tasks require high resolution of the visual targets of interest. For this study of air traffic control in the final approach area, it was necessary to differentiate between an aircraft symbol and its data block, which required a video image capable of clearly displaying each of these small features. With the state-of-the-art technology in television equipment at the time of the study, this capability was only possible when viewing a relatively small area of the 20-inch display. Prior to the start of this project, the Langley oculometer was operated by scaling to the entire PPI. The results determined which aircraft was being viewed, but could not differentiate between the aircraft symbol and its data block.

3) Eye physiology of the test subject. Individual differences in corneal curvature among subjects result in output nonlinearities, which become progressively worse at larger visual angles from the oculometer port. For that reason, compromises during calibration must often be made. When operated for full-scale coverage of the 20-inch display, the system often required major adjustments to the linearization constants in order to match lookpoints to outlying targets around the perimeter. Obtaining good accuracy for the perimeter targets often resulted, however, in mediocre accuracy within the final approach area, or vice-versa. Because the 10-inch coverage involved a relatively small total visual angle (approximately 30 degrees for a subject about 20 inches from the face of the PPI), more precise calibration was possible during the actual FASA study. Therefore, errors due to physiological differences in the test subject eyes were minimized.

2.2 Quick Look Capability

The FASA experiments were run during June through September, 1991. One of 12 subject-controllers participated each week. The tests took four days per subject. Each half-day consisted of a

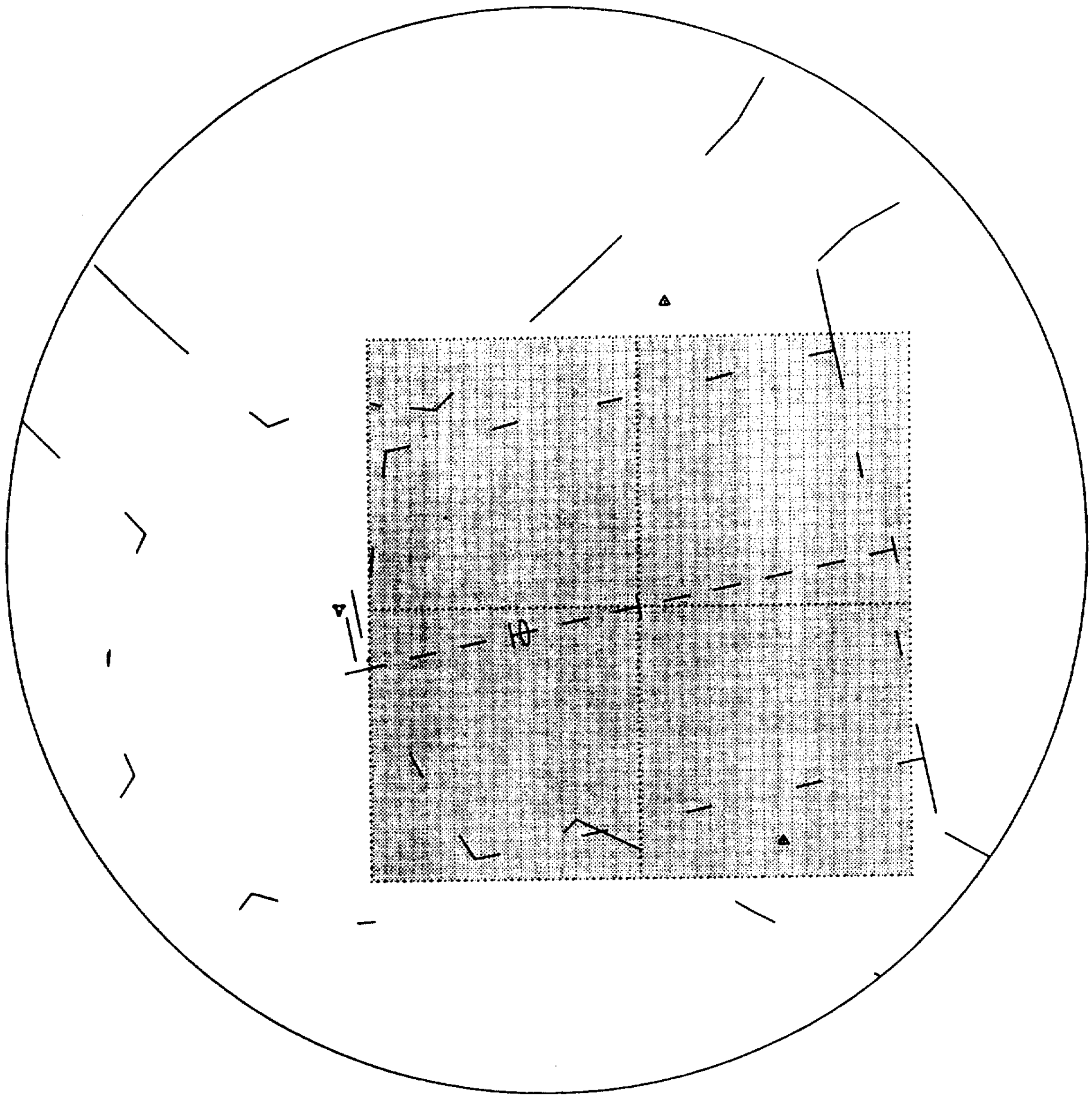


Figure 6. Area included in oculometer coverage for the FASA study.

session using one of the four display formats with one of the two approach-pattern speeds. Thus, each subject participated in eight different display-format tests (four formats times two pattern speeds) referred to as treatments. See reference 1 for a complete discussion of the design of the experiment. Each recorded test run lasted about 70 minutes. A large amount of data was acquired in a short time, which offset the fact that the equipment and personnel necessary to run this experiment were expensive and difficult to schedule.

After the experiment was started, it became apparent that a quick look feedback on the quality of the recorded data was needed. That is, the data needed to be tested immediately for quality and sufficiency. Otherwise, subsequent post experiment analysis might have been handicapped by data deficiencies. The approach taken was to try to expose any deficiencies early, while more response options were still available. Since the quick look requirement was not foreseen, a program to provide a quick look capability was quickly developed and was put into service during the third or fourth subject-week. That program, QKLOOK.BAS, is included in appendix C. It was used in conjunction with the RADAR/lookpoint display combination to monitor and tune the system.

The FASA quick look methodology is probably not directly extendible to other studies, since it is dependent on the equipment and data acquisition algorithms. Therefore, it will not be described in detail. However, the concept is directly extendible.

The quick look analysis examined three quantities: the length of time associated with a file record, the number of times certain events occurred, and pupil diameter. The following were all taken from the .DAT file.

- Total number of in-track fixations.
- Total number of out-of-track records.
- Total time of all in-track fixations.
- Total time of all out-of-track records.

They were subdivided into three classes of associated record time: 1) less than or equal to 0.1 seconds, 2) greater than 0.1 seconds and less than or equal to 0.4 seconds, and 3) greater than 0.4 seconds. In addition, they were considered as a percent of some larger class. For example, one item monitored was total time during a run for in-track events of duration greater than 0.4 seconds as a percent of total in-track time. Another example is average duration of all in-track events whose duration exceeded 0.4 seconds.

All of these measurements had to be judged on a relative basis. Certain runs were thought to be better than others. Results were compared among test runs to try to identify a serviceable discriminator in the statistics that would objectively and reliably indicate if and when the eye-scanning system was having problems. That approach was not completely successful. However, it helped the team to better understand what normal scanning behavior looked like on the monitors and how much normality varied among controllers.

For the oculometer system used in the FASA study, the number of oculometer data records (.DAT file) stored per minute proved to be a simple and useful parameter that could be easily calculated while making the run. When that rate was higher than normal, it alerted the operators that something might be functioning poorly so that attention to component performance could be increased. The parameter seemed to be sensitive to the ability of the system to track the subject's eye, noise in the system, and the style and speed of a particular controller's eye-scan pattern. The rate varied greatly among controllers. The highest average rates were three times the lowest. Figure 7 illustrates those differences. The variation among controllers as opposed to that caused by display format is discussed in section 3.0.

Figure 8 shows average pupil diameter for the twelve subjects for in-track records for which dwell time exceeded 0.4 seconds. Pupil diameter appeared to be insensitive to the display parameters being studied and was not used at all in the FASA study.

2.3 Recording, Synchronizing, and Filtering the Data

This section deals with the mechanics of defining a fixation and synchronizing it with the recorded simulation data. The problems themselves (not necessarily the solutions) have some generality, and in evaluating the results of the study reported in reference 1, some readers may want to know the details of the process.

Recording the Data. For this study, data runs were approximately 4200 seconds long and oculometer samples were taken at a rate of 30 per second for a total of about 126,000 samples per run. The time available to process a sample was about 33.3

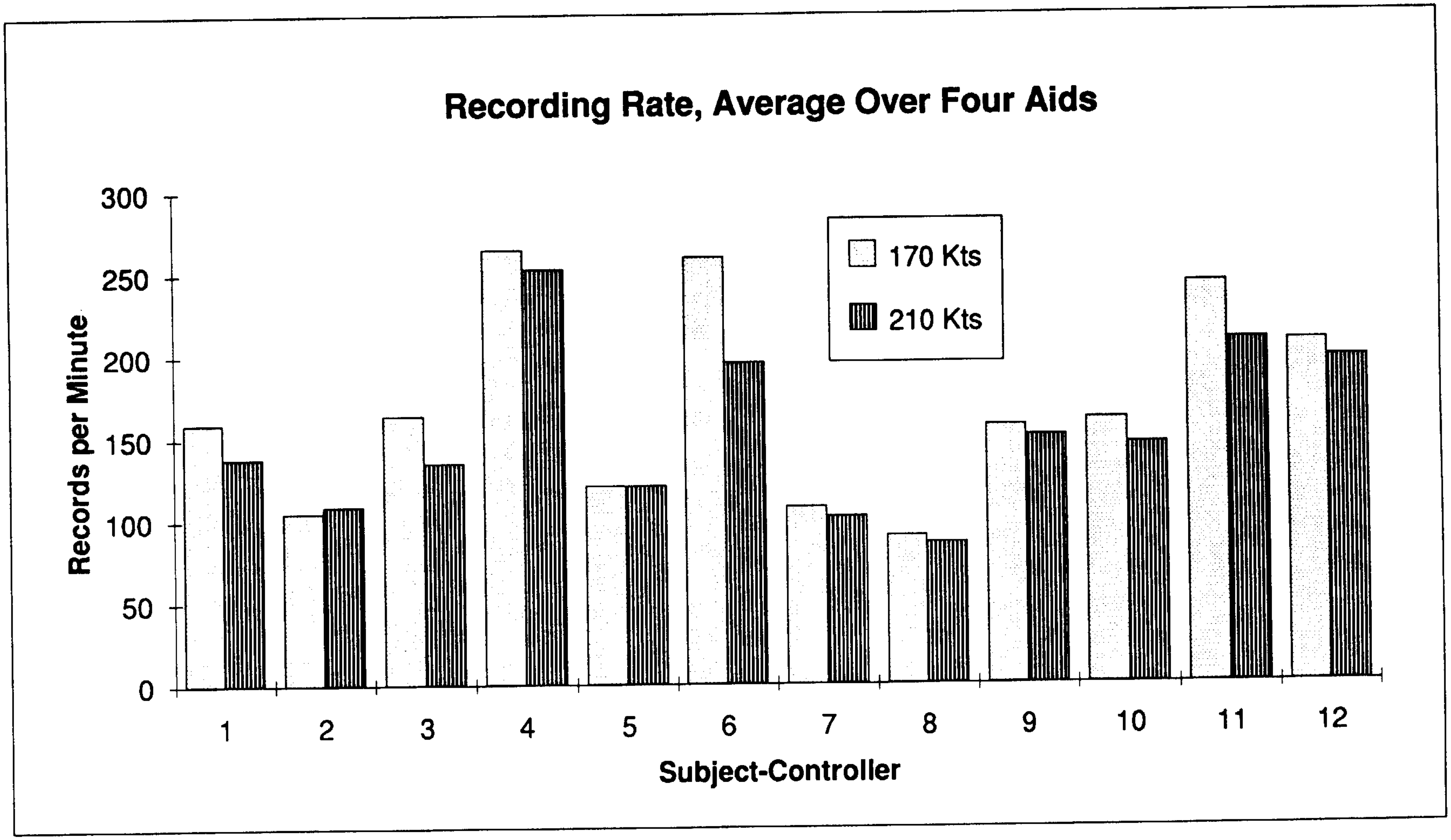


Figure 7. Comparison of average number of oculometer records per minute recorded for the FASA test subjects.

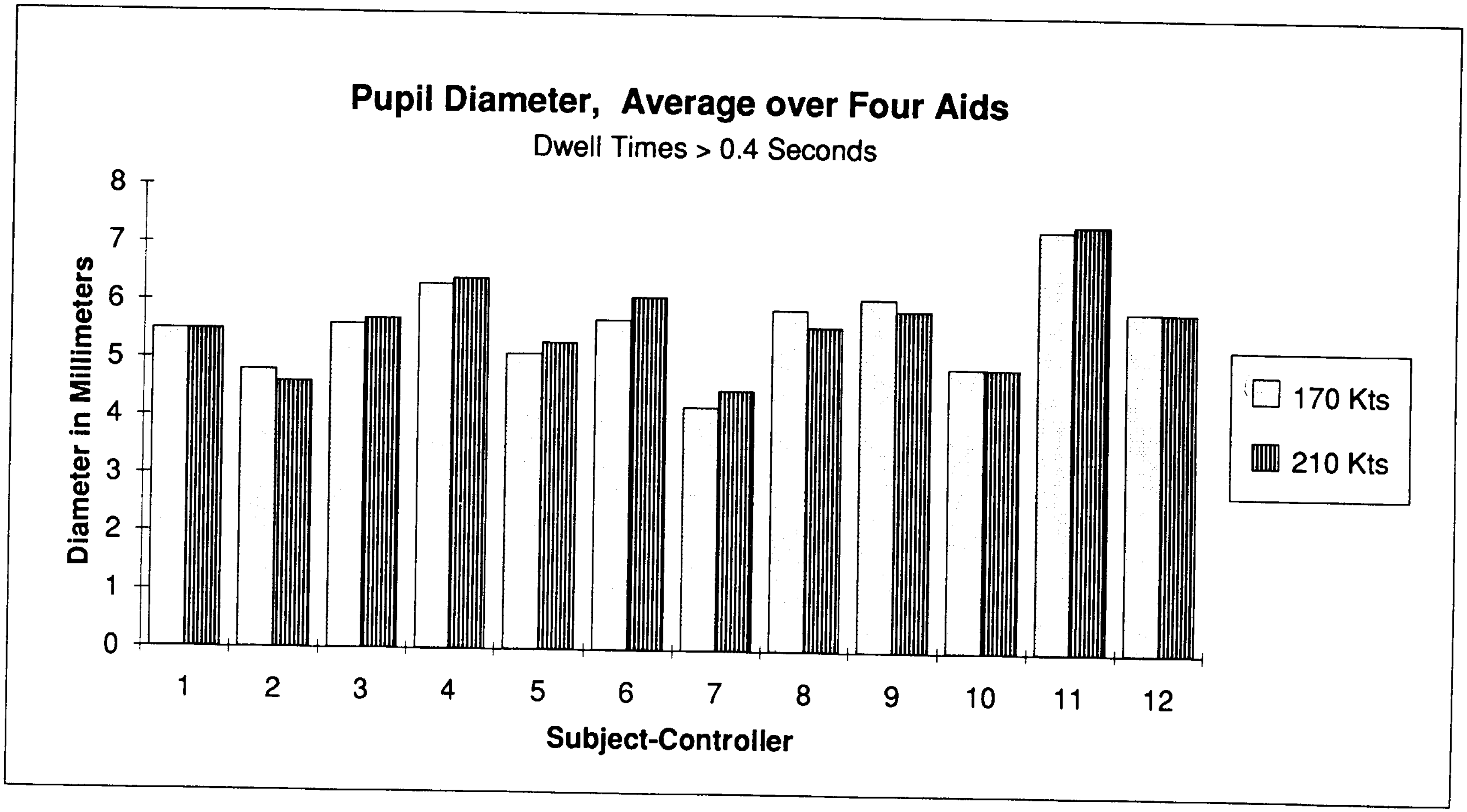


Figure 8. Comparison of average eye-pupil diameter during long fixations.

milliseconds. Because of the large number of samples, the limited size of random access memory in the data collection computer, and relatively slow hard disk access time, the researchers decided to preprocess the data. That approach decreased the number of records to be saved and allowed the data to be kept in primary memory until the end of the test, when it was stored on the disk for post-processing.

The preprocessing consisted of combining sequential samples into a single longer sample. For example, any number n of consecutive out-of-track samples were combined into a single sample of length n before being stored in the appropriate in-memory arrays. The occurrence of an in-track sample caused the storage of a previously accumulating out-of-track sample-sequence. Likewise, an out-of-track sample caused the termination and storage of an accumulating in-track sequence, i.e., a fixation.

Another part of the data compression process deals with the transition from one in-track sequence (fixation) to another. The following factors were used to determine the occurrence of a transition. The lookpoint position of a sample was compared with the preceding position. The lookpoint position of the next sample was compared with the average of the first two, etc. For each succeeding sample the lookpoint position was compared with the previous running average of lookpoint positions. If the distance between them was greater than a specified threshold (1 inch in the FASA study), the accumulated sequence was stored and a new fixation sequence was started. Otherwise, the running average was updated to include the present sample, and the sequence continued uninterrupted. This data compression prior to recording, although necessary, was of course irreversible and, therefore, added to the criticality of the acquisition process. It should be noted that the newer oculometer systems are not faced with the above data storage restrictions. In those systems, access to the raw data is provided for any post-processing that might be desired.

When an in-track sequence was stored, its lookpoint position was the average of the sampled positions. The pupil diameter was recorded as an average of the pupil diameters for the included samples. The duration of the sequence is the number of samples included multiplied later by 1/30 second. The two files created at the end of each test are referred to as the .DAT file and the .SCN file. Here as elsewhere in this paper, a file type will be referred to by its suffix in order to minimize ambiguity. There are 96 files of each type. Their file names identify 1) the controller subject, 2) the FASA display format, 3) the approach-pattern speed, and 4) the run number. For example, a file might be named MC12MN21.SCN indicating the 210 manual run numbered 12 by subject MC. Using this convention (file type suffix), one can find the data record descriptor for the file type in Appendix A and a block diagram describing how the file type is processed in appendix B. Details (source listings) of each processor are given in Appendix C.

Each data record in the .DAT file is composed of four 2-byte integers: the x and y screen coordinates, the pupil diameter, and the number of included samples. Out-of -track records are identified by a pupil diameter of zero. The number of records in the .DAT file, for a given length of time and treatment, varies significantly among subjects. For all 96 data runs, the average number of records in the files is 11,234. Since the runs lasted approximately 4200 seconds, this indicates a rate of about 2.7 records per second. The 8 run average per controller varied, however, from a low of 6,173 records to a high of 17,076 records. This large difference between subjects was easily observable during the runs. The lookpoint motion would appear slow and deliberate for one subject, but for another it would appear very rapid. The filtering described later decreased the variation across subjects, and the use of the repeated measures analysis of variance (ANOVA) prevented the differences between subjects from masking out the cross treatment variations of interest.

Synchronizing. During the test runs, the dynamics of the aircraft being simulated, their relative geometry, and the graphical interface to the pseudo pilots and controller subject were being computed on a large mainframe computer. The computer supports, and is tightly coupled to, the MOTAS facility. This large and sophisticated real-time simulation produces a time-history file (.ACP) that gives the position of all the aircraft on the controller's display every four seconds, a typical sweep-rate of an airport surveillance radar. In order to determine what the subject is looking at during any particular fixation, the frame (a set of contiguous records) of the .ACP file that describes the display at the time the fixation was recorded must be found. Then, the frame is searched for an object whose coordinates are sufficiently close (within 0.57 inches) to the lookpoint coordinates. Establishing the correspondence between each fixation and a particular display frame is referred to as synchronizing the .DAT file to the .ACP file. The search for a lookpoint object is conducted later, after the data is filtered, but the method used to synchronize the files requires that the synchronization must be the first step. It must precede any deletion or combining of records on the .DAT file.

The .SCN file is used to affect the synchronization of the fixation file with the time history file. At the beginning of each simulation update (simulated radar sweep), the simulation causes a bit to be toggled in the oculometer recording computer. This, in turn, causes the record number of the last .DAT record stored to be written to the .SCN file (in-memory array). The single integer on the .SCN record is a pointer to a record on the .DAT file. The .SCN record number or ordinal indicates time elapsed in the simulation and is in itself a pointer to a frame on the .ACP time history file. For example, the 10th record on the .SCN file corresponds to the 10th frame on the .ACP file and was written at 36 seconds of elapsed time (The first record was written at time equal to zero, the second record was written at an elapsed time of 4 seconds, etc.). The algorithm for synchronizing the .DAT and .ACP files determines which records of the .DAT fixation file correspond to frame i of the .ACP file based on the pointers contained in records i and i plus one of the .SCN file. For example, if record 500 of the .SCN file contains the integer 7123 and record 501 contains the integer 7135, this indicates that record numbers 7125 through 7136 on the .DAT file correspond to display frame 500 on the .ACP file, which was recorded at an elapsed time of 1996 seconds. This algorithm is implemented in the subroutine FIXPOINTER, which is part of the processor FIXPOINT listed in Appendix C and shown graphically in Appendix B. The processor reads the .DAT and .SCN file and assigns a frame in the .ACP file to each record on the .DAT file. The resulting .DT1 file is the same as the .DAT file except that each record has an added field, which points to the corresponding frame on the .ACP file. Because the .ACP file has a complex structure with a variable record size and a variable number of records per frame, it was indexed by the .IDX file, which was produced by the CRI8IDX process. Thus, the pointer in the .DT1 record references a record number in the .IDX file, the contents of which point to the first byte of the appropriate frame in the .ACP file. The .DT1 file is synchronized. It is one of several intermediate files produced on the way to the final .MRG file, which contains both a description of each fixation and of the display object of the fixation as well as the distance between lookpoint and object.

The synchronizing algorithm is based on the following logic. Record i of the .SCN file points to record m of the .DAT file and record i plus one of the .SCN file points to record n of the .DAT file. When record i of the .SCN was recorded, .DAT record m was already complete and record m plus one was already started. Therefore, both of those records should be assigned to a previous frame of the .ACP file. This would include records 7123 and 7124 in the example above. The records m plus 2 to n plus 1 are assigned to the current frame i of the .ACP file, records 7125 through 7136 in the example above. In a case where n equals m, no .DAT records are assigned to .ACP frame i. If n equals m plus 1, then only .DAT record n plus 1 is assigned to the current frame i of the .ACP file. This algorithm has the advantage of treating every frame individually rather than accumulating elapsed time (and error) from the beginning of the .DAT file.

Removing bad records. Once the files were synchronized the data could be cleaned up by removing bad records, combining adjacent records, and eliminating some noise records.

It was suspected that faults in the acquisition system caused some of the oculometer data records (.DAT) to be inaccurate. Occasional computer interruptions resulted in unusually long records. Extensive logs were kept by several observers during the testing and were combined afterwards into a single test log. Those logs were used together with automatic scanning (the SRCHDAT process) of all .DAT files, which was followed by selective manual scanning (the DMPDAT or DMPDT1 process) of sections of the files to identify bad records. Less than 0.02% of the million or so .DAT records were marked as suspicious, and more than half of those were associated with 2 subjects. Also more than half of the suspicious records were associated with logged problems. Only 20 of those records, all of which were associated with logged oculometer problems, were excised from the files. It was thought that the remainder of the long records would have a negligible effect on the study. Once identified, the processor CUT20 was used to excise the records from the files. As an aside, it should be emphasized that the capability to quickly and conveniently examine the data files is very important in these studies. Mistakes later in the process can probably be corrected without too much effort, but problems in data acquisition usually can only be corrected by re-testing the subjects, a very expensive prospect. It behooves the analyst to keep his eye on the data and to use quick look analyses to verify its integrity.

Filtering. Filtering was used to eliminate hardware/operator and human behavior induced problems in the data. Four filters were applied. Filters 1 through 3 consisted of combining certain sequences into a single event or record much like what was done during the recording process. The fourth filter just removed some out-of-track noise. The order that the filters were applied is significant. The salient features of the four filters are described here. More detail on the filters can be found in the source listing of the process FILTER1 in Appendix C.

Filter 1 reads three consecutive records from the .DT1 file into memory arrays A, B, and C. The condition for filtering is both A and B are in-track, the duration of B is one sample time (1/30 seconds), and C is not in-track. If the condition is true, add 1 to A's dwell time and store it in the scratch file .DT2, then advance C into A and read the next 2 records from .DT1 into B and C. Restart the process by again testing for the condition. If the condition tested false, store A into .DT2, advance B and C into A and B and read the next record from .DT1 into C. Restart the process by again testing the condition. There are, of course, tests for end of data for which appropriate action is taken. When finished the new lookpoint file is .DT2. Filter 1 is relatively unimportant and was included to compensate for a small problem in the collection software. It merely combines two consecutive in-track records, the second of which is only 1/30 of a second long.

The second filtering process (filters 2 and 3) reads three consecutive records from the .DT2 file into memory arrays A, B, and C. Condition 1 for filtering is the time duration of B is less than 13 counts, A and C are both in-track, and the display distance between A and C is 1/2 inch or less. Condition 2 is that either B is out-of-track or the duration of B is less than four sample periods. If either condition is not met, filters 2 and 3 are not applied and the process proceeds to test for filter 4. After applying filter 4, the process returns to test a new sequence, A, B, and C for filter 2 and 3 again. In this manner, the process advances through the entire file. If conditions 1 and 2 are both true, then either filter 2 or 3 is applied depending whether or not the duration of B exceeds three sample times (i.e., B is a blink). Both filters 2 and 3 combine the three records into a single record. The resulting lookpoint coordinates and pupil diameter are an average of those from A and C, weighted by their respective dwell times. The resulting dwell time for the combined record is either the sum of the dwell times of the three records (filter 2) or, if record B is a blink (out-of-track and of duration 4-12 sample times), the sum of the A and C dwell times only (filter 3). This resulting fixation is stored in A and two new records are read from the .DT2 file into B and C. The filter 2 and 3 process is then continued by testing this new sequence for condition 1 and 2. Again, there are tests for end of data for which appropriate action is taken. Filters 2 and 3 combine three records that appear to be a single fixation interrupted by a blink or a noise record into a single record.

When the process branches to filter 4, the A, B, and C arrays contain a sequence that did not qualify for filter 2 or 3 processing. If the duration of A is greater than three, A is stored on the .DT3 file, the final output of the successive filtering process. Otherwise A is discarded. In either case, B and C are moved to A and B, and a new record is read from the .DT2 file into C. Control is then moved back to the condition 1 and 2 tests for filter 2 and 3. Filter 4 removes all remaining records with duration less than four that have already survived filters 1 through 3.

The application of those filters significantly reduced the number of records that had to be processed further.

Prior to filtering: The average number of records per second for the 96 runs (12 subjects by 8 treatments) was 2.7 with a standard deviation 0.9 records per second. This resulted in .SCN files with lengths averaging (within controller but across the 8 treatments) from a low of 6,173 records to a high of 17,076. During 81.5% of the recording time, the subjects were in-track. The average length of in-track records was 0.44 seconds.

After filtering: The average number of records per second for the 96 runs (12 subjects by 8 treatments) was 1.5 with a standard deviation 0.3 records per second. This resulted in .SCN files with lengths from a low of 3579 records to a high of 8789. During 84.1% of the recording time, the subjects were in-track. The average length of in-track records was 0.73 seconds.

2.4 Target Identification

After associating each fixation with a particular TIMER simulation update frame as discussed above in Section 2.3, it is necessary to determine what the subject controller viewed during that image frame. This was done to learn something about subject eye scan behavior and in turn about the merits of proposed changes in the display. Normally there are about ten aircraft on the display at any time. While watching the test, with the computed lookpoint superimposed on the repeater displays, an observer is usually impressed by both the static and dynamic accuracy of the oculometer. The lookpoint swiftly traverses from object to object, pausing an average of less than a second on each. Just as the accuracy is obvious, so to is the problem with static resolution. Often the lookpoint will be slightly offset from the object. The observer can tell what the subject is looking at even though the projected lookpoint is not always right on the target. This type of offset is not necessarily due to anomalies in the oculometer system. Since the fovea is one degree wide, the human vision can register an item of interest without focusing directly on the object. The goal in the data analysis, therefore, is to find the nearest object to the lookpoint. The offset varies with time, subject, and position on the display, and it has a definite random component. It is minimized by careful calibration before and during the test. It should always be considered in the design of the experiment.

The word hit in this section is shorthand for associating a lookpoint with a screen object. A hit occurs when the distance between an object and a lookpoint is less than 0.57 screen inches, which for the FASA study was equivalent to about 1.2 nautical miles in the terminal area. The 0.57 inches was derived from our estimates of oculometer offset errors. If several hits occur on a given radar sweep, the lookpoint is usually associated with the closest object. The exception was the centerline slot marker and graphic marker runs, when the lookpoint was close to both the aircraft and aid. In this case, the lookpoint was assigned to a combined category, aircraft and aid, rather than to the closest of the two. This approach had more meaning in the context of the FASA study. Each aircraft on the display has an associated position symbol, normally the letter F, connected to a data block by a leader (reference 1). The nominal position of the data block (the middle of the second line) is about 3/4 of an inch from the aircraft position symbol. During the DICE runs, two lines of DICE information are added to the bottom of the data block and they are closer to the aircraft symbol than the normal data block text fields. Therefore, there is an area of intersection between the data block and aircraft position symbol hit circles. The lookpoint is assigned to the closer of the two. The third class of nonstationary objects on the display, in addition to the aircraft position symbol and data block, are the final approach spacing aids. Stationary objects on the display assigned as targets include navigational aids, the imaginary downwind flight legs (which are not displayed), the final approach course centerline, and the scheduled arrival sequence list in the upper right corner of the display.

If no assignment could be made after searching through the display, the object of the lookpoint was designated as No ID. Less than ten percent (on average) of the in-track fixations were associated with the No ID object class. Although fixations assigned to this object class represented a small percentage of the total fixations, the No ID lookpoints were displayed by themselves in hopes of discerning a pattern. They appeared randomly distributed over the area of interest. In addition, the group had a markedly lower average fixation time. It appears that this class of objects may be indicative of a characteristic of deliberate scanning behavior rather than a reflection of the eye scanning system accuracy.

Target identification is implemented by the program FILLMRG. The source code for that program is given in Appendix C. Figure B9 (Appendix B) is a block diagram of the process. The skeletal (formatted but not yet filled in) .MRG file is the product of CRE8MRG1 shown in figure B2. It contains the lookpoint data and the simulation output record pointer. FILLMRG adds the target information extracted from the .ACP file. Both the .MRG and .ACP records are discussed and illustrated in Appendix A. Since the .MRG record is both an input and output of FILLMRG, (figure B9), for safety, the output file is temporarily named as a .DUM file. When the process is complete, all the .DUM files are renamed to .MRG files. This simple device is more tolerant of error. The .MRG files are the primary eye-scanning output file. The cross check scan files (figure B5) and almost all the scanning statistics are derived from the information on these .MRG files.

Figures 9 through 14 describe the flow through the FILLMRG program. Figure 9 describes the main program. As is true in many of the programs written for this study, FILLMRG is list driven. File 1 is a list of test runs. Each run is processed until the list is exhausted. Usually the list would have 12 entries, one for each of the subject-controllers for a given test configuration. The subprogram SEARCH is called once per list entry and it processes a complete .MRG file. The flow logic for SEARCH is shown in figure 10. It uses TARGETSET once per simulation update to set up a list of possible targets of the fixation. SEARCH uses subprogram PICK to select a target from the list. PICK is called once for each in-track fixation. Note that the lookpoint coordinates are transformed to the simulation reference frame before the search begins. When a hit is made, the coordinates of the target object are transformed back to display coordinates for storage on the .DUM file. Thus, all the coordinates on the .MRG files are in the display frame of reference. If PICK does not find an object sufficiently close to the lookpoint, several other objects are checked: the final approach course centerline, the downwind lines, the scheduled aircraft list and the outer marker. If the closest object (from PICK) is a data block but it is not a hit, SEARCH tests again requiring only that the X and Y distances both be less than 0.57 inches. This minor relaxation substitutes a square with sides of 1.14 inches for a circle of diameter 1.14 inches and is used only with the data block. If subroutine SEARCH fails to find an object close enough to the lookpoint, the No ID designator is used and also stored with the lookpoint data in the .DUM file. Although not shown in the logic flow diagrams, out-of-track records are copied from the input file to the output file without processing.

TARGETSET (figure 11) makes a list of targets to be used by PICK. Knowledge of the structure of .ACP file record as illustrated and described in Appendix A is helpful in understanding TARGETSET. The first group on the list is all the displayed aircraft. They are followed by the group of data blocks, one block for each aircraft. For a graphic marker or centerline slot marker run, the data blocks are followed by a list of aids visible on that particular sweep. The list format varies with the type of aid. For the DICE runs, no entries are added to the list, but both aircraft and data block entries are modified for a particular aircraft if its DICE is active on the display. Finally, ten static navigational aids are added to the list. Their designators and positions can be found on the second page of the FILLMRG source program in Appendix C. They are in the arrays STATID, STATXS, and STATYS. The locations are given in the simulation frame of reference (nautical miles). As an aside, some other interesting constants can be found on the first page of the FILLMRG listing. SF1 = .472 is the scale factor indicating .472 inches per nautical mile. RUNOFF1 is the offset of the runway from the Y simulation axis (in nautical miles). Alpha is the angle between the two coordinate systems, etc.

Subprogram PICK (figure 12) chooses an object from the TARGETSET list. The calling parameter K is the list ordinal of the closest object. If K is returned as zero, the subprogram could not find an object within three nautical miles. For a CSM or GM run, if both the aircraft and aid are within 1.2 nautical miles of the lookpoint, the type of the closest object is changed to a combination of aircraft and aid.

The FASA study showed that the most popular target was the data block, followed by the aircraft symbol and then by the aid. Average fixation times varied widely by object. Time spent on static targets was very low.

2.5 Cross Check Scanning

Previous researchers (reference 3) have concluded that there is valuable information in scan sequencing behavior. The argument is that the pattern of transitions from object to object contains information related to relative mental loading. In this approach one searches the pattern for order or for repetitive behavior. The lack of order is referred to as entropy, which is thought to be inversely proportional to mental work load, i.e., higher entropy indicates less workload. Reference 3 suggests composing a transition matrix from the scan time history and then comparing the entropy of the matrices for the various treatments. The transition matrix is square (n x n) where n is the number of objects in the display. Element (i,k) would be proportional to the number of transitions measured between object i and object k. Element (k,i) would be different and would be proportional to the number of scan transitions from object k to object i.

The above model and associated hypothesis was not used in the FASA study. However, it was agreed that scan transition behavior should be included in the FASA analysis. Complicating factors in this study were that the display changed as

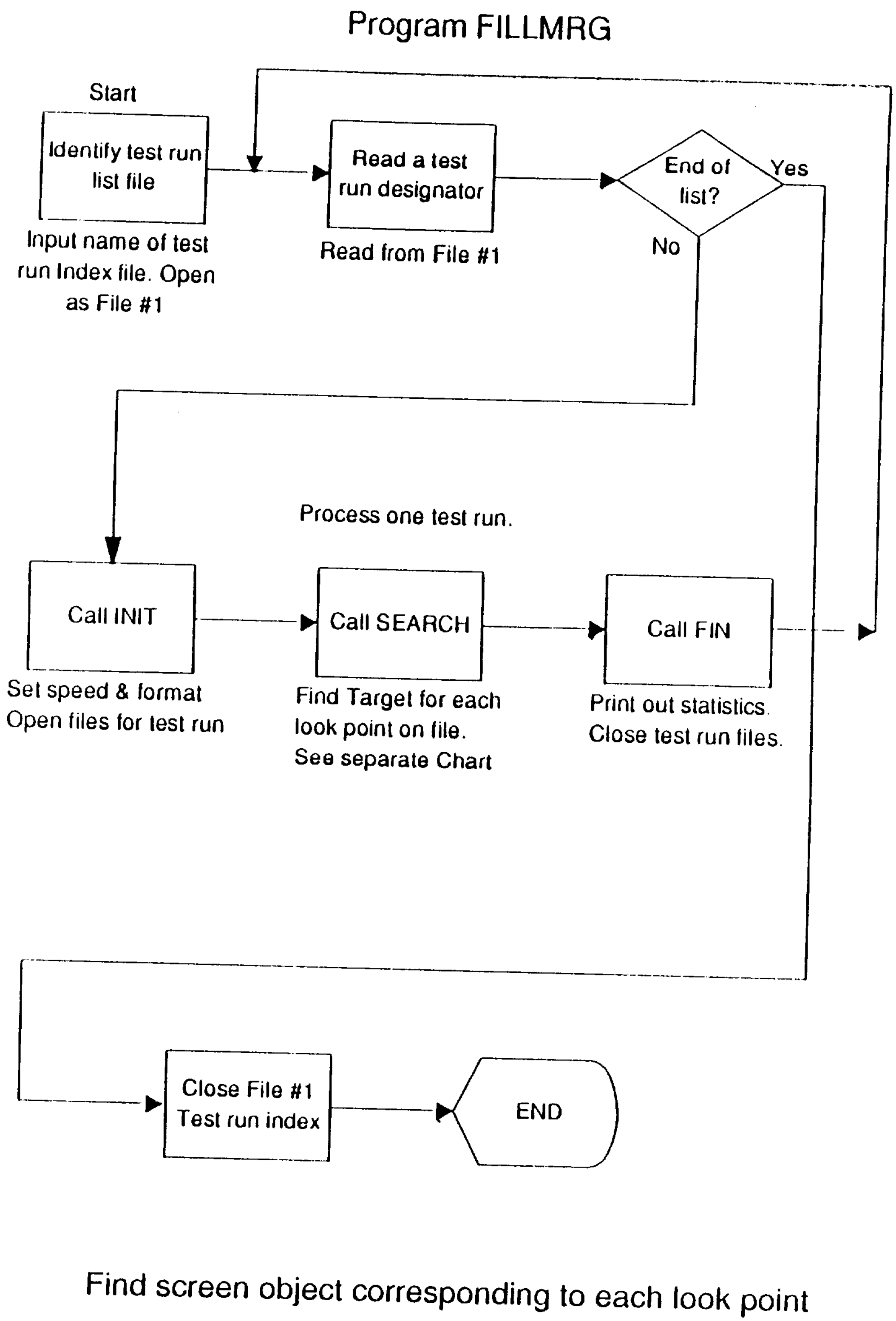


Figure 9. Flowchart of main target identification program FILLMRG.

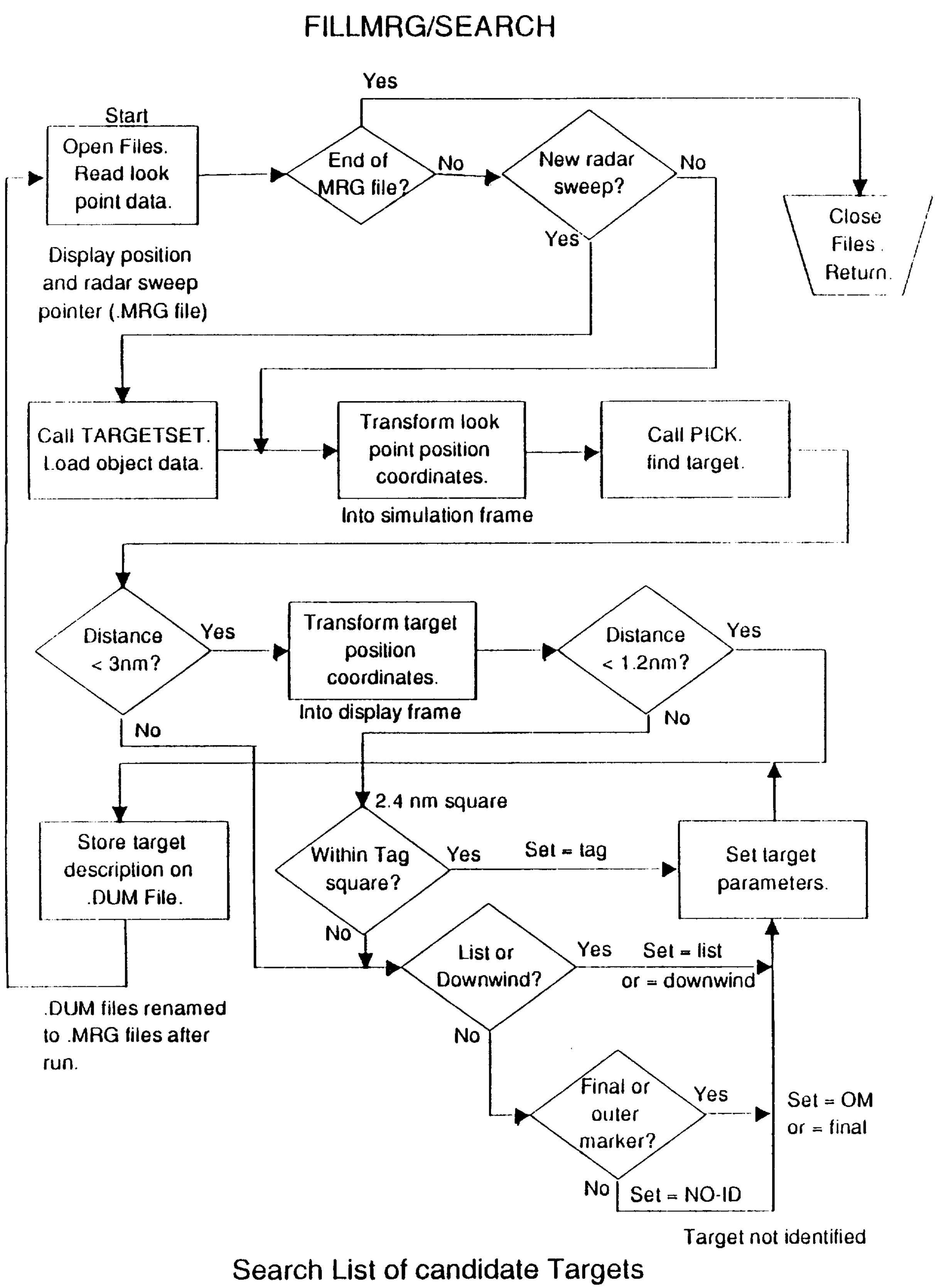


Figure 10. Flowchart of object search program used in target identification.

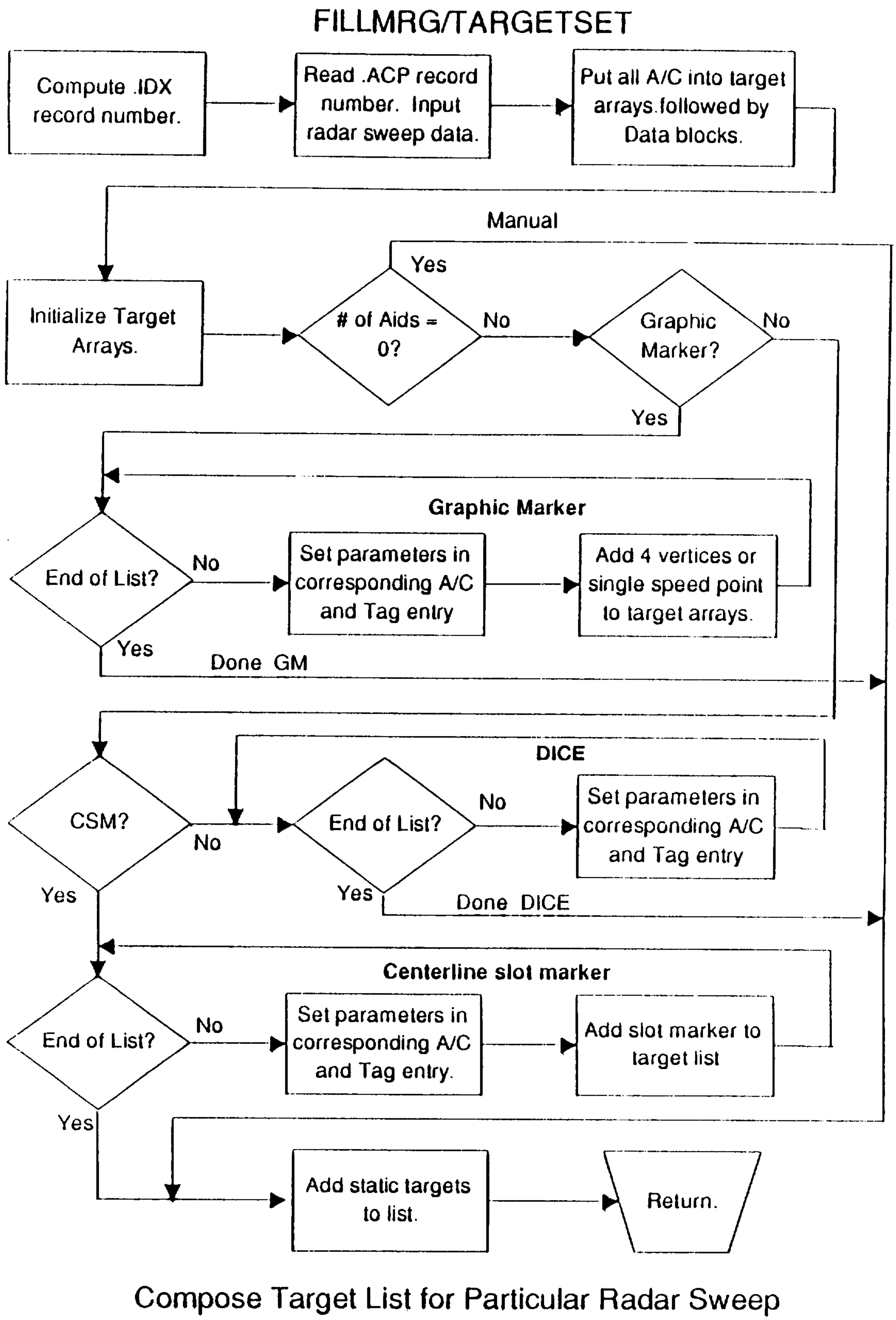


Figure 11. Flowchart of target tabulation program used in target identification.

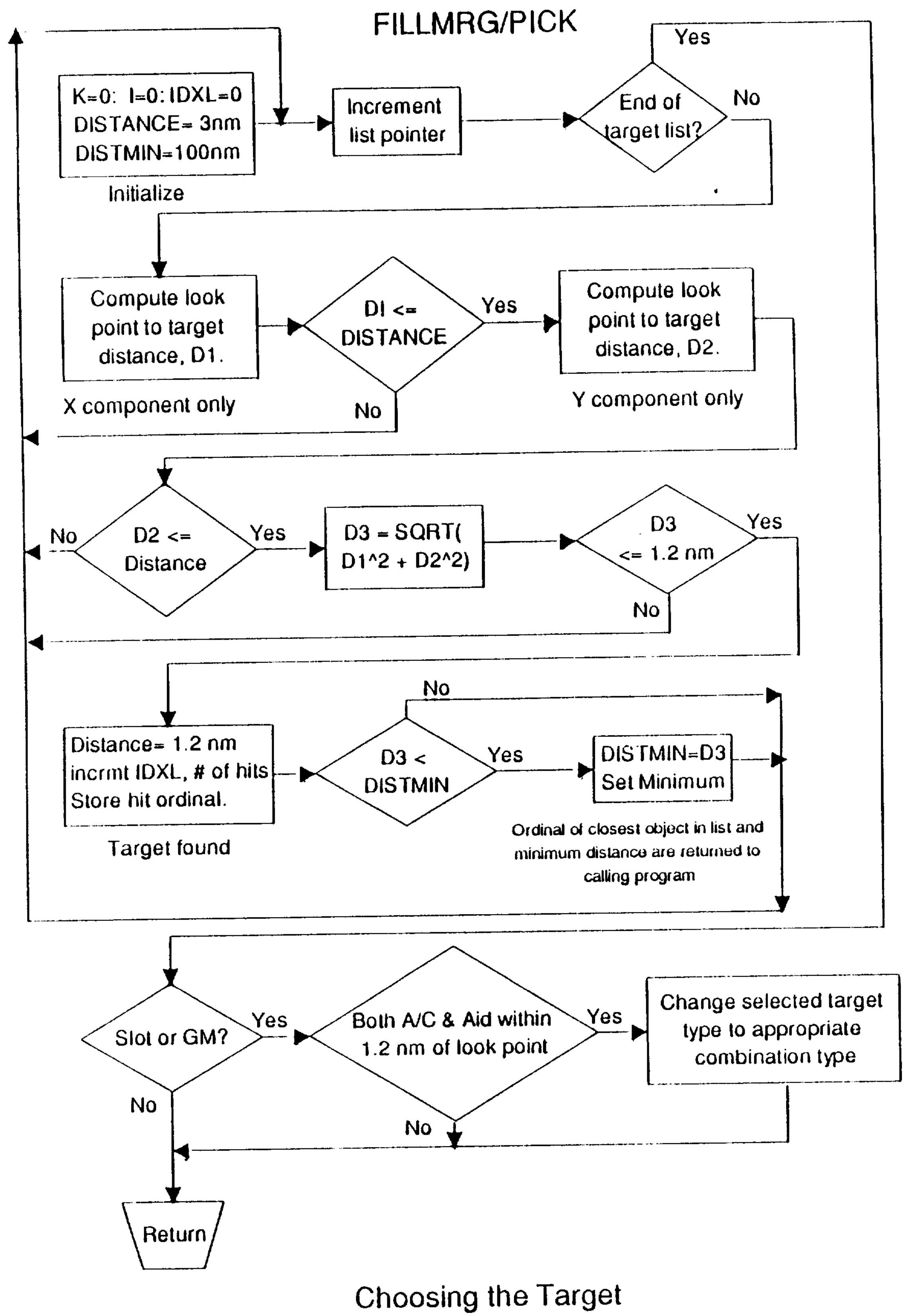


Figure 12. Flowchart of object and target association program used in target identification.

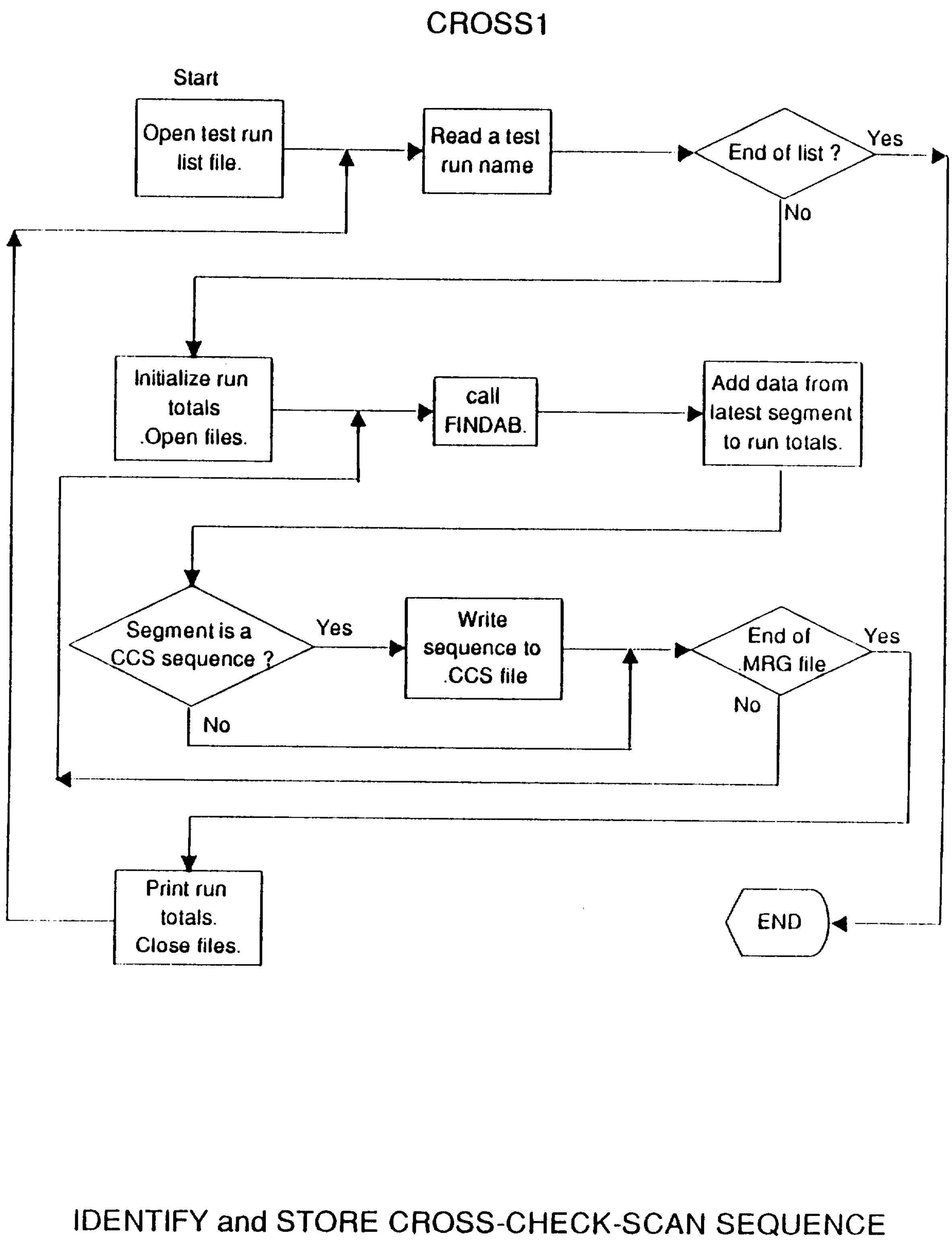


Figure 13. Flowchart of overall logic used in identifying and saving cross-check scan sequences.

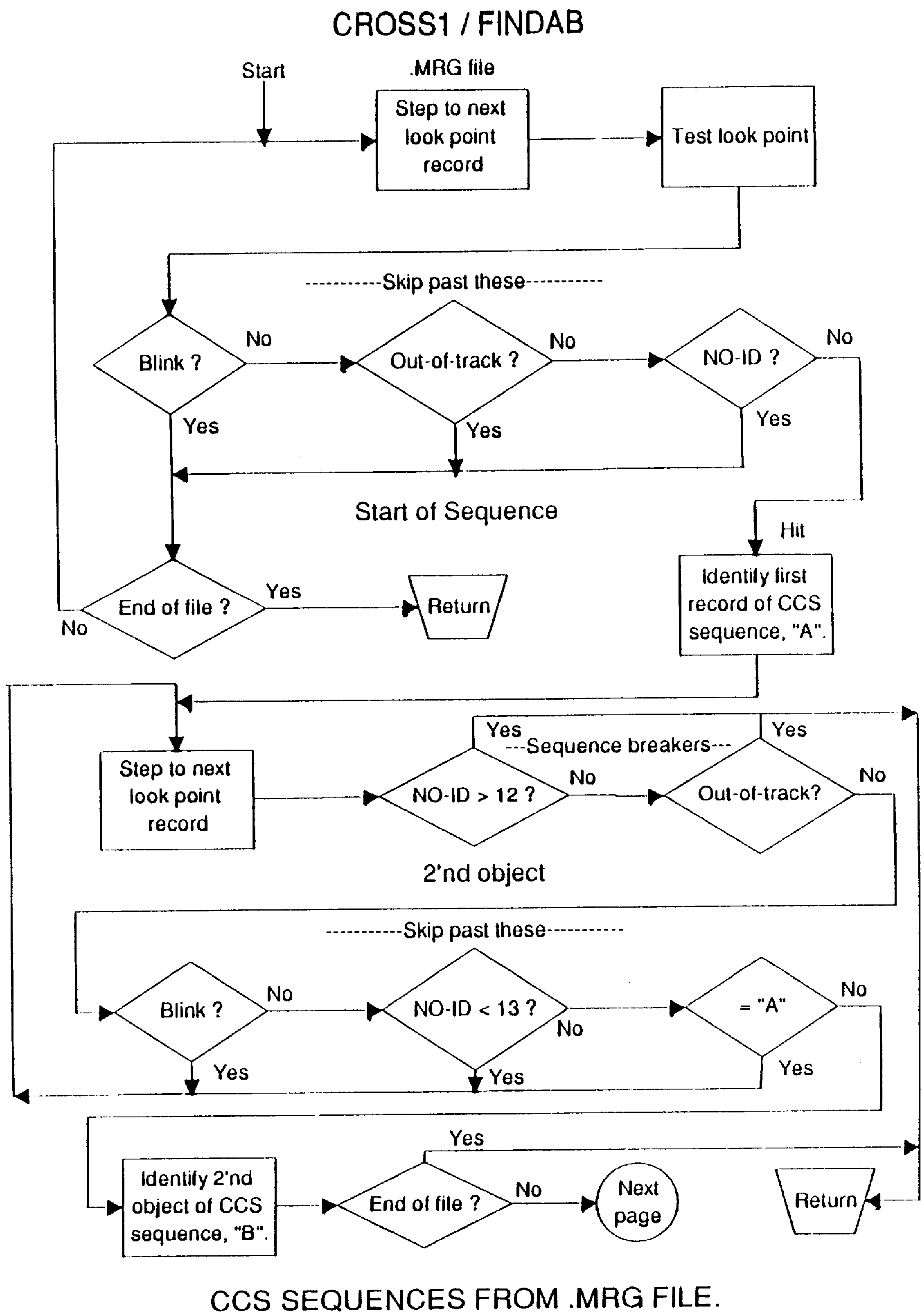
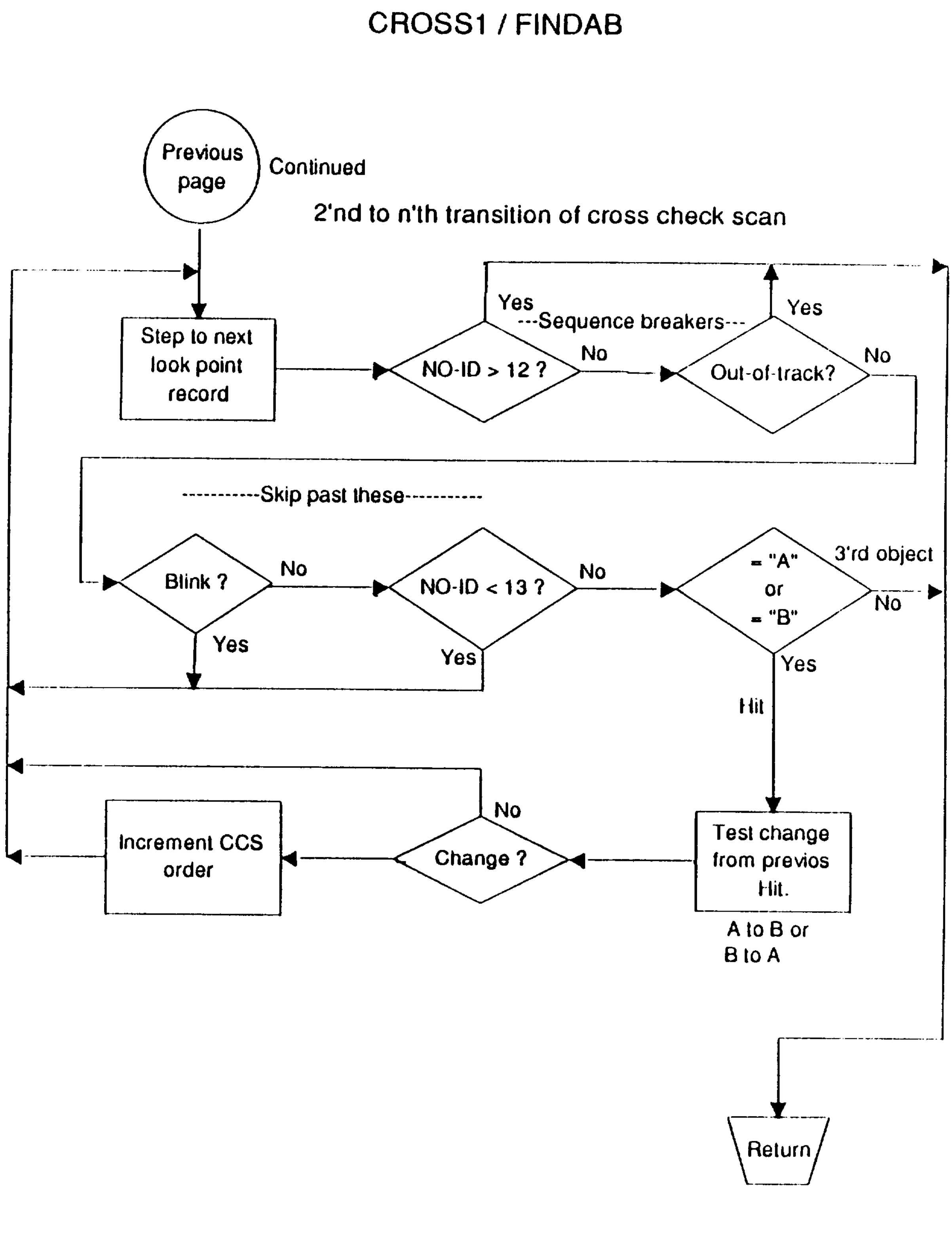


Figure 14. Flowchart of subprogram for determining the limits of a cross-check scan sequence.



CCS SEQUENCES FROM .MRG FILE (Continued).

Figure 14 (Continued).

a function of time and that at any time only a subset of all the objects were on the display.

While considering the transition matrix approach to the FASA unique display requirements, the team consulted with Dr. Randall Harris of the NASA Langley Research Center, an expert in eye scanning analysis. He suggested a simpler and arguably more powerful approach using cross check scans (CCS's). He had tested the approach with good results but had not had time to further explore it or publish his preliminary results. Cross check scanning was incorporated into the FASA study.

The working model formulated by the FASA group was that the controller performed cross-check scans to compare positions of aircraft to other aircraft as well as to geographical (or other significant) points on the display. The normal purpose of cross-checking is to either perform some control action or to monitor separation. The hypothesis was that a reduction in the number of cross-checks indicated a reduction in the amount of comparisons or judgments required to properly time a control action, assuming the amount of monitoring remained relatively constant.

For the purposes of the FASA study, a CCS is defined for no more than two objects, say A and B. Any two screen objects can be used in a particular CCS. A single scan transition from A to B (or B to A, the direction of the transition was not a factor) is termed a second order CCS. A cross check scan is an uninterrupted sequence of fixations alternating between object A and object B. Its order is one more than the number of transitions between the two objects. For example, the sequence A-A-B-A-A would be a CCS of order 3 (i.e., 2 transitions). The words uninterrupted sequence need some explanation. A third object will break the sequence. Thus, the sequence A-B-C-A would be considered three second order CCS sequences: A-B, B-C, and C-A. C breaks the A-B sequence and A breaks the B-C sequence. This example illustrates that a given fixation can belong to two CCS sequences, which complicates the computation of total time used in all CCS sequences. The other two types of records on the .MRG file that can interrupt a CCS are a long (>12/30 seconds) out-of-track or a long unassigned (No ID) fixation.

The program CROSS1 writes the .CCS file. A block diagram of the process is shown in figure B5 of Appendix B. The .MRG files were searched for CCS sequences and each record of the .CCS files describes one such sequence. The .CCS record itself is described in Appendix A. It contains the ordinals of the beginning and ending .MRG file records for the sequence. It also contains the number of transitions in the sequence and the total time (in 1/30 seconds). Other fields contain information on the two objects involved in the sequence. The CROSS1 source listing is given in Appendix C.

Figures 13 and 14 show the logic flow in CROSS1 for identifying and storing CCS sequences. The diagram for the subroutine FINDAB, (figure 14), presents the logic for identifying or defining the limits of the sequence. The diagram for CROSS1, (figure 13) shows the overall logic flow. The inner loop works its way through a single .MRG file. The outer loop goes through a list of test runs. In practice, the list usually contains the 12 subject controller runs for a given treatment.

FINDAB (figure 14) is composed of three serial stages. The first stage finds a suitable starting fixation, A. Only an end of file will cause this search to end without a hit. Neither an out-of-track nor a No ID is acceptable for A. The second stage searches for the second object of the pair. A long out-of-track or long No ID record will cause the search to end unsuccessfully. That is, a CCS sequence was not found. The search ignores short out-of-tracks and short No ID's. If and when a valid second object is found for the sequence, the subprogram proceeds to the third stage. At that point a CCS sequence of order 2 has been identified. The purpose of the third stage is to determine whether the sequence is of a higher order. The third stage (figure 14, continued) keeps track of transitions between A and B until it encounters one of three types of .MRG file records: a third object C, a long out-of-track, or a long No ID record. After exiting from the third stage the CCS sequence is recorded.

It would not be difficult to extend the logic to more than two objects corresponding to the more complex scan patterns observed to be used by the subject controllers. This was not done at this time, however, partially because of the difficulty in interpreting the results.

The data gathered on the .CCS files were statistically analyzed with respect to the number of higher order CCS sequences associated with the eight treatments. The data were broken down by type of objects in the pair and zone pairs. Reference 1 presents the zone pair results.

2.6 Display Zones

Earlier discussions focused on what the subject controllers were viewing. By defining different segments of the aircraft normal approach pattern, insight was gained as to where the subject was looking. Data were gathered on measures such as cross-check scans between zones and time spent within each zone. Figure 15 indicates the areas of the display associated with the zones. As implemented, the zone associated with an aircraft has to do with what part of the pattern it is executing rather than solely on its position. Thus, its heading gets into the calculation too. At the corners of the pattern, the heading distinguishes one zone from another depending on the progress of the turn. Also the base leg does not intersect the final at a fixed point but moves (trombones) in and out as required by the traffic. Data blocks and aids are assigned to the same zone as their corresponding aircraft except that centerline slot markers are always in zone 1. Because of the chosen implementation, some static objects such as navigational aids do not have an associated zone. As a consequence, the percentage of in-track time broken down by zones for a particular controller and type of aid do not sum to 100%. The time not included corresponds to objects that do not have an associated zone.

3.0 Description of Statistics

The eye scan portion of the FASA study described in references 1 and 2 relied, in part, on statistical analyses of the merge (.MRG) files and cross check scan (.CCS) files, both of which are discussed in the appendices and in sections 2.4 and 2.5 above. Some statistics were derived by custom programs such as A1HIST (figure B8), SEQNCE1 (figure B15) or SEQNCE2 (figure B16). Other statistics were gathered using a standard data base program. In both cases, the statistics were put into spreadsheet tables and plotted for reporting purposes.

From the beginning, the FASA study was designed to use a repeated measure analysis of variance (ANOVA) approach in order to establish the statistical significance of measured differences (in average values) as a function of the two factors of interest: display format and approach-pattern speed. The repeated measure ANOVA approach has the advantage of compensating for the wide variation among controllers. It was not unusual that the cross controller differences (within treatment) were larger than the cross treatment (within subject) differences. The ANOVA calls for all subjects to test on all treatments and, in effect, uses each subject's average performance as his own control. In the FASA study, several commercially available statistical analyses programs were used. The oculometer data analysis was done principally using the SYSTAT (SYSTAT Inc.) program, which was checked against other ANOVA programs. These programs compute the F statistic and probability (P) that the averages, measured for the various treatments, are equal. If the probability is lower than the preset level of significance, the hypothesis of equality is rejected, and the treatment is shown to have a significant effect. The ANOVA demonstrated that the differences associated with the different display formats were significant and not just statistical noise.

The form of the data tables (1 through 9) as presented in this report is the same as the form used to submit the data to the ANOVA programs. The number of rows was always twelve representing the number of controllers. The number of columns represented the number of treatments being compared. Most of the statistics computed for the FASA study used a 12X4 table for the 170 knot approach, a 12X4 table for the 210 knot approach, and the two combined in a 12X8 table to test the significance of the speed effect only. If the 12X4 ANOVA (170 or 210) showed significance ($P<0.05$), then the individual aids were contrasted against each other in a post hoc test using the Fisher PLSD method. The results of those tests were reported in reference 1. Tables 1, 4, and 7 have this straight forward 12X8 form. In table 3, four separate 12X8 tables were formed, one for each zone using both pattern speeds. Likewise in table 2, a 12x8 data block table was formed using the two 12X4 tables as shown. The target types Aid and Aid & A/C in table 2 had different dimensions (12X6 and 12X4 respectively), because the aid was not used in the manual case and the combination Aid & A/C was only defined for the graphic marker and centerline slot marker runs.

These statistical tables were used (although presented in a different form) in references 1 and 2 to support to support the FASA research. They are presented here for completeness and to illustrate the methodology.

Three measures of lookpoint behavior as functions of display format are included in table sets 1 to 9: in-track time, dwell time, and frequency of

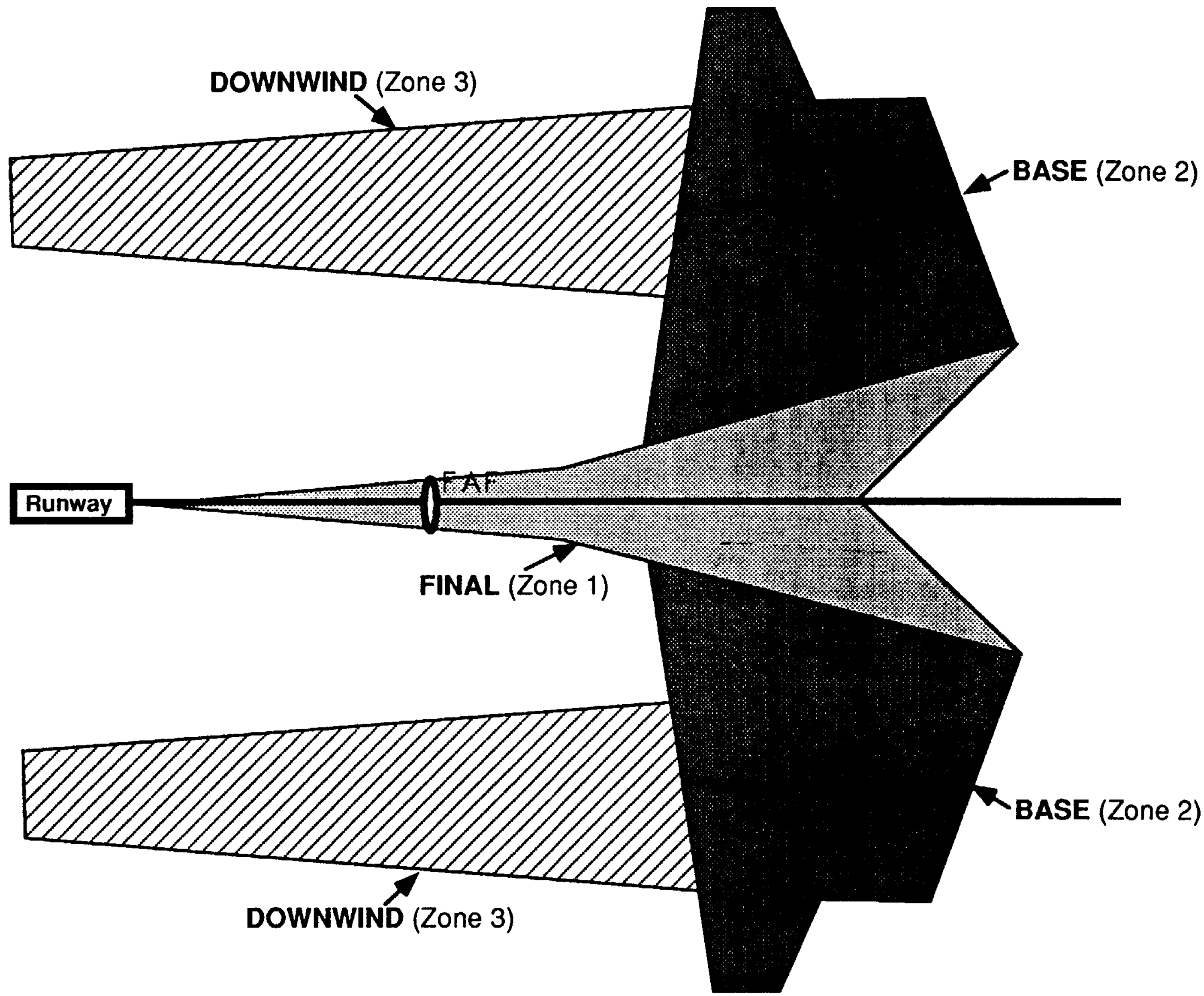


Figure 15. Final controller display zones of interest defined for the FASA study.

high-order cross-check scans. The first three table sets deal with the percentage of total time that the subject controller was being tracked by the oculometer, the total in-track time classified by types of gaze object (such as data block), and also by display zones.

Average dwell times for each controller are tabulated in tables 4 through 6. The totals in table 4 are classified by display object in table set 5, and by zones in table set 6. Tables 7 through 9 tally the number of high order cross check scans (of order 4 or greater) by controller. Section 2.5, above, has an extended explanation of cross check scans. Table 7 gives overall sums for a particular controller and treatment. In tables 8 and 9, the overall sums are classified by target pairs and zone pairs.

The three types of oculometer statistics (total time, dwell time, and frequency of cross check scans) were successfully used to discriminate between the tested display-format/pattern-speed combinations, and they generally agreed with and supported conclusions drawn from the other types of measurements made in the FASA study. The cross check scan analysis showed the most significance of the three. Other possibly interesting statistics that might be examined in future studies are the average duration of cross check scans by order and frequency of cross check scans defined on more than two (perhaps as many as five) distinct objects.

Controller Test Subject	In-Track as Percent of Total Time							
	170 Kts Procedure				210 Kts Procedure			
	MANUAL	GRAPHIC	DICE	CSM	MANUAL	GRAPHIC	DICE	CSM
1	75.6	72.7	76.7	75.2	66.5	73.4	80.6	68.4
2	84.9	83.6	81.8	88.0	85.6	81.9	79.0	93.1
3	88.7	83.7	83.9	86.4	84.7	84.2	87.9	80.9
4	84.4	81.2	79.1	82.3	67.5	80.2	79.7	78.0
5	88.3	81.6	88.5	88.9	91.3	87.1	88.6	87.3
6	83.2	55.6	59.3	67.3	82.5	69.0	77.2	82.9
7	94.1	84.1	92.7	92.6	90.7	90.9	89.3	93.9
8	90.1	91.6	94.3	90.4	95.3	93.2	93.6	95.0
9	87.6	87.4	88.7	86.4	91.3	89.2	89.1	91.2
10	87.2	83.6	86.8	85.6	87.3	90.4	87.6	91.1
11	75.9	75.3	72.2	80.7	84.8	83.4	75.2	78.2
12	80.7	83.9	81.6	90.3	89.5	86.2	86.4	91.0
Mean	85.0	80.3	82.1	84.5	84.8	84.1	84.5	85.9
St Dev	5.6	9.2	9.7	7.2	9.0	7.2	5.8	8.2

Table 1. Percentage of total test times that each subject was tracked by the oculometer in the FASA study.

Percent of In-Track Time by Targets, 170 Knot Procedure												
Test	Data Block				A/C				Aid			
Subject	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	41.7	40.2	38.6	42.1	20.7	16.5	27.1	17.1	0.0	11.7	18.2	9.6
2	45.1	31.0	32.9	36.0	39.2	33.2	32.4	19.1	0.0	7.8	21.0	10.8
3	52.1	35.5	37.3	36.7	37.8	23.6	34.2	25.3	0.0	12.2	17.7	9.2
4	44.5	39.8	29.8	39.9	36.5	17.0	36.8	24.6	0.0	12.9	17.4	6.3
5	34.8	36.5	44.9	34.8	42.9	22.9	18.4	28.6	0.0	12.7	28.5	6.1
6	57.1	37.3	32.5	42.9	29.2	24.1	33.2	22.2	0.0	8.6	17.4	8.3
7	50.1	46.0	30.4	46.2	31.1	18.3	36.0	14.6	0.0	9.3	19.9	11.4
8	53.1	38.4	42.3	44.0	36.2	26.0	26.8	20.8	0.0	8.1	24.6	8.5
9	35.2	25.2	16.4	16.2	44.8	29.2	57.2	33.2	0.0	12.8	7.9	8.5
10	52.9	45.8	36.5	46.9	36.2	20.5	30.6	23.9	0.0	9.0	22.2	8.2
11	30.4	34.3	30.6	29.1	46.0	21.9	32.5	24.8	0.0	11.3	17.3	10.0
12	51.2	43.3	40.2	39.5	27.0	15.1	21.6	18.5	0.0	13.2	23.0	11.0
Mean	45.7	37.8	34.4	37.8	35.6	22.4	32.2	22.7	0.0	10.8	19.6	9.0
St Dev	8.2	5.7	7.2	8.2	7.2	5.1	9.2	4.9	0.0	2.0	4.8	1.6

Percent of In-Track Time by Targets, 170 Knot Procedure (Cont.)												
Test	Aid & A/C				Other				Not Identified			
Subject	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	0.0	13.3	0.0	15.0	18.8	12.9	9.9	6.7	18.8	5.2	6.3	9.6
2	0.0	16.6	0.0	21.4	9.5	6.6	7.6	5.8	6.2	4.8	6.2	7.0
3	0.0	21.0	0.0	21.9	7.4	5.1	5.4	3.2	2.8	2.7	5.4	3.7
4	0.0	15.3	0.0	13.2	10.5	8.0	8.3	7.7	8.4	7.0	7.7	8.3
5	0.0	15.6	0.0	16.9	12.6	7.1	6.0	7.4	9.7	5.1	2.2	6.2
6	0.0	10.8	0.0	9.9	9.5	11.1	9.0	8.8	4.2	8.1	7.9	7.9
7	0.0	15.7	0.0	11.7	12.9	7.0	8.5	9.4	5.9	3.7	5.2	6.7
8	0.0	19.0	0.0	20.1	6.8	7.4	5.4	4.5	4.0	1.1	1.0	2.2
9	0.0	18.7	0.0	26.2	11.5	7.7	9.4	8.1	8.5	6.5	9.0	7.8
10	0.0	15.0	0.0	9.8	6.8	5.8	5.9	6.6	4.1	3.9	4.8	4.8
11	0.0	12.3	0.0	18.8	12.8	10.0	11.2	10.2	10.8	10.2	8.4	7.1
12	0.0	12.9	0.0	9.0	10.3	6.8	7.8	9.3	11.5	8.8	7.4	13.0
Mean	0.0	15.5	0.0	16.2	10.8	8.0	7.9	7.3	7.9	5.6	6.0	7.0
St Dev	0.0	2.9	0.0	5.4	3.2	2.2	1.8	2.0	4.3	2.5	2.3	2.7

Table 2a. Percentages of in-track time allocated to each type of gaze object identified for the FASA study, 170 knot approach-pattern-speed procedure.

Percent of In-Track Time by Targets, 210 Knot Procedure												
Test	Data Block				A/C				Aid			
Subject	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	51.1	34.6	27.3	39.7	24.4	15.5	31.7	13.2	0.0	17.1	25.8	13.1
2	57.9	22.5	31.5	30.3	27.1	17.4	25.1	22.2	0.0	16.0	28.3	11.3
3	29.2	37.8	33.5	25.8	55.3	15.6	25.5	27.6	0.0	14.2	24.7	8.3
4	44.6	33.8	29.9	49.3	24.9	12.7	24.3	15.4	0.0	9.7	25.9	8.3
5	41.7	21.9	22.6	26.4	38.6	30.3	33.2	29.8	0.0	14.6	36.3	9.6
6	60.4	40.6	29.2	44.2	28.2	14.0	30.1	20.3	0.0	15.7	29.1	9.1
7	45.5	39.1	18.8	40.6	32.4	15.0	35.5	22.9	0.0	15.2	32.1	7.4
8	54.1	31.6	29.7	32.1	36.0	21.2	24.2	31.5	0.0	12.6	39.2	8.8
9	46.6	19.3	16.5	23.1	38.4	18.3	45.5	25.2	0.0	23.5	21.6	10.0
10	48.7	42.6	28.9	32.6	36.0	18.0	28.3	27.2	0.0	12.3	35.2	12.0
11	39.5	29.4	16.3	23.5	32.5	25.0	49.5	23.0	0.0	12.9	18.5	9.2
12	54.6	40.1	28.5	34.0	23.5	13.5	26.0	19.5	0.0	14.5	32.1	15.0
Mean	47.8	32.8	26.1	33.5	33.1	18.0	31.6	23.1	0.0	14.9	29.1	10.2
St Dev	8.3	7.6	5.7	8.1	8.5	5.0	8.0	5.3	0.0	3.2	5.9	2.2

Percent of In-Track Time by targets, 210 Knot Procedure (Cont.)												
Test	Aid & A/C				Other				Not Identified			
Subject	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	0.0	18.2	0.0	13.0	12.0	8.4	8.1	7.5	12.5	6.2	7.1	13.5
2	0.0	26.7	0.0	24.1	9.9	10.5	8.1	4.3	5.1	6.8	7.0	7.9
3	0.0	19.8	0.0	27.8	11.0	8.5	8.5	6.3	4.5	4.2	7.8	4.2
4	0.0	12.0	0.0	9.9	13.4	10.7	9.0	6.2	17.1	21.2	10.9	11.0
5	0.0	25.7	0.0	23.6	13.2	5.6	5.5	4.3	6.6	1.9	2.5	6.3
6	0.0	15.5	0.0	9.9	8.1	7.7	7.8	9.4	3.3	6.5	3.7	7.2
7	0.0	20.4	0.0	14.5	15.9	7.6	8.6	8.1	6.2	2.8	5.0	6.5
8	0.0	29.1	0.0	22.2	7.9	4.7	6.1	3.4	2.0	0.7	0.9	2.1
9	0.0	5.0	0.0	30.7	10.6	21.6	10.3	6.0	4.4	12.4	6.2	5.1
10	0.0	20.5	0.0	18.1	11.5	4.8	5.4	5.9	3.8	1.9	2.2	4.2
11	0.0	21.2	0.0	21.2	15.4	7.9	9.4	10.0	12.7	3.6	6.3	13.0
12	0.0	19.3	0.0	15.6	11.1	6.6	7.1	5.6	10.8	6.1	6.4	10.2
Mean	0.0	19.4	0.0	19.2	11.7	8.7	7.8	6.4	7.4	6.2	5.5	7.6
St Dev	0.0	6.3	0.0	6.5	2.4	4.3	1.5	1.9	4.5	5.4	2.7	3.5

Table 2b. Percentages of in-track time allocated to each type of gaze object identified for the FASA study 210 knot approach-pattern-speed procedure.

Percent of In-Track Time by Zones, 170 Knot Procedure																
Test	ZONE 1				ZONE 2				ZONE 3				ZONE 4			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	52	45	46	62	15	29	29	23	8	12	13	3	3	7	5	2
2	55	51	42	57	26	28	37	28	11	12	13	5	4	4	4	2
3	56	42	45	65	29	39	32	23	7	13	14	7	5	4	2	2
4	52	48	43	50	30	32	34	29	7	10	13	10	2	2	3	3
5	48	45	49	50	31	36	35	33	10	12	10	8	2	2	5	2
6	58	47	35	50	26	27	35	28	8	13	19	12	5	4	3	2
7	56	42	40	59	23	38	39	25	9	12	12	7	5	5	4	2
8	49	45	51	63	30	34	32	21	13	13	11	11	4	7	5	3
9	49	42	42	61	27	33	29	22	10	14	16	5	3	5	5	2
10	47	41	38	51	25	35	38	29	19	16	15	12	3	5	3	3
11	49	46	46	60	28	27	25	20	9	11	14	8	4	5	7	4
12	40	38	43	57	33	36	33	20	12	15	13	8	4	2	4	2
Mean	51	44	43	57	27	33	33	25	10	13	14	8	4	4	4	2
St Dev	5	3	4	5	5	4	4	4	3	2	2	3	1	1	1	1

Percent of In-Track Time by Zones, 210 Knot Procedure																
Test	ZONE 1				ZONE 2				ZONE 3				ZONE 4			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	51	49	42	58	21	29	34	20	11	12	12	6	5	4	4	3
2	60	49	52	62	22	28	25	23	8	13	13	4	5	4	4	2
3	50	55	56	61	33	21	23	25	8	17	10	7	5	4	3	3
4	46	40	53	56	22	21	24	24	10	14	11	7	3	4	3	2
5	58	51	49	58	21	34	35	24	8	9	11	9	7	3	2	3
6	61	45	52	60	25	32	28	22	6	12	12	8	4	6	5	3
7	54	43	46	62	22	37	31	22	10	10	12	6	9	7	6	4
8	65	53	53	64	22	30	31	18	7	11	9	12	3	5	6	3
9	66	48	58	67	16	21	19	18	7	12	11	7	7	6	4	3
10	61	54	48	60	21	30	34	25	8	11	11	8	7	3	5	3
11	61	53	54	58	14	28	25	16	6	11	8	8	6	6	5	5
12	58	51	51	64	18	29	29	17	9	10	12	6	4	3	3	2
Mean	57	49	51	61	21	28	28	21	8	12	11	7	5	5	4	3
St Dev	6	4	4	3	5	5	5	3	2	2	1	2	2	1	1	1

Table 3. Percentages of in-track time allocated to each display zone defined for the FASA study.

Controller Test Subject	Average Fixation Time for Session (In Seconds)							
	170 Kts Procedure				210 Kts Procedure			
	MANUAL	GRAPHIC	DICE	CSM	MANUAL	GRAPHIC	DICE	CSM
1	0.66	0.64	0.67	0.74	0.61	0.70	0.95	0.75
2	0.81	0.83	0.86	0.91	0.80	0.87	0.90	1.07
3	0.68	0.73	0.72	0.73	0.72	0.77	0.92	0.87
4	0.56	0.52	0.50	0.56	0.47	0.39	0.54	0.69
5	0.71	0.73	0.86	0.82	0.78	0.77	0.86	0.81
6	0.45	0.40	0.40	0.49	0.51	0.61	0.65	0.66
7	0.84	0.79	0.78	0.87	0.81	0.82	0.91	0.94
8	0.96	1.21	1.17	1.04	0.99	1.24	1.20	1.23
9	0.67	0.70	0.65	0.84	0.68	0.70	0.72	0.81
10	0.71	0.65	0.73	0.64	0.73	0.73	0.87	0.73
11	0.44	0.49	0.50	0.55	0.48	0.58	0.54	0.50
12	0.65	0.59	0.69	0.56	0.56	0.61	0.73	0.64
Mean	0.68	0.69	0.71	0.73	0.68	0.73	0.82	0.81
St Dev	0.14	0.20	0.19	0.16	0.15	0.20	0.18	0.19

Table 4. Average length of lookpoint fixations for each subject

Fixation Time ByTargets-170 Knot Procedure												
Test	Data Block				A/C				Aid			
Subject	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	0.72	0.64	0.51	0.78	0.67	0.66	0.72	0.90	n/a	0.72	1.15	0.63
2	0.92	0.83	0.63	0.97	0.84	0.81	0.83	1.02	n/a	0.95	1.92	0.73
3	0.72	0.70	0.52	0.77	0.69	0.73	0.80	0.77	n/a	0.80	1.12	0.49
4	0.62	0.56	0.36	0.60	0.68	0.52	0.61	0.69	n/a	0.69	0.84	0.61
5	0.78	0.80	0.59	0.92	0.76	0.69	0.73	0.90	n/a	0.81	1.61	0.56
6	0.46	0.41	0.28	0.53	0.49	0.43	0.43	0.58	n/a	0.43	0.54	0.44
7	0.86	0.83	0.52	0.95	0.88	0.71	0.79	0.91	n/a	0.85	1.28	0.85
8	1.09	1.27	0.86	1.13	0.91	0.98	0.98	0.92	n/a	1.47	2.39	0.76
9	0.70	0.63	0.40	0.69	0.73	0.68	0.78	1.10	n/a	0.97	0.93	0.66
10	0.81	0.71	0.51	0.73	0.70	0.63	0.79	0.72	n/a	0.69	1.32	0.48
11	0.46	0.52	0.38	0.57	0.54	0.53	0.53	0.63	n/a	0.60	0.87	0.46
12	0.68	0.60	0.50	0.58	0.70	0.58	0.65	0.62	n/a	0.72	1.16	0.57
Mean	0.74	0.71	0.51	0.77	0.72	0.66	0.72	0.81	n/a	0.81	1.26	0.60
St Dev	0.17	0.21	0.14	0.18	0.12	0.14	0.14	0.16	n/a	0.24	0.49	0.12

Fixation Time ByTargets-210 Knot Procedure												
Test	Data Block				A/C				Aid			
Subject	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	0.69	0.73	0.56	0.79	0.66	0.75	1.09	0.85	n/a	0.72	1.73	0.76
2	0.92	0.83	0.56	1.08	0.74	0.75	0.92	1.12	n/a	0.93	1.35	0.73
3	0.73	0.82	0.58	0.84	0.79	0.74	0.92	0.89	n/a	0.84	1.29	0.59
4	0.53	1.11	0.36	0.79	0.56	0.43	0.60	0.76	n/a	0.42	0.82	0.63
5	0.82	0.80	0.40	0.85	0.83	0.81	0.78	0.85	n/a	0.69	1.45	0.54
6	0.53	0.63	0.36	0.72	0.57	0.67	0.69	0.75	n/a	0.66	0.94	0.54
7	0.87	0.85	0.40	1.05	0.85	0.73	0.93	1.03	n/a	0.89	1.45	0.69
8	1.09	1.24	0.61	1.27	0.93	0.97	0.95	1.27	n/a	1.26	1.96	0.73
9	0.67	0.64	0.33	0.64	0.82	0.73	0.81	0.94	n/a	0.90	1.01	0.62
10	0.82	0.80	0.45	0.79	0.74	0.72	0.86	0.83	n/a	0.64	1.35	0.57
11	0.51	0.60	0.28	0.50	0.55	0.55	0.60	0.60	n/a	0.61	0.74	0.41
12	0.59	0.61	0.37	0.61	0.59	0.60	0.75	0.74	n/a	0.62	0.98	0.61
Mean	0.73	0.81	0.44	0.83	0.72	0.70	0.83	0.89	n/a	0.76	1.26	0.62
St Dev	0.17	0.19	0.11	0.21	0.12	0.13	0.14	0.18	n/a	0.21	0.36	0.10

Table 5. Average length of lookpoint fixations associated with each type of gaze object.

Fixation Time ByTargets-170 Knot Procedure (Continued)								
Test	Aid & A/C				Not Identified			
Subject	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	n/a	0.86	n/a	1.15	0.58	0.34	0.36	0.47
2	n/a	1.35	n/a	1.12	0.46	0.43	0.48	0.66
3	n/a	1.05	n/a	1.22	0.30	0.32	0.42	0.31
4	n/a	0.87	n/a	0.67	0.30	0.24	0.25	0.32
5	n/a	0.90	n/a	0.95	0.48	0.44	0.39	0.55
6	n/a	0.52	n/a	0.61	0.26	0.26	0.26	0.32
7	n/a	1.18	n/a	0.89	0.58	0.38	0.40	0.58
8	n/a	1.82	n/a	1.52	0.55	0.34	0.33	0.49
9	n/a	1.22	n/a	1.24	0.43	0.39	0.42	0.45
10	n/a	0.91	n/a	0.79	0.34	0.29	0.35	0.33
11	n/a	0.70	n/a	0.71	0.26	0.29	0.29	0.31
12	n/a	0.81	n/a	0.76	0.50	0.39	0.46	0.44
Mean	n/a	1.02	n/a	0.97	0.42	0.34	0.37	0.44
St Dev	n/a	0.33	n/a	0.27	0.12	0.06	0.07	0.11

Fixation Time ByTargets-210 Knot Procedure (Continued)								
Test	Aid & A/C				Not Identified			
Subject	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	n/a	0.90	n/a	1.38	0.42	0.41	0.49	0.51
2	n/a	1.42	n/a	2.00	0.47	0.48	0.49	0.70
3	n/a	0.95	n/a	1.54	0.37	0.40	0.65	0.42
4	n/a	0.56	n/a	0.80	0.33	0.29	0.30	0.46
5	n/a	0.93	n/a	1.26	0.53	0.34	0.39	0.49
6	n/a	0.80	n/a	1.05	0.27	0.36	0.30	0.42
7	n/a	1.05	n/a	1.19	0.51	0.38	0.45	0.60
8	n/a	1.77	n/a	1.94	0.52	0.36	0.32	0.62
9	n/a	0.70	n/a	1.50	0.36	0.57	0.42	0.36
10	n/a	0.96	n/a	1.04	0.37	0.27	0.30	0.34
11	n/a	0.79	n/a	0.81	0.30	0.29	0.29	0.31
12	n/a	0.81	n/a	1.06	0.41	0.40	0.48	0.46
Mean	n/a	0.97	n/a	1.30	0.41	0.38	0.41	0.47
St Dev	n/a	0.32	n/a	0.38	0.08	0.08	0.11	0.11

Table 5 (Cont.). Average length of lookpoint fixations associated with each type of gaze object.

Fixation Time by Zones-170																
Test	ZONE 1				ZONE 2				ZONE 3				ZONE 4			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	0.70	0.65	0.62	0.81	0.70	0.70	0.87	0.85	0.55	0.67	0.77	0.47	0.78	0.80	0.76	0.63
2	0.85	0.78	0.75	0.93	0.92	1.11	1.27	1.07	0.72	0.95	0.89	0.68	0.74	0.73	0.72	0.80
3	0.72	0.68	0.67	0.82	0.70	0.88	0.90	0.74	0.60	0.78	0.80	0.57	0.78	0.66	0.62	0.53
4	0.58	0.53	0.46	0.56	0.75	0.68	0.75	0.75	0.48	0.51	0.50	0.54	0.43	0.49	0.40	0.51
5	0.73	0.74	0.76	0.80	0.86	0.82	1.07	0.99	0.61	0.70	1.02	0.74	0.56	0.56	0.91	0.72
6	0.47	0.41	0.41	0.53	0.46	0.45	0.45	0.54	0.42	0.38	0.39	0.44	0.49	0.43	0.38	0.46
7	0.89	0.73	0.66	0.90	0.92	0.99	1.07	1.03	0.67	0.76	0.77	0.69	0.81	0.73	0.69	0.69
8	0.95	1.09	1.03	1.09	1.05	1.54	1.60	1.05	0.91	1.21	1.30	0.92	1.39	1.23	1.19	1.16
9	0.76	0.66	0.59	0.97	0.68	0.91	0.90	0.91	0.57	0.71	0.72	0.57	0.62	0.64	0.57	0.57
10	0.70	0.64	0.66	0.66	0.84	0.79	0.99	0.75	0.76	0.66	0.76	0.58	0.66	0.52	0.59	0.66
11	0.45	0.51	0.49	0.59	0.58	0.62	0.70	0.60	0.39	0.46	0.50	0.49	0.42	0.52	0.50	0.58
12	0.63	0.58	0.62	0.61	0.75	0.68	0.92	0.61	0.63	0.62	0.74	0.45	0.70	0.52	0.61	0.46
Mean	0.70	0.67	0.64	0.77	0.77	0.85	0.96	0.82	0.61	0.70	0.76	0.60	0.70	0.65	0.66	0.65
St Dev	0.15	0.17	0.16	0.18	0.16	0.28	0.29	0.19	0.15	0.22	0.24	0.14	0.26	0.21	0.22	0.19

Fixation Time by Zones-210																
Test	ZONE 1				ZONE 2				ZONE 3				ZONE 4			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	0.64	0.75	0.90	0.84	0.70	0.74	1.35	0.82	0.56	0.69	1.01	0.54	0.68	0.77	0.71	0.77
2	0.84	0.93	0.93	1.25	0.87	0.98	1.12	1.02	0.68	0.86	0.90	0.64	0.93	0.73	0.78	0.73
3	0.76	0.80	0.91	0.94	0.79	0.84	1.07	0.94	0.64	0.80	1.02	0.63	0.67	0.67	0.76	0.75
4	0.53	0.43	0.60	0.72	0.56	0.50	0.70	0.87	0.44	0.39	0.51	0.61	0.41	0.38	0.39	0.56
5	0.82	0.81	0.79	0.88	0.80	0.80	1.09	0.88	0.64	0.74	0.94	0.64	0.95	0.56	0.58	0.75
6	0.53	0.69	0.68	0.74	0.55	0.64	0.73	0.66	0.45	0.54	0.66	0.54	0.56	0.58	0.61	0.58
7	0.81	0.82	0.86	1.00	0.94	0.92	1.29	1.04	0.81	0.76	0.91	0.70	0.90	0.76	0.80	0.80
8	1.02	1.26	1.15	1.32	1.06	1.28	1.44	1.21	0.81	1.23	1.13	1.03	1.06	1.22	1.18	1.25
9	0.73	0.71	0.73	0.95	0.69	0.82	0.87	0.77	0.56	0.70	0.81	0.62	0.73	0.71	0.59	0.66
10	0.75	0.77	0.83	0.77	0.83	0.78	1.13	0.82	0.69	0.73	0.87	0.66	0.84	0.67	0.70	0.64
11	0.54	0.59	0.56	0.58	0.48	0.68	0.65	0.54	0.39	0.55	0.53	0.41	0.60	0.55	0.48	0.55
12	0.60	0.64	0.71	0.71	0.58	0.64	0.87	0.62	0.49	0.56	0.72	0.49	0.57	0.57	0.68	0.60
Mean	0.71	0.77	0.80	0.89	0.74	0.80	1.03	0.85	0.60	0.71	0.83	0.63	0.74	0.68	0.69	0.72
St Dev	0.15	0.20	0.16	0.22	0.18	0.20	0.26	0.19	0.14	0.21	0.19	0.15	0.19	0.20	0.20	0.19

Table 6. Average length of lookpoint fixations associated with each display zone.

Test Subject	Number of Cross Check Scans of Order 4 or Greater By Subject							
	170 Kts Procedure				210 Kts Procedure			
	MAN	GM	DICE	CSM	MAN	GM	DICE	CSM
1	69	68	88	55	72	45	83	41
2	95	34	69	66	103	42	77	73
3	200	71	116	103	146	101	122	77
4	61	57	70	55	82	53	72	46
5	88	59	87	81	116	62	128	93
6	226	43	70	54	243	46	130	114
7	144	63	131	85	151	93	135	77
8	116	65	115	105	190	85	137	95
9	38	67	92	63	161	58	89	80
10	146	80	108	102	128	87	114	110
11	113	56	65	103	145	78	97	83
12	103	66	82	105	170	60	93	113
Mean	116.6	60.8	91.1	81.4	142.3	67.5	106.4	83.5
St Dev	55.2	12.4	21.8	21.8	47.3	20.4	23.8	23.6

Table 7. Total number of cross-check scans of order 4 or greater identified for each subject.

Number of CCS's of Order 4 or Greater by Target Pairs, 170 Knot Procedure												
Test	AIRCRAFT / AIRCRAFT				AC / OTHER TAG				TAG / TAG			
Subject	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	5	2	3	1	15	5	24	7	15	20	40	9
2	18	4	10	1	22	3	22	10	41	10	25	6
3	21	6	9	7	78	8	21	12	84	20	70	9
4	9	2	7	1	16	4	21	12	15	13	29	11
5	21	7	2	3	39	10	13	19	6	7	59	9
6	32	7	14	0	31	5	14	3	118	7	22	11
7	12	3	15	0	46	7	44	17	51	24	58	24
8	21	14	22	2	41	10	17	14	41	19	61	30
9	12	19	51	4	13	11	18	5	8	6	11	2
10	29	3	13	5	46	8	22	12	54	34	47	23
11	41	6	8	3	36	12	13	21	16	17	31	8
12	13	2	8	2	13	7	10	18	55	20	49	38
Mean	20	6	14	2	33	8	20	13	42	16	42	15
St Dev	10	5	12	2	18	3	8	5	32	8	18	11

Number of CCS's of Order 4 or Greater by Target Pairs, 210 Knot Procedure												
Test	AIRCRAFT / AIRCRAFT				AC / OTHER TAG				TAG / TAG			
Subject	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	5	1	8	0	21	2	18	3	29	13	38	8
2	6	2	4	0	32	0	18	7	43	2	49	2
3	70	5	4	5	33	14	28	3	25	21	69	15
4	8	1	6	0	7	1	12	8	30	11	18	15
5	15	8	19	5	56	9	28	15	16	8	57	2
6	29	1	17	2	37	1	19	9	150	16	72	27
7	23	2	27	6	48	5	33	15	38	19	61	19
8	42	16	22	4	50	7	20	9	73	7	82	7
9	16	2	16	3	38	1	26	5	75	2	29	10
10	17	4	9	5	39	11	11	8	48	16	77	10
11	17	9	37	2	56	14	28	9	37	11	14	2
12	8	2	4	0	29	3	19	6	106	17	59	9
Mean	21	4	14	3	37	6	22	8	56	12	52	11
St Dev	18	4	10	2	14	5	7	4	37	6	22	7

Table 8. Number of cross-check scans of order 4 or greater associated with pairs of gaze objects.

No. of CCS's of Order 4 or Greater by Target Pairs, 170 Knot Proc. (Cont.)												
Test Subject	TAG / AID				OTHER / AID				OTHER (8 PAIRS)			
	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17	MN17	GR17	DC17	SL17
1	n/a	21	n/a	22	n/a	8	n/a	5	34	12	21	11
2	n/a	7	n/a	23	n/a	5	n/a	20	14	5	12	6
3	n/a	15	n/a	31	n/a	16	n/a	32	17	6	16	12
4	n/a	12	n/a	9	n/a	10	n/a	7	21	16	13	15
5	n/a	14	n/a	17	n/a	19	n/a	15	22	2	13	18
6	n/a	10	n/a	13	n/a	2	n/a	13	45	12	20	14
7	n/a	17	n/a	15	n/a	7	n/a	10	35	5	14	19
8	n/a	6	n/a	22	n/a	10	n/a	20	13	6	15	17
9	n/a	9	n/a	6	n/a	20	n/a	30	5	2	12	16
10	n/a	17	n/a	17	n/a	6	n/a	14	17	12	26	31
11	n/a	7	n/a	19	n/a	6	n/a	37	20	8	13	15
12	n/a	17	n/a	11	n/a	9	n/a	10	22	11	15	26
Mean	n/a	13	n/a	17	n/a	10	n/a	18	22	8	16	17
St Dev	n/a	5	n/a	7	n/a	5	n/a	10	11	4	4	6

No. of CCS's of Order 4 or Greater by Target Pairs, 210 Knot Proc. (Cont.)												
Test Subject	TAG / AID				OTHER / AID				OTHER (8 PAIRS)			
	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21	MN21	GR21	DC21	SL21
1	n/a	15	n/a	22	n/a	6	n/a	3	17	8	19	5
2	n/a	9	n/a	30	n/a	17	n/a	25	22	12	6	9
3	n/a	29	n/a	9	n/a	13	n/a	36	18	19	21	9
4	n/a	19	n/a	10	n/a	3	n/a	0	37	18	36	13
5	n/a	6	n/a	23	n/a	25	n/a	37	29	6	24	11
6	n/a	18	n/a	29	n/a	6	n/a	19	27	4	22	28
7	n/a	33	n/a	18	n/a	24	n/a	5	42	10	14	14
8	n/a	18	n/a	20	n/a	29	n/a	45	25	8	13	10
9	n/a	14	n/a	18	n/a	13	n/a	31	32	26	18	13
10	n/a	26	n/a	28	n/a	13	n/a	40	24	17	17	19
11	n/a	9	n/a	23	n/a	19	n/a	31	35	16	18	16
12	n/a	15	n/a	60	n/a	9	n/a	22	27	14	11	16
Mean	n/a	18	n/a	24	n/a	15	n/a	25	28	13	18	14
St Dev	n/a	8	n/a	13	n/a	8	n/a	14	7	6	7	6

Table 8 (Cont). Number of cross-check scans of order 4 or greater associated with pairs of gaze objects.

Number of CCS's of Order 4 or Greater by Zone Pairs, 170 Knot Procedure																
Test	zone 1 / zone 1				zone 1 / zone 2				zone2 / zone 2				zone2 / zone 3			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	32	23	29	21	17	10	29	25	6	16	11	2	3	7	8	1
2	30	19	25	30	46	7	27	26	8	2	1	2	2	2	9	2
3	65	31	35	36	87	19	47	45	20	3	5	5	17	14	17	10
4	26	19	18	14	13	14	18	28	12	14	7	6	4	2	16	2
5	18	15	29	14	48	16	34	40	8	14	12	9	4	9	6	7
6	84	20	17	20	86	4	21	15	16	9	12	5	9	3	11	3
7	54	20	37	23	48	24	44	51	4	0	6	2	9	14	28	0
8	29	28	55	33	50	15	29	48	5	1	6	5	12	13	14	2
9	5	27	26	21	20	20	27	27	2	4	6	5	7	10	11	0
10	43	30	42	18	44	19	29	41	5	10	12	10	12	10	8	7
11	40	32	21	34	48	14	23	40	4	4	3	1	9	1	6	6
12	23	16	26	37	53	21	30	39	7	15	7	6	9	5	10	4
Mean	37	23	30	25	47	15	30	35	8	8	7	5	8	8	12	4
St Dev	22	6	11	8	23	6	9	11	5	6	4	3	4	5	6	3

Number of CCS's of Order 4 or Greater by Zone Pairs, 210 Knot Procedure																
Test	zone 1 / zone 1				zone 1 / zone 2				zone2 / zone 2				zone2 / zone 3			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	25	17	17	21	29	12	33	16	4	2	7	1	5	1	11	3
2	50	10	32	21	34	12	28	44	4	8	1	6	5	4	7	0
3	33	32	48	23	73	27	45	34	9	7	5	7	15	4	5	5
4	33	20	25	8	21	12	17	19	5	9	9	3	7	3	7	1
5	39	14	38	21	48	25	58	48	7	14	6	4	2	3	15	3
6	99	12	43	25	104	12	41	58	9	13	10	5	14	5	19	6
7	50	15	24	30	45	43	60	32	8	9	4	3	10	13	23	4
8	85	27	46	26	72	23	47	37	4	9	5	0	3	10	23	0
9	87	15	31	32	39	6	39	25	5	12	1	1	8	9	3	4
10	55	38	37	40	48	15	45	43	6	22	6	7	4	1	14	1
11	70	31	31	35	51	13	42	27	5	8	9	3	2	8	2	1
12	81	25	21	49	60	8	50	50	5	15	4	2	6	3	6	0
Mean	59	21	33	28	52	17	42	36	6	11	6	4	7	5	11	2
St Dev	25	9	10	11	23	10	12	13	2	5	3	2	4	4	7	2

Table 9. Number of cross-check scans of order 4 or greater associated with pairs of display zones.

No. of CCS's of Order 4 or Greater by Zone Pairs, 170 Knot Proc. (Cont)												
Test	zone1 / zone 3				zone1 / zone 4				Other			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	4	7	8	4	3	4	1	0	4	1	2	2
2	7	0	1	4	2	0	1	0	0	4	5	2
3	3	2	9	5	7	0	2	2	1	2	1	0
4	5	4	6	4	1	1	2	0	0	3	3	1
5	6	3	3	7	1	0	0	2	3	2	3	2
6	12	1	3	9	15	2	2	1	4	4	4	1
7	11	3	9	8	15	1	5	0	3	1	2	1
8	18	2	2	13	1	5	9	3	1	1	0	1
9	1	4	11	9	3	1	2	1	0	1	9	0
10	34	5	10	17	0	4	1	1	8	2	6	8
11	6	2	5	16	4	2	1	5	2	1	6	1
12	9	2	2	13	1	2	1	4	1	5	6	2
Mean	10	3	6	9	4	2	2	2	2	2	4	2
St Dev	9	2	4	5	5	2	2	2	2	1	3	2

No. of CCS's of Order 4 or Greater by Zone Pairs, 210 Knot Proc. (Cont)												
Test	zone1 / zone 3				zone1 / zone 4				Other			
Subject	MN	GR	DC	SL	MN	GR	DC	SL	MN	GR	DC	SL
1	6	7	10	0	3	3	2	0	0	3	3	0
2	6	4	7	2	4	2	1	0	0	2	1	0
3	5	17	15	6	6	9	2	1	5	5	2	1
4	7	7	9	12	5	1	2	2	4	1	3	1
5	11	3	9	13	9	3	2	2	0	0	0	2
6	9	1	5	16	4	2	7	3	4	1	5	1
7	14	5	15	7	21	4	4	1	3	4	5	0
8	22	6	9	26	2	7	7	1	2	3	0	5
9	9	10	9	9	13	4	4	6	0	2	2	3
10	8	5	6	15	5	2	3	4	2	4	3	0
11	5	4	4	10	10	11	6	3	2	3	3	4
12	9	3	7	8	3	2	1	2	6	4	4	2
Mean	9	6	9	10	7	4	3	2	2	3	3	2
St Dev	5	4	3	7	5	3	2	2	2	1	2	2

Table 9 (Cont). Number of cross-check scans of order 4 or greater associated with pairs of display zones.

4.0 Major Results and Concluding Remarks

In the FASA study, as should be the case for any experiment, careful a priori consideration was given to defining the performance measures to be used with the oculometer, the steps taken to insure data integrity (e,g., calibration and quick look), and how the data were to be recorded and analyzed. Because of system developments (hardware and software) described in this paper, the oculometer system was successfully applied to the simulated interaction between an air traffic controller and the plan view radar display. The eye scan data were used along with other measurements to evaluate the relative merits of several proposed display modifications as described in references 1 and 2. Methods were developed and implemented to increase resolution and to maintain alignment accuracy. Algorithms were developed to synchronize the oculometer data to the time history data, to filter the data, to identify the target of each individual fixation, and to identify cross check scan sequences. Some of the measurement techniques, especially with respect to cross-check scans and display zones, have not been described in previous papers. Detailed tables have been provided, which show measurements averaged over each test run, and are in the form used in ANOVA testing for significance. These tables clearly illustrate the diversity among controllers and the consistency for any given controller across experimental treatments. The ANOVA tests reported in references 1 and 2 clearly affirmed the significance of differences measured between display formats. The recorded data (.SCN, .DAT, and .ACP) files have been preserved for further analysis, if needed. The .MRG and .CCS files which resulted from the procedures described in this paper, are also available. The flow diagrams, source code, block diagrams, and file record descriptors provided in this report should ensure that the technology can be extended to future air traffic studies and that, if needed, the FASA oculometer data can be further analyzed.

References

1.) Credeur, L., Capron, W.R., Lohr, G.W., Crawford, D.J., Tang, D.A., and Rodgers, W.G., Jr. (1993), *Final Approach Spacing Aids (FASA) Evaluation For Terminal-Area, Time-Based Air Traffic Control.* NASA TP-3399, National Aeronautics and Space Administration, Langley Research Center, Hampton, VA.

2.) Credeur, L., Capron, W.R., Lohr, G.W., Crawford, D.J., Tang, D.A., and Rodgers, W.G., Jr. (July-September, 1993), "A Comparison of Final Approach Spacing Aids for Terminal ATC Automation," Air Traffic Control Quarterly, Vol 1(2) 135-178.

3.) Harris, R.L., Jr., Glover, B.J., and Spady, A.A., Jr., (1986), *Analytical Techniques of Pilot Scanning Behavior and Their Application.* NASA TP-2525, National Aeronautics and Space Administration, Langley Research Center, Hampton, VA.

4.) Kaylor, Jack T.; Simmons, Harold I.; Naftel, Patricia B.; Houck, Jacob A.; and Grove, Randall D. (1985), *The Mission Oriented Terminal Area Simulation Facility.* NASA TM-87621, National Aeronautics and Space Administration, Langley Research Center, Hampton, VA.

Appendix A

Data File Record Descriptors

Cross Check Scan File Record (.CCS)

This is an ASCII type file with 103 bytes per record. The last two bytes are a carriage-return and a line-feed. A segment of a .CCS file is shown below:

```
 988  2182 2183  1  15 10  TAG  A/C   432 433   25  1 2  0 1  0 0   25  1.2       260   0   7
 989  2183 2184  1  10 15  A/C  TAG   433 433   40  2 2  1 1  0 0   40  0.7  260 260   7   7
 990  2184 2187  2  15 15  TAG  TAG   433 432   81  2 1  1 0  0 0   26  1.9  260       7   0
 991  2194 2196  2  15 15  TAG  TAG   433 432  153  2 1  1 0  0 0   57  1.8  260       2   0
 992  2196 2199  2  15 10  TAG  A/C   433 501  285  2 3  1 0  0 0  257  4.9  260       2   0
 993  2199 2201  1  15 10  TAG  A/C   433 433  232  2 1  1 0  0 0  226  1.0  260       2   0
 994  2201 2206  3  10 15  A/C  TAG   433 432  155  1 1  0 0  0 0   34  1.2            0   0
 995  2206 2207  1  15 10  TAG  A/C   432 432   55  1 1  0 0  0 0   55  0.7            0   0
 996  2207 2208  1  10 15  A/C  TAG   432 433   79  1 1  0 0  0 0   79  2.5            0   0
 997  2208 2209  1  15 10  TAG  A/C   433 421   67  1 2  0 0  0 0   67  3.3            0   0
 998  2209 2210  1  10 10  A/C  A/C   421 501   54  2 3  0 1  0 0   35  2.2       345   0  50
 999  2212 2213  1  10 53  A/C  LINE  501 SDW   75  3 3  1 0  0 0   14  2.4  345      43   0
1000  2213 2214  1  53 15  LINE TAG   SDW 501   19  3 3  0 1  0 0   19  1.9       345   0  43
1001  2214 2215  1  15 10  TAG  A/C   501 501   43  3 3  1 1  0 0   43  0.7  345 345  43  43
1002  2215 2219  4  10 53  A/C  LINE  501 SDW  101  3 3  1 0  0 0   61  2.4  345      43   0
1003  2219 2225  1  10 15  A/C  TAG   501 433   74  3 1  1 1  0 1   23  3.7  345 170  36  -5
1004  2227 2228  1  51 15  LINE TAG   FNL 432   15  1 1  0 0  0 0   11  2.8            0   0
1005  2228 2229  1  15 15  TAG  TAG   432 433   32  1 1  0 1  0 1   11  1.8       170   0  -6
1006  2232 2234  2  15 15  TAG  TAG   433 432  111  1 1  1 0  1 0   26  1.8  170      -7   0
1007  2234 2241  4  15 51  TAG  LINE  433 FNL  213  1 1  1 0  1 0   64  1.4  170      -7   0
1008  2241 2242  1  15 15  TAG  TAG   433 432   50  1 1  0 0  0 0   50  1.7            0   0
```

The first field is a record number or sequence number for the cross check scans. The next two fields are record numbers on the merge file (.MRG) and represent the first and last record of the sequence. In the segment shown, cross-check-scan sequence number 1002 started with merge file record number 2215 and stopped on merge file record number 2219. The next .CCS field gives the number of transitions between the two objects. This number is one less than the order of the scan sequence. Thus, cross check scan sequence 1002 had 4 transitions and was therefore of order five. Fields 5 and 6 are numerical object identifiers (as defined in Table C1) and fields 7 and 8 are mnemonic object identifiers. The ninth and tenth fields are either partial flight numbers for aircraft or mnemonics such as SDW for south down wind or FNL for final. The next field (11) is the length of the sequence in sample periods; to get seconds, divide this number by 30. The next 2 fields (12 & 13) give the zones for the two objects. Note that in sequence 1000, both the data block on flight 501 and the SDW line are in zone 3. Fields 14 & 15 indicate whether the aids are displayed or not and fields 16 & 17 refer to the speed markers. Anytime a marker is on (notice 1003 and 1005-1007) the corresponding aid is on.

The next field (18) contains the amount of time included in a given sequence, which is also included in either the preceding or following sequence. To elaborate, consider the sequence of 3 objects A to B to C. In this study, a sequence was constrained to 2 objects. Therefore the sequence would be considered to be two sequences, A to B and B to C. The dwell time associated with B would be included in both sequences. As a result, when the total time for all sequences is summed, it is considerably larger than the total run time for the test. Field 18 was used to keep track of total time and overlapping time. Field 19 is the display distance in inches between the two objects. The next 4 fields (2 field pairs) are used for DICE format only. The first pair gives recommended headings (or recommended speeds) as and when they appear in the data tag. The second pair, the last two fields, includes the current DICE countdowns.

The format of the record is more precisely defined in subroutine READSEQ which appears both in programs SEQNCE1 AND SEQNCE2. It can also be seen in subroutine PRINTSEQ in the program CROSS1.

Merge File Record (.MRG)

The .MRG file is an unformatted ASCII file with variable record size. Fields are separated by commas, and each record ends with a carriage-return/line-feed. There are 19 fields in the record, three of which (spares) were not used. Because of its form, it would be difficult to show a segment of the file. Fields do not line up neatly in columns. Therefore a segment of a .PT1 file is shown; it is a .MRG file formatted using the PRNMRG program.

THE FOLLOWING DATA IS FROM THE FILE, c:\fasa\brennan\mb06dc21.mrg

The number of records on the file is : 4832

PRINT FROM RECORD # 1000 to 1020

Rec#	Tp	Typ	Fxt	PD	TgID	Dist	FrNo	TgtX	TgtY	FixX	FixY	Hdg	CD					
1000	15	TAG	13	440	631	0.17	221	-0.83	0.27	-0.67	0.25		0	Z	Z	1	0	0
1001	15	TAG	13	428	309	0.06	221	-2.54	-0.10	-2.60	-0.11		0	Z	Z	1	0	0
1002	15	TAG	9	440	309	0.07	221	-2.54	-0.10	-2.52	-0.16		0	Z	Z	1	0	0
1003	15	TAG	129	449	508	0.53	221	1.19	0.15	1.71	0.06	170	0	Z	Z	1	1	1
1004	80	BLNK	8	10	Jil	99.99	222	0.00	0.00	0.00	0.00		0	Z	Z	9	0	0
1005	53	LINE	22	439	SDW	0.10	222	2.69	-2.38	2.67	-2.29		0	Z	Z	3	0	0
1006	0	UNK	5	422	Jil	99.99	222	0.00	0.00	4.18	-4.65		0	Z	Z	9	0	0
1007	0	UNK	11	433	Jil	99.99	223	0.00	0.00	3.65	-4.58		0	Z	Z	9	0	0
1008	89	OUT	23	10	Jil	99.99	223	0.00	0.00	0.00	0.00		0	Z	Z	9	0	0
1009	15	TAG	66	441	508	0.43	223	0.96	0.19	1.35	0.02		0	Z	Z	1	0	0
1010	15	TAG	17	432	631	0.23	223	-1.01	0.22	-0.88	0.03		0	Z	Z	1	0	0
1011	15	TAG	18	433	309	0.29	224	-2.76	-0.14	-2.47	-0.17		0	Z	Z	1	0	0
1012	15	TAG	26	441	508	0.29	224	0.84	0.20	1.05	0.00		0	Z	Z	1	0	0
1013	10	A/C	36	447	970	0.43	224	1.97	-1.80	2.39	-1.88	280	10	Z	Z	2	0	1
1014	15	TAG	14	435	508	0.62	224	0.84	0.20	1.36	-0.14		0	Z	Z	1	0	0
1015	15	TAG	19	435	631	0.31	224	-1.10	0.20	-1.33	-0.01		0	Z	Z	1	0	0
1016	15	TAG	8	429	309	0.45	225	-2.84	-0.15	-2.84	-0.60		0	Z	Z	1	0	0
1017	80	BLNK	7	10	Jil	99.99	225	0.00	0.00	0.00	0.00		0	Z	Z	9	0	0
1018	53	LINE	50	436	SDW	0.45	225	2.68	-2.38	2.59	-1.94		0	Z	Z	3	0	0
1019	15	TAG	34	436	508	0.67	225	0.74	0.21	1.14	-0.32		0	Z	Z	1	0	0
1020	10	A/C	73	433	970	0.44	225	1.94	-1.67	2.38	-1.72	280	7	Z	Z	2	0	1

The first column is a record number and is not a field in the file. The first and second fields are the numerical and mnemonic object identifier as defined in table C1. The next field (3) is the length of the sequence in sample periods; to get seconds, divide this number by 30. The fourth field is pupil diameter in analog to digital converter counts; to get millimeters, multiply this by (25.4/2048). Record 1003 has a pupil diameter of 5.6 mm (449 counts). The fifth field is either a partial flight number for aircraft or a mnemonic such as SDW for south down wind or FNL for final. The mnemonic Jil in this field and the number 99.99 in the next field are examples of presets which have not been overwritten. They are used for checking the algorithms and should, for the most part, be ignored. Note that their occurrence corresponds to out-of-track records including blinks and also to in-track records where a gaze object could not be found (UNK in field 2) within the allowable 0.57 inches. Field 6 is the distance on the screen in inches between the look point and gaze object, when one is identified. The next field (7) is a pointer to the aircraft position file (.ACP) and a time stamp. It was computed by subroutine FIXPOINTER in program FIXPOINT. It gives the record number of the .ACP file that was searched for targets by subroutine SEARCH in program FILLMRG. To get run time in seconds at the beginning of a simulation update subtract one from the value in field 7 and multiply by 4, the simulation update rate. For example, .ACP record 224 referenced in .MRG record 1011 (shown above) started 892 seconds into the run.

The next four fields (8-11) are the real x,y screen coordinates of the object first and then the look point. Fields 12 and 13 are for DICE format only. Field 12 gives the recommended heading (or recommended speed) only when it appears in the displayed data block associated with the gaze object. With the same constraints, field 13 contains the current DICE countdown. Fields 14 and 15 are not used and field 16 is the zone number as supplied for the aircraft by the .ACP file. A "1" in 17 indicates the speed marker is active and a "1" in field 18 indicates that the aid is on. Field 19 is not used and is not shown above.

The array type FIXCOMB, defined in several programs including CRE8MRG1, FILLMRG, et. al., best defines the fields on this record. The subprogram PUTXX in CRE8MRG1 writes the array to the .MRG file. The common utility subroutine, GETXXA uses the FIXCOMB data structure to read one record of the file. GETXXA is called in a loop: In subroutine FILLBUF in program CROSS1, in subroutine SEARCH in program BEANCNT1, in subroutine SEARCH in program FILLMRG, and elsewhere.

Time History File Record Group (.ACP)

This file is unusual in that it has four record types in a group, the last of which types changes form with run format and pattern speed. A group of records describes the state of the controller's display during one four-second simulation interval. This file was searched (by FILLMRG) to determine what the controller was looking at for every recorded fixation. Although having a more complex structure than the other files, the .ACP file is a formatted ASCII file with each line terminated by a carriage return/line feed sequence. Two record groups are shown below. The first is a DICE display format, 210-knot approach-pattern-speed run. The second is a graphic marker format, 210 pattern-speed run. The two formats are presented to illustrate the difference in fourth record type, and the 210 pattern speed was selected to show how the speed change advisory was distinguished from the turn advisory.

```
 10 14752.12
647   -0.27    2.07     0.73    3.26   3   1
320   -0.40    5.06     0.60    6.25   4   1
674    0.37   11.68     1.37   12.87   4   1
253   -6.18    9.87    -5.18   11.06   3   3
530    4.16   13.81     5.16   15.00   1   2
929    5.86   -1.91     6.86   -0.72   4   4
783   10.01   -6.25    11.01   -5.06   4   4
856  -13.40   18.08   -12.40   19.27   2   2
943   21.53   31.91    22.53   33.10   1   4
181   34.66  -22.96    35.66  -21.77   4   4
    0     0     3
  674  S170    -1
  253   345   + 47
  530   240   + 15
```

```
 12   7492.12
536   -0.35    1.32     0.65    2.51   1   1
804   -0.25    3.62     0.75    4.81   2   1
179   -0.28    7.41     0.72    8.60   2   1
725    1.24   11.41     2.24   12.60   4   1
700   -3.53   14.28    -2.53   15.47   2   2
996    4.91   10.10     5.91   11.29   4   3
411  -12.33   16.98   -11.33   18.17   2   2
215   -6.38    0.65    -5.38    1.84   3   4
920   -6.21   -6.76    -5.21   -5.57   3   4
631   26.49  -17.17    27.49  -15.98   4   4
836   20.14   29.79    21.14   30.98   1   4
309   23.60   36.11    24.60   37.30   1   4
    3     0     0
  700     -3.80    14.30    -2.30    14.25    -1.65    13.78    -1.52    13.40
  725      1.10    10.30    99.99    99.99    99.99    99.99    99.99    99.99
  996      4.90    11.90     4.90    13.70     3.95    14.59     3.35    14.53
```

The first record of the group tells how many aircraft were on the screen during the interval. It also contains time in seconds, which is related to the traffic sample. This is not time since the beginning of the run, but rather since the beginning of the traffic sample. This time was not used in connection with the oculometer analysis. In the examples shown, there are 10 and 12 aircraft respectively. In the next group of records (starting with record 2), there is one line (or record) for each aircraft. Each record contains

- the flight identification number,
- the x and y coordinates of the aircraft given in the simulation frame of reference,
- the x and y coordinates of the data block given in the simulation frame of reference,
- a route number, and
- a zone number.

The route number indicates corner the post from which the aircraft entered the pattern. Routes 1, 2, 3, and 4 correspond to the NE, SE, SW, and NW corner posts. The zone numbers are functional as well as area indicators. Zones 1, 2, 3, and 4 correspond to final approach course, base leg, downwind leg, and everything else. To illustrate, a route/zone combination of 3/3 would indicate that the aircraft was on the south downwind coming from the west. These aircraft descriptors (the second type of record) always have the same form.

The next record type, a single record, contains three integers. The numbers indicate how many aids were active on the display during the particular interval for the graphic marker, slot marker, and DICE, respectively. In the first example above, the numbers are 0,0,3; in the second they are 3,0,0. Thus, the examples are from a DICE run and a graphic marker run. This single record is followed by that number of aid descriptors, three in each example above. For the manual format this record always contains the numbers 0,0,0 and always ends the group. For this study, no more than one type of aid was used in a run, but that was not a constraint of the system. The aid descriptors are different depending on the type of aid. For the 210 knot pattern speed graphic marker and DICE runs, the aid descriptors distinguished between turn indicators and speed indicators.

The aid descriptor for the DICE has three forms, two of which are shown in the first example above. As shown for flight 253 and 530 in the example, the descriptor gives the flight number followed by the suggested heading and the DICE countdown value. If the latter is negative the aircraft has gone beyond the recommended turn point and will be late arriving unless the controller intervenes to make up the time. Sometimes, the countdown was shown but not the heading. This occurred only when the countdown was greater than 60 seconds. The third form is illustrated with flight 674 in the first example shown above. This is the speed change advisory where the flight number is followed by the suggested speed (prefixed with the letter S) and the countdown value. For this study, the suggested speed was always 170 knots. Nominally, the controller issued the clearances when the DICE countdown values went to zero.

For the graphic marker nine numbers were furnished. The first is the flight number just as in the aircraft descriptor records. This is followed by four pairs of position coordinates (x,y) given in the simulation frame of reference. The graphic marker is three connected straight line segments. These four pairs of coordinates specify the positions of the graphic marker's vertices. Flights 700 and 996 in the second example above demonstrate normal graphic marker descriptor layout. The graphic speed marker is a single point on the screen. In the aid descriptor, each of the last six numbers is set at a constant 99.99. The flight identifier is followed by a single set of coordinates for the point location. This can be seen for flight 725 in the graphic example above. Nominally, the controller issued the clearances when the aircraft just touched the marker.

The aid descriptor for the slot marker contains two numbers, the flight number and the y coordinate in nautical miles (simulation frame of reference) of the slot marker. The x axis coordinate is not given and stays constant at -.34 nmi. The marker moves along the extended runway centerline toward the runway, which is parallel to the y simulation axis and just below it. Nominally, the controller issued clearances with the goal of placing the aircraft in the center of its slot marker as it proceeded on the final approach course.

The format of the .ACP record group is more precisely defined in subroutine TARGETSET in program FILLMRG. The file is read as #3 indexed by file #4, the .IDX file. The first three types of records in the group are read in common code but the last type of record is read using a 'SELECT CASE' structure to differentiate between the different types of FASA formats.

The Oculometer Data File (.DAT)

The .DAT file produced by the oculometer facility is a random access binary file. Each record has eight bytes (four 16-bit integers). There are no record separator bytes such as the usual carriage-return/line-feed sequence. Thus, for example, the fifth record spans bytes 33 to 40. Each record contains data on either an in-track or out-of-track event depending on whether or not the instrument was tracking the subject's eye. For in-track events, the first two fields contain the x and y position coordinates of the look point given in the display reference frame. The third field contains the pupil diameter and the fourth contains the time duration of the fixation. For out-of-track events, the first two fields contain zeros. The third field contains instrument status information not germane to this study, and the fourth contains the time duration of the event. The times are given in units of 1/30 second, i.e., divide by 30 to get seconds. The coordinates and pupil diameter are given in converter counts. To convert the coordinates to inches on the display, divide the values by 204.8 (data constant cpi! in subprogram CRE8MRGFLE of program CRE8MRG1). As was stated with reference to the .MRG file above, to convert pupil diameter to millimeters multiply the value by (25.4/2048).

The subprogram BI2 in FIXPOINT reads the .DAT file, appends a fifth integer to it, and writes (subroutine CRE8DT1) the record to the .DT1 file. The data structure DT1 defined in FIXPOINT is used with the .DT1, DT2, and DT3 files. The appended first byte is derived from the .SCN file. It contains a record pointer to the .ACP file, and it must be added prior to any filtering. See the code in FIXPOINT for further detail on these two record structures.

The Oculometer Synchronization File (.SCN)

The .SCN file produced by the oculometer facility is a random access binary file. Each record has two bytes (one 16-bit integer). There are no record separator bytes such as the usual carriage-return/line-feed sequence. The number is recorded at the beginning of each simulation update and its value is the record number (ordinal) of the last .DAT file record stored at that point in time. This is used by program FIXPOINT to synchronize the .DAT file to the .ACP file. The single integer .SCN record is read into the first column of the buffer array OCSCAN in program FIXPOINT, subprogram BI1 to be used as a record pointer to the .ACP file.

Appendix B

Program Block Diagrams

The diagrams presented in this appendix have proven themselves to be very useful during the development and maintenance of the source code presented in Appendix C. The order of the diagrams corresponds to the (almost) alphabetical order of the program listings in Appendix C. They are included here to help anyone who needs to examine the code in detail. The block diagrams show how the programs interact with the various files. The caption on each figure attempts to explain the function of the process. A PC computer was used to do this analysis. The programs were written in Microsoft Quick Basic. Certain common devices are used throughout. These will be explained in order to make the processes easier to follow.

Each run has an associated name containing information on subject, run number, format, and speed. For example, LC12DC21 would be a name associated the 12th run for subject LC, which used the DICE format at the 210 pattern speed. All files associated with this run would use this name with an appropriate suffix. Thus, LC12DC21.MRG, LC12DC21.CCS, and LC12DC21.LOG are the merge, cross-check-scan, and log file associated with that particular run. In the diagrams of this section the names of files are dropped, and the suffix is used to indicate the type of file being used, e.g., .MRG, .CCS, and .LOG. In figure B2, for example, the .DT3 file is processed by CRE8MRG1, which produces a .MRG file. During the processing, information is appended to the .LOG file for the particular run being processed. Normally, twelve files, one for each subject, were processed as a sequential group. This would represent all the runs for one display-format/pattern-speed combination. The names of the twelve runs are listed in FLEINDX1. The file TOTAL.LOG (figure B2) contains composite information for the twelve files. This semi-automated approach made it possible to use two or three computers at the same time, each processing a different treatment. The sharped numbers (e.g., #2 near the .DT3 file) indicate the file number used in the corresponding source code (Appendix C) and are included for clarity.

Hopefully the diagrams (along with their associated source listings) will shed some light on the individual processing steps. They are not, however, in temporal order, so in order to clarify the overall procedure, something needs to be said about sequence. There are 3 major parts: data acquisition, data reduction, and data analyses. Most of the analysis used a commercial program to generate repeated measures analysis of variance on statistics derived from the .MRG or .CCS files. These two files are the output of the data reduction phase, and the cross-check-scan (.CCS) file is totally derived (figure B5) from the merge (.MRG) file. The input of the data reduction phase (output from the acquisition phase) are the .SCN, .DAT, and .ACP files.

During the acquisition phase, it is critical that the oculometer data be kept carefully synchronized with the simulator data and that position accuracy be maintained through frequent calibrations. The preferred method of synchronization is to have one computer record all the data using a single time stamp. In the FASA study, because of the different processing periods (four seconds versus 1/30 second), the data were recorded on two computers and later synchronized using the .SCN file data. During the experiment, it is wise to carefully observe the data being recorded and to do quick look analyses to test its quality. In later stages of analysis, one may correct mistakes and re-analyze the data, but in the acquisition phase, an error could result in having to rerun the experiment. Therefore, one must acquire data carefully and attempt to find and correct problems immediately.

The first stage of data reduction (figure B11) is to synchronize the .DAT file records by adding to them an aircraft position record pointer (.ACP record number) derived from the .SCN file. Once synchronized, the programs SRCGDAT (figure B18) and CUT20 (figure B6) are used to prune out a few known bad records. Then, the data are filtered (figure B10) to remove noise and combine certain contiguous fixations. The filtered data are then saved (figure B2) in the merge file format. Many of the .MRG record fields are undefined at this point. Because of the complexity of the .ACP record, a file index is generated (figure

B3) for each file. The program FILLMRG (figure B9) searches the .ACP file to determine the likely object of the controller's gaze. This action involves a coordinate transformation of the lookpoint and, then, a distance computation for each object on the screen during the particular simulation interval. Once identified, target information is transformed into display coordinates and written into the .MRG record. CROSS1 (figure B5) generates the cross check file (.CCS) by searching groups of contiguous scans for the occurrence of this stylized behavior. BEANCNT1, A1HIST, SEQNCE1, and SEQNCE2 (figures B1, B8, B15, and B16) derive statistics from the .MRG, and .CCS files. They were used in addition to a commercial data base program and a statistical analysis program. CRI8MGX (figure B4) was used to produce the .MRG file index. The programs shown in figures 13 and 14 were used to make printed listings of the data, and those in figures 12 and 17 were used to plot the lookpoints. The programs in figure 7 were used to interactively examine the data.

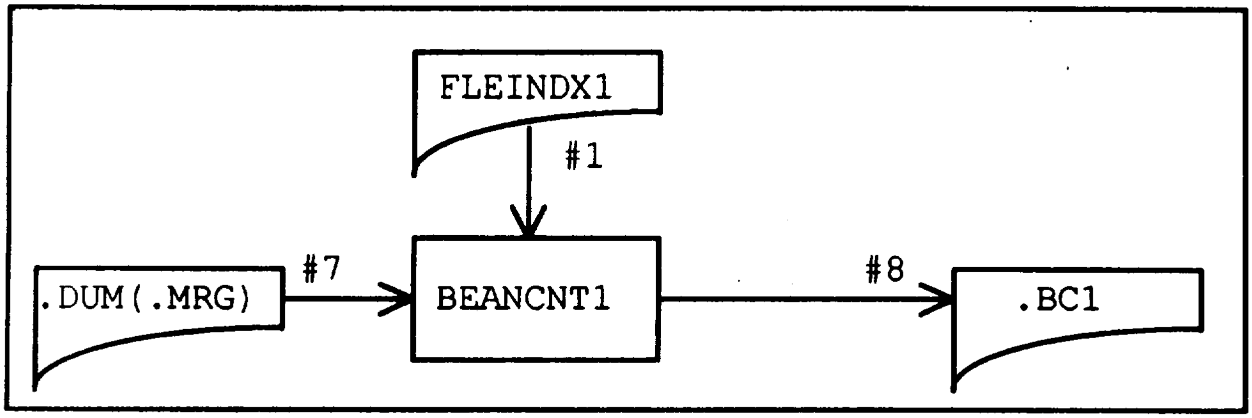


Figure B1. BEANCNT1 tallies statistics on 3 classes of oculometer objects: unidentified targets, all other in-track objects, and out-of-track objects. The statistics include very coarse frequency functions on time duration, zone, and distance between target and lookpoint.

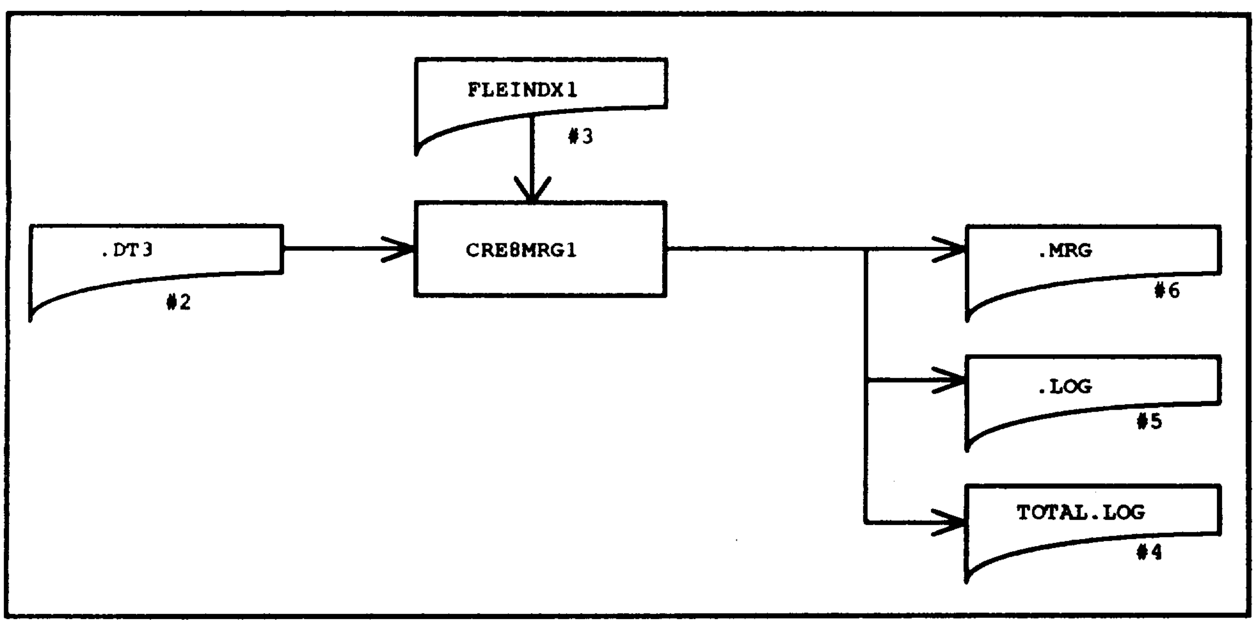


Figure B2. CRE8MRG1 is a simple program that sets up the .MRG file. After filtering and before the target search, each oculometer event becomes a record on the .MRG file. At this point, most fields have not yet been filled. Target information will be added by a later process to .MRG.

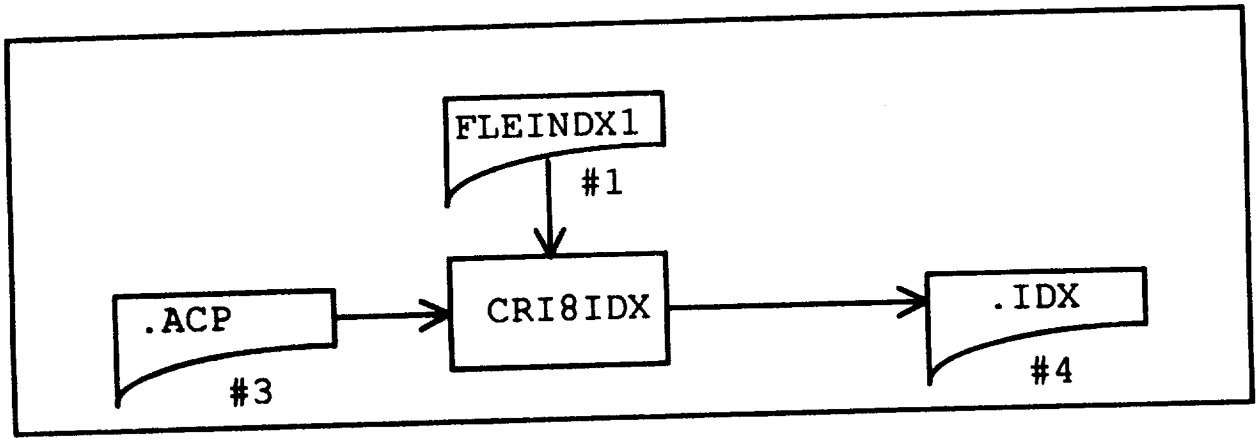


Figure B3. The .ACP file has long variable size, multiple format records. The .IDX file-record produced by CRI8IDX is a pointer (index) to the first byte of the corresponding .ACP record. Each in-track lookpoint is associated with a particular radar sweep. FILLMRG (figure B9) uses .IDX to find the associated simulation output record and to search for and extract aircraft position data from the .ACP file.

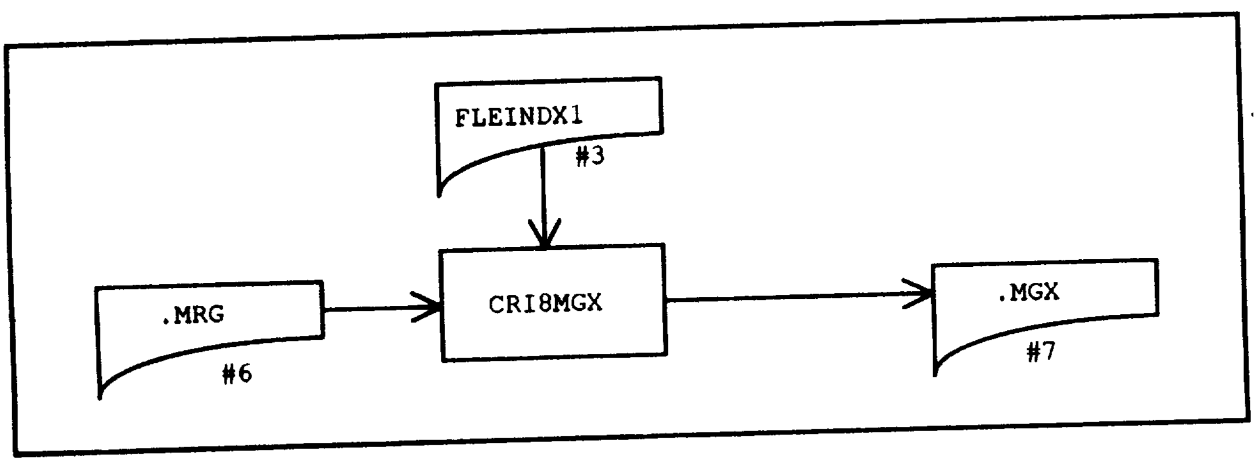


Figure B4. The .MGX index file produced by CRI8MGX is used to examine or print segments or individual records of the .MRG file. An example of its use is shown in figure B14.

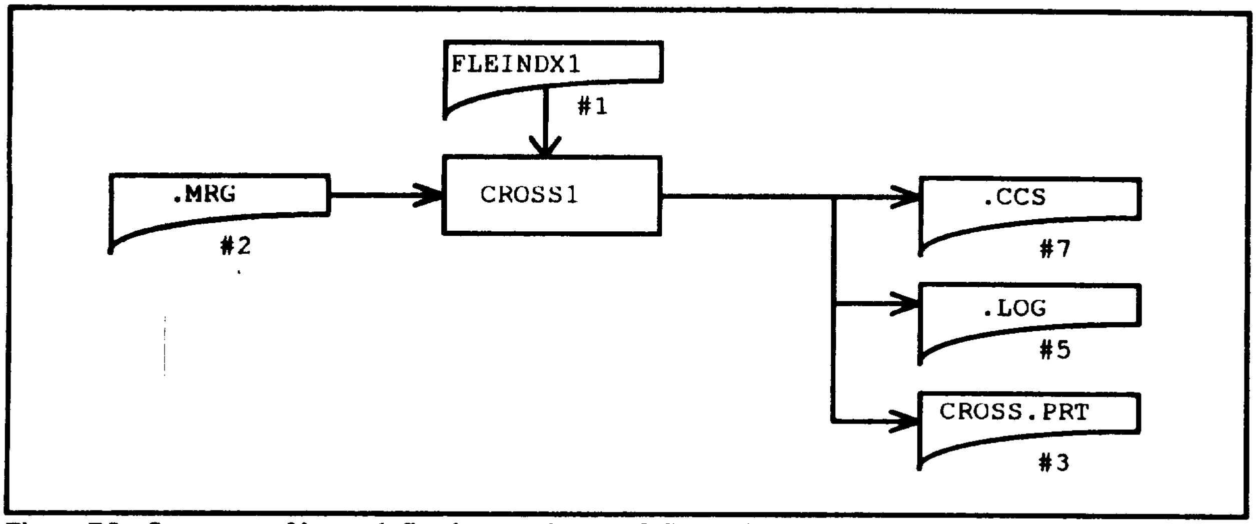


Figure B5. Sequences of in-track fixations on the .MRG file which alternate between two screen objects (such as two aircraft symbols) become a single record on the .CCS (cross check scan) file. CROSS1 does not artificially limit the order of a scan. It counts transitions until the scan is interrupted. The somewhat complex logic for identifying cross check scans is in the subroutine FINDAB. The logic can be extended to include groups of 3 or more objects.

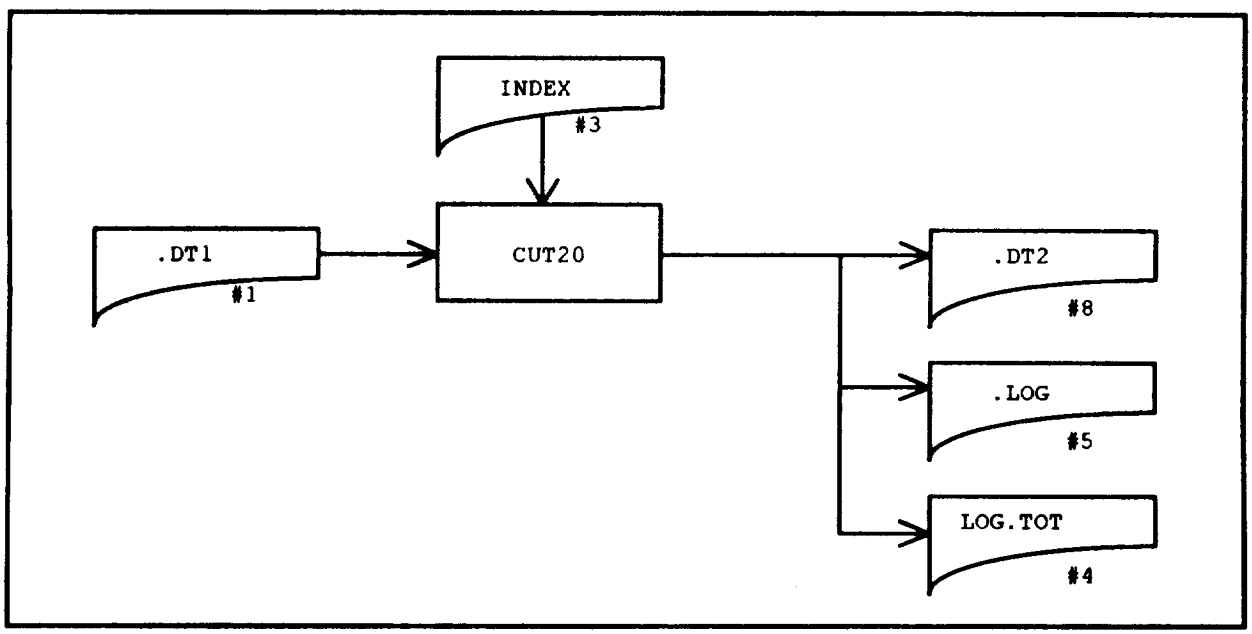


Figure B6. CUT20 simply excises a group of oculometer events (records) from a file. However it leaves an audit trail in the individual run logs and collectively in the LOG.TOT file. Index specifies a list of files including which records need to be removed.

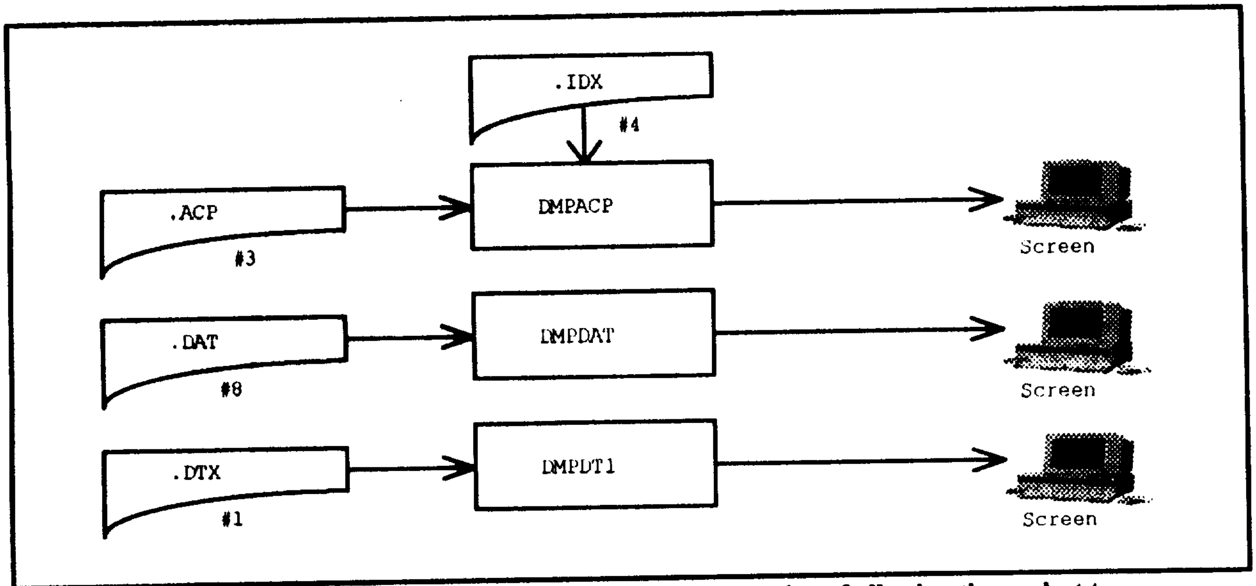


Figure B7. These three simple programs serve the important function of allowing the analyst to interactively peruse the contents of the files.

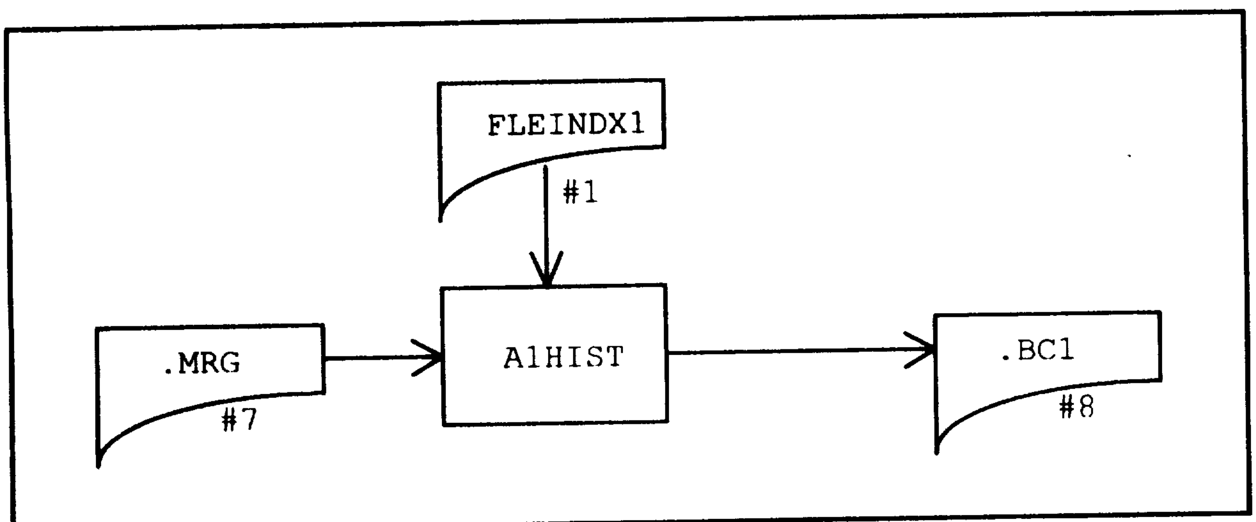


Figure B8. A1HIST makes two passes through the files to compute normalized histograms of the fixation times for each aircraft data block. It computes time duration histograms for three conditions: aid-on, aid-off and the two combined. This amounts to three histograms per run. It also produces three histograms combining all the runs listed in FLEINDX1.

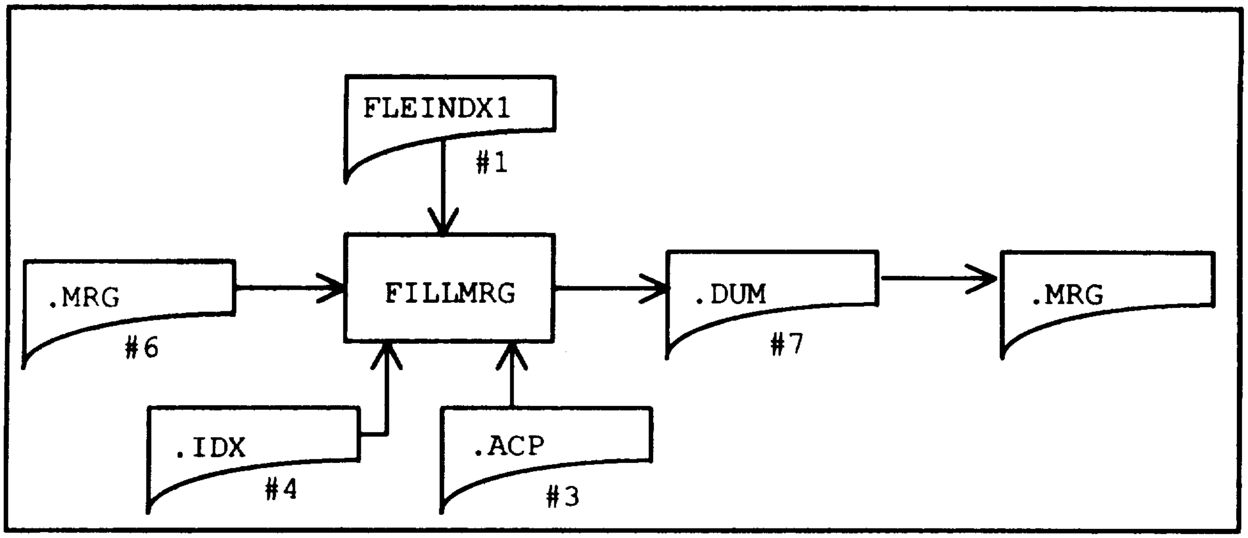


Figure B9. FILLMRG associates each lookpoint on the .MRG files with a display object recorded on the .ACP file. The resulting .MRG file has information on the lookpoint and the target as well as the distance between and whether or not the aid is on.

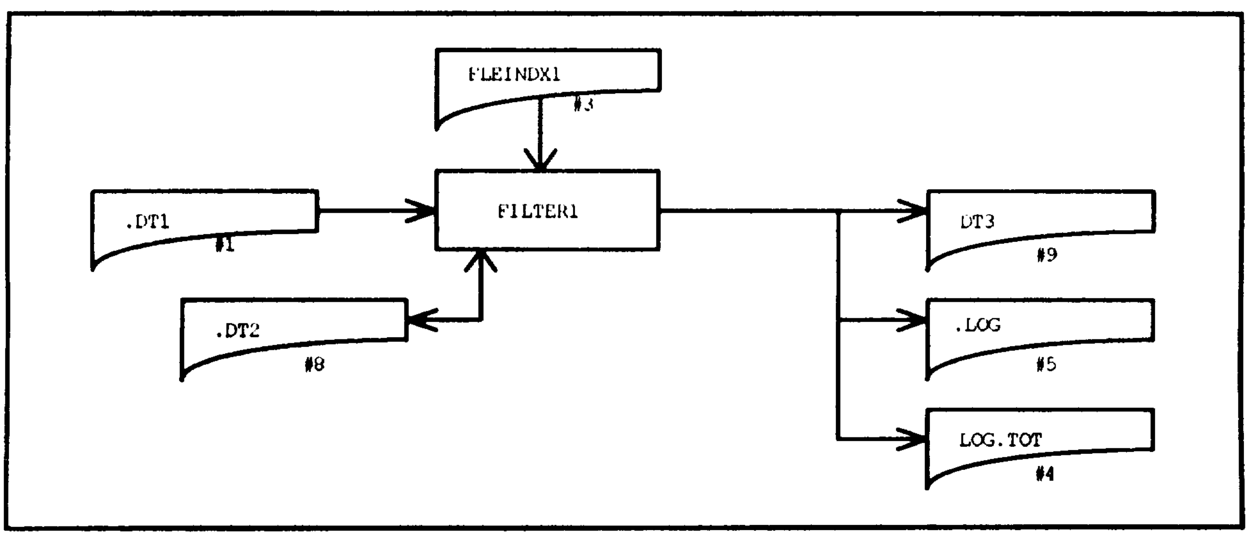


Figure B10. FILTER1 implements the four filters discussed in the paper. The input file (.DT1) is the renamed output file (.DT2) from CUT20 (Figure B6). The output file (.DT3) has significantly fewer records after filtering.

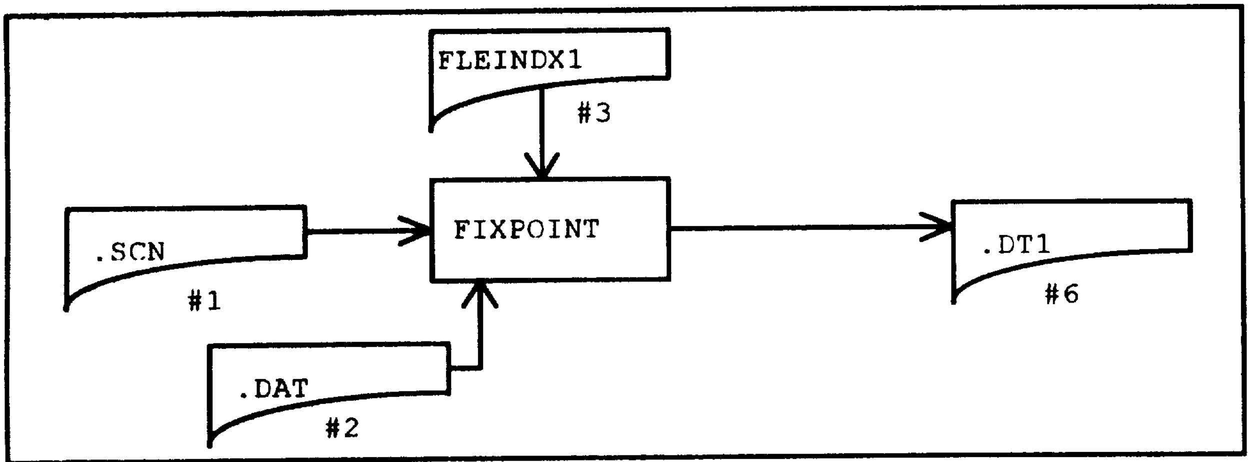


Figure B11. FIXPOINT computes the record number of the .ACP time history file corresponding to each record of the oculometer .DAT file. This number is appended as a fifth field to the four .DAT file fields and placed in .DT1. The logic for associating .DT1 records to .ACP records is in subroutine FIXPOINTER.

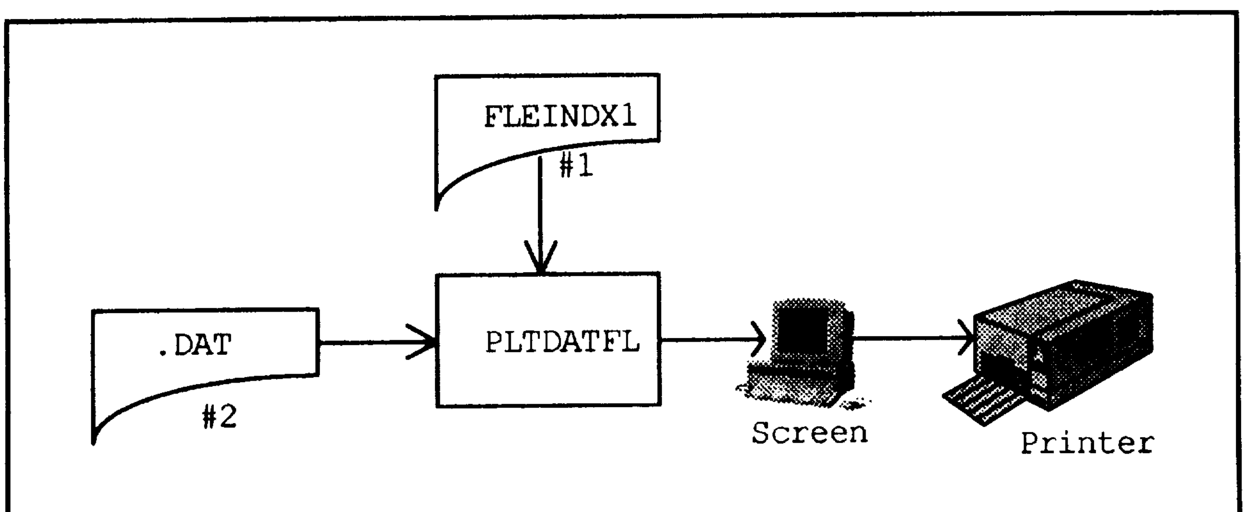


Figure B12. PLTDATFL uses the screen print interrupt to plot all the lookpoints from a given run on a single sheet. The points correspond to the positions on the controller's display. They are a graphic description of the scan pattern. With slight modification (to read the 5 byte input) this was also used to look at the data after filtering.

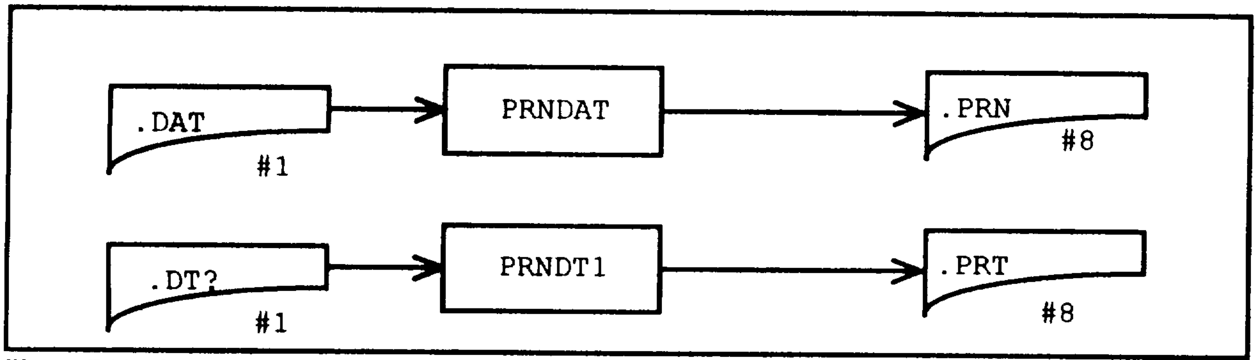


Figure B13. PRNDAT and PRNDT1 are simple but useful programs that allow one to copy segments of the data files to a printable file. It is more convenient to use the .PRN and .PRT files than to go directly to the printer. The programs are interactive with the user supplying the file name and first and last record of each segment. The .DT? files have 5 fields and the .DAT files have four.

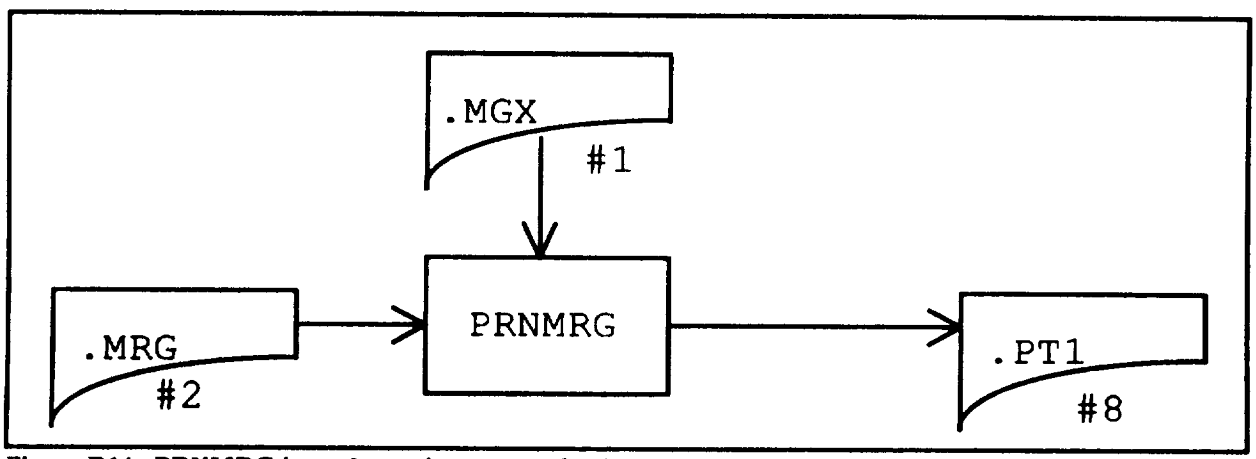


Figure B14 PRNMRG is used to write groups of adjacent .MRG file records to the .PT1 file for subsequent scrutiny. The program is interactive with the user supplying the file name and the first and last record of each segment to be copied.

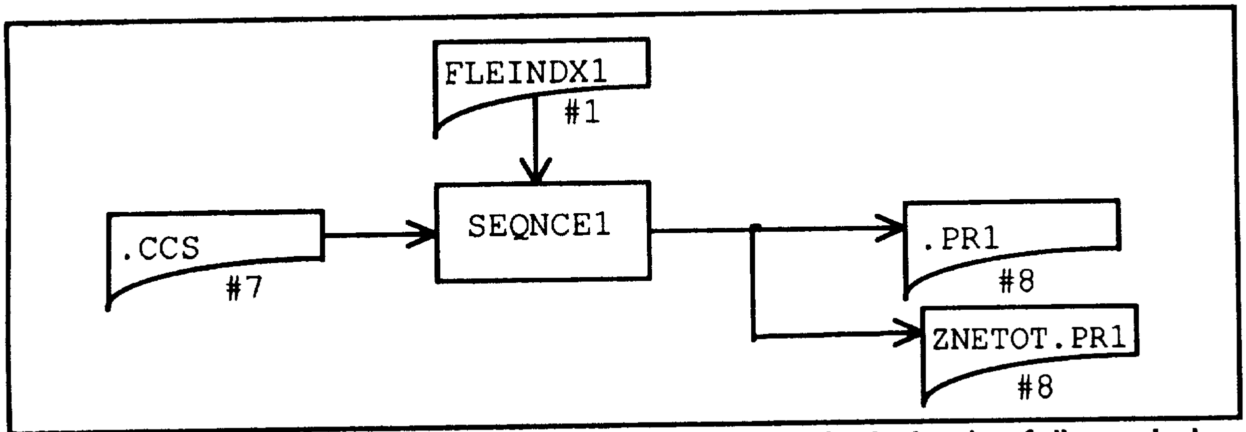


Figure B15. SEQNCE1 computes the average and standard deviation for the duration of all cross check scans and the distances between all corresponding pairs of targets. In addition to the overall values, it computes these parameters for each defined zone pair and order of cross check scan. The source program as shown uses 4 orders of cross check scans and 10 defined zone pair. The results for each run listed in FLEINDX1 are written into a corresponding .PR1 file. The accumulated results for all runs are written to the ZNETOT.PR1 file.

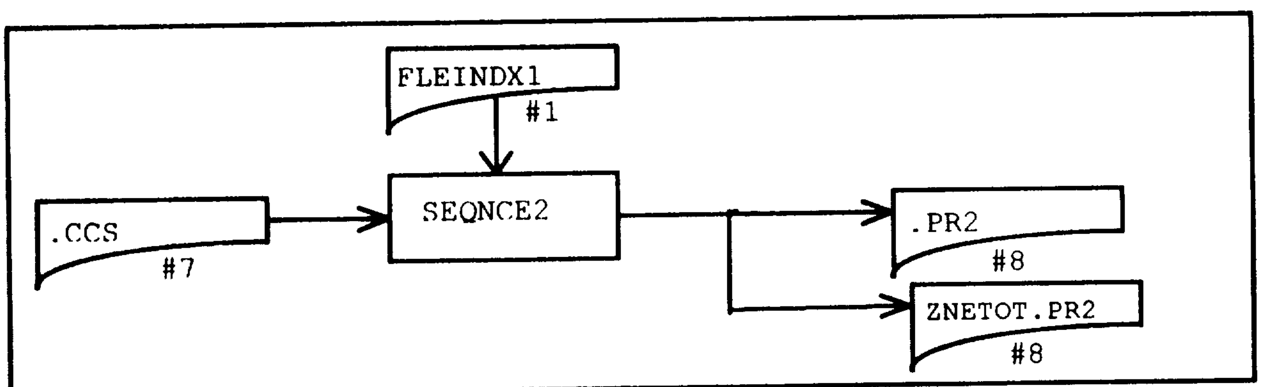


Figure B16. SEQNCE2 computes the average and standard deviation for the duration of all cross check scans and the distances between all corresponding pairs of targets. In addition to the overall values, it computes these parameters for each defined <u>target</u> pair and order of cross check scan. The source program as shown uses 4 orders of cross check scans and 20 defined <u>target</u> pairs. The results for each run listed in FLEINDX1 are written into a corresponding .PR2 file. The accumulated results for all runs are written to the ZNETOT.PR2 file. Normally, FLEINDX1 would have 1 file name for each subject for a given treatment.

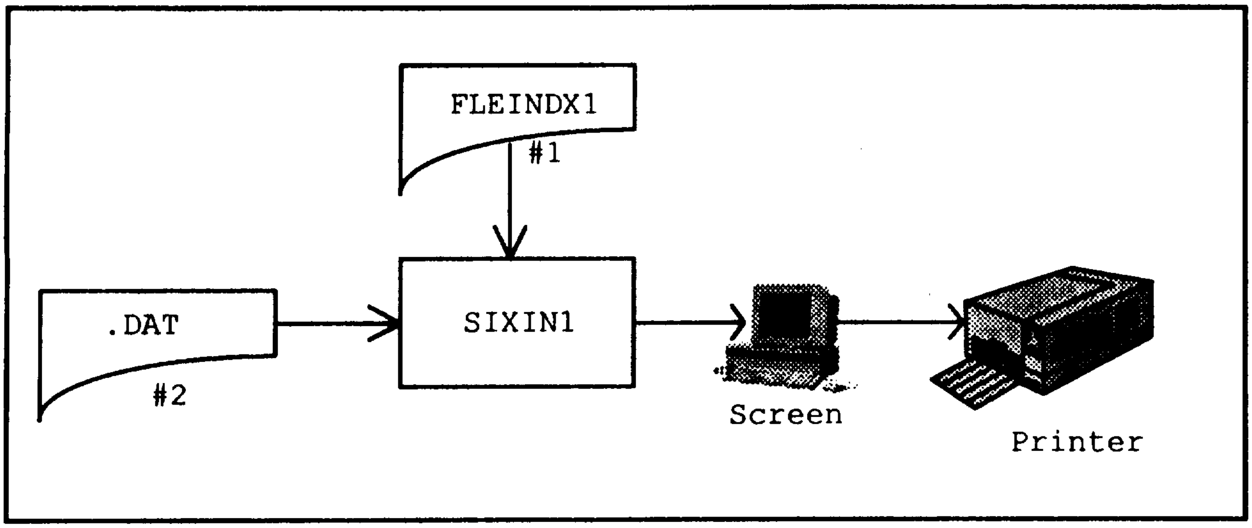


Figure B17. SIXIN1 is a plotting program almost identical to PLTDATFL (figure B12) above except that it puts six scatter plots on a single page to facilitate comparisons. Each plot depicts all lookpoint positions for a run without providing information on duration or sequence.

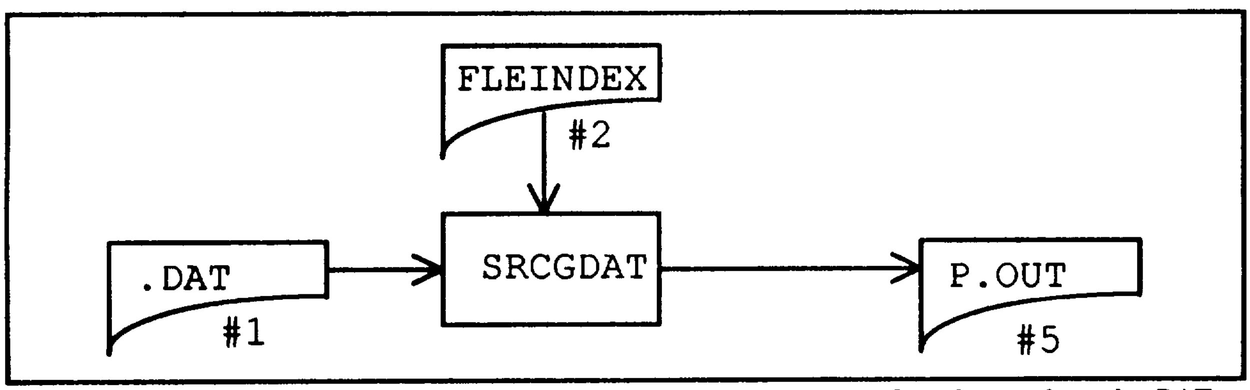


Figure B18. SRCGDAT was used to find abnormally long in-track and out-of-track records on the .DAT files. Once identified, they were checked against logs from the corresponding run and the recollections of the researchers. Some records (an extremely small amount) were then purged as false data resulting from system malfunctions.

Appendix C

Data Reduction and Analysis Source Code

Table of Contents

PROGRAM BEANCNT1 1
SUB FIN 3
SUB INIT (filename$, filename1$) 3
SUB PROUT 5
SUB SEARCH 6
SUB SRCHWRK 7
PROGRAM CRE8MRG1 10
SUB BI2 (BIP%()) 11
SUB CRE8MRGFLE 12
SUB FIN 14
SUB INIT (FILENAME$, FILENAME1$) 15
SUB PUTXX (FILENO%) 16
PROGRAM CRI8IDX 18
SUB FIN 19
PROGRAM CRI8MGX 21
PROGRAM CROSS1 22
SUB FILLBUF 24
SUB FIN 24
SUB FINDAB (IOK) 25
SUB INIT (FILENAME$, FILENAME1$) 28
SUB PRINTSEQ (LAPOVER) 29
PROGRAM CUT20 31
PROGRAM DMPACP 33
PROGRAM DMPDAT 35
PROGRAM A1HIST 36
SUB ACCUMULATE 38
SUB HISTOGRAM 39
SUB INIT (filename$, filename1$) 40
SUB PROUT2 41
SUB SEARCH 43
PROGRAM DMPDT1 45
PROGRAM FILLMRG 46
SUB FIN 47
FUNCTION HIT% (VL%, ARRAY%(), N%) 48
SUB INIT (FILENAME$, FILENAME1$) 48
SUB PICK (X!, Y!, TOTSYM%, K, DISTMIN!) 50
SUB PUTXX (FILENO%) 52
SUB SEARCH 52
SUB TARGETSET (FRAMENO%, TOTSYM%, NODICE%) 56
PROGRAM FILTER1 60

PROGRAM FIXPOINT 65
SUB BI1 (BIP%()) 66
SUB BI2 (BIP%()) 66
SUB CRE8DT1 68
SUB FIN 69
SUB FIXPOINTER 69
SUB INIT (FILENAME$, FILENAME1$) 71
PROGRAM PLTDATFL 74
PROGRAM PRNDAT 76
PROGRAM PRNDT1 77
PROGRAM PRNMRG 78
Program Qklook 80
SUB BI1 (BIP%()) 82
SUB BI2 (BIP%()) 82
SUB CRE8MRGFLE 84
SUB FIN 85
SUB FIXPOINTER 87
SUB GETXX (FILENO%) 89
SUB INIT 89
SUB PRNDAT (STRT&, NOR%) 91
SUB PUTXX (FILENO%) 92
PROGRAM SEQNCE1 93
SUB FILBLNK1 95
SUB FINDTGTPL (L) 95
SUB INIT (FILENAME$, FILENAME1$) 97
SUB PRNZONE (ZONE()) 98
SUB READSEQ 99
PROGRAM SEQNCE2 102
SUB FILBLNK1 103
SUB FINDTGTPL (L) 104
SUB INIT (FILENAME$, FILENAME1$) 106
SUB PRNTGTS (TRGTS()) 107
SUB READSEQ 109
PROGRAM SIXIN1 111
PROGRAM SRCHDAT 113
COMMON UTILITY SUBROUTINES 115
FUNCTION LOG$ (SB$, A$) 115
FUNCTION LOGS$ (SB$, A$) 115
SUB GETXXA (FILENO%) 115
SUB GETXXB (FILENO%, NEOFMRG) 116
SUB YESORNO (A$, B$) 116

PROGRAM BEANCNT1

```
DECLARE SUB PROUT ()
DECLARE SUB SRCHWRK ()
DECLARE SUB FIN ()
DECLARE SUB INIT (filename$, filename1$)
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE FUNCTION LOGS$ (SB$, A$)
DECLARE SUB SEARCH ()
DECLARE SUB GETXXA (FILENO%)
DEFINT I-N
CONST pi! = 3.14159
CONST SF! = .472, XOFF! = -5.04, YOFF! = -.9, cpi! = 204.8, alpha
                        ---->! = -11.5 * pi! / 180, runoff! = -.34
CONST big! = 3!, little! = 1!
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                         'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                              'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                       'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE      'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                       'TIME HISTORY FRAME #
     TGTX AS SINGLE                                'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                              'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS STRING * 3                                 'DICE
     COUNTDOWN AS INTEGER                                  'DICE
     CONTFIX AS STRING * 1                                 'IS THIS
                ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1                       'SPLADT  S-on, F-off
     AIDON AS STRING * 1                              'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM AAAA$
DIM OTN(1 TO 4) AS INTEGER, UNKN(1 TO 4) AS INTEGER, ITN(1 TO 4)
                        ---->AS INTEGER, DISTN(1 TO 4) AS INTEGER
DIM OTT(1 TO 7), UNKT(1 TO 7), ITT(1 TO 7) AS SINGLE, TT
DIM ZONEN(1 TO 7) AS INTEGER
DIM DISTL(1 TO 7) AS SINGLE
DIM frmt$
frmt$ = "## /  / #### #### / / ###.## #### ###.## ###.## ###.## #
                                ---->##.## ### ### ! ! //"
DIM XX AS FIXCOMB
DIM XXX$
DIM IAID AS INTEGER
DIM BI1P(1 TO 7)  AS INTEGER'Buffered Input 1
DIM BI2P(1 TO 7)  AS INTEGER'Buffered Input 2
DIM BOP(1 TO 7)  AS INTEGER                       'Buffered Output
DIM FILEDUM$, FILEMRG$, FILEBC1$
```

```
DIM FILEDUM%, FILEMRG%, FILEBC1%
DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
'    ** FILE NUMBERS **
'       #1 FILE INDEX           #2          #3     'modified 1/5/93
'       #4                      #5                        #6
'       #7 DUM                  #8 BC1                    #9
'
DIM PAGE$, FONT$
DIM SPSINV!, SB$
SPSINV! = 1 / 30: SB$ = " (BEANCNT1  "
NOSTAT% = 10
DIM STATXS(NOSTAT%), STATYS(NOSTAT%) AS SINGLE
DIM STATID(NOSTAT%) AS STRING * 4
DATA "DEN ","IOC ","OM  ","KEAN","FLTS","WIVS","BYSN","TROZ","DRK
                                                             ---->O","JASN"
FOR I = 1 TO NOSTAT%: READ STATID(I): NEXT I
DATA 2.38,-19.49,-0.3,19.44,10.92,-10.24,-23.07,-8.22,29.42,14.67
FOR I = 1 TO NOSTAT%: READ STATXS(I): NEXT I
DATA -.63,24.92,6.1,28.79,14.1,14.1,-26.08,-11.24,-19.56,-9.23
FOR I = 1 TO NOSTAT%: READ STATYS(I): NEXT I
PRTCONTROL$ = CHR$(33) + "R" + CHR$(33)
PAGE$ = PRTCONTROL$ + ";PAGE;EXIT;"                         'OFFICE
FONT$ = PRTCONTROL$ + ";RES;FONT 62; EXIT;"                 'OFFICE
'PAGE$ = CHR$(12)                                             'HOME
'FONT$ = CHR$(27) + CHR$(80)                                  'HOME
DIM SINAL!, COSAL!
SINAL! = SIN(alpha!): COSAL! = COS(alpha!)
XXX$ = "## \  \ #### #### \ \ ##.## #### ###.## ###.## ###.## ###
                                                 ---->.## ### #### ! ! \\"
AAAA$ = " MAIN-START MAIN LOOP"
INPUT " Enter full file descriptor for index file  ", index$
OPEN index$ FOR INPUT AS #1
INDEX1$ = LEFT$(index$, LEN(index$) - 4)        'modified 1/5/93
FILEBC1$ = INDEX1$ + ".BC1": FILEBC1% = 8       'modified 1/5/93
OPEN FILEBC1$ FOR OUTPUT AS #FILEBC1%           'modified 1/5/93
DO WHILE NOT EOF(1)
     INPUT #1, filename$
     IF LEN(filename$) < 8 THEN EXIT DO           'modified 1/5/93
     filename1$ = RIGHT$(UCASE$(filename$), 8)
     CALL INIT(filename$, filename1$)
     CALL SEARCH
     PRINT LOGS$(SB$, " FINISHED SEARCH " + filename1$)
     CALL PROUT
     CALL FIN                                              'Close FILES
LOOP
CLOSE 1, 8             'modified 1/5/93
PRINT AAAA$
'.............................................................
'.............................................................
END     ' MAIN PROGRAM 'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

SUB FIN

```
'##########
'Purpose................\ Close all files, scale and output a few
                                             ----> statistics
'Parameters.............\
'Other input data.......\ SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK,
                                   ---->NUMINTRACK, NUMOUTTRACK
'Input files............\
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'##########
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
    PRINT LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= "); NUMINTRAC
                           ---->K;      ' totals for .txt file
    PRINT LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS="); NUMOUTTRA
                                                         ---->CK
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
         ---->TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
        ---->TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK / 30
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
       ---->FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH / 30
    PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRACK; SUMINTRA
                                    ---->CK / NUMINTRACK / 30
    PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->OUT TRACK FIXATION IS  "); SUMOUTTRACK / NUMOUTTRACK; SUMOUT
                                ---->TRACK / NUMOUTTRACK / 30
END IF
CLOSE 7
'.................................................................
'.................................................................
END SUB     ' FIN  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB INIT (filename$, filename1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                  ---->buffers
'            \ Initialize sums to zero.  Let user choose partic-
'              \ ular run for analysis.  Determine aid type for
'              \ subsequent branching. Open FILESCN$, FILEDAT$,
'                   \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
```

```
                                                  ---->ables, sums
'                         \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED OTN() AS INTEGER, UNKN() AS INTEGER, ITN() AS INTEGER, DIS
                                              ---->TN() AS INTEGER
SHARED OTT(), UNKT(), ITT() AS SINGLE, TT
SHARED ZONEN() AS INTEGER
SHARED DISTL() AS SINGLE
SHARED BI1P()  AS INTEGER                         'BufferedInput 1
SHARED BI2P()  AS INTEGER                         'BufferedInput 2
SHARED BOP()  AS INTEGER                           'BufferedOutput
SHARED FILEDUM$, FILEMRG$, FILEBC1$
SHARED FILEDUM%, FILEMRG%, FILEBC1%
SHARED IAID AS INTEGER
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SB$ = "(INIT "
          'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                        ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1
'FOR L = 1 TO 7: print BI1P(L): NEXT L
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                         ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
I = nfixbuf: BOP(1) = I: BOP(2) = 1: BOP(3) = 1: BOP(4) = .9 * I
BOP(5) = .8 * I: BOP(6) = 0: BOP(7) = 1
'.................................................................
TT = 0
FOR I = 1 TO 4: OTN(I) = 0: UNKN(I) = 0: ITN(I) = 0: DISTN(I) = 0
                                                     ---->: NEXT I
FOR I = 1 TO 7: OTT(I) = 0: UNKT(I) = 0: ITT(I) = 0: DISTL(I) = 0
                                    ---->: ZONEN(I) = 0: NEXT I
OTT(6) = 1000000: UNKT(6) = 1000000: ITT(6) = 1000000: DISTL(6) =
                                                    ----> 1000000
'.................................................................
SELECT CASE MID$(filename1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
          IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                 ---->C,GR or SL"): PRINT : STOP
```

```
IAID1 = VAL(MID$(filename1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                     ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
'FILEDUM$ = FILENAME$ + ".DUM": FILEDUM% = 7   'modified 1/5/93
FILEDUM$ = filename$ + ".MRG": FILEDUM% = 7    ' modified 1/5/93
FILEMRG$ = filename$ + ".MRG": FILEMRG% = 6
'FILEBC1$ = filename$ + ".BC1": FILEBC1% = 8          '  COMMENT
                                  ---->: FILE NAME SHOULD LOOK LIKE
'                                     "C:\FASAFILE\CRONE\CC10SLCE"
'...................................................................
'...................................................................
'...................................................................
END SUB        ' INIT   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PROUT

```
SHARED FILEDUM$, FILEMRG$, FILEBC1$
SHARED FILEDUM%, FILEMRG%, FILEBC1%
SHARED OTN() AS INTEGER, UNKN() AS INTEGER, ITN() AS INTEGER, DIS
                                             ---->TN() AS INTEGER
SHARED OTT(), UNKT(), ITT() AS SINGLE, TT
SHARED ZONEN() AS INTEGER
SHARED DISTL() AS SINGLE
SHARED SPSINV!, SB$
SB$ = "(BEANCNT1 "
'OPEN FILEBC1$ FOR OUTPUT AS #FILEBC1%    'modified 1/5/93
PRINT #FILEBC1%, LOGS$(SB$, FILEMRG$)  'modified 1/5/93
PRINT #FILEBC1%, USING "Total time ####.#   OT ####.#   IT-UNK ##
---->##.#   IT  ####.# IN SECONDS"; TT * SPSINV!; OTT(1) * SPSINV
                          ---->!; UNKT(1) * SPSINV!; ITT(1) * SPSINV!
X1! = 100 * OTT(1) / TT: X2! = 100 * UNKT(1) / TT: X3! = 100 * IT
       ---->T(1) / TT: X4! = 100 * UNKT(1) / (UNKT(1) + ITT(1))
PRINT #FILEBC1%, USING "                    OT    ##.#%  IT-UNK
   ---->##.#%  IT    ##.#%  UNKAS%TOTT ##.#%"; X1!; X2!; X3!; X4!
'...................................................................
PRINT #FILEBC1%,
PRINT #FILEBC1%, "S*30--%T        Min      Max      Mean     SD
                                      ----> <4       4-12     >12 "
FM1$ = "  ##    #####      ##.#  ##.#    ##.#%      ##.#%  ##.#%"
FM2$ = " #.##    #.##      #.##  #.##    ##.#%      ##.#%  ##.#%"
'out of track.......................................................
X1! = OTT(1) / OTN(1): X2! = SQR(OTT(5) / OTN(1) - X1! ^ 2)
X3! = 100 * OTT(2) / OTT(1): X4! = 100 * OTT(3) / OTT(1): X5! = 1
                                        ---->00 * OTT(4) / OTT(1)
PRINT #FILEBC1%, "OT             ";
PRINT #FILEBC1%, USING FM1$; OTT(6); OTT(7); X1!; X2!; X3!; X4!;
                                                          ---->X5!
'in track target unidentified.......................................
X1! = UNKT(1) / UNKN(1): X2! = SQR(UNKT(5) / UNKN(1) - X1! ^ 2)
X3! = 100 * UNKT(2) / UNKT(1): X4! = 100 * UNKT(3) / UNKT(1): X5!
                                  ----> = 100 * UNKT(4) / UNKT(1)
PRINT #FILEBC1%, "IT-UNK         ";
```

```
PRINT #FILEBC1%, USING FM1$; UNKT(6); UNKT(7); X1!; X2!; X3!; X4!
                                                         ---->; X5!
'in track target identified.........................................
X1! = ITT(1) / ITN(1): X2! = SQR(ITT(5) / ITN(1) - X1! ^ 2)
X3! = 100 * ITT(2) / ITT(1): X4! = 100 * ITT(3) / ITT(1): X5! = 1
                                        ---->00 * ITT(4) / ITT(1)
PRINT #FILEBC1%, "IT             ";
PRINT #FILEBC1%, USING FM1$; ITT(6); ITT(7); X1!; X2!; X3!; X4!;
                                                           ---->X5!
'distance lookpoint to target.......................................
PRINT #FILEBC1%, "                              <.25    .25-.5  >.5 "
X1! = DISTL(1) / DISTN(1): X2! = SQR(DISTL(5) / DISTN(1) - X1! ^
                                                            ---->2)
X3! = 100 * (DISTN(2) / DISTN(1)): X4! = 100 * (DISTN(3) / DISTN(
                     ---->1)): X5! = 100 * (DISTN(4) / DISTN(1))
PRINT #FILEBC1%, "DIST           ";
PRINT #FILEBC1%, USING FM2$; DISTL(6); DISTL(7); X1!; X2!; X3!; X
                                                         ---->4!; X5!
'Array contents for checking........................................
PRINT #FILEBC1%,
PRINT #FILEBC1%, " # of records-Total, Time <4, 4-12, >12"
FOR II = 1 TO 4: PRINT #FILEBC1%, OTN(II); : NEXT II: PRINT #FILE
                                                     ---->BC1%, "OTN"
FOR II = 1 TO 4: PRINT #FILEBC1%, UNKN(II); : NEXT II: PRINT #FIL
                                                   ---->EBC1%, "UNKN"
FOR II = 1 TO 4: PRINT #FILEBC1%, ITN(II); : NEXT II: PRINT #FILE
                                                     ---->BC1%, "ITN"
PRINT #FILEBC1%,
PRINT #FILEBC1%, " # of records-Total, Dist <.25, .25-.5, >.5"
FOR II = 1 TO 4: PRINT #FILEBC1%, DISTN(II); : NEXT II: PRINT #FI
                                                ---->LEBC1%, "DISTN"
FOR II = 1 TO 4: PRINT #FILEBC1%, ZONEN(II); : NEXT II: PRINT #FI
                                                ---->LEBC1%, "ZONEN"
PRINT #FILEBC1%,
PRINT #FILEBC1%, " Time in counts-Total, Time <4, 4-12, >12, SSQ,
                                                   ----> Min, Max"
FOR II = 1 TO 7: PRINT #FILEBC1%, OTT(II); : NEXT II: PRINT #FILE
                                                     ---->BC1%, "OTT"
FOR II = 1 TO 7: PRINT #FILEBC1%, UNKT(II); : NEXT II: PRINT #FIL
                                                   ---->EBC1%, "UNKT"
FOR II = 1 TO 7: PRINT #FILEBC1%, ITT(II); : NEXT II: PRINT #FILE
                                                     ---->BC1%, "ITT"
FOR II = 1 TO 7: PRINT #FILEBC1%, DISTL(II); : NEXT II: PRINT #FI
                                                ---->LEBC1%, "DISTL"
PRINT #FILEBC1%, USING "##.##"; 1
PRINT #FILEBC1%,
PRINT #FILEBC1%,
PRINT #FILEBC1%,
END SUB
```

SUB SEARCH

```
'##########
```

```
'Purpose................\
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILEDUM$
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\ GETXXA,
'Comments...............\
'##########
SHARED AAAA$
SHARED FILEDUM$, FILEMRG$, FILEBC1$
SHARED FILEDUM%, FILEMRG%, FILEBC1%
SHARED XX AS FIXCOMB
SHARED SINAL!, COSAL!
SB$ = "(SEARCH "
FRMTAC$ = "### \   \ ###.## ###.##  ### \   \ ###.## ###.##"
FRMTSYMHDR$ = " #####  ### ### ###"
CLOSE 7
'OPEN FILEMRG$ FOR INPUT AS #6
OPEN FILEDUM$ FOR APPEND AS #7
IF LOF(7) = 0 THEN
     PRINT LOG$(SB$, "DUM FILE CAN NOT BE FOUND ")
     EXIT SUB
ELSE
     CLOSE 7: OPEN FILEDUM$ FOR INPUT AS #7
     AAAA$ = "SEARCH-START"
     DO WHILE NOT EOF(7)
          CALL GETXXA(7)                   'GET THE RECORD INTO XX
          CALL SRCHWRK
     LOOP
     CLOSE 6, 7
END IF
'...............................................................
'...............................................................
END SUB    'SEARCH  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB SRCHWRK

```
SHARED XX AS FIXCOMB
SHARED OTN() AS INTEGER, UNKN() AS INTEGER, ITN() AS INTEGER, DIS
                                              ---->TN() AS INTEGER
SHARED OTT(), UNKT(), ITT() AS SINGLE, TT
SHARED ZONEN() AS INTEGER
SHARED DISTL() AS SINGLE
TT = TT + XX.FIXLNGTH
SELECT CASE XX.TGTTYPEN
     CASE 80, 89                             'OUT TRACK
          OTN(1) = OTN(1) + 1: OTT(1) = OTT(1) + XX.FIXLNGTH
          TEMP = XX.FIXLNGTH: TEMP = TEMP * TEMP
          OTT(5) = OTT(5) + TEMP                                'SOSQ's
          IF XX.FIXLNGTH < OTT(6) THEN OTT(6) = XX.FIXLNGTH  'MIN
          IF XX.FIXLNGTH > OTT(7) THEN OTT(7) = XX.FIXLNGTH  'MAX
```

```
          SELECT CASE XX.FIXLNGTH
               CASE IS < 4                  'Noise          2
                    OTN(2) = OTN(2) + 1: OTT(2) = OTT(2) + XX.FIX
                                                        ---->LNGTH
               CASE IS < 13                 'Blink          3
                    OTN(3) = OTN(3) + 1: OTT(3) = OTT(3) + XX.FIX
                                                        ---->LNGTH
               CASE ELSE                    'Long out       4
                    OTN(4) = OTN(4) + 1: OTT(4) = OTT(4) + XX.FIX
                                                        ---->LNGTH
          END SELECT
     CASE 0                                    'In but can't ID target
          UNKN(1) = UNKN(1) + 1: UNKT(1) = UNKT(1) + XX.FIXLNGTH
          TEMP = XX.FIXLNGTH: TEMP = TEMP * TEMP
          UNKT(5) = UNKT(5) + TEMP                              'SOSQ's
          IF XX.FIXLNGTH < UNKT(6) THEN UNKT(6) = XX.FIXLNGTH'MIN
          IF XX.FIXLNGTH > UNKT(7) THEN UNKT(7) = XX.FIXLNGTH'MAX
          SELECT CASE XX.FIXLNGTH
               CASE IS < 4                  '          2
                    UNKN(2) = UNKN(2) + 1: UNKT(2) = UNKT(2) + XX
                                                    ---->.FIXLNGTH
               CASE IS < 13                 '          3
                    UNKN(3) = UNKN(3) + 1: UNKT(3) = UNKT(3) + XX
                                                    ---->.FIXLNGTH
               CASE ELSE                    '          4
                    UNKN(4) = UNKN(4) + 1: UNKT(4) = UNKT(4) + XX
                                                    ---->.FIXLNGTH
          END SELECT
     CASE ELSE                                    'In with target
          ITN(1) = ITN(1) + 1: ITT(1) = ITT(1) + XX.FIXLNGTH
          TEMP = XX.FIXLNGTH: TEMP = TEMP * TEMP
          ITT(5) = ITT(5) + TEMP                                'SOSQ's
          IF XX.FIXLNGTH < ITT(6) THEN ITT(6) = XX.FIXLNGTH  'MIN
          IF XX.FIXLNGTH > ITT(7) THEN ITT(7) = XX.FIXLNGTH  'MAX
          SELECT CASE XX.FIXLNGTH
               CASE IS < 4                  '          2
                    ITN(2) = ITN(2) + 1: ITT(2) = ITT(2) + XX.FIX
                                                        ---->LNGTH
               CASE IS < 13                 '          3
                    ITN(3) = ITN(3) + 1: ITT(3) = ITT(3) + XX.FIX
                                                        ---->LNGTH
               CASE ELSE                         '     4
                    ITN(4) = ITN(4) + 1: ITT(4) = ITT(4) + XX.FIX
                                                        ---->LNGTH
          END SELECT
          DISTN(1) = DISTN(1) + 1: DISTL(1) = DISTL(1) + XX.DISTA
                                                          ---->NCE
          DISTL(5) = DISTL(5) + XX.DISTANCE * XX.DISTANCE 'SOSQ's
          'PRINT XX.DISTANCE; XX.TGTTYPEN
          IF XX.DISTANCE < DISTL(6) THEN DISTL(6) = XX.DISTANCE
                                                      ---->  'MIN
          IF XX.DISTANCE > DISTL(7) THEN DISTL(7) = XX.DISTANCE
                                                      ---->  'MAX
```

```
        SELECT CASE XX.DISTANCE
             CASE IS < .25                    '            2
                  DISTN(2) = DISTN(2) + 1
             CASE IS < .5                     '            3
                  DISTN(3) = DISTN(3) + 1
             CASE ELSE                             '       4
                  DISTN(4) = DISTN(4) + 1
        END SELECT
        ZONEN(1) = ZONEN(1) + 1
        SELECT CASE XX.ZONE
             CASE " 1"                     '            2
                  ZONEN(2) = ZONEN(2) + 1
             CASE " 2"                     '            3
                  ZONEN(3) = ZONEN(3) + 1
             CASE " 3"                     '            4
                  ZONEN(4) = ZONEN(4) + 1
             CASE " 4"                     '            5
                  ZONEN(5) = ZONEN(5) + 1
             CASE " 5"                     '            6
                  ZONEN(6) = ZONEN(6) + 1
             CASE ELSE                       '      7
                  ZONEN(7) = ZONEN(7) + 1
        END SELECT
END SELECT
END SUB
```

PROGRAM CRE8MRG1

```
DEFINT I-N
DECLARE SUB BI2 (BIP%())
DECLARE SUB CRE8MRGFLE ()
DECLARE SUB FIN ()
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB INIT (FILENAME$, FILENAME1$)
DECLARE SUB PUTXX (FILENO%)
DECLARE SUB YESORNO (A$, B$)
CONST pi! = 3.14159
CONST nscanbuf = 400, nfixbuf = 500
CONST SF! = .472, XOFF! = -5.04, YOFF! = -.9, cpi! = 204.8, alpha
                       ---->! = -11.5 * pi! / 180, runoff! = -.34
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                         'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                            'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                     'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE     'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                      'TIME HISTORY FRAME #
     TGTX AS SINGLE                               'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                             'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS INTEGER                                   'DICE
     COUNTDOWN AS INTEGER                                 'DICE
     CONTFIX AS STRING * 1                                'IS THIS
                  ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1                     'SPLADT  S-on, F-off
     AIDON AS STRING * 1                            'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM XX AS FIXCOMB
DIM BI1P(1 TO 7)  AS INTEGER'Buffered Input 1
DIM BI2P(1 TO 7)  AS INTEGER'Buffered Input 2
DIM FILEDT3$, FILEMRG$
DIM FILEDT3%, FILEMRG%
DIM FIXPNTER(1 TO nfixbuf) AS INTEGER
DIM FIXLENGTH(1 TO nfixbuf)   AS INTEGER
DIM INTRACK(1 TO nfixbuf)   AS INTEGER
DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
DIM PUPDIAM(1 TO nfixbuf)   AS INTEGER
DIM SHARED NUMFIX%, NUMSCAN%, NOSTAT%, NUMMRG% '*****COMMON******
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
DIM XLOOK(1 TO nfixbuf)   AS INTEGER
DIM YLOOK(1 TO nfixbuf)   AS INTEGER
'    ** FILE NUMBERS **
'       #1                      #2 DT3                  #3 INDEX
'       #4                      #5                      #6 MRG
'       #7                      #8                      #9
```

```
'
SB$ = "(MAIN "
DIM FILETOT%, FILELOG%
PRINT : PRINT
INPUT " Enter full file descriptor for index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #3
FILETOT$ = "TOTAL.LOG": FILETOT% = 4
OPEN FILETOT$ FOR APPEND AS FILETOT%
DO WHILE NOT EOF(3)
      INPUT #3, FILENAME$
      FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
      CALL INIT(FILENAME$, FILENAME1$)
      CALL CRE8MRGFLE
      PRINT LOG$(SB$, " FINISHED CRE8MRGFLE")
      CALL FIN                                              'Close FILES
LOOP
CLOSE 3, 4
'.....................................................................
'.....................................................................
END     ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB BI2 (BIP%())

```
'##########
'Purpose................\Read the fixation data into circular buf
                                                       ---->fers and\
'                        \compute a few preliminary statistics.
'Parameters.............\BIP%() Circular buffer pointers
'Other input data.......\
'Input files............\FILEDT3% =.DT3
'Output files...........\
'Other output data......\BIP%(), XLOOK(),YLOOK(),PUPDIAM(),FIXLEN
                            ---->GTH(),INTRACK() Circular buffers
' \SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK, NUMINTRACK, NUMOUTTRACK
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED INTRACK() AS INTEGER
SHARED FIXLENGTH() AS INTEGER
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED FIXPNTER() AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED FILEDT3%, FILEMRG%
          'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
NORIB = BIP%(3) - BIP%(2)                   'If buffer low AND EOF=.F.
IF NORIB < 0 THEN NORIB = NORIB + BIP%(1)
IF NORIB < BIP%(4) AND NOT EOF(FILEDT3%) THEN
     FOR I = 1 TO BIP%(5)                               'Load buffer
     GET FILEDT3%, , FIXPNTER(BIP%(3))
```

```
     GET FILEDT3%, , XLOOK(BIP%(3)): GET FILEDT3%, , YLOOK(BIP%(3
                        ---->))                              'read record
     IF NOT EOF(FILEDT3%) THEN
          GET FILEDT3%, , PUPDIAM(BIP%(3)): GET FILEDT3%, , FIXLE
                                                   ---->NGTH(BIP%(3))
          SUMFIXLENGTH = SUMFIXLENGTH + FIXLENGTH(BIP%(3))
                              ---->                'total of fixations
          INTRACK(BIP%(3)) = 1
          IF XLOOK(BIP%(3)) = 0 AND YLOOK(BIP%(3)) = 0 AND PUPDIA
                         ---->M(BIP%(3)) < 11 THEN ' out of track
               INTRACK(BIP%(3)) = 0
               SUMOUTTRACK = SUMOUTTRACK + FIXLENGTH(BIP%(3))'tot
                                                 ---->al out of track
               NUMOUTTRACK = NUMOUTTRACK + 1
          ELSE                           ' in track
               SUMINTRACK = SUMINTRACK + FIXLENGTH(BIP%(3))
                         ---->                          'total intrack
               NUMINTRACK = NUMINTRACK + 1
          END IF
          BIP%(3) = BIP%(3) + 1
          IF BIP%(3) > BIP%(1) THEN BIP%(3) = 1
     ELSE
     EXIT FOR
     END IF
     NEXT I
END IF
BIP%(2) = BIP%(2) + 1: IF BIP%(2) > BIP%(1) THEN BIP%(2) = 1'Incr
                                                    ---->ement first
'.....................................................................
'.....................................................................
END SUB      'BI2  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB CRE8MRGFLE

```
'##########
'Purpose................\ Initial creation of the .MRG file using
                                                    ----> data from
'          \ .DAT and time history pointer array from subroutine
'                       \ FIXPOINTER
'Parameters.............\ none
'Other input data.......\ NUMMRG%, PUPDIAM, FIXPNTER, XLOOK, CPI!
                                                        ---->, YLOOK
'.......................\ FIXLENGTH,
'Input files............\ FILEDT3$= .DAT
'Output files...........\ FILEMRG$= .MRG
'Other output data......\ none
'Function calls.........\ LOG$
'Subroutine calls.......\ BI2, PUTXX
'Comments...............\ Target type is set to 0, "UNK" for in-t
                                                      ---->racks or
'        \ 80, "OUT" for out-tracks. Other fields are initialized
'                       \ to unrealistic constants.
'##########
```

```
SHARED BI2P()  AS INTEGER                          'BufferedInput 2
SHARED FILEDT3$, FILEMRG$
SHARED FILEDT3%, FILEMRG%
SHARED FIXLENGTH() AS INTEGER
SHARED FIXPNTER() AS INTEGER
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED XX AS FIXCOMB
'..............................................................
SB$ = "(CRE8MRGFLE "
CLOSE FILEMRG%
OPEN FILEMRG$ FOR APPEND AS #FILEMRG% ' FIXATION, TIME HISTORY ME
                                                           ---->RGE
NUMMRG& = LOF(FILEMRG%)                        'can the file be found
IF NUMMRG& <> 0 THEN
     PRINT
     PRINT LOG$(SB$, FILEMRG$ + " is not empty and you are trying
                                  ----> to OPEN it for output")
     PRINTLOG$(SB$, "NUMFIX% = "); NUMFIX%; "NUMMRG& = "; NUMMRG&
     A$ = "Do you want to SKIP " + FILEMRG$ + " and exit CRE8MRGF
                                         ---->LE subroutine"
     A$ = A$ + "  If (no) then old data is purged"
     CALL YESORNO(A$, B$)
     IF B$ = "Y" THEN
          CLOSE FILEMRG%: EXIT SUB
     ELSE
          CLOSE FILEMRG%: OPEN FILEMRG$ FOR OUTPUT AS #FILEMRG%
     END IF
END IF
'..............................................................
XX.TGTTYPEN = 0: XX.TGTTYPEC = "UNK": XX.FIXLNGTH = 0: XX.PUPDIAM
                                   ---->% = 0: XX.TGTID = "Jil"
XX.DISTANCE = 99.99: XX.FRAMENO = 9999: XX.TGTX = 0: XX.TGTY = 0:
                                ----> XX.FIXX = 0: XX.FIXY = 0
XX.HEADING = 999: XX.COUNTDOWN = 0: XX.CONTFIX = "Z": XX.CROSSCHE
                                   ---->CK = "Z": XX.ZONE = "99"
XX.SPEED = "F": XX.AIDON = "F": XX.SPARE = "        "
SEEK #FILEDT3%, 1          'REWIND FILE & RESET BUFFER           '
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                          ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
'..............................................................
FOR I = 1 TO NUMFIX%
     CALL BI2(BI2P())
     II = BI2P(2)
     XX.PUPDIAM = PUPDIAM(II): XX.FRAMENO = FIXPNTER(II): XX.FIXX
                                     ----> = XLOOK(II) / cpi!
     XX.FIXY = YLOOK(II) / cpi!: XX.FIXLNGTH = FIXLENGTH(II)
     IF XLOOK(II) = 0 AND YLOOK(II) = 0 AND PUPDIAM(II) < 11 THEN
                                   ---->       ' out of track
          XX.TGTTYPEN = 89: XX.TGTTYPEC = "OUT"
          IF FIXLENGTH(II) < 13 THEN XX.TGTTYPEN = 80: XX.TGTTYPE
```

```
                                                ---->C = "BLNK"
      END IF
      CALL PUTXX(FILEMRG%)
      XX.TGTTYPEN = 0: XX.TGTTYPEC = "UNK"
NEXT I
CLOSE FILEMRG%
'...............................................................
'...............................................................

END SUB      'CRESMRGFLE  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB FIN

```
'###########
'Purpose................\ Close all files, scale and output a few
                                                ----> statistics
'Parameters.............\
'Other input data.......\ SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK,
                                   ---->NUMINTRACK, NUMOUTTRACK
'Input files............\
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'###########
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED FILETOT%, FILELOG%
SB$ = "(FIN "
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
      PRINT #FILELOG%, LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= ")
                     ---->; NUMINTRACK     ' totals for .txt file
      PRINT #FILELOG%, LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS=")
                                                ---->; NUMOUTTRACK
      PRINT #FILELOG%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
      PRINT #FILELOG%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK /
                                                          ---->30
      PRINT #FILELOG%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH
                                                        ---->/ 30
      PRINT #FILELOG%, USING "& ###.## OR ##.## SECONDS"; LOG$(SB$
---->, "AVERAGE IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRAC
                         ---->K; SUMINTRACK / NUMINTRACK / 30
      PRINT #FILELOG%, USING "& ###.## OR ##.## SECONDS"; LOG$(SB$
---->, "AVERAGE OUT TRACK FIXATION IS  "); SUMOUTTRACK / NUMOUTTR
                      ---->ACK; SUMOUTTRACK / NUMOUTTRACK / 30
      '
      PRINT #FILETOT%, LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= ")
                     ---->; NUMINTRACK     ' totals for .txt file
      PRINT #FILETOT%, LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS=")
```

```
                                               ---->; NUMOUTTRACK
     PRINT #FILETOT%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
     PRINT #FILETOT%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK /
                                                          ---->30
     PRINT #FILETOT%, USING "& ###### OR ####.## SECONDS"; LOG$(S
---->B$, "TOTAL FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH
                                                        ---->/ 30
     PRINT #FILETOT%, USING "& ###.## OR ##.## SECONDS"; LOG$(SB$
---->, "AVERAGE IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRAC
                          ---->K; SUMINTRACK / NUMINTRACK / 30
     PRINT #FILETOT%, USING "& ###.## OR ##.## SECONDS"; LOG$(SB$
---->, "AVERAGE OUT TRACK FIXATION IS  "); SUMOUTTRACK / NUMOUTTR
                       ---->ACK; SUMOUTTRACK / NUMOUTTRACK / 30
END IF
CLOSE 2, 5, 6
'...............................................................
'...............................................................
END SUB      ' FIN  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose.................\Initialize parameters on circular buffer
'               \ Initialize sums to zero.  Let user choose partic-
'                  \ ular run for analysis.  Determine aid type for
'                        \ subsequent branching. Open  FILEDT3$,
'                        \  and store their lengths.
'Parameters..............\ none
'Other input data........\
'Input files.............\  FILEDT3$,
'Output files............\
'Other output data.......\ File names & unit #'s. Initialized vari
                                                  ---->ables, sums
                         \ and pointers
'Function calls..........\ LOG$
'Subroutine calls........\ none
'Comments................\
'##########
SHARED BI2P()  AS INTEGER                         'BufferedInput 2
SHARED FILEDT3$, FILEMRG$
SHARED FILEDT3%, FILEMRG%
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED FILETOT%, FILELOG%
SB$ = "(INIT "
        'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                         ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
NUMFIX% = 0: NUMSCAN% = 0
SUMFIXLENGTH = 0: SUMINTRACK = 0: SUMOUTTRACK = 0
```

```
NUMINTRACK = 0: NUMOUTTRACK = 0
'...............................................................
IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
FILEDT3$ = FILENAME$ + ".DT3": FILEDT3% = 2    'append extension
FILEMRG$ = FILENAME$ + ".MRG": FILEMRG% = 6
FILELOG$ = FILENAME$ + ".LOG": FILELOG% = 5
'                     COMMENT: FILE NAME SHOULD LOOK LIKE
'                               "C:\FASAFILE\CRONE\CC10SLCE"
'...............................................................
OPEN FILEDT3$ FOR BINARY AS #2 'oculometer .dat file
NUMFIX% = LOF(2) / 10      'can the file be found
IF NUMFIX% = 0 THEN
    STOP
ELSE
    PRINT LOG$(SB$, "NUMBER OF FIXATIONS IS  "); NUMFIX%
END IF
OPEN FILELOG$ FOR APPEND AS FILELOG%
IF LOF(FILELOG%) = 0 THEN PRINT "Log file is empty "; FILENAME$:
                                                        ---->STOP
PRINT #FILELOG%, "**********************************************
                                      ---->*************************"
PRINT #FILELOG%, LOG$(SB$, " Making Merge file from .DT3 after fi
                                        ---->ltering is complete")
PRINT #FILELOG%, LOG$(SB$, FILENAME$ + "  NUMBER OF FIXATIONS IS
                                                 ----> "); NUMFIX%
PRINT #FILETOT%, "**********************************************
                                      ---->*************************"
PRINT #FILETOT%, LOG$(SB$, " Making Merge file from .DT3 after fi
                                        ---->ltering is complete")
PRINT #FILETOT%, LOG$(SB$, FILENAME$ + "  NUMBER OF FIXATIONS IS
                                                 ----> "); NUMFIX%
'...............................................................
'...............................................................
END SUB       ' INIT   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PUTXX (FILENO%)

```
'##########
'Purpose................\ Writes a record from XX to the appropri
                                                    ---->ate file.
'Parameters.............\ FILENO%
'Other input data.......\ XX
'Input files............\ none
'Output files...........\ FILEMRG$
'Other output data......\ none
'Function calls.........\ none
'Subroutine calls.......\ none
'Comments...............\ This makes it easier to modify record f
                                                        ---->orm.
'##########
'Write the array XX to a record on the FILEMRG$ file.
SHARED XX AS FIXCOMB
WRITE #FILENO%, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM
```

```
---->, XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FI
     ---->XX, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.C
          ---->ROSSCHECK, XX.ZONE, XX.SPEED, XX.AIDON, XX.SPARE
'..............................................................
'..............................................................
END SUB       ' PUTXX     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

PROGRAM CRI8IDX

```
DECLARE SUB FIN ()
DECLARE FUNCTION LOG$ (SB$, A$)
DEFINT I-N
'##########
'Purpose................\ Creates FILEIDX$ which contains pointer
                                                         ---->s at the
'            \ first byte of each time history record on FILEACP$
'Parameters.............\
'Other input data.......\ File names and unit numbers
'Input files............\ FILEACP$
'Output files...........\ FILEIDX$
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\ Assumes .ACP is open and pointer is at
                                                        ---->the first
'          \ byte of the time history i.e. beyond the file title.
'##########
DIM IDX(1 TO 2000) AS LONG
INPUT " Enter full file descriptor for index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #1
DO WHILE NOT EOF(1)
     INPUT #1, FILENAME$
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     'INPUT " ENTER OCULOMETER FILE NAME, NO EXTENSION   ", FILEN
                                                            ---->AME$
     'FILENAME$ = COMMAND$
     IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
     SB$ = "(CRI8IDX "
     FILEACP$ = FILENAME$ + ".ACP": FILEACP% = 3'append extension
     FILEIDX$ = FILENAME$ + ".IDX": FILEIDX% = 4
     '..........................................................
     OPEN FILEACP$ FOR INPUT AS FILEACP%
     NOBACP& = LOF(FILEACP%)        'can the file be found
     IF NOBACP& = 0 THEN
          PRINT LOG$(SB$, FILEACP$); "FILE NOT FOUND"
                                               ---->  ' fix this test
          CALL FIN
     ELSE : PRINT LOG$(SB$, "NUMBER OF ACP BYTES IS  "); NOBACP&
     END IF
     INPUT #FILEACP%, FILEHDR$
     INPUT #FILEACP%, FILEHDR$
     PRINT LOG$(SB$, FILEHDR$)
     '..........................................................
     CLOSE FILEIDX%
     OPEN FILEIDX$ FOR APPEND AS #FILEIDX%
     IF LOF(FILEIDX%) = 0 THEN
          IDX(1) = SEEK(FILEACP%)
           ----> 'assumes INIT set the pointer at the 1'st record
          PRINT #FILEIDX%, USING "########"; IDX(1)      'write
                                   ---->byte position of 1'st record
          I = 1
```

```
        DO WHILE NOT EOF(FILEACP%)
             I = I + 1
             INPUT #FILEACP%, NOAC%, T%
             IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " A***EOF(FI
                                 ---->LEACP%)***"); I; J: EXIT DO
             FOR J = 1 TO NOAC%
                  LINE INPUT #FILEACP%, A$        'INPUT AIRCRAFT
                  IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " B***E
                             ---->OF(FILEACP%)***"); I; J: EXIT FOR
             NEXT J
             IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " C***EOF(FI
                                 ---->LEACP%)***"); I; J: EXIT DO
             INPUT #FILEACP%, NOTURNS%, NOSLOTS%, NODICE%
             IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " D***EOF(FI
                                 ---->LEACP%)***"); I; J: EXIT DO
             NOSYM% = NOTURNS% + NOSLOTS% + NODICE%
             IF NOSYM% <> 0 THEN
                  FOR J = 1 TO NOSYM%
                       K = J + NOAC%
                       LINE INPUT #FILEACP%, A$  'INPUT SYMBOLS
                       IF EOF(FILEACP%) THEN PRINT LOG$(SB$, "
                      ---->E***EOF(FILEACP%)***"); I; J: EXIT FOR
                  NEXT J
             END IF
             IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " F***EOF(FI
                                 ---->LEACP%)***"); I; J: EXIT DO
             IDX(I) = SEEK(FILEACP%)                    'write byt
                                    ---->e position of i'th record
             PRINT #FILEIDX%, USING "########"; IDX(I)
        LOOP
        PRINT LOG$(SB$, "; COMPLETED GENERATION OF TIME HISTORY
                        ----> INDEX FILE "); I; " RECORDS "; ""
   ELSE
        PRINT LOG$(SB$, " ERROR-TIME HISTORY INDEX FILE ALREADY
                            ----> EXISTS.  FILE NOT GENERATED")
        CALL FIN
   END IF
   CLOSE FILEIDX%, FILEACP%
LOOP
'..............................................................
'..............................................................
END      ' MAIN PROGRAM
```

SUB FIN

```
'##########
'Purpose................\ Close all files, scale and output a few
                                                ----> statistics
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
```

```
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'##########
SB$ = "(FIN "
CLOSE 1, 2, 3, 4, 5, 6, 7, 9
'..............................................................
'..............................................................
END SUB        ' FIN  'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

PROGRAM CRI8MGX

```
DEFINT I-N
DECLARE FUNCTION LOG$ (SB$, A$)
'    ** FILE NUMBERS **
'       #1                    #2                       #3 FLEINDX1
'       #4                    #5                       #6 MRG
'       #7 MGX                #8                       #9
'
XXX$ = "## \  \ #### #### \ \ ##.## #### ###.## ###.## ###.## ###
                                   ---->.## ### #### !  ! \\"
SB$ = " CRI8MGX ("
INPUT " Enter full file descriptor for index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #3
DO WHILE NOT EOF(3)
     INPUT #3, FILENAME$
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     FILEMRG$ = FILENAME$ + ".MRG": FILEMRG% = 6
     OPEN FILEMRG$ FOR INPUT AS #FILEMRG% '  MERGE FILE
     NUMFIX& = LOF(FILEMRG%)        'can the file be found
     IF NUMFIX& = 0 THEN
          PRINT LOG$(SB$, FILEMRG$); "FILE NOT FOUND"
                                          ----> ' fix this test
          EXIT DO
     ELSE : PRINT LOG$(SB$, "NUMBER OF BYTES ON .MRG FILE IS  ");
                                                  ----> NUMFIX&
     END IF
     FILEMGX$ = FILENAME$ + ".MGX": FILEMGX% = 7
     OPEN FILEMGX$ FOR APPEND AS #FILEMGX%
     IF LOF(FILEMGX%) <> 0 THEN
          PRINT "Index file already exists", FILENAME$
          EXIT DO
     ELSE
          CLOSE FILEMGX%
          OPEN FILEMGX$ FOR RANDOM AS FILEMGX% LEN = 4
     END IF
     DO WHILE NOT EOF(FILEMRG%)
          N& = SEEK(FILEMRG%)
          PUT #FILEMGX%, , N&
          LINE INPUT #FILEMRG%, A$
     LOOP
     NOR = LOF(FILEMGX%) / 4
     CLOSE #FILEMRG%, FILEMGX%
     PRINT LOG$(SB$, FILENAME1$ + "  Number of records ="); NOR
LOOP
CLOSE 3
'................................................................
'................................................................
END     ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

PROGRAM CROSS1

```
DEFINT I-N
DECLARE SUB PRINTSEQ (LAPOVER%)
DECLARE SUB FINDAB (IOK%)
DECLARE SUB FILLBUF ()
DECLARE SUB FIN ()
DECLARE SUB INIT (FILENAME$, FILENAME1$)
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB GETXXB (FILENO%, NEOFMRG%)
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                              'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                                 'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                            'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE       'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                         'TIME HISTORY FRAME #
     TGTX AS SINGLE                                  'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                                'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS STRING * 3                                    'DICE
     COUNTDOWN AS INTEGER                                     'DICE
     CONTFIX AS STRING * 1                                    'IS THIS
               ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1                         'SPLADT  S-on, F-off
     AIDON AS STRING * 1                                 'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM AAAA$
DIM DTEMP1 AS FIXCOMB
DIM NBUFPARM(4), ITRIG
'  SIZE, FIRST, LAST, EOF
DIM FLAGFUL AS INTEGER, NXRECNO AS INTEGER
ITRIG = 5: NBUFPARM(1) = 100
DIM FIXBUF(NBUFPARM(1) + 1) AS FIXCOMB
DIM ASTORE AS FIXCOMB, ARECNO AS INTEGER, ENDRECNO AS INTEGER, BS
                    ---->TORE AS FIXCOMB, ENDSTORE AS FIXCOMB
DIM LSEQ, LASTREC, ISEQORD
DIM APSTORE AS FIXCOMB, BPSTORE AS FIXCOMB, APRECNO, ENDPRECNO, L
                              ---->PSEQ, ISEQTP, ITLAP, ITLAPP
DIM IAT, IET, ISEQT, NONSEQT, TOTSEQT, TOTLPV, TOTNSEQT  '*TIME
DIM IAID AS INTEGER
DIM FILEMRG$
DIM FILEMRG%
'DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
'DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
'    ** FILE NUMBERS **
'       #1 FILE INDEX #2 MRG                     #3 CROSS.PRT(PROUT)
'       #4                      #5 LOG                   #6
'       #7 CCS                  #8                       #9
```

```
'
SPSINV! = 1 / 30: SB$ = " (CROSS1  "
'NOSTAT% = 10
AAAA$ = " MAIN-START MAIN LOOP"
HEADING$ = "  SEQ#  STRT-FNSH TOG  TYPE     TYPE      TGTID   TIM
                            ---->E  ZONE   AID   SPD   LPOV  DSTAB"
INPUT " Enter full file descriptor for Index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #1
DO WHILE NOT EOF(1)
     INPUT #1, FILENAME$
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     CALL INIT(FILENAME$, FILENAME1$)
     OPEN FILENAME$ + ".CCS" FOR OUTPUT AS #7
     'PRINT #7, HEADING$
     LENDRECNO = 0: ISEQORD = 0: TOTSEQT = 0: TOTLPV = 0: TOTNSEQ
                                                     ---->T = 0
     SUM = 0
     PRINT LOG$(SB$, " Start"): PRINT #5, LOG$(SB$, " Start")
     DO
     CALL FINDAB(IOK)
     TOTSEQT = TOTSEQT + ISEQT
     TOTNSEQT = TOTNSEQT + NONSEQT
     LAPOVER = O: IF ARECNO = LENDRECNO AND IOK = 1 THEN LAPOVER
                                                      ---->= 1
     IF LAPOVER = 1 THEN SUM = SUM + ASTORE.FIXLNGTH
     IF IOK = 1 THEN CALL PRINTSEQ(LAPOVER)
     LENDRECNO = ENDRECNO
     IF LASTREC = 1 THEN EXIT DO
     LOOP
     CALL PRINTSEQ(0)           'FLUSH BUFFER
     PRINT " # of MRG records processed", NXRECNO
     PRINT #5, " # of MRG records processed", NXRECNO
     PRINT " # OF resulting sequences", ISEQORD
     PRINT #5, " # OF resulting sequences", ISEQORD
     PRINT " Total sequence time", TOTSEQT
     PRINT #5, " Total sequence time", TOTSEQT
     PRINT " Minus Lap over time", -SUM
     PRINT #5, " Minus Lap over time", -SUM
     PRINT " Plus time not in sequence", TOTNSEQT
     PRINT #5, " Plus time not in sequence", TOTNSEQT
     PRINT " Equals total time", TOTSEQT - SUM + TOTNSEQT
     PRINT #5, " Equals total time", TOTSEQT - SUM + TOTNSEQT
     PRINT LOG$(SB$, " FINISHED SEARCH"): PRINT #5, LOG$(SB$, " F
                                            ---->INISHED SEARCH")
     CLOSE 2, 3, 5, 7
     'CALL FIN                                    'Close FILES
LOOP
CLOSE
PRINT AAAA$
'.............................................................
'.............................................................
END    ' MAIN PROGRAM
```

SUB FILLBUF

```
'##########
'Purpose................\ Fill the Merge file buffer and set fina
                                                    ---->l record
'                         pointer
'Parameters.............\
'Other input data.......\ filemrg%, size, first, last in "NBUFPAR
                                                         ---->M()"
'Input files............\ FILEMRG$
'Output files...........\
'Other output data......\ last, final in "NBUFPARM()"
'Function calls.........\
'Subroutine calls.......\ GETXXB
'Comments...............\
'##########
SHARED NBUFPARM(), ITRIG
'  SIZE, FIRST, LAST, EOF
SHARED FIXBUF() AS FIXCOMB
SHARED DTEMP1 AS FIXCOMB
SHARED FILEMRG$
SHARED FILEMRG%
IF NBUFPARM(3) >= NBUFPARM(2) THEN
     NIB = (NBUFPARM(3) - NBUFPARM(2) + 1) MOD (NBUFPARM(1) + 1)
ELSE
     NIB = (NBUFPARM(1) - NBUFPARM(2) + 2 + NBUFPARM(3)) MOD (NBU
                                              ---->FPARM(1) + 1)
END IF
IF (NIB < (.8 * NBUFPARM(1))) AND NBUFPARM(4) = 0 THEN 'NOT EOF A
                                                 ---->ND NOT FULL
     NEOFMRG = 0
     ITEMP = NBUFPARM(3)
     FOR I = 1 TO NBUFPARM(1) - NIB
     L = ((ITEMP + I - 1) MOD (NBUFPARM(1) + 1)) + 1
     CALL GETXXB(FILEMRG%, NEOFMRG)
     FIXBUF(L) = DTEMP1
     NBUFPARM(3) = L
     IF NEOFMRG = 1 THEN NBUFPARM(4) = L: EXIT FOR
     NEXT I
END IF
'..........................................................
'..........................................................
END SUB      ' FILLBUF'
```

SUB FIN

```
'##########
'Purpose................\ Close all files, scale and output a few
                                                 ----> statistics
'Parameters.............\
'Other input data.......\ SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK,
                                     ---->NUMINTRACK, NUMOUTTRACK
'Input files............\
'Output files...........\
```

```
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'##########
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
     PRINT LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= "); NUMINTRAC
                                    ---->K;      ' totals for .txt file
     PRINT LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS="); NUMOUTTRA
                                                            ---->CK
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
           ---->TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
         ---->TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK / 30
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
       ---->FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH / 30
     TT1 = SUMINTRACK / NUMINTRACK / 30
     PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
     ---->IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRACK; TT1
     TT1 = SUMOUTTRACK / NUMOUTTRACK / 30
     PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
    ---->OUT TRACK FIXATION IS "); SUMOUTTRACK / NUMOUTTRACK; TT1
END IF
CLOSE 2, 3, 4, 5, 6, 7
'...............................................................
'...............................................................
END SUB      ' FIN
```

SUB FINDAB (IOK)

```
SHARED ASTORE AS FIXCOMB, ARECNO AS INTEGER, ENDRECNO AS INTEGER,
                          ----> BSTORE AS FIXCOMB, ENDSTORE AS FIXCOMB
SHARED LSEQ, LASTREC, ISEQORD
SHARED NBUFPARM(), ITRIG
SHARED IAT, IET, ISEQT, NONSEQT, TOTSEQT, TOTLPV, TOTNSEQT   '***
                                                         ---->*TIME
'  SIZE, FIRST, LAST, EOF
SHARED FIXBUF() AS FIXCOMB
SHARED FLAGFUL AS INTEGER, NXRECNO AS INTEGER
IAT = O: IET = 0: ISEQT = O: NONSEQT = 0: LTEBT = 0: LTEET = 0 '*
                                                       ---->***TIME
IOK = 0: LSEQ = 0: ARECNO = 0: ENDRECNO = 0: IHITA = 0: LASTREC =
                                                            ----> 0
DO                                     ' FIND A
     IF FLAGFUL <> 1 THEN CALL FILLBUF: FLAGFUL = 1
     J = 0
     FOR I = NBUFPARM(2) TO NBUFPARM(2) + NBUFPARM(1) - 1
     L = ((I - 1) MOD (NBUFPARM(1) + 1)) + 1
     J = J + 1
     ITEST = FIXBUF(L).TGTTYPEN <> 0 AND FIXBUF(L).TGTTYPEN <> 80
                                   ----> AND FIXBUF(L).TGTTYPEN <> 89
```

```
     NONSEQT = NONSEQT + FIXBUF(L).FIXLNGTH                  '****TIME
     IF ITEST THEN IHITA = 1: IAT = FIXBUF(L).FIXLNGTH: NONSEQT =
                              ----> NONSEQT - IAT        '****TIME
     IF L = NBUFPARM(4) THEN LASTREC = 1
     IF IHITA = 1 OR LASTREC = 1 THEN EXIT FOR                ' Found
                                                  ---->A OR LAST REC
     IF L = NBUFPARM(3) THEN EXIT FOR
     NEXT I
     NXRECNO = NXRECNO + J      ' A not found, Load another buffer
     NBUFPARM(2) = (L MOD (NBUFPARM(1) + 1)) + 1: FLAGFUL = 0
     IF IHITA = 1 OR LASTREC = 1 THEN EXIT DO                ' Found A
                                                   ----> OR LAST REC
LOOP
IF LASTREC = 1 THEN      '****TIME   ' HIT LAST RECORD BEFORE A
     IF IHITA = 1 THEN NONSEQT = NONSEQT + IAT
     EXIT SUB
END IF
ARECNO = NXRECNO - 1: ASTORE = FIXBUF(L)
'.....................................................................
'
IHITB = 0
IBREAK = 0
DO                                   ' FIND B
     IF FLAGFUL <> 1 THEN CALL FILLBUF: FLAGFUL = 1
     J = 0
     FOR I = NBUFPARM(2) TO NBUFPARM(2) + NBUFPARM(1) - 1
     L = ((I - 1) MOD (NBUFPARM(1) + 1)) + 1
     J = J + 1
     LTEBT = LTEBT + FIXBUF(L).FIXLNGTH                '****TIME
     IF L = NBUFPARM(4) THEN LASTREC = 1
     ITEST3 = FIXBUF(L).TGTTYPEN = 89 OR (FIXBUF(L).TGTTYPEN = 0
                             ---->AND FIXBUF(L).FIXLNGTH > 12)
     IF ITEST3 THEN IBREAK = 1
     'no sequence, Start again with A
     ITEST1 = FIXBUF(L).TGTTYPEN = 80 OR (FIXBUF(L).TGTTYPEN = 0
                             ---->AND FIXBUF(L).FIXLNGTH < 13)
     'blink or unk < 13
     ITEST2 = FIXBUF(L).TGTTYPEN = ASTORE.TGTTYPEN AND FIXBUF(L).
                                         ---->TGTID = ASTORE.TGTID
     'Back to A
     IF NOT ITEST1 AND NOT ITEST2 AND NOT ITEST3 THEN IHITB = 1
                                                 ---->     ' Found B
     IF IHITB = 1 OR LASTREC = 1 OR IBREAK = 1 THEN EXIT FOR
     IF L = NBUFPARM(3) THEN EXIT FOR
     NEXT I
     NXRECNO = NXRECNO + J     ' B not found, Load another buffer.
     NBUFPARM(2) = (L MOD (NBUFPARM(1) + 1)) + 1: IF J > ITRIG TH
                                                ---->EN FLAGFUL = 0
     IF ITEST3 OR IHITB = 1 OR LASTREC = 1 THEN EXIT DO
LOOP
            ' END FIND B
IF IHITB = 1 THEN
     NXRECNO = NXRECNO - 1
```

```
    NBUFPARM(2) = 1 + (NBUFPARM(2) + NBUFPARM(1) - 1) MOD (NBUFP
                                             ---->ARM(1) + 1)
    ENDRECNO = NXRECNO: BSTORE = FIXBUF(L)
    IOK = 1: LSEQ = 1
    ENDSTORE = BSTORE
    ISEQT = IAT + LTEBT                                  '****TIME
ELSE
    NONSEQT = NONSEQT + LTEBT + IAT                       '****TIME
    'IF LASTREC <> 1 THEN NONSEQT = NONSEQT - FIXBUF(L).FIXLNGTH
END IF
IF LASTREC = 1 OR IBREAK = 1 THEN EXIT SUB
'.............................................................
DO
    IF FLAGFUL <> 1 THEN CALL FILLBUF: FLAGFUL = 1
    J = 0
    FOR I = NBUFPARM(2) + 1 TO NBUFPARM(2) + NBUFPARM(1) - 1
    L = ((I - 1) MOD (NBUFPARM(1) + 1)) + 1
    J = J + 1
    IF L = NBUFPARM(4) THEN LASTREC = 1
    LTEET = LTEET + FIXBUF(L).FIXLNGTH
    ITEST1 = FIXBUF(L).TGTTYPEN = 80 OR (FIXBUF(L).TGTTYPEN = 0
                              ---->AND FIXBUF(L).FIXLNGTH < 13)
    ITEST2 = FIXBUF(L).TGTTYPEN = ASTORE.TGTTYPEN AND FIXBUF(L).
                                    ---->TGTID = ASTORE.TGTID
    ITEST3 = FIXBUF(L).TGTTYPEN = 89 OR (FIXBUF(L).TGTTYPEN = 0
                              ---->AND FIXBUF(L).FIXLNGTH > 12)
    ITEST4 = FIXBUF(L).TGTTYPEN = BSTORE.TGTTYPEN AND FIXBUF(L).
                                    ---->TGTID = BSTORE.TGTID
    IF LASTREC <> 1 AND NOT (ITEST2 OR ITEST3 OR ITEST4) THEN
        IF NOT ITEST1 THEN     ' Third target.
        EXIT SUB
        END IF
    ELSE
        EXIT FOR  'EOF or HIT or BREAK
    END IF
    IF L = NBUFPARM(3) THEN EXIT FOR  'buffer empty
    NEXT I
    NXRECNO = NXRECNO + J
    NBUFPARM(2) = L: IF J > ITRIG THEN FLAGFUL = 0
    IF (ITEST2 OR ITEST4) THEN              ' A or B
    ITEST5 = FIXBUF(L).TGTTYPEN = ENDSTORE.TGTTYPEN AND FIXBUF(L
                                 ---->).TGTID = ENDSTORE.TGTID
    IF NOT ITEST5 THEN LSEQ = LSEQ + 1: ENDSTORE = FIXBUF(L)
    ISEQT = ISEQT + LTEET: LTEET = 0                  '****TIME
    IF LASTREC = 1 THEN EXIT SUB
    ELSE
    NONSEQT = NONSEQT + LTEET     '****TIME
    NXRECNO = NXRECNO + 1                    'advance poinnters
    NBUFPARM(2) = 1 + NBUFPARM(2) MOD (NBUFPARM(1) + 1)
    EXIT SUB
    END IF
    ENDRECNO = ENDRECNO + J
LOOP
```

```
PRINT "NXRECNO = ENDRECNO      ' ."
PRINT "EXIT SUB"
'...............................................................
'...............................................................
END SUB   ' FINDAB
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                    ---->buffers
'              \ Initialize sums to zero.  Let user choose partic-
'                 \ ular run for analysis.  Determine aid type for
'                 \ subsequent branching. Open FILESCN$, FILEDAT$,
'                        \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                ---->ables, sums
'                     \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED NBUFPARM(), ITRIG
'  SIZE, FIRST, LAST, EOF
SHARED FLAGFUL AS INTEGER, NXRECNO AS INTEGER
SHARED FILEMRG$
SHARED FILEMRG%
SHARED IAID AS INTEGER
STAR$ = " *******************************************************
                                                ---->**********"
SB$ = "(INIT "
NBUFPARM(2) = 1: NBUFPARM(3) = NBUFPARM(1) + 1: NBUFPARM(4) = 0
NXRECNO = 1
'...............................................................
'...............................................................
SELECT CASE MID$(FILENAME1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
          IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                  ---->C,GR or SL"): PRINT : STOP
```

```
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                    ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT STAR$: PRINT FILENAME$, IAID
FILEMRG$ = FILENAME$ + ".MRG": FILEMRG% = 2
OPEN FILEMRG$ FOR INPUT AS #FILEMRG%
LOGFILE$ = FILENAME$ + ".LOG": LOGFILE% = 5
OPEN LOGFILE$ FOR APPEND AS #LOGFILE%
PRINT #5, STAR$: PRINT #5, FILENAME$, IAID
OPEN "CROSS.PRT" FOR OUTPUT AS #3
CALL FILLBUF: FLAGFUL = 1
'.............................................................
'.............................................................
END SUB       ' INIT
```

SUB PRINTSEQ (LAPOVER)

```
SHARED IAID AS INTEGER
SHARED ASTORE AS FIXCOMB, ARECNO AS INTEGER, ENDRECNO AS INTEGER,
                    ----> BSTORE AS FIXCOMB, ENDSTORE AS FIXCOMB
SHARED APSTORE AS FIXCOMB, BPSTORE AS FIXCOMB, APRECNO, ENDPRECNO
                             ---->, LPSEQ, ISEQTP, ITLAP, ITLAPP
SHARED LSEQ, LASTREC, ISEQORD
SHARED IAT, IET, ISEQT, NONSEQT, TOTSEQT, TOTLPV, TOTNSEQT  '****
                                                        ---->TIME
XXX1$ = "  ####  #### ####  ##  ## ##"
XXX2$ = "  \  \ \  \  \ \ \ \  ####"
XXX3$ = "  ! !  ! !"
XXX4$ = "  ! !  ####  ##.#"
XXX5$ = "  \ \ \ \  #### ####"
ITLAP = 0
ISEQORD = ISEQORD + 1
IF LAPOVER = 1 THEN
     ITLAP = ASTORE.FIXLNGTH
     ITLAPP = ITLAPP + ASTORE.FIXLNGTH
END IF
IF ISEQORD > 1 THEN
     D = SQR((APSTORE.TGTX - BPSTORE.TGTX) ^ 2 + (APSTORE.TGTY -
                                      ---->BPSTORE.TGTY) ^ 2)
     ZA$ = LEFT$(APSTORE.ZONE, 1): ZB$ = LEFT$(BPSTORE.ZONE, 1)
     PRINT #7, USING XXX1$; ISEQORD - 1; APRECNO; ENDPRECNO; LPSE
                    ---->Q; APSTORE.TGTTYPEN; BPSTORE.TGTTYPEN;
     PRINT #7, USING XXX2$; APSTORE.TGTTYPEC; BPSTORE.TGTTYPEC; A
                          ---->PSTORE.TGTID; BPSTORE.TGTID; ISEQTP;
     PRINT #7, USING XXX3$; ZA$; ZB$; APSTORE.AIDON; BPSTORE.AIDO
                                                          ---->N;
     PRINT #7, USING XXX4$; APSTORE.SPEED; BPSTORE.SPEED; ITLAPP;
                                                        ----> D;
     PRINT #7, USING XXX5$; APSTORE.HEADING; BPSTORE.HEADING; APS
                          ---->TORE.COUNTDOWN; BPSTORE.COUNTDOWN
     TOTLPV = TOTLPV + ITLAPP
END IF
APSTORE = ASTORE: BPSTORE = BSTORE: APRECNO = ARECNO: ENDPRECNO =
```

```
                                                     ----> ENDRECNO
LPSEQ = LSEQ: ISEQTP = ISEQT: ITLAPP = ITLAP
END SUB
```

PROGRAM CUT20

```
'DEFINT I-N
START:
TYPE DDAT
     TT AS INTEGER
     XX AS INTEGER
     YY AS INTEGER
     PD AS INTEGER
     FT AS INTEGER
END TYPE
DIM X1 AS DDAT
DIM X2 AS DDAT
DIM X3 AS DDAT
AA$ = "**************************************************************
AB$ = "  CUT20.BAS  " + DATE$ + " " + TIME$
AC$ = "  In accordance with the 8/28/91 memo from DJC to RBF 20 r
                                                    ---->ecords from"
AD$ = "  14 files need to be cut out. The record(s) taken from th
                                                    ---->is file are"
AE$ = "  listed here. "
INPUT " Enter full file descriptor for index file ", INDEX$
OPEN INDEX$ FOR INPUT AS #3                       'Input directives
LGTOT$ = LEFT$(INDEX$, LEN(INDEX$) - 8) + "LOG.TOT"
OPEN LGTOT$ FOR OUTPUT AS #4                  'Concatenated log file
DO WHILE NOT EOF(3)        ' Loop through one subject's files   #3
     PRINT : PRINT
     INPUT #3, FLE$                         ' FILE NAME
     IF FLE$ = "" THEN EXIT DO
     INPUT #3, NOD                          ' NUMBER OF DELETIONS
     FOR I = 1 TO NOD
     INPUT #3, NOR(I)                   'Record #'s to be deleted
     NEXT I
     NOR(NOD + 1) = 0
     NN$ = FLE$ + ".DT2"
     N1$ = FLE$ + ".DT1"
     LG$ = FLE$ + ".LOG"
     OPEN LG$ FOR APPEND AS #5
     PRINT NN$
     T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN N1$ FOR INPUT AS #1     'Can find .DT1 file??    #1
     ON ERROR GOTO 0
     IF T% = 1 THEN GOTO START
     CLOSE 1
     OPEN "R", #1, N1$, 10              ' Open .DT1 in random mode
     LENFLE% = LOF(1) / 10
     DN1 = LENFLE%                       '  Floating point
     OPEN NN$ FOR RANDOM AS #8 LEN = 10
     IF LOF(8) <> 0 THEN STOP     '  Open DT2 in random mode    #8
     PRINT #4, AA$: PRINT #4, AB$, FLE$: PRINT #4,
     PRINT #4, AC$: PRINT #4, AD$: PRINT #4, AE$     ' Preamble #4
     PRINT #5, AA$: PRINT #5, AB$, FLE$: PRINT #5,
     PRINT#5, AC$: PRINT #5, AD$: PRINT #5, AE$     ' Preamble  #5
```

```
     K = 1
'.............................................................
     FOR I = 1 TO LENFLE%' FILTER #1 INTO #8
     GET #1, , X1
     IF I <> NOR(K) THEN
          PUT #8, , X1
     ELSE
          K = K + 1
          PRINT I; X1.TT; X1.XX; X1.YY; X1.PD; X1.FT
          PRINT "TO SAVE THIS RECORD TYPE IN NO.  OTHERWISE C/R"
          INPUT A$
          IF UCASE$(LEFT$(A$, 1)) = "N" THEN
          PUT #8, , X1
          ELSE
          PRINT #4, USING "The following record , ##### ,has been
                                                   ----> deleted"; I
          PRINT #4, I; X1.TT; X1.XX; X1.YY; X1.PD; X1.FT
          PRINT #5, USING "The following record , ##### ,has been
                                                   ----> deleted"; I
          PRINT #5, I; X1.TT; X1.XX; X1.YY; X1.PD; X1.FT
          END IF
     END IF
     NEXT I
CLOSE 1, 5, 8
LOOP
CLOSE 3, 4
'.............................................................
END
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM DMPACP

```
DECLARE FUNCTION LOG$ (SB$, A$)
DEFINT I-N
NOR% = 1
START:
PRINT " This program lists records from FASA Time History files."
DO
     PRINT : PRINT                              'CLS ?
     INPUT " ENTER TIME HISTORY FILE NAME, NO EXTENSION   ", FILE
                                                        ---->NAME$
     IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
     SB$ = "(DMPIDX "
     FILEACP$ = FILENAME$ + ".ACP": FILEACP% = 3'append extension
     FILEIDX$ = FILENAME$ + ".IDX": FILEIDX% = 4
     '...........................................................
     T% = 0
     CLOSE
     ON ERROR GOTO NOSUCHFILE
     OPEN FILEACP$ FOR INPUT AS FILEACP%
     OPEN FILEIDX$ FOR INPUT AS FILEIDX%
     ON ERROR GOTO 0
     IF T% = 1 THEN GOTO START
     LENFLE% = LOF(FILEIDX%) / 10
     PRINT "The number of 4-second print outs on the file is : ";
                                                     ----> LENFLE%

     DO
     PRINT
     INPUT "STARTING RECORD # ? "; RNI%
     IF RNI% <= 0 OR RNI% > LENFLE% THEN
          PRINT "  Record number out of range."
          EXIT DO
     END IF
     FOR I% = 1 TO NOR%
          SEEK FILEIDX%, (RNI% + I% - 2) * 10 + 1
          INPUT #FILEIDX%, ACPPOS&
          SEEK #FILEACP%, ACPPOS&
          INPUT #FILEACP%, NOAC%, T%
          IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " A***EOF(FILEACP
                                       ---->%)***"); I; J: EXIT DO
          PRINT NOAC%; " Aircraft "; "  Time is  "; T%
          FOR J = 1 TO NOAC%
               LINE INPUT #FILEACP%, A$           'INPUT AIRCRAFT
               IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " B***EOF(FI
                                 ---->LEACP%)***"); I; J: EXIT FOR
               PRINT A$
          NEXT J
          IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " C***EOF(FILEACP
                                       ---->%)***"); I; J: EXIT DO
          INPUT #FILEACP%, NOTURNS%, NOSLOTS%, NODICE%
          IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " D***EOF(FILEACP
                                       ---->%)***"); I; J: EXIT DO
          PRINT NOTURNS%, NOSLOTS%, NODICE%
          NOSYM% = NOTURNS% + NOSLOTS% + NODICE%
```

```
          IF NOSYM% <> 0 THEN
               FOR J = 1 TO NOSYM%
                    K = J + NOAC%
                    LINE INPUT #FILEACP%, A$          'INPUT SYMBOLS
                    IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " E***E
                           ---->OF(FILEACP%)***"); I; J: EXIT FOR
                    PRINT A$
               NEXT J
          END IF
          IF EOF(FILEACP%) THEN PRINT LOG$(SB$, " F***EOF(FILEACP
                                          ---->%)***"); I; J: EXIT DO
     NEXT I%
     LOOP
     CLOSE 1

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM DMPDAT

```
START:
PRINT " This program lists records from oculometer .DAT files."
DO
      PRINT : PRINT
      INPUT "enter file name"; FLE$
      IF FLE$ = "" THEN END
      T% = 0
      ON ERROR GOTO NOSUCHFILE
      OPEN FLE$ FOR INPUT AS #1
      ON ERROR GOTO 0
      IF T% = 1 THEN GOTO START
      CLOSE 1
      OPEN "R", #1, FLE$, 8
      FIELD #1, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
      LENFLE% = LOF(1) / 8
      PRINT "The number of records on the file is : "; LENFLE%
      DO
      PRINT
      INPUT "STARTING RECORD # "; RNI%
      IF RNI% = 0 THEN EXIT DO
      INPUT "LAST RECORD # "; RNMAX%
      IF RNMAX% = 0 THEN EXIT DO
      IF RNMAX% < 0 OR RNMAX% > LENFLE% THEN RNMAX% = LENFLE%
      IF RNI% < 0 OR RNI% > RNMAX% THEN RNI% = RNMAX%
      FOR I% = RNI% TO RNMAX%
      GET #1, I%
      PRINT I%, CVI(A$), CVI(B$), CVI(C$), CVI(D$)
      NEXT I%
      LOOP
      CLOSE 1

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM A1HIST

```
'Started 1/6/93 to generate histograms of fixation times on Tag
DECLARE SUB HISTOGRAM ()
DECLARE SUB PROUT2 ()
DECLARE SUB ACCUMULATE ()
DECLARE SUB INIT (filename$, filename1$)
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE FUNCTION LOGS$ (SB$, A$)
DECLARE SUB SEARCH ()
DECLARE SUB GETXXA (FILENO%)
DEFINT I-N
CONST pi! = 3.14159
CONST SF! = .472, XOFF! = -5.04, YOFF! = -.9, cpi! = 204.8, alpha
                        ---->! = -11.5 * pi! / 180, runoff! = -.34
CONST big! = 3!, little! = 1!
TYPE FIXCOMB
     TGTTYPEN AS INTEGER              'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4           'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3              'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE        'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER               'TIME HISTORY FRAME #
     TGTX AS SINGLE                   'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                   'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS STRING * 3            'DICE
     COUNTDOWN AS INTEGER             'DICE
     CONTFIX AS STRING * 1            'IS THIS A CONTINUATION OF TH
                                           ---->E PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1              'SPLADT  S-on, F-off
     AIDON AS STRING * 1              'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM AAAA$
DIM frmt$
frmt$ = "## /  / #### #### / / ###.## #### ###.## ###.## ###.## #
                                      ---->##.## ### ### ! ! //"
DIM XX AS FIXCOMB
DIM XXX$
DIM IAID AS INTEGER
DIM BI1P(1 TO 7)  AS INTEGER'Buffered Input 1
DIM BI2P(1 TO 7)  AS INTEGER'Buffered Input 2
DIM BOP(1 TO 7)  AS INTEGER       'Buffered Output
DIM FILEDUM$, FILEMRG$, filebc1$
DIM FILEDUM%, FILEMRG%, filebc1%
DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
'    ** FILE NUMBERS **
'       #1 FILE INDEX         #2          #3    'modified 1/5/93
```

```
'       #4                          #5                          #6
'       #7 DUM                      #8 BC1                      #9
'
DIM PAGE$, FONT$
DIM SPSINV!, SB$
SPSINV! = 1 / 30: SB$ = " (A1HIST  "
NOSTAT% = 10
DIM STATXS(NOSTAT%), STATYS(NOSTAT%) AS SINGLE
DIM STATID(NOSTAT%) AS STRING * 4
DATA "DEN ","IOC ","OM  ","KEAN","FLTS","WIVS","BYSN","TROZ","DRK
                                                      ---->O","JASN"
FOR I = 1 TO NOSTAT%: READ STATID(I): NEXT I
DATA 2.38,-19.49,-0.3,19.44,10.92,-10.24,-23.07,-8.22,29.42,14.67
FOR I = 1 TO NOSTAT%: READ STATXS(I): NEXT I
DATA -.63,24.92,6.1,28.79,14.1,14.1,-26.08,-11.24,-19.56,-9.23
FOR I = 1 TO NOSTAT%: READ STATYS(I): NEXT I
PRTCONTROL$ = CHR$(33) + "R" + CHR$(33)
PAGE$ = PRTCONTROL$ + ";PAGE;EXIT;"                    'OFFICE
FONT$ = PRTCONTROL$ + ";RES;FONT 62; EXIT;"            'OFFICE
DIM SINAL!, COSAL!
SINAL! = SIN(alpha!): COSAL! = COS(alpha!)
XXX$ = "## \  \ #### #### \ \ ##.## #### ###.## ###.## ###.## ###
                                          ---->.## ### #### ! ! \\"
'.....................................................................
DIM NCTR AS INTEGER, NORT AS INTEGER, NOBKT AS INTEGER
NOBKT = 15: NOC = 15
DIM AACOUNT(NOC, 3) AS LONG, AASUM(NOC, 3) AS LONG, AASUMSQ(NOC,
                                                      ---->3) AS LONG
DIM AAALL(NOC, NOBKT%) AS LONG, AANOTON(NOC, NOBKT%) AS LONG, AAO
                                          ---->N(NOC, NOBKT%) AS LONG
DIM BBCOUNT(3) AS LONG, BBSUM(3) AS LONG, BBSUMSQ(3) AS LONG
DIM BBALL(NOBKT%) AS LONG, BBNOTON(NOBKT%) AS LONG, BBON(NOBKT%)
                                                        ---->AS LONG
DIM AVG AS SINGLE, STD AS SINGLE
FOR I = 1 TO 3: BBCOUNT(I) = 0: BBSUM(I) = 0: BBSUMSQ(I) = 0: NEX
                                                            ---->T I
FOR I = 1 TO NOBKT%: BBALL(I) = 0: BBNOTON(I) = 0: BBON(I) = 0: N
                                                          ---->EXT I
NCTR = 0
'.....................................................................
AAAA$ = " MAIN-START MAIN LOOP"
INPUT " Enter full file descriptor for index file  ", index$
OPEN index$ FOR INPUT AS #1
INDEX1$ = LEFT$(index$, LEN(index$) - 4)       'modified 1/5/93
filebc1$ = INDEX1$ + ".BC1": filebc1% = 8      'modified 1/5/93
OPEN filebc1$ FOR OUTPUT AS #filebc1%          'modified 1/5/93
PRINT #filebc1%, LOGS$(SB$, " ")
DO WHILE NOT EOF(1)
    INPUT #1, filename$
    IF LEN(filename$) < 8 THEN EXIT DO          'modified 1/5/93
    filename1$ = RIGHT$(UCASE$(filename$), 8)
    CALL INIT(filename$, filename1$)
    NCTR = NCTR + 1
```

```
     FOR I = 1 TO 3
     AACOUNT(NCTR, I) = 0: AASUM(NCTR, I) = 0: AASUMSQ(NCTR, I) =
                                                         ----> 0: NEXT I
     FOR I = 1 TO NOBKT%
     AAALL(NCTR, I) = 0: AANOTON(NCTR, I) = 0: AAON(NCTR, I) = 0:
                                                            ----> NEXT I
     CALL SEARCH
     PRINT #filebcl%, filename1$; " # of rec's= "; NORT;
     PRINT #filebcl%, " Tag Avg & Std= "; AVG; STD
     PRINT filename1$; " # of rec's= "; NORT;
     PRINT " Tag Avg & Std= "; AVG; STD
LOOP
CALL PROUT2
PRINT #filebcl%, "*****************************************************
                                                   ---->*************"
CLOSE 1, 8              'modified 1/5/93
PRINT AAAA$
'.................................................................
'.................................................................
END      ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB ACCUMULATE

```
SHARED XX AS FIXCOMB
'.................................................................
SHARED AACOUNT() AS LONG, AASUM() AS LONG, AASUMSQ() AS LONG
SHARED AAALL() AS LONG, AANOTON() AS LONG, AAON() AS LONG
SHARED BBCOUNT() AS LONG, BBSUM() AS LONG, BBSUMSQ() AS LONG
SHARED BBALL() AS LONG, BBNOTON() AS LONG, BBON() AS LONG
SHARED AVG AS SINGLE, STD AS SINGLE
SHARED NCTR AS INTEGER, NORT AS INTEGER, NOBKT AS INTEGER
'.................................................................
AACOUNT(NCTR, 1) = AACOUNT(NCTR, 1) + 1
AASUM(NCTR, 1) = AASUM(NCTR, 1) + XX.FIXLNGTH        'Sum
DUMLONG& = XX.FIXLNGTH: DUMLONG& = DUMLONG& * DUMLONG&
AASUMSQ(NCTR, 1) = AASUMSQ(NCTR, 1) + DUMLONG&       'Sum of ^2
BBCOUNT(1) = BBCOUNT(1) + 1: BBSUM(1) = BBSUM(1) + XX.FIXLNGTH
                                                           ---->'Sum
BBSUMSQ(1) = BBSUMSQ(1) + DUMLONG&                   'Sum of ^2
IF XX.AIDON = "1" THEN
     AACOUNT(NCTR, 3) = AACOUNT(NCTR, 3) + 1
     AASUM(NCTR,3) = AASUM(NCTR, 3) + XX.FIXLNGTH             'Sum
     AASUMSQ(NCTR, 3) = AASUMSQ(NCTR, 3) + DUMLONG&        'SOSQ'S
     BBCOUNT(3) = BBCOUNT(3) + 1: BBSUM(3) = BBSUM(3) + XX.FIXLNG
                                                    ---->TH   'Sum
     BBSUMSQ(3) = BBSUMSQ(3) + DUMLONG&                 'Sum of ^2
ELSE
     AACOUNT(NCTR, 2) = AACOUNT(NCTR, 2) + 1
     AASUM(NCTR, 2) = AASUM(NCTR, 2) + XX.FIXLNGTH           'Sum
     AASUMSQ(NCTR, 2) = AASUMSQ(NCTR, 2) + DUMLONG&        'SOSQ'S
     BBCOUNT(2) = BBCOUNT(2) + 1: BBSUM(2) = BBSUM(2) + XX.FIXLNG
                                                      ---->TH 'Sum
     BBSUMSQ(2) = BBSUMSQ(2) + DUMLONG&                 'Sum of ^2
```

```
END IF
END SUB
```

SUB HISTOGRAM

```
SHARED XX AS FIXCOMB
SHARED IAID AS INTEGER
'.............................................................
SHARED AACOUNT() AS LONG, AASUM() AS LONG, AASUMSQ() AS LONG
SHARED AAALL() AS LONG, AANOTON() AS LONG, AAON() AS LONG
SHARED BBCOUNT() AS LONG, BBSUM() AS LONG, BBSUMSQ() AS LONG
SHARED BBALL() AS LONG, BBNOTON() AS LONG, BBON() AS LONG
SHARED AVG AS SINGLE, STD AS SINGLE
SHARED NCTR AS INTEGER, NORT AS INTEGER, NOBKT AS INTEGER
'.............................................................
FT = XX.FIXLNGTH: FT = FT / 30: FT = (FT - AVG) / STD'NORMALIZE F
                                                ---->IXATION TIME

SELECT CASE FT
     CASE IS < -3.25
     I = 1                'CENTER ON -3.5
     CASE IS < -2.75
     I = 2                'CENTER ON -3.0
     CASE IS < -2.25
     I = 3                'CENTER ON -2.5
     CASE IS < -1.75
     I = 4                'CENTER ON -2.0
     CASE IS < -1.25
     I = 5                'CENTER ON -1.5
     CASE IS < -.75
     I = 6                'CENTER ON -1.0
     CASE IS < -.25
     I = 7                'CENTER ON -0.5
     CASE IS < .25
     I = 8                'CENTER ON  0.0
     CASE IS < .75
     I = 9                'CENTER ON  0.5
     CASE IS < 1.25
     I = 10                'CENTER ON  1.0
     CASE IS < 1.75
     I = 11                'CENTER ON  1.5
     CASE IS < 2.25
     I = 12                'CENTER ON  2.0
     CASE IS < 2.75
     I = 13                'CENTER ON  2.5
     CASE IS < 3.25
     I = 14                'CENTER ON  3.0
     CASE IS >= 3.25
     I = 15                'CENTER ON  3.5
     END SELECT
AAALL(NCTR, I) = AAALL(NCTR, I) + 1: BBALL(I) = BBALL(I) + 1
IF XX.AIDON = "1" THEN
AAON(NCTR, I) = AAON(NCTR, I) + 1: BBON(I) = BBON(I) + 1
ELSE
```

```
AANOTON(NCTR, I) = AANOTON(NCTR, I) + 1: BBNOTON(I) = BBNOTON(I)
                                                          ---->+ 1
END IF
END SUB
```

SUB INIT (filename$, filename1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                     ---->buffers
'             \ Initialize sums to zero.  Let user choose partic-
'               \ ular run for analysis.  Determine aid type for
'               \ subsequent branching. Open FILESCN$, FILEDAT$,
'                    \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                 ---->ables, sums
'                     \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED BI1P()  AS INTEGER                         'BufferedInput 1
SHARED BI2P()  AS INTEGER                         'BufferedInput 2
SHARED BOP()  AS INTEGER                           'BufferedOutput
SHARED FILEDUM$, FILEMRG$, filebc1$
SHARED FILEDUM%, FILEMRG%, filebc1%
SHARED IAID AS INTEGER
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SB$ = "(INIT "
        'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                         ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1
'FOR L = 1 TO 7: print BI1P(L): NEXT L
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                          ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
I = nfixbuf: BOP(1) = I: BOP(2) = 1: BOP(3) = 1: BOP(4) = .9 * I
BOP(5) = .8 * I: BOP(6) = 0: BOP(7) = 1
'.................................................................
SELECT CASE MID$(filename1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
```

```
            IAID = 4
      CASE ELSE
            IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                              ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(filename1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                            ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
'FILEDUM$ = FILENAME$ + ".DUM": FILEDUM% = 7   'modified 1/5/93
FILEDUM$ = filename$ + ".MRG": FILEDUM% = 7    ' modified 1/5/93
FILEMRG$ = filename$ + ".MRG": FILEMRG% = 6
'FILEBC1$ = filename$ + ".BC1": FILEBC1% = 8           '  COMMENT
                                         ---->: FILE NAME SHOULD LOOK LIKE
'                                             "C:\FASAFILE\CRONE\CC10SLCE"
'..............................................................
'..............................................................
'..............................................................
END SUB        ' INIT    'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

SUB PROUT2

```
SHARED FILEDUM$, FILEMRG$, filebc1$
SHARED FILEDUM%, FILEMRG%, filebc1%
SHARED SPSINV!, SB$
'..............................................................
SHARED AACOUNT() AS LONG, AASUM() AS LONG, AASUMSQ() AS LONG
SHARED AAALL() AS LONG, AANOTON() AS LONG, AAON() AS LONG
SHARED BBCOUNT() AS LONG, BBSUM() AS LONG, BBSUMSQ() AS LONG
SHARED BBALL() AS LONG, BBNOTON() AS LONG, BBON() AS LONG
SHARED AVG AS SINGLE, STD AS SINGLE
SHARED NCTR AS INTEGER, NORT AS INTEGER, NOBKT AS INTEGER
'..............................................................
SB$ = "(PROUT2 "
PRINT #filebc1%,
F301$ = "             ALL      ": F302$ = "           OFF       "
F303$ = "           ON         "
F3$ = "    n       Sum       Sumsq  "
PRINT #filebc1%, F301$, F302$, F303$
PRINT #filebc1%, F3$; F3$; F3$
F4A$ = " #####.  ######.  ########."
FOR J = 1 TO NCTR
     FOR I = 1 TO 3
     PRINT #filebc1%, USING F4A$; AACOUNT(J, I); AASUM(J, I); AAS
                                                      ---->UMSQ(J, I);

     NEXT I
     PRINT #filebc1%,
NEXT J
PRINT #filebc1%,
PRINT #filebc1%, " Composite Tag Avg & Std= "
FOR I = 1 TO 3
     IF BBCOUNT(I) > 0 THEN
```

```
     AVG = 0: STD = 0
     FOR J = 1 TO NCTR: AVGT = AASUM(J, I): AVGT = AVGT / AACOUNT
                                                          ---->(J, I)
     AVG = AVG + AVGT / 30: SS = AASUMSQ(J, I): SS = SS / AACOUNT
                                                          ---->(J, I)
     SS = SQR(SS - AVGT * AVGT) / 30
     STD = STD + SS: NEXT J
     AVG = AVG / NCTR: STD = STD / NCTR
     PRINT #filebcl%, AVG; STD; "    ";
     END IF
NEXT I
PRINT #filebcl%,
FOR I = 1 TO 3: PRINT #filebcl%, USING F4A$; BBCOUNT(I); BBSUM(I)
                                    ---->; BBSUMSQ(I); : NEXT I
PRINT #filebcl%,
     AGTOTALL& = 0:  AGTOTNOT& = 0: AGTOTON& = 0
FOR I = 1 TO NCTR
     ATOTALL& = 0: ATOTNOT& = 0: ATOTON& = 0
     FOR J = 1 TO NOBKT
     ATOTALL& = ATOTALL& + AAALL(I, J)
     ATOTNOT& = ATOTNOT& + AANOTON(I, J)
     ATOTON& = ATOTON& + AAON(I, J)
     NEXT J
     AGTOTALL& = AGTOTALL& + ATOTALL&
     AGTOTNOT& = AGTOTNOT& + ATOTNOT&
     AGTOTON& = AGTOTON& + ATOTON&
NEXT I
BTOTALL& = 0: BTOTNOT& = 0: BTOTON& = 0
FOR J = 1 TO NOBKT
     BTOTALL& = BTOTALL& + BBALL(J)
     BTOTNOT& = BTOTNOT& + BBNOTON(J)
     BTOTON& = BTOTON& + BBON(J)
NEXT J
PRINT #filebcl%,
PRINT #filebcl%, "A,B Counts ="; AGTOTALL&; BTOTALL&, AGTOTNOT&;
                                                    ---->BTOTNOT&,
PRINT #filebcl%, AGTOTON&; BTOTON&
PRINT #filebcl%,
PRINT #filebcl%, "Each pair should be equal and the 2'nd two shou
                                    ---->ld add up to the 1'st."
PRINT #filebcl%,
PRINT #filebcl%, "Histograms for each run in 1/2 STD intervals.";
PRINT #filebcl%, "Last entry is check sum."
F1$ = ".###"
F2$ = "    -3.0    -2.0    -1.0     0.0     1.0     2.0     3.0"
PRINT #filebcl%, F2$
FOR I = 1 TO NCTR
     PROBSUM = 0
     FOR J = 1 TO NOBKT
     PROB = AAALL(I, J): PROB = PROB / AACOUNT(I, 1)
     PROBSUM = PROBSUM + PROB
     PRINT #filebcl%, USING F1$; PROB;
     NEXT J
```

```
     PRINT #filebc1%, PROBSUM
NEXT I
PROBSUM = 0
PRINT #filebc1%,
PRINT #filebc1%, "Cumulative histograms in 1/2 STD intervals.   "
PRINT #filebc1%, F2$
FOR J = 1 TO NOBKT
     PROB = BBALL(J): PROB = PROB / BBCOUNT(1)
     PROBSUM = PROBSUM + PROB
     PRINT #filebc1%, USING F1$; PROB;
NEXT J
PRINT #filebc1%, PROBSUM
PROBSUM = 0
IF BBCOUNT(2) <> 0 THEN
     FOR J = 1 TO NOBKT
     PROB = BBNOTON(J): PROB = PROB / BBCOUNT(2)
     PROBSUM = PROBSUM + PROB
     PRINT #filebc1%, USING F1$; PROB;
     NEXT J
PRINT #filebc1%, PROBSUM
END IF
PROBSUM = 0
IF BBCOUNT(3) <> 0 THEN
     FOR J = 1 TO NOBKT
     PROB = BBON(J): PROB = PROB / BBCOUNT(3)
     PROBSUM = PROBSUM + PROB
     PRINT #filebc1%, USING F1$; PROB;
     NEXT J
     PRINT #filebc1%, PROBSUM
END IF
END SUB
```

SUB SEARCH

```
'##########
'Purpose................\
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILEDUM$
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\ GETXXA,
'Comments...............\
'##########
SHARED AAAA$
SHARED FILEDUM$, FILEMRG$, filebc1$
SHARED FILEDUM%, FILEMRG%, filebc1%
SHARED XX AS FIXCOMB
SHARED SINAL!, COSAL!
'.............................................................
SHARED AACOUNT() AS LONG, AASUM() AS LONG, AASUMSQ() AS LONG
SHARED AAALL() AS LONG, AANOTON() AS LONG, AAON() AS LONG
```

```
SHARED BBCOUNT() AS LONG, BBSUM() AS LONG, BBSUMSQ() AS LONG
SHARED BBALL() AS LONG, BBNOTON() AS LONG, BBON() AS LONG
SHARED AVG AS SINGLE, STD AS SINGLE
SHARED NCTR AS INTEGER, NORT AS INTEGER, NOBKT AS INTEGER

'............................................................
SB$ = "(SEARCH "
FRMTAC$ = "### \   \ ###.## ###.##  ### \   \ ###.## ###.##"
FRMTSYMHDR$ = " #####  ### ### ###"
CLOSE 7
OPEN FILEDUM$ FOR APPEND AS #7
IF LOF(7) = 0 THEN
     PRINT LOG$(SB$, "DUM FILE CAN NOT BE FOUND ")
     EXIT SUB
ELSE
     CLOSE 7: OPEN FILEDUM$ FOR INPUT AS #7
     AAAA$ = "SEARCH-START 1"
     NORT = 0
     DO WHILE NOT EOF(7)
          CALL GETXXA(7)          'GET THE RECORD INTO XX
          IF XX.TGTTYPEN = 15 THEN CALL ACCUMULATE      ' If Tag
          NORT = NORT + 1
     LOOP
     AVG = AASUM(NCTR, 1): AVG = AVG / AACOUNT(NCTR, 1)
     AVG = AVG / 30: AVG2 = AVG * AVG
     STD = AASUMSQ(NCTR, 1): STD = STD / AACOUNT(NCTR, 1)
     STD = STD / 900: STD = SQR(STD - AVG2)
     CLOSE 7: OPEN FILEDUM$ FOR INPUT AS #7
     AAAA$ = "SEARCH-START 2"
     NORT1 = 0
     DO WHILE NOT EOF(7)
          CALL GETXXA(7)          'GET THE RECORD INTO XX
          IF XX.TGTTYPEN = 15 THEN CALL HISTOGRAM       ' If Tag
          NORT1 = NORT1 + 1
     LOOP
     CLOSE 7
END IF
'............................................................
'............................................................
END SUB    'SEARCH   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

PROGRAM DMPDT1

```
START:
PRINT " This program lists records from oculometer .DAT files."
DO
     PRINT : PRINT
     INPUT "enter file name"; FLE$
     IF FLE$ = "" THEN END
     T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN FLE$ FOR INPUT AS #1
     ON ERROR GOTO 0
     IF T% = 1 THEN GOTO START
     CLOSE 1
     OPEN "R", #1, FLE$, 10
     FIELD #1, 2 AS T$, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
     LENFLE% = LOF(1) / 10
     PRINT "The number of records on the file is : "; LENFLE%
     DO
     PRINT
     INPUT "STARTING RECORD # "; RNI%
     IF RNI% = 0 THEN EXIT DO
     INPUT "LAST RECORD # "; RNMAX%
     IF RNMAX% = 0 THEN EXIT DO
     IF RNMAX% < 0 OR RNMAX% > LENFLE% THEN RNMAX% = LENFLE%
     IF RNI% < 0 OR RNI% > RNMAX% THEN RNI% = RNMAX%
     FOR I% = RNI% TO RNMAX%
     GET #1, I%
     PRINT I%, CVI(T$), CVI(A$), CVI(B$), CVI(C$), CVI(D$)
     NEXT I%
     LOOP
     CLOSE 1

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM FILLMRG

```
DEFINT I-N
DECLARE SUB FIN ()
DECLARE FUNCTION HIT% (VL%, ARRAY%(), N%)
DECLARE SUB INIT (F$, G$)
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB PICK (X!, Y!, TOTSYM%, K%, DISTANCE!)
DECLARE SUB SEARCH ()
DECLARE SUB TARGETSET (FRAMENO%, TOTSYM%, NODICE%)
DECLARE SUB GETXXA (FILENO%)
DECLARE SUB PUTXX (FILENO%)
CONST pi! = 3.14159
CONST SF! = .472, XOFF! = -5.04, YOFF! = -.9, cpi! = 204.8, alpha
                       ---->! = -11.5 * pi! / 180, runoff! = -.34
CONST big! = 3!, little! = 1.2
CONST tagwdth = .5, taght = .5, taglit = little! * SF! + .25, dw
                                                           ---->= 6
CONST xlistlm = 4.3, xlistbm = 3, omx = 6.1
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                          'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                              'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                       'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE      'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                        'TIME HISTORY FRAME #
     TGTX AS SINGLE                                 'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                               'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS STRING * 3                                     'DICE
     COUNTDOWN AS INTEGER                                     'DICE
     CONTFIX AS STRING * 1                                    'IS THIS
                 ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1                       'SPLADT  S-on, F-off
     AIDON AS STRING * 1                               'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM AAAA$
DIM XX AS FIXCOMB
DIM XXX$
NOTG = 60
DIM ACID(1 TO NOTG) AS INTEGER
DIM ACX(1 TO NOTG)  AS SINGLE
DIM ACY(1 TO NOTG)  AS SINGLE
DIM AIDON(1 TO NOTG) AS STRING * 1
DIM COUNTDN(1 TO NOTG) AS INTEGER
DIM HEAD(1 TO NOTG) AS STRING * 3
DIM ROUTE(1 TO NOTG) AS INTEGER
DIM SPEEDON(1 TO NOTG) AS STRING * 1
DIM TIEPEC(1 TO NOTG) AS STRING * 4
```

```
DIM TIEPEN(1 TO NOTG) AS INTEGER
DIM ZONE(1 TO NOTG) AS INTEGER
DIM dicehead%(1 TO 20)
DIM dicetime%(1 TO 20)
DIM IAID AS INTEGER
DIM FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEMRG$, FILESCN$
DIM FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEMRG%, FILESCN%
DIM SHARED NUMFIX%, NUMSCAN%, NOSTAT%, NUMMRG% '*****COMMON******
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
DIM IDX(1 TO 2000) AS LONG
'    ** FILE NUMBERS **
'       #1 FILE INDEX           #2 DAT                  #3 ACP
'       #4 IDX                  #5                      #6 MRG
'       #7 DUM                  #8                      #9
'
DIM PAGE$, FONT$
NOSTAT% = 10
DIM STATXS(NOSTAT%), STATYS(NOSTAT%) AS SINGLE
DIM STATID(NOSTAT%) AS STRING * 3
DATA "DEN","IOC","OM ","KEN","FLS","WIV","BYS","TRZ","DRK","JSN"
FOR I = 1 TO NOSTAT%: READ STATID(I): NEXT I
DATA 2.38,-19.49,-0.3,19.44,10.92,-10.24,-23.07,-8.22,29.42,14.67
FOR I = 1 TO NOSTAT%: READ STATXS(I): NEXT I
DATA -.63,24.92,6.1,28.79,14.1,14.1,-26.08,-11.24,-19.56,-9.23
FOR I = 1 TO NOSTAT%: READ STATYS(I): NEXT I
SB$ = "(MAIN "
DIM SINAL!, COSAL!
SINAL! = SIN(alpha!): COSAL! = COS(alpha!)
XXX$ = "## \  \ #### #### \ \ ##.## #### ###.## ###.## ###.## ###
                                   ---->.## ### #### ! ! \\"
AAAA$ = " MAIN-START MAIN LOOP"
INPUT " Enter full file descriptor for index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #1
DO WHILE NOT EOF(1)
     INPUT #1, FILENAME$
     IF FILENAME$ = "" THEN EXIT DO
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     CALL INIT(FILENAME$, FILENAME1$)
     CALL SEARCH
     PRINT LOG$(SB$, " FINISHED SEARCH")
     'CALL PRNMRG(5767, 174)
     CALL FIN                                          'Close FILES
LOOP
CLOSE 1
PRINT AAAA$
'..............................................................
'..............................................................
END     ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB FIN

```
'##########
'Purpose................\ Close all files, scale and output a few
```

```
                                              ----> statistics
'Parameters.............\
'Other input data.......\ SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK,
                                   ---->NUMINTRACK, NUMOUTTRACK
'Input files............\
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'##########
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SB$ = "(FIN "
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
     PRINT LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= "); NUMINTRAC
                              ---->K;      ' totals for .txt file
     PRINT LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS="); NUMOUTTRA
                                                          ---->CK
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
          ---->TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
        ---->TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK / 30
     PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
       ---->FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH / 30
     PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRACK; SUMINTRA
                                     ---->CK / NUMINTRACK / 30
     PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->OUT TRACK FIXATION IS  "); SUMOUTTRACK / NUMOUTTRACK; SUMOUT
                                  ---->TRACK / NUMOUTTRACK / 30
END IF
CLOSE 2, 3, 4, 6, 7
'.............................................................
'.............................................................
END SUB      ' FIN  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

FUNCTION HIT% (VL%, ARRAY%(), N%)

```
HITL% = 0
FOR K = 1 TO N%
IF ARRAY%(K) = VL% THEN HITL% = K: EXIT FOR
NEXT K
HIT% = HITL%
END FUNCTION
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                    ---->buffers
'               \ Initialize sums to zero.  Let user choose partic-
'                  \ ular run for analysis.  Determine aid type for
'                  \ subsequent branching. Open FILESCN$, FILEDAT$,
```

```
'                           \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                      ---->ables, sums
'                        \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEMRG$, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEMRG%, FILESCN%
SHARED IAID AS INTEGER
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SB$ = "(INIT "
NUMFIX% = 0: NUMSCAN% = 0
SUMFIXLENGTH = 0: SUMINTRACK = 0: SUMOUTTRACK = 0
NUMINTRACK = 0: NUMOUTTRACK = 0
'...............................................................
SELECT CASE MID$(FILENAME1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
          IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                    ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                      ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
FILEACP$ = FILENAME$ + ".ACP": FILEACP% = 3   'append extension
FILEDUM$ = FILENAME$ + ".DUM": FILEDUM% = 7
FILEIDX$ = FILENAME$ + ".IDX": FILEIDX% = 4
FILEMRG$ = FILENAME$ + ".MRG": FILEMRG% = 6
'                        COMMENT: FILE NAME SHOULD LOOK LIKE
'                                   "C:\FASAFILE\CRONE\CC10SLCE"
'...............................................................
OPEN FILEACP$ FOR INPUT AS #3
NOBACP& = LOF(3)          'can the file be found
IF NOBACP& = 0 THEN
     PRINT LOG$(SB$, FILEACP$); "FILE NOT FOUND"  ' fix this test
     EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF ACP BYTES IS  "); NOBACP&
```

```
END IF
INPUT #3, FILEHDR$
INPUT #3, FILEHDR$
PRINT LOG$(SB$, FILEHDR$)
'..................................................................
'..................................................................
'..................................................................
END SUB         ' INIT   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PICK (X!, Y!, TOTSYM%, K, DISTMIN!)

```
'##########
'Purpose................\ Picks a target, the K'th entry in the a
                                                    ---->rrays ACX()
'             \ and ACY() to correspond to the look point position
'              \ (X!, Y!). TOTSYM% is the number of targets in the
'          \ array.  If the subject is looking at both an aircraft
'              \ and turn marker or both an aircraft and slot then
'                  \ the type is changed to 38/"ACTR" or 39/"ACSP"
'                        \ or 28/"ACSL"
'                        \ respectively
'Parameters.............\ input (X!, Y!, TOTSYM%),  output(K, DIS
                                                       ---->TANCE!)
'Other input data.......\ ACX(), ACY(), IAID, TIEPEC(), TIEPEN(),
                                                       ----> ACID()
'Input files............\ none
'Output files...........\ none
'Other output data......\ TIEPEC(K), TIEPEN(K)  selected target o
                                                            ---->nly
'Function calls.........\ none
'Subroutine calls.......\ none
'Comments...............\ Formerly stopped search when distance <
                                                     ---->= little!
'             \ Now search continues for minimum over all targets.
'##########
SHARED ACID()   AS INTEGER
SHARED ACX()    AS SINGLE
SHARED ACY()    AS SINGLE
SHARED AIDON()  AS STRING * 1
SHARED COUNTDN() AS INTEGER
SHARED HEAD() AS STRING * 3
SHARED ROUTE()  AS INTEGER
SHARED SPEEDON() AS STRING * 1
SHARED TIEPEC()  AS STRING * 4
SHARED TIEPEN()  AS INTEGER
SHARED ZONE()  AS INTEGER
SHARED IAID AS INTEGER
DIM IDX(1 TO 20), IDXL AS INTEGER
DISTANCE! = big!: DISTMIN! = 100: K = 0: IDXL = 0
FOR I = 1 TO TOTSYM%
     D1! = ABS(X! - ACX(I))
     IF D1! <= DISTANCE! THEN
          D2! = ABS(Y! - ACY(I))
```

```
        IF D2! <= DISTANCE! THEN                    'HIT ON BIG
             D3! = SQR(D1! * D1! + D2! * D2!)
             IF D3! <= little! THEN
                  DISTANCE! = little!
                  IDXL = IDXL + 1
                  IDX(IDXL) = I
             END IF
             IF D3! < DISTMIN! THEN K = I: DISTMIN! = D3!
        END IF
    END IF
NEXT I
'===============================CHANGE TYPE ???==================
                                              ---->================

IF IDXL > 1 AND (IAID = 31 OR IAID = 41 OR IAID = 32 OR IAID = 42
                         ---->) THEN                 'TURN, SLOT ONLY
    FOR I = 1 TO IDXL
         J = IDX(I)
         IF J <> K AND ACID(J) = ACID(K) THEN'DIF TARGET SAME ID
              SELECT CASE TIEPEN(K)
                   CASE 10
                        SELECT CASE TIEPEN(J)
                             CASE 32, 33, 34, 35
                             TIEPEN(K) = 38: TIEPEC(K) = "ACTR"
                             EXIT FOR
                             CASE 36
                             TIEPEN(K) = 39: TIEPEC(K) = "ACSP"
                             EXIT FOR
                             CASE 20
                             TIEPEN(K) = 28: TIEPEC(K) = "ACSL"
                             EXIT FOR
                        END SELECT
                   CASE 32, 33, 34, 35
                        IF TIEPEN(J) = 10 THEN
                             TIEPEN(K) = 38: TIEPEC(K) = "ACTR"
                             EXIT FOR
                        END IF
                   CASE 36
                        IF TIEPEN(J) = 10 THEN
                             TIEPEN(K) = 39: TIEPEC(K) = "ACSP"
                             EXIT FOR
                        END IF
                   CASE 20
                        IF TIEPEN(J) = 10 THEN
                             TIEPEN(K) = 28: TIEPEC(K) = "ACSL"
                             EXIT FOR
                        END IF
              END SELECT
         END IF
    NEXT I
END IF
'.................................................................
'.................................................................
```

```
END SUB      ' PICK     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PUTXX (FILENO%)

```
'##########
'Purpose................\ Writes a record from XX to the appropri
                                                  ---->ate file.
'Parameters.............\ FILENO%
'Other input data.......\ XX
'Input files............\ none
'Output files...........\ FILEMRG$
'Other output data......\ none
'Function calls.........\ none
'Subroutine calls.......\ none
'Comments...............\ This makes it easier to modify record f
                                                       ---->orm.
'##########
'Write the array XX to a record on the FILEMRG$ file.
SHARED XX AS FIXCOMB
WRITE #FILENO%, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM
---->, XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FI
     ---->XX, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.C
            ---->ROSSCHECK, XX.ZONE, XX.SPEED, XX.AIDON, XX.SPARE
'.............................................................
'.............................................................
END SUB      ' PUTXX     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB SEARCH

```
'##########
'Purpose................\
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILEIDX$, FILEACP$, FILEMRG$
'Output files...........\ FILEDUM$ becomes FILEMRG$
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\ GETXXA, PICK, PUTXX, TARGETSET
'Comments...............\
'##########
SHARED AAAA$
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEMRG$, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEMRG%, FILESCN%
SHARED XX AS FIXCOMB
SHARED SINAL!, COSAL!
SHARED ACID()   AS INTEGER
SHARED ACX()    AS SINGLE
SHARED ACY()    AS SINGLE
SHARED AIDON()  AS STRING * 1
SHARED COUNTDN() AS INTEGER
SHARED HEAD() AS STRING * 3
SHARED ROUTE()  AS INTEGER
SHARED SPEEDON() AS STRING * 1
SHARED TIEPEC()  AS STRING * 4
```

```
SHARED TIEPEN()  AS INTEGER
SHARED ZONE()  AS INTEGER
SHARED dicehead%()
SHARED dicetime%()
SHARED STATID() AS STRING * 3
SB$ = "(SEARCH "
FRMTAC$ = "### \   \ ###.## ###.##  ### \   \ ###.## ###.##"
FRMTSYMHDR$ = " #####  ### ### ###"
CLOSE 3, 4, 6, 7
OPEN FILEIDX$ FOR INPUT AS #4
OPEN FILEACP$ FOR INPUT AS #3
OPEN FILEMRG$ FOR INPUT AS #6
OPEN FILEDUM$ FOR OUTPUT AS #7
IF LOF(3) = 0 OR LOF(4) = 0 OR LOF(6) = 0 OR LOF(7) <> 0 THEN
     PRINT LOG$(SB$, "LENGTHS OF ACP, IDX, MRG, DUMMY = "); LOF(3
                                        ---->); LOF(4); LOF(6); LOF(7)
     PRINT LOG$(SB$, "AT LEAST ONE FILE CAN NOT BE FOUND OR DUMMY
                                                  ----> FILE EXISTS")
     EXIT SUB
ELSE
     AAAA$ = "SEARCH-START"
     FRMNOM1% = 0
     DO WHILE NOT EOF(6)
          CALL GETXXA(6)                      'GET THE RECORD INTO XX
          IF XX.TGTTYPEC = "UNK " THEN
               IF FRMNOM1% <> XX.FRAMENO THEN
                    AAAA$ = "SEARCH-CALL TO TARGETSET"
                    CALL TARGETSET(XX.FRAMENO, TOTSYM%, NODICE%)
                        ---->                       'POSSIBLE TARGETS
                    AAAA$ = "SEARCH-RETURN FROM TARGETSET"
                    FRMNOM1% = XX.FRAMENO
               END IF
               XOCS1! = (XX.FIXX - XOFF!) / SF!
               YOCS1! = (XX.FIXY - YOFF!) / SF!     'TRANSFORM FIX
                                                          ---->ATION
               XOCS! = SINAL! * XOCS1! + COSAL! * YOCS1!
                                    ---->        'TO SYSTEM COORDTS
               YOCS! = COSAL! * XOCS1! - SINAL! * YOCS1!
               CALL PICK(XOCS!, YOCS!, TOTSYM%, IDXTARG, DISTANCE
                                    ---->!)     'PICK FROM POSBLES
               AAAA$ = "SEARCH-RETURN FROM PICK"
               IF IDXTARG <> 0 THEN
                    XSOC1! = ACX(IDXTARG) * SF!
                    YSOC1! = ACY(IDXTARG) * SF!
                         ---->             'TRANSFORM TGT POSITION
                    XSOC! = SINAL! * XSOC1! + COSAL! * YSOC1! + X
                        ---->OFF                'TO SCREEN COORDTS
                    YSOC! = COSAL! * XSOC1! - SINAL! * YSOC1! + Y
                                                            ---->OFF
                    IF DISTANCE! > little THEN
                         IF TIEPEN(IDXTARG) <> 15 THEN
                              IDXTARG = 0: DISTANCE! = 100
                         ELSE
```

```
                DX = XX.FIXX - XSOC!: DY = XX.FIXY
                                        ---->- YSOC!
                IF DX < -taglit OR DX > taglit OR D
               ---->Y < -taglit OR DY > taglit THEN
                IDXTARG = 0: DISTANCE! = 100
                END IF
            END IF
        END IF
END IF
IF IDXTARG = 0 THEN
XX.SPEED = "0": XX.AIDON = "0"
XX.HEADING = "    ": XX.COUNTDOWN = 0
IF XX.FIXX >= xlistlm AND XX.FIXY >= xlistbm THEN
     XX.DISTANCE = SF! * little
     XX.TGTTYPEC = "LIST": XX.TGTTYPEN = 56
     XX.ZONE = "4": XX.TGTID = "LST"
     XX.TGTX = xlistlm          'TO SCREEN COORDTS
     XX.TGTY = xlistbm
ELSE
     DISFRMFIN = XOCS! - runoff!
     SELECT CASE DISFRMFIN
     CASE (-dw - little) TO (-dw + little)    'SOU
                                    ---->TH DOWNWIND
          XX.DISTANCE = SF! * ABS(DISFRMFIN + dw)
          XX.TGTTYPEC = "LINE": XX.TGTTYPEN = 53
          XX.ZONE = "3": XX.TGTID = "SDW"
          XSOC1! = (-dw + runoff) * SF!
          YSOC1! = YOCS! * SF!
              ---->         'TRANSFORM TGT POSITION
          XX.TGTX = SINAL! * XSOC1! + COSAL! * YSO
  ---->C1! + XOFF                 'TO SCREEN COORDTS
          XX.TGTY = COSAL! * XSOC1! - SINAL! * YSO
                                      ---->C1! + YOFF
     CASE -little TO little                   'FINAL
     IF YOCS! >= omx - little AND YOCS! <= omx + 1
                                      ---->ittle THEN
          XX.DISTANCE = SF! * ABS(DISFRMFIN + dw)
                                ---->    'OUTER MARKER
          XX.TGTTYPEC = "OMRK": XX.TGTTYPEN = 50
          XX.ZONE = "1": XX.TGTID = "OMK"
          XSOC1! = runoff! * SF!
          YSOC1! = omx * SF!
              ---->         'TRANSFORM TGT POSITION
          XX.TGTX = SINAL! * XSOC1! + COSAL! * YSO
  ---->C1! + XOFF                 'TO SCREEN COORDTS
          XX.TGTY = COSAL! * XSOC1! - SINAL! * YSO
                                      ---->C1! + YOFF
          TMP1 = (XX.FIXX - XX.TGTX) ^ 2: TMP2 = (
                        ---->XX.FIXY - XX.TGTY) ^ 2
          XX.DISTANCE = SQR(TMP1 + TMP2)
     ELSE                                     ' final
          XX.DISTANCE = SF! * ABS(DISFRMFIN)
          XX.TGTTYPEC = "LINE": XX.TGTTYPEN = 51
```

```
                    XX.ZONE = "1": XX.TGTID = "FNL"
                    XSOC1! = runoff * SF!
                    YSOC1! = YOCS! * SF!
                       ---->          'TRANSFORM TGT POSITION
                    XX.TGTX = SINAL! * XSOC1! + COSAL! * YSO
              ---->C1! + XOFF               'TO SCREEN COORDTS
                    XX.TGTY = COSAL! * XSOC1! - SINAL! * YSO
                                                ---->C1! + YOFF
                END IF
                CASE (dw - little) TO (dw + little)       'NOR
                                              ---->TH DOWN WIND
                    XX.DISTANCE = SF! * ABS(DISFRMFIN - dw)
                    XX.TGTTYPEC = "LINE": XX.TGTTYPEN = 52
                    XX.ZONE = "3": XX.TGTID = "NDW"
                    XSOC1! = (dw + runoff) * SF!
                    YSOC1! = YOCS! * SF!
                       ---->          'TRANSFORM TGT POSITION
                    XX.TGTX = SINAL! * XSOC1! + COSAL! * YSO
              ---->C1! + XOFF               'TO SCREEN COORDTS
                    XX.TGTY = COSAL! * XSOC1! - SINAL! * YSO
                                                ---->C1! + YOFF
                CASE ELSE
                END SELECT
            END IF
            END IF
            IF IDXTARG <> 0 THEN
                XX.DISTANCE= SF! * DISTANCE!: XX.TGTX = XSOC!
                XX.TGTY = YSOC!
                IF TIEPEN(IDXTARG) <> 55 THEN
                    XX.TGTID= RIGHT$(STR$(ACID(IDXTARG)), 3)
                ELSE
                    XX.TGTID = STATID(ACID(IDXTARG))
                END IF
                XX.TGTTYPEC = TIEPEC(IDXTARG): XX.TGTTYPEN =
                                           ---->TIEPEN(IDXTARG)
                XX.ZONE = RIGHT$(STR$(ZONE(IDXTARG)), 1) + RI
          ---->GHT$(STR$(ROUTE(IDXTARG)), 1)'SET ZONE etc. here
                XX.AIDON = AIDON(IDXTARG): XX.SPEED = SPEEDON
                                                ---->(IDXTARG)
                XX.HEADING = HEAD(IDXTARG): XX.COUNTDOWN = CO
                                             ---->UNTDN(IDXTARG)
            END IF
        ELSE
        XX.SPEED = "0": XX.AIDON = "0"
        XX.HEADING = "    ": XX.COUNTDOWN = 0
        END IF
        CALL PUTXX(7)
    LOOP
    IDUM = 0: IF LOF(7) >= LOF(6) THEN IDUM = 1
    CLOSE 3, 4, 6, 7
    'IF IDUM = 1 THEN KILL FILEMRG$: NAME FILEDUM$ AS FILEMRG$
END IF
'.............................................................
```

```
'...........................................................
END SUB    'SEARCH  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB TARGETSET (FRAMENO%, TOTSYM%, NODICE%)

```
'##########
'Purpose................\ Sets up an array of targets (actually s
                                                        ---->everal
'                \ arrays are used) which will be compared to the
'            \ lookpoint to determine what the subject is looking
'              \ at.  This includes: aircraft, tags, aids, static
'                \ targets and lines.  Aids and tags are assigned
'             \ to the same zone as their corresponding aircraft.
'               \ For each aid turned on, the "aidon" flag is set
'                 \ for the corresponding aircraft and tag.  If a
'               \ speed advisory is encountered, then the "speed"
'           \ flag is set for the corresponding aircraft and tag.
'Parameters.............\ input-(FRAMENO%), output-(TOTSYM%, NODI
                                                           ---->CE%)
'Other input data.......\  IAID
'Input files............\ .IDX AS #4, .ACP AS #3
'Output files...........\
'Other output data......\  ACID(),ACX(),ACY(),ROUTE(),ZONE(),TIEP
                                                 ---->EC(),TIEPEN()
'                       \  dicehead%(),dicetime%()
'Function calls.........\
'Subroutine calls.......\
'Comments...............\ Assumes .IDX has been opened as #4 and
                                                 ---->.ACP has been
'                       \ opened as #3.
'##########
SHARED ACID()   AS INTEGER
SHARED ACX()    AS SINGLE
SHARED ACY()    AS SINGLE
SHARED AIDON()   AS STRING * 1
SHARED COUNTDN() AS INTEGER
SHARED HEAD() AS STRING * 3
SHARED ROUTE()   AS INTEGER
SHARED SPEEDON() AS STRING * 1
SHARED TIEPEC()   AS STRING * 4
SHARED TIEPEN()   AS INTEGER
SHARED ZONE()   AS INTEGER
SHARED dicehead%()
SHARED dicetime%()
SHARED IAID AS INTEGER
SHARED STATXS(), STATYS() AS SINGLE
SHARED STATID() AS STRING * 3
DIM GRX(1 TO 4), GRY(1 TO 4) AS SINGLE
JJ = (FRAMENO% - 1) * 10 + 1
IF JJ > LOF(4) - 9 THEN JJ = LOF(4) - 9
SEEK #4, JJ                              'BYTE OFFSET FOR INDEX FILE
INPUT #4, NBYTE&
SEEK #3, NBYTE&                        'BYTE OFFSET FOR TIME HISTORY
```

```
INPUT #3, NOAC%, T%
FOR J = 1 TO NOAC%
     INPUT #3, ACID(J), ACX(J), ACY(J), ACX(J + NOAC%), ACY(J + N
                    ---->OAC%), ROUTE(J), ZONE(J)'INPUT AIRCRAFT
     ROUTE(J + NOAC%) = ROUTE(J): ZONE(J + NOAC%) = ZONE(J)
     TIEPEC(J) = "A/C": TIEPEC(J + NOAC%) = "TAG": ACID(J + NOAC%
                                                  ---->) = ACID(J)

     TIEPEN(J) = 10: TIEPEN(J + NOAC%) = 15
     AIDON(J) = "0": AIDON(J + NOAC%) = "0"
     SPEEDON(J) = "0": SPEEDON(J + NOAC%) = "0"
     HEAD(J) = "    ": HEAD(J + NOAC%) = "    "
     COUNTDN(J) = 0: COUNTDN(J + NOAC%) = 0
NEXT J
K = 2 * NOAC%
INPUT #3, NOTURNS%, NOSLOTS%, NODICE%
NOSYM% = NOTURNS% + NOSLOTS% + NODICE%
'PRINT NOAC%; NOTURNS%; NOSLOTS%; NODICE%; T%; FRAMENO%; J;
IF NOSYM% <> 0 THEN
     SELECT CASE IAID                              'GRAPHIC TURN MARKER
          CASE 31, 32
               FOR JJ = 1 TO NOTURNS%
                    K = K + 1
                    INPUT #3, ACID(K), ACX(K), ACY(K), GRX(1), GR
                  ---->Y(1), GRX(2), GRY(2), GRX(3), GRY(3)'INPUT
                    LLL = HIT%(ACID(K), ACID(), NOAC%)  'Find cor
                                             ---->responding A/C

                    IF LLL <> 0 THEN
                    AIDON(LLL) = "1": AIDON(LLL + NOAC%) = "1"  'S
                                                ---->et A/C & Tag AIDON
                    ZONE(K) = ZONE(LLL): ROUTE(K) = ROUTE(LLL): A
                                                   ---->IDON(K) = "1"

                    SPEEDON(K) = "0"
                         IF GRX(1) = 99.99 THEN          'SPLAT
                              TIEPEC(K) = "SPLT": TIEPEN(K) = 36
                              SPEEDON(K) = "1"
                              SPEEDON(LLL) = "1": SPEEDON(LLL + N
                                                   ---->OAC%) = "1"

                         ELSE
                         TIEPEC(K) = "TRN1": TIEPEC(K + 1) = "TRN
                                                           ---->2"

                         TIEPEC(K + 2) = "TRN3": TIEPEC(K + 3) =
                                                      ---->"TRN4"

                         TIEPEN(K) = 32: TIEPEN(K + 1) = 33
                         TIEPEN(K + 2) = 34: TIEPEN(K + 3) = 35
                         FOR L = 1 TO 3
                              ACID(L + K) = ACID(K): ACX(L + K) =
                                                      ----> GRX(L)

                              ACY(L + K) = GRY(L): AIDON(L + K) =
                                                         ----> "1"

                              ZONE(L + K) = ZONE(LLL): ROUTE(L +
                                              ---->K) = ROUTE(LLL)

                              SPEEDON(L + K) = "0"
                         NEXT L
```

```
                    K = K + 3
                    END IF
                END IF
            NEXT JJ
        CASE 41, 42                                'SLOT MARKER-BUBBLE
            FOR JJ = 1 TO NOSLOTS%
                K = K + 1
                TIEPEC(K) = "SLOT"
                TIEPEN(K) = 20
                ACX(K) = runoff!                       ' -.34 See page 1
                INPUT #3, ACID(K), ACY(K)               'INPUT SYMBOLS
                LLL = HIT%(ACID(K), ACID(), NOAC%)   'Find cor
                                                 ---->responding A/C
                IF LLL <> 0 THEN
                AIDON(LLL) = "1": AIDON(LLL + NOAC%) = "1" 'S
                                              ---->et A/C & Tag AIDON
                ZONE(K) = ZONE(LLL): ROUTE(K) = ROUTE(LLL): A
                                                 ---->IDON(K) = "1"
                SPEEDON(K) = "0"
                END IF
            NEXT JJ
        CASE 21, 22                                'DICE
            FOR JJ = 1 TO NODICE%
                LINE INPUT #3, DICELINE$             'INPUT SYMBOLS
                ACID(K) = VAL(LEFT$(DICELINE$, 5))
                LLL = HIT%(ACID(K), ACID(), NOAC%)   'Find cor
                                                 ---->responding A/C
                IF LLL <> 0 THEN
                COUNTDN(LLL) = VAL(MID$(DICELINE$, 12, 5))
                                                 ---->   'COUNTDOWN
                COUNTDN(LLL + NOAC%) = COUNTDN(LLL)
                HEAD(LLL) = MID$(DICELINE$, 8, 3)     'HEADING
                HEAD(LLL + NOAC%) = HEAD(LLL)
                AIDON(LLL) = "1": AIDON(LLL + NOAC%) = "1"
                IF UCASE$(MID$(DICELINE$, 7, 1)) = "S" THEN
                    PRINT FRAMENO%; DICELINE$; HEAD(LLL)
                    SPEEDON(LLL) = "1": SPEEDON(LLL + NOAC%)
                                                      ----> = "1"
                END IF
                END IF
            NEXT JJ
    END SELECT
END IF
    IF NOSTAT% <> 0 THEN
        FOR JJ = 1 TO NOSTAT%
            K = K + 1
            TIEPEC(K) = "STAT"
            TIEPEN(K) = 55
            ACID(K) = JJ                              'INPUT SYMBOLS
            ACX(K) = STATXS(JJ): ACY(K) = STATYS(JJ)
            ZONE(K) = 4: ROUTE(K) = 0: AIDON(K) = "0"
            SPEEDON(K) = "0"
            HEAD(K) = "   ": COUNTDN(K) = 0
```

```
            NEXT JJ
        END IF
TOTSYM% = K: 'PRINT K
'............................................................
'............................................................
END SUB        ' TARGETSET      'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

PROGRAM FILTER1

```
'DEFINT I-N
START:
TYPE DDAT
     TT AS INTEGER
     XX AS INTEGER
     YY AS INTEGER
     PD AS INTEGER
     FT AS INTEGER
END TYPE
DIM X1 AS DDAT
DIM X2 AS DDAT
DIM X3 AS DDAT
DIST = 102.4: DIST2 = DIST * DIST                       'CPI=204.8
AA$ = "***************************************************************
AB$ = "  FILTER1.BAS  " + DATE$ + " " + TIME$
AC$ = "  .DT1 IS A COMBINATION OF .SCN AND .DAT.  This program us
                                                  ---->es 4 filters"
AD$ = "  proposed by Randy Harris in Oct,91 to reduce the number
                                                    ---->of records"
AE$ = "  on the file. Input-.DT1, Output-.DT3"
INPUT " Enter full file descriptor for index file ", INDEX$
OPEN INDEX$ FOR INPUT AS #3          'List of files for one subject
LGTOT$ = LEFT$(INDEX$, LEN(INDEX$) - 8) + "LOG.TOT"
OPEN LGTOT$ FOR OUTPUT AS #4                'Concatenated log file
DO WHILE NOT EOF(3)        ' Loop through one subject's files   #3
     PRINT : PRINT
     INPUT #3, FLE$
     IF FLE$ = "" THEN END
     NN$ = FLE$ + ".DT2"
     N1$ = FLE$ + ".DT1"
     LG$ = FLE$ + ".LOG"
     OPEN LG$ FOR APPEND AS #5
     PRINT NN$
     T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN N1$ FOR INPUT AS #1      'Can find .DT1 file??   #1
     ON ERROR GOTO 0
     IF T% = 1 THEN GOTO START
     CLOSE 1
     OPEN "R", #1, N1$, 10              ' Open .DT1 in random mode
     LENFLE% = LOF(1) / 10
     DN1 = LENFLE%                      '  Floating point
     OPEN NN$ FOR RANDOM AS #8 LEN = 10
     IF LOF(8) <> 0 THEN STOP     '  Open DT2 in random mode    #8
     PRINT #4, AA$: PRINT #4, AB$, FLE$: PRINT #4,
     PRINT #4, AC$: PRINT #4, AD$: PRINT #4, AE$      ' Preamble #4
     PRINT #5, AA$: PRINT #5, AB$, FLE$: PRINT #5,
     PRINT#5, AC$: PRINT #5, AD$: PRINT #5, AE$     ' Preamble  #5
     FIL1 = 0: FIL2 = 0: FIL3 = 0: FIL4 = 0
     T1 = 0: T2 = 0: T3 = 0: T4 = 0: T5 = 0: T6 = 0     ' Initiali
                                              ---->ze accumulators
     GET #1, , X1                '#1 & #8 OPEN,  FILTER #1 TO #8
```

```
GET #1, , X2                    ' DT1 TO DT2
GET #1, , X3
T1 = T1 + X1.FT + X2.FT + X3.FT
RECRED% = 3                     ' NOR read from .DT1
'...............................................................
DO WHILE RECRED% <= 30000                ' FILTER #1 INTO #8
RECLFT%= LENFLE% - RECRED%                  ' Remaining records
X1IT% = X1.XX <> 0 OR X1.YY <> 0 OR X1.PD > 10
X2IT% = X2.XX <> 0 OR X2.YY <> 0 OR X2.PD > 10
X3IT% = X3.XX <> 0 OR X3.YY <> 0 OR X3.PD > 10
IF X2.FT = 1 AND X2IT% AND (NOT X3IT%) AND X1IT% THEN 'REMOV
                                                  ---->E RECORD

     X1.FT = X1.FT + 1
     PUT #8, , X1
     X1 = X3
     FIL1 = FIL1 + 1             ' Increment NOR accumulator
     IF RECLFT% > 1 THEN         '  2 or more .DT1 records left
          GET #1, , X2
          GET #1, , X3
          T1= T1 + X2.FT + X3.FT  ' Increment time, filter 1
          RECRED% = RECRED% + 2
     ELSE
          PUT #8, , X1            ' 1 or 0 .DT1 records left
          IF RECLFT = 1 THEN
          GET #1, , X1: PUT #8, , X1
          T1 = T1 + X1.FT
          RECRED% = RECRED% + 1
          END IF
          EXIT DO
     END IF
ELSE                                     ' No filter 1
     PUT #8, , X1
     X1 = X2: X2 = X3
     IF RECLFT% > 0 THEN                  ' Not EOF .DT1
          GET #1, , X3
          T1 = T1 + X3.FT
          RECRED% = RECRED% + 1
     ELSE                                  ' EOF .DT1
          PUT #8, , X1: PUT #8, , X2
          EXIT DO
     END IF
END IF
LOOP
CLOSE 1
'...............................................................
NN3$ = FLE$ + ".DT3"
OPEN NN3$ FOR RANDOM AS #9 LEN = 10     ' #8 AND #9 OPEN, FIL
                                          ---->TER #8 INTO #9

LOF9 = LOF(9): LOF8 = LOF(8) / 10       ' DT2 TO DT3
IF LOF9 <> 0 THEN STOP
SEEK #8, 1
GET #8, , X1
GET #8, , X2
```

```
     GET #8, , X3
     RECRED% = 3
     DO
     RECLFT% = LOF8 - RECRED%
     X2IT% = X2.XX <> 0 OR X2.YY <> 0 OR X2.PD > 10
     IF (X2.FT > 12) OR (X2IT% AND X2.FT > 3) THEN GOTO FILTER4
                                                       ---->' B too long
     X1IT% = X1.XX <> 0 OR X1.YY <> 0 OR X1.PD > 10
     X3IT% = X3.XX <> 0 OR X3.YY <> 0 OR X3.PD > 10
     IF NOT X1IT% OR NOT X3IT% THEN GOTO FILTER4        ' A or C no
                                                       ---->t in track
     D13% = ((X1.XX - X3.XX) ^ 2 + (X1.YY - X3.YY) ^ 2) <= DIST2
     IF NOT D13% THEN GOTO FILTER4     ' A to C distance too great
     XX1 = X1.XX: YY1 = X1.YY: XX3 = X3.XX: PD1 = X1.PD' 3 INTO 1
     YY3 = X3.YY: FT1 = X1.FT: FT3 = X3.FT: PD3 = X3.PD
     X1.XX = ((XX1 * FT1) + (XX3 * FT3)) / (FT1 + FT3)
     X1.YY = ((YY1 * FT1) + (YY3 * FT3)) / (FT1 + FT3)
     X1.PD = ((PD1 * FT1) + (PD3 * FT3)) / (FT1 + FT3)
     X1.FT = X1.FT + X3.FT
     IF X2.FT < 4 THEN
          X1.FT = X1.FT + X2.FT
     ELSE
          IF NOT X2IT% THEN T5 = T5 + X2.FT
     END IF
     IF X2IT% THEN
     FIL2 = FIL2 + 1
     ELSE
     FIL3 = FIL3 + 1
     END IF
     IF RECLFT% > 1 THEN
     GET #8, , X2: GET #8, , X3
     RECRED% = RECRED% + 2
     ELSE                                 ' END OF FILE
     IF X1.FT > 3 THEN                                  ' Filter4 A??
          PUT #9, , X1: T2 = T2 + X1.FT
     ELSE
          FIL4 = FIL4 + 1:
          T6 = T6 + X1.FT
     END IF
     IF RECLFT = 1 THEN
          GET #8, , X1: RECRED% = RECRED% + 1
          IF X1.FT > 3 THEN                            ' Filter4 A??
               PUT #9, , X1: T2 = T2 + X1.FT
          ELSE
               T6 = T6 + X1.FT: FIL4 = FIL4 + 1
          END IF
     END IF
     EXIT DO
     END IF
     GOTO BOTTOM
FILTER4:
     IF X1.FT > 3 THEN                                  ' Filter4 A??
          PUT #9, , X1: T2 = T2 + X1.FT
```

```
    ELSE
         FIL4 = FIL4 + 1:
         T6 = T6 + X1.FT
    END IF
    X1 = X2: X2 = X3
    IF RECLFT% > 0 THEN                          ' Not EOF .DT2
         GET #8, , X3
         RECRED% = RECRED% + 1
    ELSE                                         'END OF FILE .DT2
    IF X1.FT > 3 THEN                                     ' Filter4 A??
         PUT #9, , X1: T2 = T2 + X1.FT
    ELSE
         T6 = T6 + X1.FT: FIL4 = FIL4 + 1
    END IF
    IFX2.FT > 3 THEN                                      ' Filter4 B??
         PUT #9, , X2: T2 = T2 + X2.FT
    ELSE
         T6 = T6 + X2.FT: FIL4 = FIL4 + 1
    END IF
    EXIT DO
    END IF
BOTTOM:
    LOOP
'.....................................................................
    N2 = LOF(9) / 10
    CLOSE 8, 9
    N3 = FIL1: PN3 = 100 * N3 / DN1: N4 = FIL2 * 2: PN4 = 100 *
                                                         ---->N4 / DN1
    N5 = FIL3 * 2: PN5 = 100 * N5 / DN1: N6 = FIL4: PN6 = 100 *
                                                         ---->N6 / DN1
    PRINT : PRINT FIL1, FIL2, FIL3, FIL4
    PRINT #4,
    PRINT #4, USING "  Number of records .DT1 #####,   .DT3 #####
---->            Percent Excised ###.#%"; DN1; N2; 100 * (DN1 -
                                                         ---->N2) / DN1
    PRINT #4, "  Number of records excised for each of 4 filters
                                         ----> as % of .DTI records "
    PRINT #4, SPACE$(11);
    PRINT #4, USING " ##### ###.#% "; N3; PN3; N4; PN4; N5; PN5;
                                                         ----> N6; PN6
    T3 = 0: PT3 = 100 * T3 / T1:  PT4 = 100 * T4 / T1
    PT5 = 100 * T5 / T1:  PT6 = 100 * T6 / T1
    PRINT #4,
    PRINT #4, USING "  TOTAL TIME .DT1 ######,   .DT3 ######"; T1
                                                             ---->; T2
    PRINT #4, "  Time excised for each of 4 filters as % of .DTI
                                                  ----> total time "
    PRINT #4, SPACE$(11);
    PRINT #4, USING " ##### ###.#% "; T3; PT3; T4; PT4; T5; PT5;
                                                         ----> T6; PT6
    PRINT #4,
    PRINT #4, USING "  TOTAL TIME in seconds .DT1 ####,   .DT3 ##
---->##          Percent Excised ###.#%"; T1 / 30; T2 / 30; 100
```

```
                                              ---->* (T1 - T2) / T1
     PRINT #4, "  Time excised in seconds for each of 4 filters a
                                        ---->s % of .DTI total time "
     PRINT #4, SPACE$(11);
     PRINT #4, USING " #### ###.#% "; T3 / 30; PT3; T4 / 30; PT4;
                                     ----> T5 / 30; PT5; T6 / 30; PT6
     PRINT #5,
     PRINT #5, USING "  Number of records .DT1 #####,  .DT3 #####
---->             Percent Excised ###.#%"; DN1; N2; 100 * (DN1 -
                                                       ---->N2) / DN1
     PRINT #5, "  Number of records excised for each of 4 filters
                                        ----> as % of .DTI records "
     PRINT #5, SPACE$(11);
     PRINT #5, USING " ##### ###.#% "; N3; PN3; N4; PN4; N5; PN5;
                                                       ----> N6; PN6
     PRINT #5,
     PRINT #5, USING "  TOTAL TIME .DT1 ######,  .DT3 ######"; T1
                                                           ---->; T2
     PRINT #5, "  Time excised for each of 4 filters as % of .DTI
                                                   ----> total time "
     PRINT #5, SPACE$(11);
     PRINT #5, USING " ##### ###.#% "; T3; PT3; T4; PT4; T5; PT5;
                                                       ----> T6; PT6
     PRINT #5,
     PRINT #5, USING "  TOTAL TIME in seconds .DT1 ####,  .DT3 ##
---->##           Percent Excised ###.#%"; T1 / 30; T2 / 30; 100
                                              ---->* (T1 - T2) / T1
     PRINT #5, "  Time excised in seconds for each of 4 filters a
                                        ---->s % of .DTI total time "
     PRINT #5, SPACE$(11);
     PRINT #5, USING " #### ###.#% "; T3 / 30; PT3; T4 / 30; PT4;
                                     ----> T5 / 30; PT5; T6 / 30; PT6
     CLOSE 5
LOOP
CLOSE 3, 4
END
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM FIXPOINT

```
DEFINT I-N
DECLARE SUB CRE8DT1 ()
DECLARE SUB BI2 (BIP%())
DECLARE SUB BI1 (BIP%())
DECLARE SUB FIN ()
DECLARE SUB FIXPOINTER ()
DECLARE SUB INIT (F$, G$)
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB YESORNO (A$, B$)
CONST nscanbuf = 400, nfixbuf = 5000
TYPE DT1
     TIME AS INTEGER
     XL AS INTEGER
     YL AS INTEGER
     PUP AS INTEGER
     LENGTH AS INTEGER
END TYPE
DIM XDT1 AS DT1
DIM BI1P(1 TO 7)  AS INTEGER'Buffered Input 1
DIM BI2P(1 TO 7)  AS INTEGER'Buffered Input 2
DIM FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEDT1$, FILESCN$
DIM FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
DIM FIXPNTER(26000) AS INTEGER
DIM FIXLENGTH(1 TO nfixbuf)   AS INTEGER
DIM INTRACK(1 TO nfixbuf)   AS INTEGER
DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
DIM OCSCAN(1 TO nscanbuf, 1 TO 4) AS INTEGER
DIM PUPDIAM(1 TO nfixbuf)   AS INTEGER
DIM SHARED NUMFIX%, NUMSCAN%, NOSTAT%, NUMMRG% '*****COMMON******
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
DIM XLOOK(1 TO nfixbuf)   AS INTEGER
DIM YLOOK(1 TO nfixbuf)   AS INTEGER
'     ** FILE NUMBERS **
'        #1 SCN                #2 DAT                #3 INDEX
'        #4                    #5                    #6 DT1
'        #7                    #8                    #9
'
NOSTAT% = 10
SB$ = "(MAIN "
INPUT " Enter full file descriptor for index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #3     ' e.g. O:KISER\FLEINDX1
DO WHILE NOT EOF(3)
      INPUT #3, FILENAME$
      FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
      CALL INIT(FILENAME$, FILENAME1$)
      CALL FIXPOINTER
      PRINT LOG$(SB$, " FINISHED FIXPOINTER")
      CALL CRE8DT1                                        'Close FILES
      CALL FIN
LOOP
CLOSE 3
'.........................................................................
```

```
'..............................................................
END      ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB BI1 (BIP%())

```
'##########
'Purpose................\ Reads FILESCN$ using circular buffer OC
                                                      ---->SCAN()
'                        \ and manages pointers, BIP%().
'Parameters.............\ BIP%() Circular buffer pointers
'Other input data.......\
'Input files............\ FILESCN$ = .SCN
'Output files...........\
'Other output data......\ OCSCAN()
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED OCSCAN()  AS INTEGER
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
         'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
NORIB = BIP%(3) - BIP%(2)                   'If buffer low AND EOF=.F.
IF NORIB < 0 THEN NORIB = NORIB + BIP%(1)
IF NORIB < BIP%(4) AND BIP%(6) = 0 THEN
     FORI = 1 TO BIP%(5)                              'Load buffer
     GET FILESCN%, , OCSCAN(BIP%(3), 1)
     BIP%(3) = BIP%(3) + 1
     IF BIP%(3) > BIP%(1) THEN BIP%(3) = 1
     IF EOF(FILESCN%) THEN
     BIP%(6) = BIP%(3)                             'Points at last
     EXIT FOR
     END IF
     NEXT I
END IF
JDUM = BIP%(2)
BIP%(2) = BIP%(2) + 1: IF BIP%(2) > BIP%(1) THEN BIP%(2) = 1 'Inc
                                                  ---->rement first
IF BIP%(2) = BIP%(6) THEN                         'READ BEYOND DATA
     BIP%(2) = JDUM
     PRINT "****** ERROR READING EOF(FILESCN%) IN BI1 ******"
END IF
'..............................................................
'..............................................................
END SUB    ' BI1   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB BI2 (BIP%())

```
'##########
'Purpose................\Read the fixation data into circular buf
                                                    ---->fers and\
'                        \compute a few preliminary statistics.
'Parameters.............\BIP%() Circular buffer pointers
'Other input data.......\
'Input files............\FILEDAT% =.DAT
```

```
'Output files...........\
'Other output data......\BIP%(), XLOOK(),YLOOK(),PUPDIAM(),FIXLEN
                         ---->GTH(),INTRACK() Circular buffers
' \SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK, NUMINTRACK, NUMOUTTRACK
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED INTRACK() AS INTEGER
SHARED FIXLENGTH() AS INTEGER
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
        'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
NORIB = BIP%(3) - BIP%(2)                    'If buffer low AND EOF=.F.
IF NORIB < 0 THEN NORIB = NORIB + BIP%(1)
IF NORIB < BIP%(4) AND NOT EOF(FILEDAT%) THEN
                                                        'Load buffer
    FOR I = 1 TO BIP%(5)
    GET FILEDAT%, , XLOOK(BIP%(3)): GET FILEDAT%, , YLOOK(BIP%(3
                ---->))                                 'read record
    IF NOT EOF(FILEDAT%) THEN
        GET FILEDAT%, , PUPDIAM(BIP%(3)): GET FILEDAT%, , FIXLE
                                                ---->NGTH(BIP%(3))
        SUMFIXLENGTH = SUMFIXLENGTH + FIXLENGTH(BIP%(3))
                     ---->                      'total of fixations
        INTRACK(BIP%(3)) = 1
        IF XLOOK(BIP%(3)) = 0 AND YLOOK(BIP%(3)) = 0 AND PUPDIA
                    ---->M(BIP%(3)) < 11 THEN ' out of track
            INTRACK(BIP%(3)) = 0
            SUMOUTTRACK = SUMOUTTRACK + FIXLENGTH(BIP%(3))'tot
                                            ---->al out of track
            NUMOUTTRACK = NUMOUTTRACK + 1
                                    ' in track
        ELSE
            SUMINTRACK = SUMINTRACK + FIXLENGTH(BIP%(3))
                    ---->                               'total intrack
            NUMINTRACK = NUMINTRACK + 1
        END IF
        BIP%(3) = BIP%(3) + 1
        IF BIP%(3) > BIP%(1) THEN BIP%(3) = 1
    ELSE
    EXIT FOR
    END IF
    NEXT I
END IF
BIP%(2) = BIP%(2) + 1: IF BIP%(2) > BIP%(1) THEN BIP%(2) = 1'Incr
                                                ---->ement first
'..............................................................
'..............................................................
END SUB      'BI2   'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

SUB CRE8DT1

```
'##########
'Purpose................\ Initial creation of the .MRG file using
                                                  ----> data from
'            \ .DAT and time history pointer array from subroutine
'                         \ FIXPOINTER
'Parameters.............\ none
'Other input data.......\ NUMMRG%, PUPDIAM, FIXPNTER, XLOOK, CPI!
                                                       ---->, YLOOK
'.......................\ FIXLENGTH,
'Input files............\ FILEDAT$= .DAT
'Output files...........\ FILEDT1$= .DT1
'Other output data......\ none
'Function calls.........\ LOG$
'Subroutine calls.......\ BI2
'Comments...............\ Target type is set to 0, "UNK" for in-t
                                                  ---->racks or
'        \ 80, "OUT" for out-tracks. Other fields are initialized
'                         \ to unrealistic constants.
'##########
SHARED BI2P()  AS INTEGER                           'BufferedInput 2
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEDT1$, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
SHARED FIXLENGTH() AS INTEGER
SHARED FIXPNTER() AS INTEGER
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED XDT1 AS DT1
'...............................................................
SB$ = "(CRE8DT1 "
CLOSE FILEDT1%
OPEN FILEDT1$ FOR RANDOM AS #FILEDT1% LEN = 10' FIXATION, TIME HI
                                                  ---->STORY MERGE
NUMMRG& = LOF(FILEDT1%)                         'can the file be found
IF NUMMRG& <> 0 THEN
     PRINT LOG$(SB$, FILEDT1$ + " is not empty and you are trying
                                      ----> to open it for output")
     PRINTLOG$(SB$, "NUMFIX% = "); NUMFIX%; "NUMMRG& = "; NUMMRG&
     A$ = "Do you want to close " + FILEDT1$ + " and exit CRE8DT1
                                                 ----> subroutine"
     CALL YESORNO(A$, B$)
     IF B$ = "Y" THEN CLOSE FILEDT1%: EXIT SUB
END IF
'...............................................................
SEEK #FILEDAT%, 1          'REWIND FILE & RESET BUFFER           '
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                          ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
'...............................................................
FOR I = 1 TO NUMMRG%
     CALL BI2(BI2P())
     II = BI2P(2)
```

```
    XDT1.TIME = FIXPNTER(I): XDT1.XL = XLOOK(II): XDT1.YL = YLOO
                                                         ---->K(II)
    XDT1.PUP = PUPDIAM(II): XDT1.LENGTH = FIXLENGTH(II)
    PUT #FILEDT1%, I, XDT1
NEXT I
'CLOSE FILEDT1%
'..................................................................
'..................................................................
END SUB      'CRESDT1
```

SUB FIN

```
'##########
'Purpose................\ Close all files, scale and output a few
                                              ----> statistics
'Parameters.............\
'Other input data.......\ SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK,
                                  ---->NUMINTRACK, NUMOUTTRACK
'Input files............\
'Output files...........\
'Other output data......\
'Function calls.........\ LOG$
'Subroutine calls.......\
'Comments...............\
'##########
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SB$ = "(FIN "
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
    PRINT LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= "); NUMINTRAC
                              ---->K;       ' totals for .txt file
    PRINT LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS="); NUMOUTTRA
                                                         ---->CK
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
          ---->TIME IN TRACK IS  "); SUMINTRACK; SUMINTRACK / 30
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
        ---->TIME OUT TRACK IS "); SUMOUTTRACK; SUMOUTTRACK / 30
    PRINT USING "& ###### OR ####.## SECONDS"; LOG$(SB$, "TOTAL
      ---->FIXATION TIME IS  "); SUMFIXLENGTH; SUMFIXLENGTH / 30
    PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->IN TRACK FIXATION IS   "); SUMINTRACK / NUMINTRACK; SUMINTRA
                                   ---->CK / NUMINTRACK / 30
    PRINT USING "& ###.## OR ##.## SECONDS"; LOG$(SB$, "AVERAGE
---->OUT TRACK FIXATION IS  "); SUMOUTTRACK / NUMOUTTRACK; SUMOUT
                                ---->TRACK / NUMOUTTRACK / 30
END IF
CLOSE 1, 2, 4, 5, 6, 7, 9
'..................................................................
'..................................................................
END SUB     ' FIN  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB FIXPOINTER

```
'##########
'Purpose...............\ This routine assigns a time history rec
                                                 ---->ord #, (I), to
'          \ each fixation. The contents of FIXPNTER(N) indicates
'            \ which time record the N'th fixation is associated
'          \ with and therefore where one should seek the target
'                      \ of the fixation.
'
'Parameters.............\
'Other input data.......\ OCSCAN(), BI1P(), PAGE$, FONT$
'Input files............\ FILESCN$ thru BI1 only.
'Output files...........\ #9 a print out of .ACP record #'s for e
                                                 ---->ach fixation

'Other output data......\ FIXPNTER(), NUMMRG%
'Function calls.........\ LOG$,
'Subroutine calls.......\ BI1
'Comments...............\ The first scan # is recorded at time =
                                                 ---->4 seconds.
'               \ Therefore if the third scan # is 3 and the 4'th
'              \ scan # is 10, then .DAT records 5 through 11 are
'              \ associated with targets recorded at t=12 seconds
'             \ or record 4 on the .ACP file since the first .ACP
'                      \ record corresponds to t=0.
'
'***
'           \ The first entry on the time history file is time=0.
'                      \ The second entry is time=4 seconds.
'            \ The first scan toggle precedes slightly this 2'nd
'          \ entry i.e. ~4 seconds into the run. Thus spake Buddy
'       \ after careful consideration and investigation on 4/3/91.
'***
'         \ If the 1'st scan # is k, then fixations <= (k+1) are
'                 \ pointed at time history frame 1, i.e. time=0.
'
'***
'                     \ The last time history frame is not used.
'              \ NUMMRG%<= NUMFIX%. A few fixations (<20) during
'                 \ the last partial 4 second frame are dropped.
'
'***
'            \ If the scan # does not increase, then drop through
'                      \ the loop until it does.
'
'##########
SHARED FIXPNTER() AS INTEGER
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
                                                 'Buffered Input1
SHARED BI1P()  AS INTEGER
SHARED OCSCAN()  AS INTEGER
SHARED PAGE$, FONT$
SB$ = "(FIXPOINTER "
SEEK #FILESCN%, 1             'REWIND FILE & RESET BUFFER
          'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                 ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1'nscanbuf is a constant
SCNPREV% = -1
FORI = 1 TO NUMSCAN%              ' I is the record# on the ACP file
     CALL BI1(BI1P())             ' NUMSCAN% defined in INIT
```

```
    SCN% = OCSCAN(BI1P(2), 1)      'scn%= fixation# from SCN file
    IF SCN% >= 1 AND SCN% <= NUMFIX% THEN     'out-of-range error
        IF SCN% > SCNPREV% THEN  'If equal loop otherwise error
            FOR J = SCNPREV% + 2 TO SCN% + 1
                IF J <= NUMFIX% THEN
                    FIXPNTER(J) = I
                    JTEMP = J
                END IF
            NEXT J
            SCNPREV% = SCN%
        ELSE
            IF SCN% < SCNPREV% THEN     'not monotonic
                PRINT LOG$(SB$, "********* ERROR SCAN POINTER
                                   ---->DECREASING  "); I, SCN%

            END IF
        END IF
    ELSE
        PRINT LOG$(SB$, "FIXATION POINTER OUT OF RANGE  "); I;
                                                        ---->BI1P(2)

    END IF
NEXT I
                                                  ' NUMMRG% DEFINED
NUMMRG% = JTEMP
PRINT LOG$(SB$, "NUMBER OF MERGE FILE FIXATIONS IS  "); NUMMRG%
'=====================================================================
                                  ---->=========================

'...................................................................
'...................................................................
END SUB       ' FIXPOINTER     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                         ---->buffers
'             \ Initialize sums to zero.  Let user choose partic-
'                \ ular run for analysis.  Determine aid type for
'                \ subsequent branching. Open FILESCN$, FILEDAT$,
'                       \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                      ---->ables, sums
'                    \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
                                                   'BufferedInput 1
SHARED BI1P()  AS INTEGER
                                                   'BufferedInput 2
SHARED BI2P()  AS INTEGER
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEIDX$, FILEDT1$, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEIDX%, FILEDT1%, FILESCN%
```

```
SHARED IAID AS INTEGER
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SB$ = "(INIT "
            'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                            ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1
'FOR L = 1 TO 7: print BI1P(L): NEXT L
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                             ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
NUMFIX% = 0: NUMSCAN% = 0
SUMFIXLENGTH = 0: SUMINTRACK = 0: SUMOUTTRACK = 0
NUMINTRACK = 0: NUMOUTTRACK = 0
'...............................................................
IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
SELECT CASE MID$(FILENAME1$, 5, 1)
      CASE "M"
            IAID = 1
      CASE "D"
            IAID = 2
      CASE "G"
            IAID = 3
      CASE "S"
            IAID = 4
      CASE ELSE
            IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                  ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                         ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
FILEDAT$ = FILENAME$ + ".DAT": FILEDAT% = 2    'append extension
FILEDT1$ = FILENAME$ + ".DT1": FILEDT1% = 6
FILESCN$ = FILENAME$ + ".SCN": FILESCN% = 1
'                           COMMENT: FILE NAME SHOULD LOOK LIKE
'                                     "C:\FASAFILE\CRONE\CC10SLCE"
'...............................................................
OPEN FILESCN$ FOR BINARY AS #1      ' oculometer scan file
NUMSCAN% = LOF(1) / 2         'can the file be found
IF NUMSCAN% = 0 THEN
      PRINT LOG$(SB$, FILESCN$); " FILE NOT FOUND" ' fix this test
      EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF 4 SECOND SCANS IS  "); NUMSCAN%
END IF
'...............................................................
OPEN FILEDAT$ FOR BINARY AS #2 'oculometer .dat file
NUMFIX% = LOF(2) / 8       'can the file be found
IF NUMFIX% = 0 THEN
```

```
        PRINT LOG$(SB$, FILEDAT$); "FILE NOT FOUND"   ' fix this test
        EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF FIXATIONS IS   "); NUMFIX%

END IF
'.....................................................................
'.....................................................................
END SUB         ' INIT   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

PROGRAM PLTDATFL

```
TYPE REGTYPE
     AX AS INTEGER
     BX AS INTEGER
     CX AS INTEGER
     DX AS INTEGER
     BP AS INTEGER
     SI AS INTEGER
     DI AS INTEGER
     FLAGS AS INTEGER
     DS AS INTEGER
     ES AS INTEGER
END TYPE
DIM INREG AS REGTYPE
DIM OUTREG AS REGTYPE
'...........................................................
CPI! = 204.8:
DOTSIZE! = .01
TITLE$ = " Look Point Positions In Oculometer Coordinates"
T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN "fleindx1" FOR INPUT AS #1
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find INDEX file :": END
DO
     INPUT #1, FILENAME$
     IFFILENAME$ = "" THEN PRINT "BLANK LINE INDEX FILE": EXIT DO
     RUN$ = RIGHT$(FILENAME$, 8)
     DAT$ = FILENAME$ + ".DAT"
     PRINT RUN$, DAT$
     T% = 0
     ON ERROR GOTO NOSUCHFILE
          OPEN DAT$ FOR INPUT AS #2
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find .OCULOMETER DATA FILE :" +
                                                  ---->DAT$: EXIT DO
     CLOSE 2
     OPEN "R", #2, DAT$, 8
     FIELD #2, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
     LENFLE% = LOF(2) / 8       'OK THE .DAT FILE IS OPEN FOR RAND
                                                       ---->OM INPUT
'...........................................................
     SCREEN 12
     CLS 1
     BLX = -5: BLY = -5: TRX = 5: TRY = 5   ' CORNERS OF THE WINDO
                                                   ---->W, INCHES
     TLX1 = 110: TLY1 = 50: BLX1 = 610: BLY1 = 430 ' CORNERS IN P
                                                       ---->IXELS
     WINDOW (BLX, BLY)-(TRX, TRY)
     VIEW (TLX1, TLY1)-(BLX1, BLY1), , 1
'...........................................................
     SFX! = 1 / CPI!
     SFY! = 1 / CPI!
```

```
    XO! = 0
    YO! = 0
    LOCATE 2, 23
    PRINT TITLE$
'...............................................................
    J% = 0
    FOR I% = 1 TO LENFLE%
         GET #2, I%
         A% = CVI(A$): B% = CVI(B$): C% = CVI(C$): D% = CVI(D$)
         IF A% <> 0 OR B% <> 0 OR C% > 10 THEN
              J% = J% + 1
              X! = SFX! * A% + XO!: Y! = SFY! * B% + YO!
              CIRCLE (X!, Y!), DOTSIZE
         END IF
    NEXT I%
         LOCATE 3, 30
         ID$ = DATE$ + "  " + LEFT$(TIME$, 5) + "  " + RUN$ + "
                                               ----> " + STR$(J%)

         PRINT ID$
    '...........................................................
    CLOSE 2
    'CALL INTERRUPT(&H5, INREG, OUTREG)  '&H5 is print screen fu
                                                     ---->nction

    SCREEN 0
LOOP WHILE NOT EOF(1)
CLOSE 1
END
NOSUCHFILE:
T% = 1
RESUME NEXT
'...............................................................
```

PROGRAM PRNDAT

```
START:
PRINT " This program lists records from oculometer .DAT files."
DO
      PRINT : PRINT
      INPUT "enter file name"; FLE$
      IF FLE$ = "" THEN END
      NN$ = LEFT$(RIGHT$(FLE$, 12), 8) + ".PRN"
      PRINT NN$
      T% = 0
      ON ERROR GOTO NOSUCHFILE
      OPEN FLE$ FOR INPUT AS #1
      ON ERROR GOTO 0
      IF T% = 1 THEN GOTO START
      CLOSE 1
      OPEN "R", #1, FLE$, 8
      FIELD #1, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
      LENFLE% = LOF(1) / 8
      OPEN NN$ FOR OUTPUT AS #8
      PRINT #8, : PRINT #8,
      PRINT #8, "THE FOLLOWING DATA IS FROM THE FILE, "; FLE$
      PRINT #8, "The number of records on the file is : "; LENFLE%
      PRINT , "The number of records on the file is : "; LENFLE%,
                                                           ---->FLE$
      DO
      PRINT
      INPUT "STARTING RECORD # "; RNI%
      IF RNI% = 0 THEN EXIT DO
      INPUT "LAST RECORD # "; RNMAX%
      IF RNMAX% = 0 THEN EXIT DO
      IF RNMAX% < 0 OR RNMAX% > LENFLE% THEN RNMAX% = LENFLE%
      IF RNI% < 0 OR RNI% > RNMAX% THEN RNI% = RNMAX%
      PRINT #8, "PRINT FROM RECORD # "; RNI%; "  to "; RNMAX%
      PRINT #8, "Record #", "  x", "  y", "Pupdiam", "Length"
      FOR I% = RNI% TO RNMAX%
      GET #1, I%
      PRINT #8, I%, CVI(A$), CVI(B$), CVI(C$), CVI(D$)
      NEXT I%
      LOOP
      CLOSE 1, 8

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM PRNDT1

```
START:
PRINT " This program lists records from oculometer .DAT files."
DO
     PRINT : PRINT
     INPUT "enter file name"; FLE$
     IF FLE$ = "" THEN END
     NN$ = LEFT$(RIGHT$(FLE$, 12), 8) + ".PRT"
     PRINT NN$
     T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN FLE$ FOR INPUT AS #1
     ON ERROR GOTO 0
     IF T% = 1 THEN GOTO START
     CLOSE 1
     OPEN "R", #1, FLE$, 10
     FIELD #1, 2 AS T$, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
     LENFLE% = LOF(1) / 10
     OPEN NN$ FOR OUTPUT AS #8
     PRINT #8, : PRINT #8,
     PRINT #8, "THE FOLLOWING DATA IS FROM THE FILE, "; FLE$
     PRINT #8, "The number of records on the file is : "; LENFLE%
     PRINT , "The number of records on the file is : "; LENFLE%;
                                                        ---->FLE$
     DO
     PRINT
     INPUT "STARTING RECORD # "; RNI%
     IF RNI% = 0 THEN EXIT DO
     INPUT "LAST RECORD # "; RNMAX%
     IF RNMAX% = 0 THEN EXIT DO
     IF RNMAX% < 0 OR RNMAX% > LENFLE% THEN RNMAX% = LENFLE%
     IF RNI% < 0 OR RNI% > RNMAX% THEN RNI% = RNMAX%
     PRINT #8, "PRINT FROM RECORD # "; RNI%; "  to "; RNMAX%
     PRINT #8, "Record #", "T", "  x", "  y", "Pupdiam", "Length"
     FOR I% = RNI% TO RNMAX%
     GET #1, I%
     PRINT #8, I%, CVI(T$), CVI(A$), CVI(B$), CVI(C$), CVI(D$)
     NEXT I%
     LOOP
     CLOSE 1, 8

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

PROGRAM PRNMRG

```
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                               'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                                  'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                             'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE        'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                              'TIME HISTORY FRAME #
     TGTX AS SINGLE                                       'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                                     'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS STRING * 3                                         'DICE
     COUNTDOWN AS INTEGER                                         'DICE
     CONTFIX AS STRING * 1                                        'IS THIS
                    ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
     SPEED AS STRING * 1                          'SPLADT  S-on, F-off
     AIDON AS STRING * 1                                   'A-on, F-off
     SPARE AS STRING * 8
END TYPE
DIM frmt$
DIM XX AS FIXCOMB
DIM XXX$
frmt1$ = "## \  \ #### #### \ \ ###.## #### "
frmt2$ = "###.## ###.## ###.## ###.## \ \ ### ! !"
frmt3$ = "! ! \\ ! ! ! ! \      \"
START:
PRINT " This program lists records from oculometer .MRG files."
DO
     PRINT : PRINT
     INPUT "enter fULL FILE DESCRIPTOR WITH EXTENSION"; FLE$
     IF FLE$ = "" THEN END
     FLE1$ = LEFT$(FLE$, LEN(FLE$) - 4)
     OPEN FLE1$ + ".MGX" FOR RANDOM AS #1 LEN = 4
     IF LOF(1) = 0 THEN PRINT " can't find index file "; FLE1$ +
                                          ---->".MGX": CLOSE 1: STOP
     LENMGX% = LOF(1) / 4: PRINT "NUMBER OF RECORDS ON MERGE FILE
                                          ----> ="; LENMGX%; FLE$
     OPEN FLE$ FOR INPUT AS #2
     OPEN FLE1$ + ".PT1" FOR OUTPUT AS #8
     PRINT #8, : PRINT #8,
     PRINT #8, "THE FOLLOWING DATA IS FROM THE FILE, "; FLE$
     PRINT #8, "The number of records on the file is : "; LENMGX%
     DO
     PRINT
     INPUT "STARTING RECORD # "; RNI%
     IF RNI% = 0 THEN EXIT DO
     INPUT "LAST RECORD # "; RNMAX%
     IF RNMAX% = 0 THEN EXIT DO
     IF RNMAX% < 0 OR RNMAX% > LENMGX% THEN RNMAX% = LENMGX%
```

```
     IF RNI% < 0 OR RNI% > RNMAX% THEN RNI% = RNMAX%
     PRINT #8, "PRINT FROM RECORD # "; RNI%; "  to "; RNMAX%: PRI
                                                        ---->NT #8,
     PRINT #8, "Rec# Tp Typ   Fxt  PD TgID   Dist FrNo   TgtX   T
                                    ---->gtY   FixX   FixY Hdg  CD"
     GET #1, RNI%, N&
     SEEK 2, N&
     FOR I% = RNI% TO RNMAX%
     INPUT #2, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM,
----> XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FIX
     ---->X, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.CR
             ---->OSSCHECK, XX.ZONE, XX.SPEED, XX.AIDON, XX.SPARE
     PRINT #8, USING "#### "; I%;
     PRINT #8, USING frmt1$; XX.TGTTYPEN; XX.TGTTYPEC; XX.FIXLNGT
           ---->H; XX.PUPDIAM; XX.TGTID; XX.DISTANCE; XX.FRAMENO;
     PRINT #8, USING frmt2$; XX.TGTX; XX.TGTY; XX.FIXX; XX.FIXY;
                         ---->XX.HEADING; XX.COUNTDOWN; XX.CONTFIX;
     PRINT #8, USING frmt3$; XX.CROSSCHECK; XX.ZONE; XX.SPEED; XX
                                              ---->.AIDON; XX.SPARE
     NEXT I%
     LOOP
     CLOSE 1, 2, 8

LOOP
NOSUCHFILE:
PRINT "Couldn't find input file  "; FLE$
T% = 1
RESUME NEXT
```

Program Qklook

```
'?r?;RES;FONT 62;EXIT;
DEFINT I-N
DECLARE SUB BI1 (BIP%())
DECLARE SUB BI2 (BIP%())
DECLARE SUB CRESMRGFLE ()
DECLARE SUB FIN ()
DECLARE SUB FIXPOINTER ()
DECLARE SUB GETXX (FILENO%)
DECLARE SUB INIT ()
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE FUNCTION LOGS$ (SB$, A$)
DECLARE SUB PUTXX (FILENO%)
DECLARE SUB YESORNO (A$, B$)
CONST pi! = 3.14159
CONST nscanbuf = 400, nfixbuf = 5000
CONST SF! = .472, XOFF! = -3.27, YOFF! = -2.1, cpi! = 102.4, alph
                      ---->a! = -11.5 * pi! / 180, runoff! = -.34
CONST big! = 3!, little! = 1!
TYPE FIXCOMB
     TGTTYPEN AS INTEGER                         'NON ZERO MEANS HIT
     TGTTYPEC AS STRING * 4                            'TARGET TYPE
     FIXLNGTH AS INTEGER
     PUPDIAM AS INTEGER
     TGTID AS STRING * 3                      'ID OF CLOSEST TARGET
     DISTANCE AS SINGLE      'BETWEEN CLOSEST TARGET AND FIXATION
     FRAMENO AS INTEGER                       'TIME HISTORY FRAME #
     TGTX AS SINGLE                                'TARGET POSITION
     TGTY AS SINGLE
     FIXX AS SINGLE                              'FIXATION POSITION
     FIXY AS SINGLE
     HEADING AS INTEGER                                     'DICE
     COUNTDOWN AS INTEGER                                   'DICE
     CONTFIX AS STRING * 1                                  'IS THIS
                 ---->A CONTINUATION OF THE PREVIOUS FIXATION
     CROSSCHECK AS STRING * 1
     ZONE AS STRING * 2   'WHAT AREA OF THE TUBE IS THE FIXATION?
END TYPE
DIM frmt$
frmt$ = "## /  / #### #### / / ###.## #### ###.## ###.## ###.## #
                                 ---->##.## ### ### ! ! //"

DIM XX AS FIXCOMB
DIM XXX$
DIM ACID(1 TO 50) AS INTEGER
DIM ROUTE(1 TO 50) AS INTEGER
DIM ZONE(1 TO 50) AS INTEGER
DIM ACX(1 TO 50)  AS SINGLE
DIM ACY(1 TO 50)  AS SINGLE
DIM dicehead%(1 TO 20)
DIM dicetime%(1 TO 20)
DIM IAID AS INTEGER
DIM TIEPEC(1 TO 50) AS STRING * 4
DIM TIEPEN(1 TO 50) AS INTEGER
```

```
DIM BI1P(1 TO 7)  AS INTEGER'Buffered Input 1
DIM BI2P(1 TO 7)  AS INTEGER'Buffered Input 2
DIM BOP(1 TO 7)  AS INTEGER                              'Buffered Output
DIM FILEACP$, FILEDAT$, FILEDUM$, FILEDX2$, FILEIDX$, FILEMRG$, F
                                                       ---->ILESCN$
DIM FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%, F
                                                       ---->ILESCN%
DIM FIXPNTER(26000) AS INTEGER
DIM FIXLENGTH(1 TO nfixbuf)   AS INTEGER
DIM INTRACK(1 TO nfixbuf)   AS INTEGER
DIM NUMINTRACK, NUMOUTTRACK AS INTEGER
DIM N4SS, N5SS, N6SS, N7SS, N8SS, N9SS AS INTEGER
DIMT4SS, T5SS, T6SS, T7SS, T8SS, T9SS, P1SS, P2SS, P3SS AS SINGLE
DIM OCSCAN(1 TO nscanbuf, 1 TO 4) AS INTEGER
DIM PUPDIAM(1 TO nfixbuf)   AS INTEGER
DIM SHARED NUMFIX%, NUMSCAN%, NOSTAT%, NUMMRG%, FILENAME1$
                  ---->                         '*****COMMON******
DIM SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
DIM XLOOK(1 TO nfixbuf)   AS INTEGER
DIM YLOOK(1 TO nfixbuf)   AS INTEGER
DIM IDX(1 TO 2000) AS LONG
'    ** FILE NUMBERS **
'       #1 SCN                 #2 DAT                  #3 ACP
'       #4 IDX                 #5 DX2                  #6 MRG
'  #7 DUM                 #8                      #9 PRINTER.FLE
'
DIM PAGE$, FONT$
NOSTAT% = 10
DIM STATXS(NOSTAT%), STATYS(NOSTAT%) AS SINGLE
DIM STATID(NOSTAT%) AS STRING * 4
DATA "DEN ","IOC ","OM  ","KEAN","FLTS","WIVS","BYSN","TROZ","DRK
                                                    ---->O","JASN"
FOR I = 1 TO NOSTAT%: READ STATID(I): NEXT I
DATA 2.38,-19.49,-0.3,19.44,10.92,-10.24,-23.07,-8.22,29.42,14.67
FOR I = 1 TO NOSTAT%: READ STATXS(I): NEXT I
DATA -.63,24.92,6.1,28.79,14.1,14.1,-26.08,-11.24,-19.56,-9.23
FOR I = 1 TO NOSTAT%: READ STATYS(I): NEXT I
SB$ = "(MAIN "
PRTCONTROL$ = CHR$(33) + "R" + CHR$(33)
PAGE$ = PRTCONTROL$ + ";PAGE;EXIT;"                    'OFFICE
FONT$ = PRTCONTROL$ + ";RES;FONT 62; EXIT;"            'OFFICE
'PAGE$ = CHR$(12)                                        'HOME
'FONT$ = CHR$(27) + CHR$(80)                             'HOME
DIM SINAL!, COSAL!
SINAL! = SIN(alpha!): COSAL! = COS(alpha!)
XXX$ = "## \  \ #### #### \ \ ##.## #### ###.## ###.## ###.## ###
                                  ---->.## ### #### ! ! \\"
'XX.TGTTYPEN = 0: XX.TGTTYPEC = "UNK": XX.FIXLNGTH = 0: XX.PUPDIM
                                 ----> = 0: XX.TGTID = "Jil"
'XX.DISTANCE = 9999: XX.FRAMENO = 9999: XX.TGTX = 0: XX.TGTY = 0:
                              ----> XX.FIXX = 0: XX.FIXY = 0
'XX.HEADING = 999: XX.COUNTDOWN = 0: XX.CONTFIX = 0: XX.ZONE = 99
OPEN "BRENNAN" FOR INPUT AS #12
```

```
DO
CALL INIT
CALL FIXPOINTER
PRINT LOG$(SB$, " FINISHED FIXPOINTER")
CALL CRE8MRGFLE
PRINT LOG$(SB$, " FINISHED CRE8MRGFLE")                    'Close FILES
CALL FIN
LOOP
'..................................................................
'..................................................................
END     ' MAIN PROGRAM 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB BI1 (BIP%())

```
                                      'Increment 1'st INPUT file
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
SHARED OCSCAN()  AS INTEGER
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%
                                                  ---->, FILESCN%

        'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
NORIB = BIP%(3) - BIP%(2)                  'If buffer low AND EOF=.F.
IF NORIB < 0 THEN NORIB = NORIB + BIP%(1)
IF NORIB < BIP%(4) AND BIP%(6) = 0 THEN                  'Load buffer
     FORI = 1 TO BIP%(5)
     GET FILESCN%, , OCSCAN(BIP%(3), 1)
     BIP%(3) = BIP%(3) + 1
     IF BIP%(3) > BIP%(1) THEN BIP%(3) = 1
     IF EOF(FILESCN%) THEN
     BIP%(6) = BIP%(3)                               'Points at last
     EXIT FOR
     END IF
     NEXT I
END IF
JDUM = BIP%(2)
BIP%(2) = BIP%(2) + 1: IF BIP%(2) > BIP%(1) THEN BIP%(2) = 1 'Inc
                                              ---->rement first
                                              'READ BEYOND DATA
IF BIP%(2) = BIP%(6) THEN
     BIP%(2) = JDUM
     PRINT "****** ERROR READING EOF(FILESCN%) IN BI1 ******"
END IF
'..................................................................
'..................................................................
END SUB     ' BI1   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB BI2 (BIP%())

```
                         'Increment 2'nd INPUT file
'##########
'Parameters.............\ BIP%(), circular buffer pointers
'Other input data.......\
'Input files............\FILEDAT% =.DAT
'Other output data......\BIP%(), XLOOK(),YLOOK(),PUPDIAM(),FIXLEN
                                        ---->GTH(),INTRACK()
' \SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK, NUMINTRACK, NUMOUTTRACK
'Output files...........\
'Function...............\Read the fixation data into circular buf
                                                  ---->fers and
'                        \compute a few preliminary statistics.
'Comments...............\
'##########
SHARED INTRACK() AS INTEGER
SHARED FIXLENGTH() AS INTEGER
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED N4SS, N5SS, N6SS, N7SS, N8SS, N9SS AS INTEGER
SHARED T4SS, T5SS, T6SS, T7SS, T8SS, T9SS, P1SS, P2SS, P3SS AS SI
                                                      ---->NGLE
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%
                                                 ---->, FILESCN%
       'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
NORIB = BIP%(3) - BIP%(2)                'If buffer low AND EOF=.F.
IF NORIB < 0 THEN NORIB = NORIB + BIP%(1)
IF NORIB < BIP%(4) AND NOT EOF(FILEDAT%) THEN
    FOR I = 1 TO BIP%(5)                               'Load buffer
    GET FILEDAT%, , XLOOK(BIP%(3)): GET FILEDAT%, , YLOOK(BIP%(3
                        ---->))                        'read record
    IF NOT EOF(FILEDAT%) THEN
        GET FILEDAT%, , PUPDIAM(BIP%(3)): GET FILEDAT%, , FIXLE
                                            ---->NGTH(BIP%(3))
        SUMFIXLENGTH = SUMFIXLENGTH + FIXLENGTH(BIP%(3))
                            ---->                'total of fixations
        INTRACK(BIP%(3)) = 1
        IF XLOOK(BIP%(3)) = 0 AND YLOOK(BIP%(3)) = 0 AND PUPDIA
                          ---->M(BIP%(3)) < 11 THEN ' out of track
            INTRACK(BIP%(3)) = 0
            SUMOUTTRACK = SUMOUTTRACK + FIXLENGTH(BIP%(3))'tot
                                          ---->al out of track
            NUMOUTTRACK = NUMOUTTRACK + 1
            SELECT CASE FIXLENGTH(BIP%(3))
                CASE IS <= 3
                    N4SS = N4SS + 1: T4SS = T4SS + FIXLENGTH
                                                ---->(BIP%(3))
                CASE IS <= 12
                    N5SS = N5SS + 1: T5SS = T5SS + FIXLENGTH
                                                ---->(BIP%(3))
                CASE IS > 12
```

```
                    N6SS = N6SS + 1: T6SS = T6SS + FIXLENGTH
                                                  ---->(BIP%(3))
               END SELECT
          ELSE                       ' in track
               SUMINTRACK = SUMINTRACK + FIXLENGTH(BIP%(3))
                    ---->                          'total intrack
               NUMINTRACK = NUMINTRACK + 1
               SELECT CASE FIXLENGTH(BIP%(3))
                    CASE IS <= 3
                         N7SS = N7SS + 1: T7SS = T7SS + FIXLENGTH
                                                  ---->(BIP%(3))
                         P1SS = P1SS + PUPDIAM(BIP%(3))
                    CASE IS <= 12
                         N8SS = N8SS + 1: T8SS = T8SS + FIXLENGTH
                                                  ---->(BIP%(3))
                         P2SS = P2SS + PUPDIAM(BIP%(3))
                    CASE IS > 12
                         N9SS = N9SS + 1: T9SS = T9SS + FIXLENGTH
                                                  ---->(BIP%(3))
                         P3SS = P3SS + PUPDIAM(BIP%(3))
               END SELECT
          END IF
          BIP%(3) = BIP%(3) + 1
          IF BIP%(3) > BIP%(1) THEN BIP%(3) = 1
     ELSE
     EXIT FOR
     END IF
     NEXT I
END IF
BIP%(2) = BIP%(2) + 1: IF BIP%(2) > BIP%(1) THEN BIP%(2) = 1'Incr
                                                 ---->ement first
'.................................................................
'.................................................................
END SUB      'BI2  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB CRE8MRGFLE

```
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
SHARED BI2P() AS INTEGER                         'BufferedInput 2
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEDX2$, FILEIDX$, FILEMRG$
                                                ---->, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%
                                                ---->, FILESCN%
SHARED FIXLENGTH() AS INTEGER
SHARED FIXPNTER() AS INTEGER
```

```
SHARED XLOOK() AS INTEGER
SHARED YLOOK() AS INTEGER
SHARED PUPDIAM() AS INTEGER
SHARED XX AS FIXCOMB
'.................................................................
SB$ = "(CRE8MRGFLE "
'.................................................................
XX.TGTTYPEN = 0: XX.TGTTYPEC = "UNK": XX.FIXLNGTH = 0: XX.PUPDIAM
                                   ---->% = 0: XX.TGTID = "Jil"
XX.DISTANCE = 99.99: XX.FRAMENO = 9999: XX.TGTX = 0: XX.TGTY = 0:
                                 ----> XX.FIXX = 0: XX.FIXY = 0
XX.HEADING = 999: XX.COUNTDOWN = 0: XX.CONTFIX = "Z": XX.CROSSCHE
                                   ---->CK = "Z": XX.ZONE = "99"
SEEK #FILEDAT%, 1        'REWIND FILE & RESET BUFFER           '
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                          ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
'.................................................................
FOR I = 1 TO NUMMRG%
     CALL BI2(BI2P())
     II = BI2P(2)
     XX.PUPDIAM = PUPDIAM(II): XX.FRAMENO = FIXPNTER(I): XX.FIXX
                                       ---->= XLOOK(II) / cpi!
     XX.FIXY = YLOOK(II) / cpi!: XX.FIXLNGTH = FIXLENGTH(II)
     IF XLOOK(II) = 0 AND YLOOK(II) = 0 AND PUPDIAM(II) < 11 THEN
                                     ---->       ' out of track
          XX.TGTTYPEN = 89: XX.TGTTYPEC = "OUT"
          IF FIXLENGTH(II) < 13 THEN XX.TGTTYPEN = 80: XX.TGTTYPE
                                              ---->C = "BLNK"
     END IF
'         CALL PUTXX(FILEMRG%)
     XX.TGTTYPEN = 0: XX.TGTTYPEC = "UNK"
NEXT I
'CLOSE FILEMRG%
'.................................................................
'.................................................................

END SUB      'CRE8MRGFLE  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB FIN

```
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
PO = 1
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
```

```
SHARED N4SS, N5SS, N6SS, N7SS, N8SS, N9SS AS INTEGER
SHARED T4SS, T5SS, T6SS, T7SS, T8SS, T9SS, P1SS, P2SS, P3SS AS SI
                                                          ---->NGLE
CN! = 1 / 30: T4SS = CN! * T4SS: T5SS = CN! * T5SS: T6SS = CN! *
                                                          ---->T6SS
T7SS = CN! * T7SS: T8SS = CN! * T8SS: T9SS = CN! * T9SS
CN1! = 25.4 / 2048: P1SS = P1SS * CN1! / N7SS: P2SS = P2SS * CN1!
                        ----> / N8SS: P3SS = P3SS * CN1! / N9SS
SB$ = "(FIN "
IF NUMINTRACK <> 0 AND NUMOUTTRACK <> 0 THEN
     F1$ = "& ####.## SECONDS"
     F2$ = "& ##.## SECONDS"
     F3$ = "& ####.## ####.## ####.## ####.## ####.## ####.##"
     F4$ = "& ###.## ###.## ###.##"
     PRINT LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= "); NUMINTRAC
                         ---->K;        ' totals for .txt file
     PRINT LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS="); NUMOUTTRA
                                                          ---->CK
     PRINT USING F1$; LOG$(SB$, "TOTAL TIME IN TRACK IS  "); SUMI
                                                ---->NTRACK / 30
     PRINT USING F1$; LOG$(SB$, "TOTAL TIME OUT TRACK IS "); SUMO
                                               ---->UTTRACK / 30
     PRINT USING F1$; LOG$(SB$, "TOTAL FIXATION TIME IS  "); SUMF
                                              ---->IXLENGTH / 30
     PRINT USING F2$; LOG$(SB$, "AVERAGE IN TRACK FIXATION IS  "
                         ---->); SUMINTRACK / NUMINTRACK / 30
     PRINT USING F2$; LOG$(SB$, "AVERAGE OUT TRACK FIXATION IS  "
                       ---->); SUMOUTTRACK / NUMOUTTRACK / 30
     PRINT LOG$(SB$, "N4 THROUGH N9 = "); N4SS; N5SS; N6SS; N7SS;
                                              ----> N8SS; N9SS
     PRINT USING F3$; LOG$(SB$, "T4 THRU T9 = "); T4SS; T5SS; T6S
                                  ---->S; T7SS; T8SS; T9SS
     PRINT USING F4$; LOG$(SB$, "PUP DIAM 1 TO 3 = "); P1SS; P2SS
                                                   ---->; P3SS
     IF PO = 1 THEN
          PRINT #8, "@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
                              ---->@@@@@@@@@@@@@@@@@@@@@"
          PRINT #8, FILENAME1$, FILENAME1$, FILENAME1$, FILENAME1
                                              ---->$, FILENAME1$
          PRINT #8, LOG$(SB$, "NUMBER OF FIXATIONS IS  (-N1-)");
                                                   ---->NUMFIX%
          PRINT #8, LOG$(SB$, "NUMBER OF IN TRACK FIXATIONS= (-N2
               ---->-)"); NUMINTRACK       ' totals for .txt file
          PRINT #8, LOG$(SB$, "NUMBER OF OUT TRACK FIXATIONS=(-N3
                                        ---->-)"); NUMOUTTRACK
          PRINT #8, LOG$(SB$, "(-N4 THROUGH N9-) = "); N4SS; N5SS
                               ---->; N6SS; N7SS; N8SS; N9SS
          PRINT #8,
          PRINT #8, USING F1$; LOG$(SB$, "TOTAL FIXATION TIME IS
                              ----> (-T1-)"); SUMFIXLENGTH / 30
          PRINT #8, USING F1$; LOG$(SB$, "TOTAL TIME IN TRACK IS
                                ----> (-T2-)"); SUMINTRACK / 30
          PRINT #8, USING F1$; LOG$(SB$, "TOTAL TIME OUT TRACK IS
```

```
                                ----> (-T3-)"); SUMOUTTRACK / 30
        PRINT #8, USING F3$; "(-T4 THRU T9-)"; T4SS; T5SS; T6SS
                                          ---->; T7SS; T8SS; T9SS
        PRINT #8,
        PRINT #8, USING F4$; LOG$(SB$, "(-PUP DIAM 1 TO 3-) = "
                                          ---->); P1SS; P2SS; P3SS
        PRINT #8, : PRINT #8,
        PRINT #8, USING F2$; LOG$(SB$, "AVERAGE IN TRACK FIXATI
                  ---->ON IS   "); SUMINTRACK / NUMINTRACK / 30
        PRINT #8, USING F2$; LOG$(SB$, "AVERAGE OUT TRACK FIXAT
                 ---->ION IS  "); SUMOUTTRACK / NUMOUTTRACK / 30
        PRINT #8, LOG$(SB$, "NUMBER OF 4 SECOND SCANS IS  "); N
                                                   ---->UMSCAN%
        PRINT #8, LOG$(SB$, "NUMBER OF MERGE FILE FIXATIONS IS
                                               ----> "); NUMMRG%
    END IF
END IF
CLOSE 1, 2, 3, 4, 5, 6, 7, 8, 9
'.....................................................
'.....................................................
END SUB      ' FIN  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB FIXPOINTER

```
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
'***
'***       This routine assigns a time history record #, (I), to
'***        each fixation. The contents of FIXPNTER(N) indicates
'***          which time record the N'th fixation is associated
'***         with and therefore where one should seek the target
'***          of the fixation.
'***
SHARED FIXPNTER() AS INTEGER
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%
                                                ---->, FILESCN%
SHARED BI1P()  AS INTEGER                       'Buffered Input1
SHARED OCSCAN()  AS INTEGER
SHARED PAGE$, FONT$
'***
'***           The first scan # is recorded at time = 4 seconds.
'***           Therefore if the third scan # is 3 and the 4'th
'***           scan # is 10, then scans 5 through 11 point at
'***           targets recorded at t=12 seconds or i=4 since
'***           i=1 corresponds to t=0.
'***
```

```
'***          The first entry on the time history file is time=0.
'***            The second entry is time=4 seconds.
'***            The first scan toggle precedes slightly this 2'nd
'***          entry i.e. ~4 seconds into the run. Thus spake Buddy
'***      after careful consideration and investigation on 4/3/91.
'***
'***          If the 1'st scan # is k, then fixations <= (k+1) are
'***            pointed at time history frame 1, i.e. time=0.
'***
'***            The last time history frame is not used.
'***
SB$ = "(FIXPOINTER "
SEEK #FILESCN%, 1                    'REWIND FILE & RESET BUFFER          '
        'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                              ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1'nscanbuf is a constant
SCNPREV% = -1
FORI = 1 TO NUMSCAN%                  ' I is the record# on the ACP file
     CALL BI1(BI1P())                  ' NUMSCAN% defined in INIT
     scn% = OCSCAN(BI1P(2), 1)       'scn%= fixation# from SCN file
     IF scn% >= 1 AND scn% <= NUMFIX% THEN
          IF scn% > SCNPREV% THEN
               FOR J = SCNPREV% + 2 TO scn% + 1
                    IF J <= NUMFIX% THEN
                         FIXPNTER(J) = I
                         JTEMP = J
                    END IF
               NEXT J
               SCNPREV% = scn%
          ELSE
               IF scn% < SCNPREV% THEN
                    PRINT LOG$(SB$, "********* ERROR SCAN POINTER
                                        ---->DECREASING  "); I, scn%
               END IF
          END IF
     ELSE
          PRINT LOG$(SB$, "FIXATION POINTER OUT OF RANGE"); I; BI
                                                    ---->1P(2); scn%
     END IF
NEXT I
NUMMRG% = JTEMP                                         ' NUMMRG% DEFINED
PRINT LOG$(SB$, "NUMBER OF MERGE FILE FIXATIONS IS  "); NUMMRG%
'The remainder of this routine prints out the time history record
                                                  ----> corresponding
'to a fixation. It does this for the first and last "NOP" fixatio
                                                  ---->ns. Its set up
'the DISC Kyocera printer. Clean all this out later.
PRINT #9, PAGE$
PRINT #9, FONT$
PRINT #9, : PRINT #9, LOGS$(SB$, " FIRST PART OF FILE............
                                        ---->............ "): L = 3
NOP = 1600
```

```
FOR J = 1 TO NOP
     PRINT #9, USING "#####"; FIXPNTER(J);
     IF J MOD 24 = 0 THEN
          PRINT #9, : L = L + 1
     END IF
     IF L = 60 THEN
     PRINT #9, PAGE$: PRINT #9, : PRINT #9,
     L = 3
     END IF
NEXT J
L = 3
PRINT #9, PAGE$: PRINT #9,
PRINT #9, LOGS$(SB$, " LAST PART OF FILE.........................
                                                   ---->... "); L = 5
FOR J = NUMFIX% - NOP + 1 TO NUMFIX%
     PRINT #9, USING "#####"; FIXPNTER(J);
     IF J MOD 24 = 0 THEN
          PRINT #9, : L = L + 1
     END IF
     IF L = 60 THEN
          PRINT #9, PAGE$: PRINT #9, : PRINT #9,
          L = 3
     END IF
NEXT J

'.................................................................
'.................................................................
END SUB      ' FIXPOINTER     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB GETXX (FILENO%)

```
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
'Read the array XX from a record on the FILEMRG$ file.
SHARED XX AS FIXCOMB
INPUT #FILENO%, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM
---->, XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FI
     ---->XX, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.C
                                           ---->ROSSCHECK, XX.ZONE
'.................................................................
'.................................................................
END SUB      ' GETXX     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB INIT

```
'##########
'Parameters.............\
```

```
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
SHARED BI1P()  AS INTEGER                                          'BufferedInput 1
SHARED BI2P()  AS INTEGER                                          'BufferedInput 2
SHARED BOP()  AS INTEGER                                            'BufferedOutput
SHARED FILEACP$, FILEDAT$, FILEDUM$, FILEDX2$, FILEIDX$, FILEMRG$
                                                               ---->, FILESCN$
SHARED FILEACP%, FILEDAT%, FILEDUM%, FILEDX2%, FILEIDX%, FILEMRG%
                                                               ---->, FILESCN%
SHARED IAID AS INTEGER
SHARED NUMINTRACK, NUMOUTTRACK AS INTEGER
SHARED N4SS, N5SS, N6SS, N7SS, N8SS, N9SS AS INTEGER
SHARED T4SS, T5SS, T6SS, T7SS, T8SS, T9SS, P1SS, P2SS, P3SS AS SI
                                                                     ---->NGLE
SHARED SUMFIXLENGTH, SUMINTRACK, SUMOUTTRACK AS LONG
SHARED XX AS FIXCOMB
SB$ = "(INIT "
          'SIZE,FIRST,LAST,TRIG,NREC,NEOF,P1
I = nscanbuf: BI1P(1) = I: BI1P(2) = 0: BI1P(3) = 1: BI1P(4) = .2
                                                                     ----> * I
BI1P(5) = .7 * I: BI1P(6) = 0: BI1P(7) = 1
'FOR L = 1 TO 7: print BI1P(L): NEXT L
I = nfixbuf: BI2P(1) = I: BI2P(2) = 0: BI2P(3) = 1: BI2P(4) = .2
                                                                      ---->* I
BI2P(5) = .7 * I: BI2P(6) = 0: BI2P(7) = 1
I = nfixbuf: BOP(1) = I: BOP(2) = 1: BOP(3) = 1: BOP(4) = .9 * I
BOP(5) = .8 * I: BOP(6) = 0: BOP(7) = 1
NUMFIX% = 0: NUMSCAN% = 0
SUMFIXLENGTH = 0: SUMINTRACK = 0: SUMOUTTRACK = 0
NUMINTRACK = 0: NUMOUTTRACK = 0
N4SS = 0: N5SS = 0: N6SS = 0: N7SS = 0: N8SS = 0: N9SS = 0
T4SS = 0!: T5SS = 0!: T6SS = 0!: T7SS = 0!: T8SS = 0!: T9SS = 0!
'..................................................................
'INPUT " ENTER OCULOMETER FILE NAME   ", FILENAME$
'FILENAME$ = COMMAND$
LINE INPUT #12, FILENAME$
IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
SELECT CASE MID$(FILENAME1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
```

```
          IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                            ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                     ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
PR$ = "PRINTER.FLE"
OPEN PR$ FOR OUTPUT AS #9
FILEACP$ = FILENAME$ + ".ACP": FILEACP% = 3   'append extension
FILEDAT$ = FILENAME$ + ".DAT": FILEDAT% = 2    'append extension
FILEDUM$ = FILENAME$ + ".DUM": FILEDUM% = 7
FILEDX2$ = FILENAME$ + ".DX2": FILEDX2% = 5
FILEIDX$ = FILENAME$ + ".IDX": FILEIDX% = 4
FILEMRG$ = FILENAME$ + ".MRG": FILEMRG% = 6
FILEOUT$ = FILENAME$ + ".OUT"
FILESCN$ = FILENAME$ + ".SCN": FILESCN% = 1
FILEQK$ = FILENAME1$ + "QCK": FILEQK% = 8
OPEN FILEQK$ FOR OUTPUT AS FILEQK%
'                       COMMENT: FILE NAME SHOULD LOOK LIKE
'                                  "C:\FASAFILE\CRONE\CC10SLCE"
'                       COMMENT: FILE NAME SHOULD LOOK LIKE
'                                  "C:\FASAFILE\CRONE\CC10SLCE"
'.............................................................
OPEN FILESCN$ FOR BINARY AS #1      ' oculometer scan file
NUMSCAN% = LOF(1) / 2       'can the file be found
IF NUMSCAN% = 0 THEN
     PRINT LOG$(SB$, FILESCN$); " FILE NOT FOUND" ' fix this test
     EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF 4 SECOND SCANS IS  "); NUMSCAN%
END IF
'.............................................................
OPEN FILEDAT$ FOR BINARY AS #2 'oculometer .dat file
NUMFIX% = LOF(2) / 8      'can the file be found
IF NUMFIX% = 0 THEN
     PRINT LOG$(SB$, FILEDAT$); "FILE NOT FOUND"  ' fix this test
     EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF FIXATIONS IS  "); NUMFIX%

END IF
'.............................................................
'.............................................................
'OPEN FILEIDX$ FOR INPUT AS #4
'.............................................................
'.............................................................
END SUB       ' INIT   'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PRNDAT (STRT&, NOR%)

```
'##########
'Parameters.............\
'Other input data.......\
```

```
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
'INPUT " ENTER OCULOMETER FILE NAME   ", FILENAME$
'IF FILENAME$ = "" THEN PRINT : PRINT : PRINT : STOP
FILENAME$ = "C:\FASAFILE\CRONE\CC10SLCE"
FILEDAT$ = FILENAME$ + ".DAT"        'append extension
OPEN FILEDAT$ FOR BINARY AS #2 'oculometer .dat file
NUMFIX% = LOF(2) / 8       'can the file be found
IF NUMFIX% = 0 THEN
     PRINT LOG$(SB$, FILEDAT$); "FILE NOT FOUND"  ' fix this test
     EXIT SUB
ELSE : PRINT LOG$(SB$, "NUMBER OF FIXATIONS IS  "); NUMFIX%
END IF
SEEK 2, (STRT& - 1) * 8 + 1
FOR I = 1 TO NOR%
     IF NOT EOF(2) THEN
          GET 2, , X1%: GET 2, , X2%: GET 2, , X3%: GET 2, , X4%
          PRINT USING "####"; X1%; X2%; X3%; X4%
     END IF
NEXT I
CLOSE 2
'..............................................................
'..............................................................
END SUB      ' PRNDAT     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB PUTXX (FILENO%)

```
'##########
'Parameters.............\
'Other input data.......\
'Input files............\
'Other output data......\
'Output files...........\
'Function...............\
'Comments...............\
'##########
'Write the array XX to a record on the FILEMRG$ file.
SHARED XX AS FIXCOMB
WRITE #FILENO%, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM
---->, XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FI
     ---->XX, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.C
                                            ---->ROSSCHECK, XX.ZONE
'..............................................................
'..............................................................
END SUB      ' PUTXX     'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

PROGRAM SEQNCE1

```
DEFINT I-N
DECLARE SUB FINDTGTPL (L%)
DECLARE SUB INIT (FILENAME$, FILENAME1$)
DECLARE SUB FILBLNK1 ()
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB PRNZONE (ZONE!())
DECLARE SUB READSEQ ()
CONST NOCCS = 4, lg1 = 5, lg2 = (NOCCS + 1) * lg1
'Max order CCS's,  # of accumulators each- N, t, t*t, D, D*D
CONST NOZ = 4, NOZP = 10, NOTGT = 6, NOTP = 16'# of rows in zone
                                                      ---->pair array,
                        ' # of target pairs...
DIM ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, TYPE
                                                      ---->A$, TYPEB$
DIM IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEEDA$,
                                                      ----> SPEEDB$
DIM IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
DIM AAAA$
DIM IAID AS INTEGER
DIM BLANK1(lg2)
DIM ZNE$(NOZP), TGTS$(NOTP), SLOT$(6)
DIM ZONE(NOZP, lg2), TGTS(NOTP, lg2), SLOT(6, 15)
DIM ZONED(NOZP, lg2), TGTSD(NOTP, lg2)
DATA "1/1","1/2","1/3","2/3","2/2","1/4","3/3","OTH","BTBK","ALL"
FOR I = 1 TO NOZP: READ ZNE$(I): NEXT I
DATA 1,2,3,6,2,5,4,8,3,4,7,8,6,8,8,8
FOR I = 1 TO NOZ: FOR J = 1 TO NOZ: READ I2ZN(I, J): NEXT J: NEXT
                                                      ----> I
DATA "ALL","AC/AC","AC/OWNTG","AC/OTHTAG","TAG/TAG","AC/OM","TAG/
                                                      ---->OM","AC/FNL"
DATA "TAG/FNL","AC/AID","TAG/AID","AID/AID","AC/OTH","TAG/OTH","O
                                                      ---->TH","NOTA"
FOR I = 1 TO NOTP: READ TGTS$(I): NEXT I
DIM ITGTN(NOTGT, NOTGT)
DATA2,4,10,6,8,13,4,5,11,7,9,14,10,11,12,15,15,15,6,7,15,15,15,15
DATA 8,9,15,15,15,15,13,14,15,15,15,15
FOR I = 1 TO NOTGT: FOR J = 1 TO NOTGT: READ ITGTN(I, J): NEXT J:
                                                      ----> NEXT I
DATA "SL/XX","AC/OWNSL","AC/OTHSL","TAG/OWNSL","TAG/OTHSL","SL/OT
                                                      ---->H"
FOR I = 1 TO 6: READ SLOT$(I): NEXT I
'    ** FILE NUMBERS **
'       #1 FILE INDEX          #2                    #3
'       #4                     #5                    #6
'       #7 CCS                 #8 PR1                #9
'
SPSINV! = 1 / 30: SB$ = " (CNTSEQ_1  "
AAAA$ = " Finished-End of list file"
HEADING$ = "  SEQ#  STRT-FNSH TOG  TYPE     TYPE      TGTID   TIM
                         ---->E  ZONE   AID   SPD   LPOV  DSTAB"
INPUT " Enter full file descriptor for Index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #1
```

```
nof = 0
DO WHILE NOT EOF(1)
     INPUT #1, FILENAME$
     IF LEN(FILENAME$) <= 4 THEN EXIT DO
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     CALL INIT(FILENAME$, FILENAME1$)
     FLSTRNG$ = FLSTRNG$ + FILENAME1$
     OPEN FILENAME$ + ".CCS" FOR INPUT AS #7
     OPEN FILENAME$ + ".PR1" FOR OUTPUT AS #8
     PRINT #8, LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(24);
                                                            ---->FILENAME1$
     PRINT LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(24); FILE
                                                                ---->NAME1$
     PRINT LOG$(SB$, " Start"); EOF(1)
     nof = nof + 1
     DO
     CALL READSEQ
     CALL FILBLNK1
     IZ = VAL(ZONEA$): JZ = VAL(ZONEB$)
     IF IZ >= 1 AND IZ <= NOCCS AND JZ >= 1 AND JZ <= NOCCS THEN
          L = I2ZN(IZ, JZ)
     ELSE
          L = NOZP - 1                'Bit Bucket
          PRINT " ZBB"; IZ; JZ;
     END IF
     'Accumulate BLANK array in ZONE and ZONED
     FORK = 1 TO lg2: ZONE(L, K) = ZONE(L, K) + BLANK1(K): NEXT K
     FOR K = 1 TO lg2: ZONE(NOZP, K) = ZONE(NOZP, K) + BLANK1(K):
                                                               ----> NEXT K
     FOR K = 1 TO lg2: ZONED(L, K) = ZONED(L, K) + BLANK1(K): NEX
                                                                   ---->T K
     FOR K = 1 TO lg2: ZONED(NOZP, K) = ZONED(NOZP, K) + BLANK1(K
                                                              ---->): NEXT K
     IF EOF(7) THEN EXIT DO
     LOOP
     CALL PRNZONE(ZONE())
     CLOSE 7, 8
     PRINT LOG$(SB$, " FINISHED SEARCH"); EOF(1)
     'CALL FIN                                             'Close FILES
LOOP
OPEN "ZNETOT" + LTRIM$(STR$(IAID)) + ".PR1" FOR OUTPUT AS #8
PRINT #8, LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(10); "Numb
                                              ---->er of Files= "; nof
PRINT #8, FLSTRNG$
PRINT #8,
CALL PRNZONE(ZONED())
CLOSE
PRINT AAAA$
'..............................................................
'..............................................................
END     ' MAIN PROGRAM
```

SUB FILBLNK1

```
'##########
'Purpose................\
'                       \
'                       \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                        ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                             ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
SHARED BLANK1()
SB$ = "(FILBLNK1 "
FOR I = 1 TO lg2: BLANK1(I) = 0!: NEXT I
I = ISWTCHNO: IF I > NOCCS THEN I = NOCCS      'Choose # of Scans
IF I < 1 THEN PRINT LOG$(SB$, "ILLEGAL CCS ORDER < 1 "), I: STOP
J = lg1 * (I - 1) + 1
BLANK1(J) = 1: BLANK1(J + 1) = ISEQT: BLANK1(J + 2) = BLANK1(J +
                                    ---->1) * BLANK1(J + 1)
BLANK1(J + 3) = DIST: BLANK1(J + 4) = DIST * DIST
J = (NOCCS * lg1) + 1
BLANK1(J) = 1: BLANK1(J + 1) = ISEQT: BLANK1(J + 2) = BLANK1(J +
                                    ---->1) * BLANK1(J + 1)
BLANK1(J + 3) = DIST: BLANK1(J + 4) = DIST * DIST
'.......................................................
'.......................................................
END SUB       'FILBLNK1
```

SUB FINDTGTPL (L)

```
'##########
'Purpose................\
'                       \
'                       \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
```

```
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                                    ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                                    ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
SHARED ITGTN()
J1 = 7: J2 = 7
SELECT CASE ITYPENOA
      CASE 10                                           'AC
      J1 = 1
      CASE 15                                           'TAG
      J1 = 2
      CASE 20, 28, 32, 33, 34, 35, 36, 38, 39           'SLGR
      J1 = 3
      CASE 50                                           'OM
      J1 = 4
      CASE 51                                           'FNL
      J1 = 5
      CASE 52, 53, 55, 56                 'OTH, LINE,LIST OR OM
      J1 = 6
      CASE ELSE                               'SHOULD BE EMPTY
      J1 = 7
END SELECT
SELECT CASE ITYPENOB
      CASE 10                                           'AC
      J2 = 1
      CASE 15                                           'TAG
      J2 = 2
      CASE 20, 28, 32, 33, 34, 35, 36, 38, 39           'SLGR
      J2 = 3
      CASE 50                                           'OM
      J2 = 4
      CASE 51                                           'FNL
      J2 = 5
      CASE 52, 53, 55, 56                 'OTH, LINE,LIST OR OM
      J2 = 6
      CASE ELSE                               'SHOULD BE EMPTY
      J2 = 7
END SELECT
IF J1 = 6 AND ITYPENOA = 55 AND LEFT$(IDA$, 2) = "OM" THEN J1 = 4
IF J2 = 6 AND ITYPENOB = 55 AND LEFT$(IDB$, 2) = "OM" THEN J2 = 4
IF J1 = 7 OR J2 = 7 THEN
      L = 16
      PRINT ISEQNUM; TYPEA$; TYPEB$
ELSE
      L = ITGTN(J1, J2)
END IF
IF L = 4 AND IDA$ = IDB$ THEN L = 3
'..........................................................
'..........................................................
END SUB     'FINDTGTPL
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose...............\ Initialize parameters on both circular
                                                    ---->buffers
'             \ Initialize sums to zero.  Let user choose partic-
'                \ ular run for analysis.  Determine aid type for
'                \ subsequent branching. Open FILESCN$, FILEDAT$,
'                       \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                ---->ables, sums
'                    \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED IAID AS INTEGER
SHARED ZONE(), TGTS(), SLOT()
SHARED BLANK1()
SB$ = "(INIT "
'...........................................................
'...........................................................
SELECT CASE MID$(FILENAME1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
          IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                   ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                 ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
FOR J = 1 TO lg2: FOR I = 1 TO NOZP: ZONE(I, J) = 0: NEXT I: NEXT
                                                          ----> J
FOR J = 1 TO lg2: FOR I = 1 TO NOTP: TGTS(I, J) = 0: NEXT I: NEXT
                                                          ----> J
SELECT CASE IAID
     CASE 11, 12                                    'Manual's
     CASE 21, 22                                    'DICE
     CASE 31, 32                                    'GRAPHIC
```

```
    CASE 41, 42                                               'SLOTS
    FOR J = 1 TO 3: FOR I = 1 TO 6: SLOT(I, J) = 0: NEXT I: NEXT
                                                            ----> J
    CASE ELSE
    PRINT LOG$(SB$, "ILLEGAL AID "), IAID: STOP
END SELECT
'.....................................................................
'.....................................................................
END SUB       ' INIT
```

SUB PRNZONE (ZONE())

```
'###########
'Purpose................\
'                       \
'                       \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'###########
SHARED ZNE$(), TGTS$(), SLOT$()
A1$ = "  ##.#"                    'TBAR'S
A2$ = "  #.#"                     'DBARS
A3$ = " ###### ##### #### #### ######"          'N's
A4$ = " ####.# ###.# ##.# ##.# ####.#"      'T's SECONDS
A5$ = "  ###.# ##.# #.# #.# ###.#"          'D INCHES
A6$ = " ###### ###### ##### ##### ######"
B1$ = SPACE$(14) + "Zone Pairs by order of cross check scan"
'B2$ = SPACE$(10) + "Average Duration" + SPACE$(7) + "Average dis
                   ---->tance A/B" + SPACE$(9) + "# OF CCS"
B2$ = SPACE$(10) + "Average Duration" + SPACE$(12) + "Average dis
                                                  ---->tance A/B"
'B3$ = "Pairs  1    2    3    4   ALL    1    2    3    4   ALL"
B3$ = "Pairs  12    3    4   ALL         1    2    3    4   ALL "
PRINT #8, B1$: PRINT #8, B2$: PRINT #8, B3$
FOR L = 1 TO NOZP
    PRINT #8, USING "\  \"; ZNE$(L);
    FOR I = 1 TO lg2 STEP lg1
    X = 0: IF ZONE(L, I) <> 0 THEN X = ZONE(L, I + 1) / ZONE(L,
                                                    ---->I) / 30
    PRINT #8, USING A2$; X;
    NEXT I
    PRINT #8, "      ";
    FOR I = 1 TO lg2 STEP lg1
    X = 0: IF ZONE(L, I) <> 0 THEN X = ZONE(L, I + 3) / ZONE(L,
                                                        ---->I)
    PRINT #8, USING A2$; X;
```

```
        NEXT I
        PRINT #8,
  NEXT L
  PRINT #8,
  'B4$ = SPACE$(10) + "Std Dev Duration" + SPACE$(7) + "Std Dev dis
                        ---->tance A/B" + SPACE$(9) + "Time, Sec's"
  'B5$ = "Pairs  1    2    3    4   ALL   1    2    3    4   ALL"
  B4$ = SPACE$(10) + "Std Dev Duration" + SPACE$(12) + "Std Dev dis
                                                    ---->tance A/B"
  B5$ = "Pairs1    2    3    4   ALL        1    2    3    4   ALL"
  PRINT #8, B4$: PRINT #8, B5$
  FOR L = 1 TO NOZP
        PRINT #8, USING "\  \"; ZNE$(L);
        FORI = 1 TO lg2 STEP lg1                    'STD SEQUENCE TIME
        IF ZONE(L, I) > 1 THEN
              X = ZONE(L, I + 1) / ZONE(L, I) / 30
              Y= SQR(ZONE(L, I + 2) / 900 / (ZONE(L, I) - 1) - X * X)
        ELSE
              Y = 0
        END IF
        PRINT #8, USING A2$; Y;
        NEXT I
        PRINT #8, SPC(5);
        FOR I = 1 TO lg2 STEP lg1                   'STD DISTANCE A/B
        IF ZONE(L, I) > 1 THEN
              X = ZONE(L, I + 3) / ZONE(L, I)
              Y = SQR(ZONE(L, I + 4) / (ZONE(L, I) - 1) - X * X)
        ELSE
              Y = 0
        END IF
        PRINT #8, USING A2$; Y;
        NEXT I
        PRINT #8,
NEXT L
PRINT #8,
PRINT #8, SPACE$(14) + "# OF CCS" + SPACE$(29) + "Time, Sec's"
PRINT #8, "           1    2    3    4    ALL         1     2
                                          ---->  3    4     ALL"
FOR L = 1 TO NOZP
      PRINT #8, USING "\  \"; ZNE$(L);
      PRINT #8, USING A3$; ZONE(L, 1); ZONE(L, 6); ZONE(L, 11); ZO
                                  ---->NE(L, 16); ZONE(L, 21);
      PRINT #8, "     ";
      PRINT #8, USING A6$; ZONE(L, 2) / 30; ZONE(L, 7) / 30; ZONE(
            ---->L, 12) / 30; ZONE(L, 17) / 30; ZONE(L, 22) / 30;
      PRINT #8,
NEXT L
'.............................................................
'.............................................................
END SUB         'PRNZONE
```

SUB READSEQ

```
'##########
'Purpose...............\
'                      \
'                      \
'Parameters............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                                    ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                                    ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
LINE INPUT #7, A$
ISEQNUM = VAL(MID$(A$, 3, 4))
IRCNOA = VAL(MID$(A$, 9, 4))
IRCNOEND = VAL(MID$(A$, 14, 4))
ISWTCHNO = VAL(MID$(A$, 20, 2))
ITYPENOA = VAL(MID$(A$, 24, 2))
ITYPENOB = VAL(MID$(A$, 27, 2))
TYPEA$ = MID$(A$, 31, 4)
TYPEB$ = MID$(A$, 36, 4)
IDA$ = MID$(A$, 42, 3)
IDB$ = MID$(A$, 46, 3)
ISEQT = VAL(MID$(A$, 51, 4))
ZONEA$ = MID$(A$, 57, 1)
IF (ITYPENOA = 20 OR ITYPENOA = 28) THEN ZONEA$ = "1"
ZONEB$ = MID$(A$, 59, 1)
IF (ITYPENOB = 20 OR ITYPENOB = 28) THEN ZONEB$ = "1"
AIDONA$ = MID$(A$, 62, 1)
AIDONB$ = MID$(A$, 64, 1)
SPEEDA$ = MID$(A$, 67, 1)
SPEEDB$ = MID$(A$, 69, 1)
IOVERLAPT = VAL(MID$(A$, 72, 4))
DIST = VAL(MID$(A$, 78, 4))
HEADA$ = MID$(A$, 84, 3)
HEADB$ = MID$(A$, 88, 3)
ICNTA = VAL(MID$(A$, 93, 4))
ICNTB = VAL(MID$(A$, 98, 4))
XXX1$ = "  ####  #### ####  ##  ## ##"
XXX2$ = "  \  \ \  \  \ \ \ \  ####"
XXX3$ = "  ! !  ! !"
XXX4$ = "  ! !  ####  ##.#"
XXX5$ = "  \ \ \ \  #### ####"
'     PRINT #8, USING XXX1$; ISEQNUM; IRCNOA; IRCNOEND; ISWTCHNO;
                                      ----> ITYPENOA; ITYPENOB;
```

```
'       PRINT #8, USING XXX2$; TYPEA$; TYPEB$; IDA$; IDB$; ISEQT;
'       PRINT #8, USING XXX3$; ZONEA$; ZONEB$; AIDONA$; AIDONB$;
'       PRINT #8, USING XXX4$; SPEEDA$; SPEEDB$; IOVERLAPT; DIST;
'       PRINT #8, USING XXX5$; HEADA$; HEADB$; ICNTA; ICNTB
'.............................................................
'.............................................................
END SUB                 'READSEQ
```

PROGRAM SEQNCE2

```
'9/29/92 This program was modified. the old program is on file SE
                                                    ---->Q2OLD.bas
'1 target type and 4 target pairs were added.
DEFINT I-N
DECLARE SUB FINDTGTPL (L%)
DECLARE SUB INIT (FILENAME$, FILENAME1$)
DECLARE SUB FILBLNK1 ()
DECLARE FUNCTION LOG$ (SB$, A$)
DECLARE SUB PRNTGTS (ZONE!())
DECLARE SUB READSEQ ()
CONST NOCCS = 4, lg1 = 5, lg2 = (NOCCS + 1) * lg1
'Max order CCS's,  # of accumulators each- N, t, t*t, D, D*D
CONST NOZ = 4, NOZP = 10, NOTGT = 7, NOTP = 20'# of rows in zone
                                                    ---->pair array,
                          ' # of target pairs...
DIM ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, TYPE
                                                    ---->A$, TYPEB$
DIM IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEEDA$,
                                                    ----> SPEEDB$
DIM IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
DIM AAAA$
DIM IAID AS INTEGER
DIM BLANK1(lg2)
DIM ZNE$(NOZP), TGTS$(NOTP), SLOT$(6)
DIM ZONE(NOZP, lg2), TGTS(NOTP, lg2), SLOT(6, 15)
DIM ZONED(NOZP, lg2), TGTSD(NOTP, lg2)
DATA "1/1","1/2","1/3","2/3","2/2","1/4","3/3","OTH","BTBK","ALL"
FOR I = 1 TO NOZP: READ ZNE$(I): NEXT I
DATA 1,2,3,6,2,5,4,8,3,4,7,8,6,8,8,8
FOR I = 1 TO NOZ: FOR J = 1 TO NOZ: READ I2ZN(I, J): NEXT J: NEXT
                                                    ----> I
DATA "ALL","AC/AC","AC/OWNTG","AC/OTHTAG","TAG/TAG","AC/OM","TAG/
                                                    ---->OM","AC/FNL"
DATA "TAG/FNL","AC/AID","TAG/AID","AID/AID","AC/OTH","TAG/OTH","O
                                                    ---->TH","&/AC"
DATA "&/TAG","&/AID","&/OTH","NOTA"
FOR I = 1 TO NOTP: READ TGTS$(I): NEXT I
DIM ITGTN(NOTGT, NOTGT)
DATA 2,4,10,6,8,13,16,4,5,11,7,9,14,17,10,11,12,15,15,15,18,6,7,1
                                                    ---->5,15,15,15,19
DATA 8,9,15,15,15,15,19,13,14,15,15,15,15,19,16,17,18,19,19,19,18
FOR I = 1 TO NOTGT: FOR J = 1 TO NOTGT: READ ITGTN(I, J): NEXT J:
                                                    ----> NEXT I
DATA "SL/XX","AC/OWNSL","AC/OTHSL","TAG/OWNSL","TAG/OTHSL","SL/OT
                                                    ---->H"
FOR I = 1 TO 6: READ SLOT$(I): NEXT I
'    ** FILE NUMBERS **
'       #1 FILE INDEX           #2                       #3
'       #4                      #5                       #6
'       #7 CCS                  #8 PR2                   #9
'
SPSINV! = 1 / 30: SB$ = " (CNTSEQ_1  "
```

```
AAAA$ = " Finished-End of list file"
HEADING$ = "  SEQ#  STRT-FNSH TOG  TYPE     TYPE      TGTID   TIM
                              ---->E  ZONE   AID    SPD    LPOV  DSTAB"
INPUT " Enter full file descriptor for Index file  ", INDEX$
OPEN INDEX$ FOR INPUT AS #1
nof = 0
DO WHILE NOT EOF(1)
     INPUT #1, FILENAME$
     IF LEN(FILENAME$) <= 4 THEN EXIT DO
     FILENAME1$ = RIGHT$(UCASE$(FILENAME$), 8)
     CALL INIT(FILENAME$, FILENAME1$)
     FLSTRNG$ = FLSTRNG$ + FILENAME1$
     OPEN FILENAME$ + ".CCS" FOR INPUT AS #7
     OPEN FILENAME$ + ".PR2" FOR OUTPUT AS #8
     PRINT #8, LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(24);
                                                      ---->FILENAME1$
     PRINT LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(24); FILE
                                                          ---->NAME1$
     PRINT LOG$(SB$, " Start"); EOF(1)
     nof = nof + 1
     DO
     CALL READSEQ
     CALL FILBLNK1      ' Blank1(1 X Lg2) contains increment in
              ' appropriate positions
     CALL FINDTGTPL(L)  ' L = ordinal of target pair
     'Accumulate BLANK array in TGTS and TGTSD
     FORK = 1 TO lg2: TGTS(L, K) = TGTS(L, K) + BLANK1(K): NEXT K
     FORK = 1 TO lg2: TGTS(1, K) = TGTS(1, K) + BLANK1(K): NEXT K
     FOR K = 1 TO lg2: TGTSD(L, K) = TGTSD(L, K) + BLANK1(K): NEX
                                                               ---->T K
     FOR K = 1 TO lg2: TGTSD(1, K) = TGTSD(1, K) + BLANK1(K): NEX
                                                               ---->T K
     IF EOF(7) THEN EXIT DO
     LOOP
     CALL PRNTGTS(TGTS())
     CLOSE 7, 8
     PRINT LOG$(SB$, " FINISHED SEARCH"); EOF(1)
     'CALL FIN                                          'Close FILES
LOOP
OPEN "c:\fasa\ZNETOT" + LTRIM$(STR$(IAID)) + ".PR2" FOR OUTPUT AS
                                                              ----> #8
PRINT #8, LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5); SPC(10); "Numb
                                   ---->er of Files= "; nof, FLSTRNG$
CALL PRNTGTS(TGTSD())
CLOSE
PRINT AAAA$
'.............................................................
'.............................................................
END     ' MAIN PROGRAM
```

SUB FILBLNK1

```
'##########
```

```
'Purpose...............\ Fill Blank1(LG2) array.  It contains LG
                                                          ---->1 X
'                  \ (ACCS + 1) buckets.  LG1 is the number of ac-
'                 \ cumulators: 1, t, t*t, d d*d.  One set for each
'Parameters.............\ Order of CCS and one set for total
'Other input data.......\ ISWTCHNO= order of CCS .
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                               ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                               ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
SHARED BLANK1()
SB$ = "(FILBLNK1 "
FOR I = 1 TO lg2: BLANK1(I) = 0!: NEXT I
I = ISWTCHNO: IF I > NOCCS THEN I = NOCCS      'Choose # of Scans
IF I < 1 THEN PRINT LOG$(SB$, "ILLEGAL CCS ORDER < 1 "), I: STOP
J = lg1 * (I - 1) + 1
BLANK1(J) = 1: BLANK1(J + 1) = ISEQT: BLANK1(J + 2) = BLANK1(J +
                                        ---->1) * BLANK1(J + 1)
BLANK1(J + 3) = DIST: BLANK1(J + 4) = DIST * DIST
J = (NOCCS * lg1) + 1
BLANK1(J) = 1: BLANK1(J + 1) = ISEQT: BLANK1(J + 2) = BLANK1(J +
                                        ---->1) * BLANK1(J + 1)
BLANK1(J + 3) = DIST: BLANK1(J + 4) = DIST * DIST
'...............................................................
'...............................................................
END SUB       'FILBLNK1
```

SUB FINDTGTPL (L)

```
'##########
'Purpose...............\ FIND L THE ORDINAL OF THE APPROPRIATE P
                                                        ---->AIR
'                      \
'                      \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
```

```
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                                    ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                                      ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
SHARED ITGTN()
J1 = 7: J2 = 7
SELECT CASE ITYPENOA
     CASE 10                                               'AC
     J1 = 1
     CASE 15                                               'TAG
     J1 = 2
     CASE 20, 32, 33, 34, 35, 36, 16, 17    'Slot,graph, tag "on"
'.............................................................
'Added case 16 and 17 on 10/1/92 to make pairs containing tag sho
                                                            ---->w the
'difference whether the count is activated "Dice on" or not. If o
                                                           ---->n, the
'tag will be counted as an aid
     J1 = 3
     CASE 28, 38, 39                                  'SLAC,GRAC,SPAC
     J1 = 7
     CASE 50                                               'OM
     J1 = 4
     CASE 51                                               'FNL
     J1 = 5
     CASE 52, 53, 55, 56                      'OTH, LINE,LIST OR OM
     J1 = 6
     CASE ELSE                                       'SHOULD BE EMPTY
     J1 = 8
END SELECT
SELECT CASE ITYPENOB
     CASE 10                                               'AC
     J2 = 1
     CASE 15                                               'TAG
     J2 = 2
     CASE 20, 32, 33, 34, 35, 36, 16, 17    'Slot,graph, tag "on"
'.............................................................
'Added case 16 and 17 on 10/1/92 to make pairs containing tag sho
                                                            ---->w the
'difference whether the count is activated "Dice on" or not. If o
                                                           ---->n, the
'tag will be counted as an aid
     J2 = 3
     CASE 28, 38, 39                                  'SLAC,GRAC,SPAC
     J2 = 7
     CASE 50                                               'OM
     J2 = 4
     CASE 51                                               'FNL
     J2 = 5
     CASE 52, 53, 55, 56                      'OTH, LINE,LIST OR OM
```

```
     J2 = 6
     CASE ELSE                                              'SHOULD BE EMPTY
     J2 = 8
END SELECT
IF J1 = 6 AND ITYPENOA = 55 AND LEFT$(IDA$, 2) = "OM" THEN J1 = 4
IF J2 = 6 AND ITYPENOB = 55 AND LEFT$(IDB$, 2) = "OM" THEN J2 = 4
IF J1 = 8 OR J2 = 8 THEN
     L = 20
     PRINT ISEQNUM; TYPEA$; TYPEB$
ELSE
     L = ITGTN(J1, J2)
END IF
IF L = 4 AND IDA$ = IDB$ THEN L = 3
'...........................................................
'...........................................................
END SUB      'FINDTGTPL
```

SUB INIT (FILENAME$, FILENAME1$)

```
'##########
'Purpose................\ Initialize parameters on both circular
                                                        ---->buffers
'               \ Initialize sums to zero.  Let user choose partic-
'                 \ ular run for analysis.  Determine aid type for
'                 \ subsequent branching. Open FILESCN$, FILEDAT$,
'                       \ FILEACP$ and store their lengths.
'Parameters.............\ none
'Other input data.......\
'Input files............\ FILESCN$, FILEDAT$, FILEACP$
'Output files...........\
'Other output data......\ File names & unit #'s. Initialized vari
                                                     ---->ables, sums
'                     \ and pointers and the branch variable IAID
'Function calls.........\ LOG$
'Subroutine calls.......\ none
'Comments...............\ I don't think I'm using this BOP stuff.
'##########
SHARED IAID AS INTEGER
SHARED ZONE(), TGTS(), SLOT()
SHARED BLANK1()
SB$ = "(INIT "
'...........................................................
'...........................................................
SELECT CASE MID$(FILENAME1$, 5, 1)
     CASE "M"
          IAID = 1
     CASE "D"
          IAID = 2
     CASE "G"
          IAID = 3
     CASE "S"
          IAID = 4
     CASE ELSE
```

```
        IAID = 9
END SELECT
IF IAID = 9 THEN PRINT LOG$(SB$, "CASE FROM FILENAME MUST BE MN,D
                                  ---->C,GR or SL"): PRINT : STOP
IAID1 = VAL(MID$(FILENAME1$, 7, 1))
IF IAID1 <> 1 AND IAID1 <> 2 THEN PRINT LOG$(SB$, "CASE FROM FILE
                     ---->NAME MUST BE 170 OR 210"): PRINT : STOP
IAID = IAID * 10 + IAID1: PRINT IAID
FOR J = 1 TO lg2: FOR I = 1 TO NOZP: ZONE(I, J) = 0: NEXT I: NEXT
                                                           ----> J
FOR J = 1 TO lg2: FOR I = 1 TO NOTP: TGTS(I, J) = 0: NEXT I: NEXT
                                                           ----> J
SELECT CASE IAID
    CASE 11, 12                                        'Manual's
    CASE 21, 22                                        'DICE
    CASE 31, 32                                        'GRAPHIC
    CASE 41, 42                                        'SLOTS
    FOR J = 1 TO 3: FOR I = 1 TO 6: SLOT(I, J) = 0: NEXT I: NEXT
                                                           ----> J
    CASE ELSE
    PRINT LOG$(SB$, "ILLEGAL AID "), IAID: STOP
END SELECT
'...................................................................
'...................................................................
END SUB       ' INIT
```

SUB PRNTGTS (TRGTS())

```
'##########
'Purpose................\
'                       \
'                       \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                       \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED ZNE$(), TGTS$(), SLOT$()
A1$ = "  ##.#"                  'TBAR'S
A2$ = "  #.#"                  'DBARS
A3$ = " ###### ##### #### #### ######"          'N's
A4$ = " ####.# ###.# ##.# ##.# ####.#"      'T's SECONDS
A5$ = "  ###.# ##.# #.# #.# ###.#"          'D INCHES
A6$ = " ###### ###### ##### ##### ######"
B1$ = SPACE$(19) + "Target Pairs by order of cross check scan"
B2$ = SPACE$(15) + "Average Duration" + SPACE$(12) + "Average dis
                                                    ---->tance A/B"
B3$ = "Pairs       1    2    3    4   ALL1    2    3    4   ALL "
```

```
PRINT #8, B1$: PRINT #8, B2$: PRINT #8, B3$
FOR L = 1 TO NOTP
     PRINT #8, USING "\        \"; TGTS$(L);
     FOR I = 1 TO lg2 STEP lg1
     X = 0: IF TRGTS(L, I) <> 0 THEN X = TRGTS(L, I + 1) / TRGTS(
                                                    ---->L, I) / 30

     PRINT #8, USING A2$; X;
     NEXT I
     PRINT #8, "      ";
     FOR I = 1 TO lg2 STEP lg1
     X = 0: IF TRGTS(L, I) <> 0 THEN X = TRGTS(L, I + 3) / TRGTS(
                                                         ---->L, I)

     PRINT #8, USING A2$; X;
     NEXT I
     PRINT #8,
NEXT L
B4$ = SPACE$(15) + "Std Dev Duration" + SPACE$(12) + "Std Dev dis
                                                   ---->tance A/B"

B5$ = "Pairs1    2    3    4   ALL        1    2    3    4   ALL"
PRINT #8, B4$: PRINT #8, B5$
FOR L = 1 TO NOTP
     PRINT #8, USING "\        \"; TGTS$(L);
     FORI = 1 TO lg2 STEP lg1                       'STD SEQUENCE TIME
     IF TRGTS(L, I) > 1 THEN
          X = TRGTS(L, I + 1) / TRGTS(L, I) / 30
          Y = SQR(TRGTS(L, I + 2) / 900 / (TRGTS(L, I) - 1) - X *
                                                          ----> X)

     ELSE
          Y = 0
     END IF
     PRINT #8, USING A2$; Y;
     NEXT I
     PRINT #8, SPC(5);
     FOR I = 1 TO lg2 STEP lg1                      'STD DISTANCE A/B
     IF TRGTS(L, I) > 1 THEN
          X = TRGTS(L, I + 3) / TRGTS(L, I)
          Y = SQR(TRGTS(L, I + 4) / (TRGTS(L, I) - 1) - X * X)
     ELSE
          Y = 0
     END IF
     PRINT #8, USING A2$; Y;
     NEXT I
     PRINT #8,
NEXT L
PRINT #8, SPACE$(19) + "# OF CCS" + SPACE$(29) + "Time, Sec's"
PRINT #8, "                    1     2    3    4     ALL          1
                                    ---->  2     3     4     ALL"

FOR L = 1 TO NOTP
     PRINT #8, USING "\        \"; TGTS$(L);
     PRINT #8, USING A3$; TRGTS(L, 1); TRGTS(L, 6); TRGTS(L, 11);
                                  ----> TRGTS(L, 16); TRGTS(L, 21);

     PRINT #8, "      ";
     PRINT #8, USING A6$; TRGTS(L, 2) / 30; TRGTS(L, 7) / 30; TRG
```

```
       ---->TS(L, 12) / 30; TRGTS(L, 17) / 30; TRGTS(L, 22) / 30;
     PRINT #8,
NEXT L
'..............................................................
'..............................................................
END SUB          'PRNTGTS
```

SUB READSEQ

```
'##########
'Purpose................\
'                        \
'                        \
'                        \
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'                        \
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
SHARED IAID AS INTEGER
SHARED ISEQNUM, IRCNOA, IRCNOEND, ISWTCHNO, ITYPENOA, ITYPENOB, T
                                              ---->YPEA$, TYPEB$
SHARED IDA$, IDB$, ISEQT, ZONEA$, ZONEB$, AIDONA$, AIDONB$, SPEED
                                                  ---->A$, SPEEDB$
SHARED IOVERLAPT, DIST, HEADA$, HEADB$, ICNTA, ICNTB
LINE INPUT #7, A$
ISEQNUM = VAL(MID$(A$, 3, 4))
IRCNOA = VAL(MID$(A$, 9, 4))
IRCNOEND = VAL(MID$(A$, 14, 4))
ISWTCHNO = VAL(MID$(A$, 20, 2))
ITYPENOA = VAL(MID$(A$, 24, 2))
ITYPENOB = VAL(MID$(A$, 27, 2))
TYPEA$ = MID$(A$, 31, 4)
TYPEB$ = MID$(A$, 36, 4)
IDA$ = MID$(A$, 42, 3)
IDB$ = MID$(A$, 46, 3)
ISEQT = VAL(MID$(A$, 51, 4))
ZONEA$ = MID$(A$, 57, 1)
ZONEB$ = MID$(A$, 59, 1)
AIDONA$ = MID$(A$, 62, 1)
AIDONB$ = MID$(A$, 64, 1)
SPEEDA$ = MID$(A$, 67, 1)
SPEEDB$ = MID$(A$, 69, 1)
'..........................................................
'Added 10/1/92 to make pairs containing tag show the difference
'whether the count is activated "Dice on" or not. If on, the
'tag will be counted as an aid
IF (IAID = 21 OR IAID = 22) AND ITYPENOA = 15 THEN
    IF AIDONA$ = "1" THEN ITYPENOA = 16
```

```
     IF SPEEDA$ = "1" THEN ITYPENOA = 17
END IF
IF (IAID = 21 OR IAID = 22) AND ITYPENOB = 15 THEN
     IF AIDONB$ = "1" THEN ITYPENOB = 16
     IF SPEEDB$ = "1" THEN ITYPENOB = 17
END IF
'...........................................................................
IOVERLAPT = VAL(MID$(A$, 72, 4))
DIST = VAL(MID$(A$, 78, 4))
HEADA$ = MID$(A$, 84, 3)
HEADB$ = MID$(A$, 88, 3)
ICNTA = VAL(MID$(A$, 93, 4))
ICNTB = VAL(MID$(A$, 98, 4))
XXX1$ = "  ####  #### ####  ##  ## ##"
XXX2$ = "  \  \ \  \  \ \ \ \  ####"
XXX3$ = "  ! !  ! !"
XXX4$ = "  ! !  ####  ##.#"
XXX5$ = "  \ \ \ \  #### ####"
'     PRINT #8, USING XXX1$; ISEQNUM; IRCNOA; IRCNOEND; ISWTCHNO;
                                        ----> ITYPENOA; ITYPENOB;
'     PRINT #8, USING XXX2$; TYPEA$; TYPEB$; IDA$; IDB$; ISEQT;
'     PRINT #8, USING XXX3$; ZONEA$; ZONEB$; AIDONA$; AIDONB$;
'     PRINT #8, USING XXX4$; SPEEDA$; SPEEDB$; IOVERLAPT; DIST;
'     PRINT #8, USING XXX5$; HEADA$; HEADB$; ICNTA; ICNTB
'...........................................................................
'...........................................................................
END SUB                'READSEQ
```

PROGRAM SIXIN1

```
TYPE REGTYPE
     AX AS INTEGER
     BX AS INTEGER
     CX AS INTEGER
     DX AS INTEGER
     BP AS INTEGER
     SI AS INTEGER
     DI AS INTEGER
     FLAGS AS INTEGER
     DS AS INTEGER
     ES AS INTEGER
END TYPE
DIM INREG AS REGTYPE
DIM OUTREG AS REGTYPE
DIM ATLX1(1 TO 6), ATLY1(1 TO 6)
'.............................................................
CPI! = 204.8:
DOTSIZE! = .01
EDGE = 120: HSTEP = 320: VSTEP = 160: ATLX0 = 100: ATLY0 = 20
ATLX1(1) = ATLX0: ATLX1(2) = ATLX0: ATLX1(3) = ATLX0:
ATLX1(4) = ATLX0 + HSTEP: ATLX1(5) = ATLX0 + HSTEP: ATLX1(6) = AT
                                                  ---->LX0 + HSTEP:
ATLY1(1) = ATLY0: ATLY1(2) = ATLY0 + VSTEP: ATLY1(3) = ATLY0 + 2
                                                       ---->* VSTEP
ATLY1(4) = ATLY0: ATLY1(5) = ATLY0 + VSTEP: ATLY1(6) = ATLY0 + 2
                                                       ---->* VSTEP
TITLE$ = " Look Point Positions In Oculometer Coordinates"
T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN "fleindx1" FOR INPUT AS #1
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find INDEX file :": END
     LL = 0
     SCREEN 12
     CLS 1
 DO
     LL = LL + 1
     INPUT #1, FILENAME$
     IFFILENAME$ = "" THEN PRINT "BLANK LINE INDEX FILE": EXIT DO
     RUN$ = RIGHT$(FILENAME$, 8)
     DAT$ = FILENAME$ + ".DAT"
     'PRINT RUN$, DAT$
     T% = 0
     ON ERROR GOTO NOSUCHFILE
          OPEN DAT$ FOR INPUT AS #2
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find .OCULOMETER DATA FILE :" +
                                                  ---->DAT$: EXIT DO

     CLOSE 2
     OPEN "R", #2, DAT$, 8
     FIELD #2, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
     LENFLE% = LOF(2) / 8        'OK THE .DAT FILE IS OPEN FOR RAND
```

```
'..............................................................----->OM INPUT
     BLX = -5: BLY = -5: TRX = 5: TRY = 5   ' CORNERS OF THE WINDO
                                                         ---->W, INCHES
     TLX1 = ATLX1(LL): TLY1 = ATLY1(LL)
     BRX1 = TLX1 + EDGE: BRY1 = TLY1 + EDGE
     WINDOW (BLX, BLY)-(TRX, TRY)
     VIEW (TLX1, TLY1)-(BRX1, BRY1), , 1
'..................................................................
     SFX! = 1 / CPI!
     SFY! = 1 / CPI!
     XO! = 0
     YO! = 0
     LOCATE 2, 23
     'PRINT TITLE$
'..................................................................
     J% = 0
     FOR I% = 1 TO LENFLE%
          GET #2, I%
          A% = CVI(A$): B% = CVI(B$): C% = CVI(C$): D% = CVI(D$)
          IF (A% <> 0 OR B% <> 0 OR C% > 10) AND D% > 3 THEN
               J% = J% + 1
               X! = SFX! * A% + XO!: Y! = SFY! * B% + YO!
               CIRCLE (X!, Y!), DOTSIZE
          END IF
     NEXT I%
          LOCATE TLY1 / 16, TLX1 / 8
          'ID$ = DATE$ + "   " + LEFT$(TIME$, 5) + "   " + RUN$ + "
                                                   ---->   " + STR$(J%)
          ID$ = RUN$ + "   " + STR$(J%)
          PRINT ID$
     '.............................................................
     CLOSE 2
LOOP WHILE NOT EOF(1)
CALL INTERRUPT(&H5, INREG, OUTREG)  '&H5 is print screen function
CLOSE 1
SCREEN 0
END
NOSUCHFILE:
T% = 1
RESUME NEXT
'..................................................................
```

PROGRAM SRCHDAT

```
OPEN "P.OUT" FOR OUTPUT AS #5
T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN "fleindex" FOR INPUT AS #2
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find INDEX file :": END
INPUT #2, inmax%, outmax%
PRINT #5, : PRINT #5, : PRINT #5, "      In-track trigger > "; in
          ---->max%, "Out-track trigger > "; outmax%: PRINT #5,
PRINT : PRINT : PRINT "      In-track trigger > "; inmax%, "Out-t
                              ---->rack trigger > "; outmax%: PRINT
PRINT #5, "        Run           Time   Rec #      X       Y
                                          ---->  PD     Counts"
PRINT "        Run Time   Rec #      X       Y       PD     Counts"
DO
     INPUT #2, FLE$
     IF EOF(2) THEN END
     IF FLE$ = "" THEN END
     ID$ = LEFT$(RIGHT$(FLE$, 12), 8)
     T% = 0
     ON ERROR GOTO NOSUCHFILE
     OPEN FLE$ FOR INPUT AS #1
     ON ERROR GOTO 0
     IF T% = 1 THEN PRINT "Can't find file :"; FLE$: GOTO LOOP1
     CLOSE 1
     OPEN "R", #1, FLE$, 8
     FIELD #1, 2 AS A$, 2 AS B$, 2 AS C$, 2 AS D$
     LENFLE% = LOF(1) / 8
     PRINT #5, : PRINT #5, "      "; ID$
     PRINT : PRINT "      "; ID$
     SUM& = 0!
     FOR I% = 1 TO LENFLE%
     GET #1, I%
     A% = CVI(A$): B% = CVI(B$): C% = CVI(C$): D% = CVI(D$)
     MINUTE% = SUM& \ 1800
     DUM& = SUM& - (1800& * MINUTE%)
     SECOND% = DUM& \ 30
     SUM& = SUM& + D%
     IF A% = 0 AND B% = 0 AND C% <= 10 THEN
          IF D% > outmax% THEN
               PRINT #5, "      ";
               PRINT #5, ID$; "  OUT  ";
               PRINT #5, USING "###"; MINUTE%; SECOND%;
               PRINT #5, USING "#######."; I%; A%; B%; C%; D%
               PRINT "      ";
               PRINT ID$; "  OUT  ";
               PRINT USING "###"; MINUTE%; SECOND%;
               PRINT USING "#######."; I%; A%; B%; C%; D%
          END IF
     ELSE
          IF D% > inmax% THEN
               PRINT #5, "      ";
```

```
                PRINT #5, ID$; "   IN    ";
                PRINT #5, USING "###"; MINUTE%; SECOND%;
                PRINT #5, USING "#######."; I%; A%; B%; C%; D%
                PRINT "        ";
                PRINT ID$; "   IN    ";
                PRINT USING "###"; MINUTE%; SECOND%;
                PRINT USING "#######."; I%; A%; B%; C%; D%
            END IF
        END IF
        NEXT I%
CLOSE 1
LOOP1:
LOOP
END
NOSUCHFILE:
T% = 1
RESUME NEXT
```

COMMON UTILITY SUBROUTINES

FUNCTION LOG$ (SB$, A$)

```
'##########
'Purpose................\ Tags message with time & source for log
                                                          ---->ging.
'Parameters.............\ SB$ is usually the subroutine name.
'                        \ A$ is the message string.
'Other input data.......\ none
'Input files............\ none
'Output files...........\ none
'Other output data......\ The combined string, LOG$, with the sub
'                                                   ---->routine name,
                         \ date, time and message for logging.
'Function calls.........\ none
'Subroutine calls.......\ none
'Comments...............\ LOG$ beeps
'##########
LOG$ = SB$ + LEFT$(DATE$, 5) + " " + LEFT$(TIME$, 5) + ")" + A$
BEEP
'.....................................................................
'.....................................................................
END FUNCTION       'LOG  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

FUNCTION LOGS$ (SB$, A$)

```
'##########
'Purpose................\ Tags message with time & source for log
                                                          ---->ging.
'Parameters.............\ SB$ is usually the subroutine name.
'                        \ A$ is the message string.
'Other input data.......\ none
'Input files............\ none
'Output files...........\ none
'Other output data......\ The combined string, LOGS$, with the su
'                                                  ---->broutine name,
                         \ date, time and message for logging.
'Function calls.........\ none
'Subroutine calls.......\ none
'Comments...............\ Exactly the same as LOG$ but no beep.
'##########
LOGS$ = SB$ + DATE$ + " " + LEFT$(TIME$, 5) + ")" + A$  'modified
'                                                       ----> 1/5/93
'.....................................................................
'.....................................................................
END FUNCTION       'LOGS  'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

SUB GETXXA (FILENO%)

```
'##########
'Purpose................\ Reads a record from the appropriate fil
                                                      ---->e into XX
'Parameters.............\ FILENO%
'Other input data.......\
```

```
'Input files............\  FILEMRG$
'Output files...........\
'Other output data......\ XX
'Function calls.........\
'Subroutine calls.......\
'Comments...............\ This makes it easier to modify record f
                                                         ---->orm.

'##########
'Read the array XX from a record on the FILEMRG$ file.
SHARED XX AS FIXCOMB
INPUT #FILENO%, XX.TGTTYPEN, XX.TGTTYPEC, XX.FIXLNGTH, XX.PUPDIAM
---->, XX.TGTID, XX.DISTANCE, XX.FRAMENO, XX.TGTX, XX.TGTY, XX.FI
     ---->XX, XX.FIXY, XX.HEADING, XX.COUNTDOWN, XX.CONTFIX, XX.C
           ---->ROSSCHECK, XX.ZONE, XX.SPEED, XX.AIDON, XX.SPARE
'................................................................
'................................................................
END SUB       ' GETXXA      'DUMMYPAGE$    ?r?;PAGE;EXIT;
```

SUB GETXXB (FILENO%, NEOFMRG)

```
SUB GETXXB (FILENO%, NEOFMRG)
'##########
'Purpose................\ Reads a record from the appropriate fil
                                              ---->e into DTEMP1

'Parameters.............\ FILENO%, XX() AS FIXCOMB, N
'Other input data.......\
'Input files............\  FILEMRG$
'Output files...........\
'Other output data......\ DTEMP1
'Function calls.........\
'Subroutine calls.......\
'Comments...............\ This makes it easier to modify record f
                                                         ---->orm.

'##########
'Read the array DTEMP1 from a record on the FILEMRG$ file.
SHARED DTEMP1 AS FIXCOMB
INPUT #FILENO%, DTEMP1.TGTTYPEN, DTEMP1.TGTTYPEC, DTEMP1.FIXLNGTH
                                            ---->, DTEMP1.PUPDIAM

INPUT #FILENO%, DTEMP1.TGTID, DTEMP1.DISTANCE, DTEMP1.FRAMENO, DT
                                                   ---->EMP1.TGTX

INPUT #FILENO%, DTEMP1.TGTY, DTEMP1.FIXX, DTEMP1.FIXY, DTEMP1.HEA
                                                        ---->DING

INPUT #FILENO%, DTEMP1.COUNTDOWN, DTEMP1.CONTFIX, DTEMP1.CROSSCHE
                                              ---->CK, DTEMP1.ZONE

INPUT #FILENO%, DTEMP1.SPEED, DTEMP1.AIDON, DTEMP1.SPARE
IF EOF(FILENO%) THEN NEOFMRG = 1
'................................................................
'................................................................
END SUB       ' GETXXB
```

SUB YESORNO (A$, B$)

```
'##########
'Purpose................\
```

```
'Parameters.............\
'Other input data.......\
'Input files............\
'Output files...........\
'Other output data......\
'Function calls.........\
'Subroutine calls.......\
'Comments...............\
'##########
PRINT A$
DO
     INPUT " YES or NO ???", C$
     d$ = LEFT$(UCASE$(C$), 1)
LOOP UNTIL d$ = "Y" OR d$ = "N"
B$ = d$
'...........................................................................
'...........................................................................
END SUB   'YES OR NO 'DUMMYPAGE$   ?r?;PAGE;EXIT;
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
December 1993

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Techniques Used for the Analysis of Oculometer Eye-Scanning Data Obtained from an Air Traffic Control Display

5. FUNDING NUMBERS
C NAS1-19000
WU 505-64-13-01

6. AUTHOR(S)
Daniel J. Crawford, Daniel W. Burdette, and William R. Capron

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Engineering & Sciences Company
Langley Program Office
144 Research Drive, Hampton, VA 23666

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-191559

11. SUPPLEMENTARY NOTES
Langley Task Monitor: Leonard Credeur

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 04

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This report documents the methodology and techniques used to collect and analyze look-point position data from a real-time ATC display-format comparison experiment. That study compared the delivery precision and controller workload of three final approach spacing aid display formats. Using an oculometer, controller lookpoint position data were collected, associated with gaze objects (e.g., moving aircraft) on the ATC display, and analyzed to determine eye-scan behavior. The equipment involved and algorithms for saving, synchronizing with the ATC simulation output, and filtering the data are described. Target (gaze object) and cross-check scanning identification algorithms are also presented. Data tables are provided of total dwell times, average dwell times, and cross-check scans. Flow charts, block diagrams, file record descriptors, and source code are included. The techniques and data presented are intended to benefit researchers in other studies that incorporate non-stationary gaze objects and oculometer equipment.

14. SUBJECT TERMS
Eye-scan behavior
Oculometer
ATC display study

15. NUMBER OF PAGES
186

16. PRICE CODE
A09

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	