

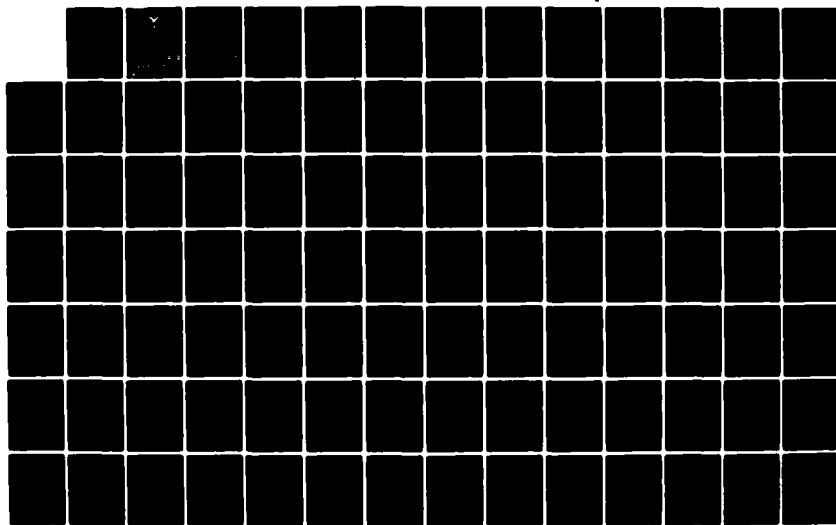
AD-A124 700 DEVELOPMENT OF AN OCULOMETER DATA COLLECTION SUBSYSTEM

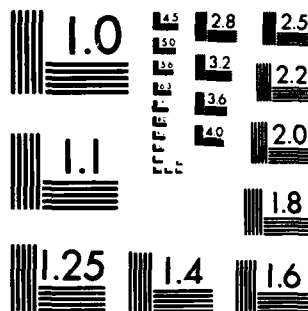
1/2

(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
SCHOOL OF ENGINEERING N L WOOD DEC 82

UNCLASSIFIED AFIT/GE/EE/82D-72

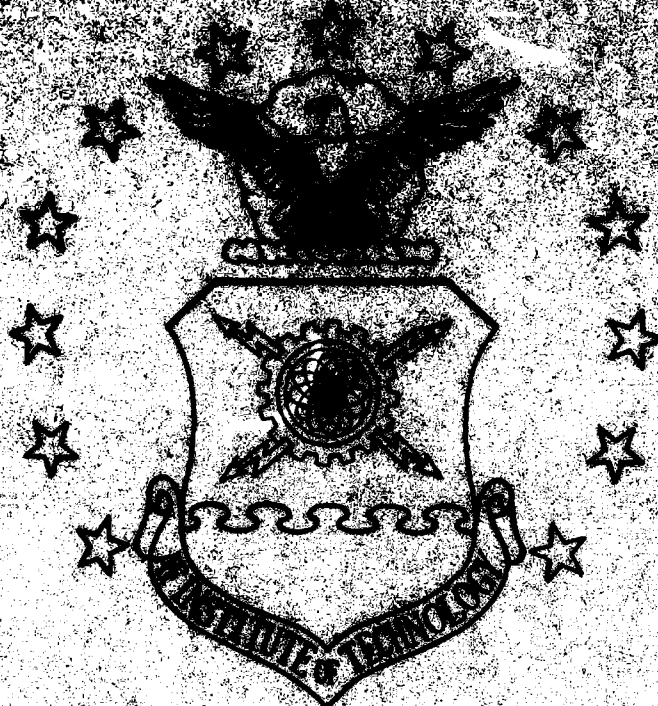
F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ADA 124700



THIS DOCUMENT HAS BEEN REPRODUCED
EXACTLY AS RECEIVED FROM THE
PERSON OR ORGANIZATION ORIGINATING IT

REPRODUCED BY THE AIR FORCE

DTIC
ELECTE
FEB 1983

S D

AFIT/GE/EE/82D-72

DEVELOPMENT OF AN OCULOMETER
DATA COLLECTION SUBSYSTEM

THESIS

AFIT/GE/EE Nancy L. Wood
 Capt USAF

OTIC
FEB 22 1983

Approved for public release; distribution unlimited

DEVELOPMENT OF AN OCULOMETER
DATA COLLECTION SUBSYSTEM

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Nancy L. Wood, B.S.
Capt USAF

Graduate Electrical Engineering

Dec 1982



Accession For	
NTIS CERI	☒
DTIC TAB	☐
Unannounced	☐
Justification	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special

A

Approved for public release: distribution unlimited

Preface

The Flight Dynamics Lab at Wright-Patterson AFB had procured an oculometer and two SYM-1's and various support equipment to collect data from the oculometer and present the reduced data in readable form. One SYM-1 system had been set up and initially tested but the oculometer was marginally operational. They needed someone to get acquainted with the SYM-1 and write the necessary software for the oculometer data collection subsystem. The project was ideal, because I wanted to do my thesis with the Flight Dynamics Lab and I also desired a software oriented thesis.

It was unknown from the start of the thesis whether the oculometer would be operating correctly or not. Because it was not operating correctly at thesis completion, the software was developed using simulated oculometer data. The data was as realistic as possible and the software written to be easily integrated into a system using the actual oculometer.

I would like to thank the Flight Dynamics Lab, specifically Mr. Bill Klotzback, for making the thesis possible and providing all the support and equipment I needed. Thanks also to Lt Jay Kirchoff and especially Lt Rick Benken, without whose encouragement, advice, and help the SYM-1 wouldn't have been so cooperative and those rough spots would have been a lot rougher.

My deepest thanks and gratitude go to my fiancée, Bruce Crowley, for listening to all my ravings when everything was going wrong and supported me through it all. Thank-you, Bruce.

Contents

Preface	ii
List of Figures	vi
List of Tables	vii
Abstract	viii
I. Introduction	1
Background	1
Problem	4
Scope	5
Assumptions	6
General Approach	6
Sequence of Presentation	7
II. Equipment	8
III. Development of the Program	10
General Development	10
Module: Main Subsystem	15
Module: Initialization	15
Module: Store Data	18
Module: Add Timing	19
Module: Determine Instrument Number	21
Module: Compare Data Samples	23
Module: Create Table	24
Module: Print Results	27
IV. Module Validation	29
Module: Main Subsystem	31
Module: Initialization	31
Module: Store Data	32
Module: Add Timing	32
Module: Determine Instrument Number	33
Module: Compare Data Samples	33
Module: Create Table	34
Module: Print Results	34
V. Subsystem Validation	36
VI. Subsystem Timing	38
VII. Users Guide	39

VIII. Recommendations	47
Bibliography	48
Appendix A: SYM-1 Memory Map	50
Appendix B: Oculometer Connector for Digital Data Output	51
Appendix C: Pascal-Like Description of Modules	52
Appendix D: Assembly Language Implementation of Modules	64
Appendix E: Module Validation Data	125
Appendix F: Simulated Oculometer Input for Subsystem Validation	132
Appendix G: Final Data Tables and Outputs	134

List of Figures

<u>Figure</u>		<u>Page</u>
1	Subsystem Data Flow Diagram	1
2	Structure Chart for Main Module	13
3	Structure Chart for Print Results Module . .	14
4	Instrument Boundaries	22
5	Simulated Instrument Boundaries for Subsystem Validation	37
6	Test Instrument Boundaries	127

List of Tables

<u>Table</u>		<u>Page</u>
1	Subsystem Variable Initialization	17
2	Final Data Table Row Organization	25
3	Final Data Table Column Organization	25
4	Module Worst Case Timing	38
5	Division Validation Data	125
6	Add Timing Validation Data	126
7	Determine Instrument Number Validation Data .	127
8	Compare Data Samples Validation Data	128
9	Create Table Validation Data	129
10	Create Table Validation Results	130

Abstract

A SYM-1 microprocessor with dual 5 1/4 inch disk drives was used to develop software to gather and reduce data from a Cubic-Foot Remote Oculometer built by Honeywell, Inc. The primary function of the oculometer is to measure the look direction of a pilot's eye in a ground cockpit simulator. The output of the oculometer used for this effort is eye lookpoint in azimuth and elevation and whether the oculometer is tracking the eye or not. The line-of-sight measurement covers a viewing field of plus and minus 30 degrees in azimuth and zero to plus 30 degrees in elevation. This viewing field is broken into instruments whose boundaries are defined by the data collection subsystem.

The following performance measures are printed out at the end of the data mission:

1. Total dwell time on each instrument.
2. Mean dwell time on each instrument.
3. Proportion of dwell time on each instrument.
4. Proportion of fixations on each instrument.
5. Transition probability from one instrument to another.
6. Number of fixes per minute for each instrument.

The software for the SYM-1 was developed modularly with each module tested separately and then the whole subsystem tested. Simulated oculometer data was used to test the

software. The data collection subsystem was designed to run with minimal knowledge and interaction required by the user.

I. Introduction

Background

This thesis involves the development of software to gather and reduce the output of a device called an oculometer. This is accomplished with a SYM-1 microprocessor system. The following paragraphs briefly describe the operation and uses of the oculometer, and the specific application addressed by the thesis.

An oculometer is an electro-optical device that tracks the eye line-of-sight in two axes without interfering with the subject. The subject, for Air Force applications, is a pilot. The oculometer can be used either in a ground simulator or in an actual aircraft in flight. Uses of the oculometer include two-dimensional tracking where the eye line-of-sight is used for fire control system control, the evaluation of different types of cockpit landing displays, quantifying pilot scan patterns during various phases of flight, and testing and evaluating different types of aircraft cockpit displays.

The oculometer used for this thesis was the "Cubic-Foot Remote Oculometer" built by Honeywell, Inc of Lexington Massachusetts (Ref 1).

The primary function of the oculometer is to measure the look direction of the subjects eye without interfering

with any of his activities. Information on pupil diameter and blink rate can also be obtained.

A sensing subsystem illuminates the subject's eye and collects information on the eye's orientation. A near-infrared (IR) illumination source and an IR-sensitive TV camera are used to obtain a video signal which contains the necessary information about the eye. The IR illumination causes no physical discomfort to the eye and does not cause the pupil diameter to decrease appreciably.

The measurement principle of the eye lookpoint is based on the fact that the angular direction of the eye is proportional to the position of the corneal reflection of the IR light source relative to the center of the pupil. The oculometer can locate the eye and track it throughout a one cubic foot motion box.

The line-of-sight measurement of the oculometer covers a viewing field of plus and minus 30 degrees in azimuth and zero to plus 30 degrees in elevation. The average error in eye tracking is approximately one degree. The oculometer must be calibrated for each subject to eliminate nonlinearities in the viewing field. The parameters for up to six pilots can be stored, so the subject can be tested again without full calibration of the oculometer.

The output of the oculometer is eye lookpoint in azimuth and elevation, pupil diameter, and whether the oculometer is tracking the eye or not. The oculometer loses track of the eye when the pilot moves his eye out of the

range of the tracking equipment. For example, the oculometer will lose track of the eye if the pilot looks behind him. This loss of track is called an out-of-track condition. If the eye image is lost, automatic search procedures are initiated to reacquire it (Ref 2,3).

Oculometer measurements are useful in testing integrated aircraft displays and, coupled with flight path and control input data, to detect subtle differences in piloting techniques during simulated flight (Ref 4). Another use for the oculometer is in the placement of cockpit instruments. Ground and/or flight data can be collected which indicates which instrument needs to be in the center of the pilot's field of view (i.e. the instrument most often looked at) and which instruments need to be in close proximity (i.e. two instruments which have a high number of transitions from one to the other) (Ref 5). The oculometer has been used to evaluate individual displays in the cockpit, and to evaluate changes in displays (Ref 6,7).

The Flight Dynamics Lab at Wright-Patterson Air Force Base, Ohio would like to use the oculometer during 12-18 minute ground data missions in flight simulators to gather data on the pilots lookpoint with respect to cockpit instruments, or segments of instruments. The data will be collected and analyzed by an oculometer data collection subsystem. This data will be used for cockpit design and analysis. Presently, the oculometer data is not being

gathered or reduced. The lab requires thesis completion before the oculometer output can be utilized.

Problem

The problem is to develop an oculometer data collection subsystem that will collect, reduce, print out, and store the data from a data mission.

The following performance measures need to be calculated and stored for each mission:

1. Total dwell time on each instrument or instrument section.
2. Mean dwell time on each instrument or instrument section.
3. Proportion of dwell time spent on each instrument or instrument section.
4. Proportion of fixations on each instrument or instrument section.
5. Transition probability from one instrument to another.
6. Number of fixes per minute for each instrument or instrument section.

The performance measures will be printed out between data missions.

Scope

The oculometer data collection subsystems is to be used in a ground environment in a cockpit simulator. The use of the subsystem in the airborne environment is not addressed.

The cockpit can be subdivided into rectangular or square subsections which can be instruments, segments of instruments, or any area within the viewing field of the oculometer. The subsections are defined by specifying two diagonal corners. The oculometer subsystem design allows for a maximum of 24 instrument subsections to be defined. For an additional seven subsections to be defined, there would be a 25 percent increase in memory space required and in program complexity. Another consideration is that the amount of time between oculometer inputs is 16.67 msec. The more instrument subsections defined, the longer is the time to determine the instrument number for each oculometer input.

The data runs are planned to be from 12-18 minutes long. The data runs are broken into time increments, with each time increment equalling one millisecond. One millisecond was chosen because it was small enough to allow accurate timing of the time between oculometer data samples, which is a minimum of 16.67 milliseconds. A data run of 18 minutes requires 1.08×10^5 time increments. This requires a minimum of three SYM-1 eight bit words, or bytes. Therefore, three bytes were used for the storage of timing information. These three bytes allow for data missions up

to 279 minutes long (3 bytes = 16,777,215 time increments = 279 minutes).

Assumptions

The assumptions are that the oculometer used will be the Honeywell Cubic Foot Oculometer with a digital X and Y direction and track/no track outputs, and that the data collection subsystem is to be used in a ground cockpit simulator.

General Approach

Two SYM-1 single board microprocessors were available for the data collection subsystem. The original concept was to have one SYM-1 gather the data from the oculometer, do some preliminary data reduction, and pass the data to the second SYM-1. This SYM-1 would complete the data reduction to produce the final tabular form. As the algorithms were developed, it became apparent that there was adequate time available between data inputs from the oculometer to do all data reduction. There was also enough random access memory (RAM) available on one SYM to store the data reduction programs and the final reduced data table. Therefore, one SYM-1 was used for the data collection subsystem.

The data collection subsystem algorithms were designed using the concepts of structured design. Structured design is a disciplined approach to algorithm design which

simplifies system design by partitioning the system into "black boxes" and organizes these boxes hierarchically. Structured design produces systems that are easy to understand, reliable, flexible, long-lasting, and efficient to operate (Ref 8).

Good design uses the concepts of partitioning and organizing the pieces of the system. Partitioning refers to the division of the problem into small subproblems. Highly interrelated parts of the problem should be in a single black box. A black box is a component with known inputs, known outputs, and generally, a known transform but the contents of the box don't have to be known (Ref 9). The modules were tested with simulated data, since the oculometer was not operational.

Sequence of Presentation

Chapter II describes the equipment used and the configuration of the equipment. The memory map for the SYM-1 is discussed. Chapter III describes how the subsystem was developed using Pascal-like statements, a data flow diagram for the overall program and a structure chart. Each program module is discussed. Chapter IV details the data used to test each module and Chapter V describes the testing of the overall data collection subsystems. Chapter VI describes subsystem timing. Chapter VII details the users guide for operation of the subsystem during ground data missions.

II. Equipment

The following equipment was used to develop the oculometer data collection subsystem:

One SYM-1 single board microprocessor

One Hudson Digital Electronics (HDE) 15 slot 19 inch
cage with power supply

Four HDE 8K RAM boards

One HDE 8K EPROM board

One parallel input/output board

One HDE dual 5-1/4 inch disk drive unit

One Hazeltine 1500 CRT terminal

Software:

Editor/assembler in ROM

Disk operating system

Editor on disk

This equipment was purchased for the subsystem before thesis start. Appendix A shows the memory map of the SYM-1.

The unassembled modules are loaded into the Resident Assembler/Editor from disk at memory locations 0200 (hex) to 14FF (hex). When assembled, the modules reside at memory locations 0000 (hex) to 002F hex), 1500 (hex) to 224C (hex) and 6000 (hex) to 7000 (hex). The final data table is from 6000 (hex) to 63FF (hex), and the simulated oculometer input data starts at 6490 (hex). The Resident Assembler Editor and the SYM-1 stack uses memory locations 0A00 (hex) to

0F00 (hex). The memory locations used from 0000 (hex) to 0024 (hex) are used for zero page addressing. The actual program is not stored here. The modules were placed back-to-back, with approximately 20 (hex) memory locations between them.

There is unused memory from 224D (hex) to 5FFF (hex), 7000 (hex) to 7FFF (hex), and 9000 (hex) to 9FFF (hex).

III. Development of the Program

General Development

This section describes the overall subsystem logic and each module. The program was developed by breaking it down into a series of modules, and developing each module into a subroutine. Therefore, the main program is merely a series of calls to subroutines.

The data flow diagram (DFD) is used to partition a system and is a major tool of structured analysis. A DFD is a network representation of a system showing the active components and the data interfaces (Ref 8). The DFD is a tool used to define the proposed logical system. It is useful in identifying the functions of a system and the resultant data transformations (Ref 9).

The basic elements of a DFD are called transforms and are represented by circles. A transform represents transformations of data from one form to another. The data elements themselves are labeled arrows going into the transforms (Ref 10).

Figure 1 shows a data flow diagram for the subsystem. Each module is represented by a transform with the data passed between modules represented by the data elements.

Structure charts are an integral part of structured design methodology. A structure chart shows the partitioning of a system into modules, the hierarchy and

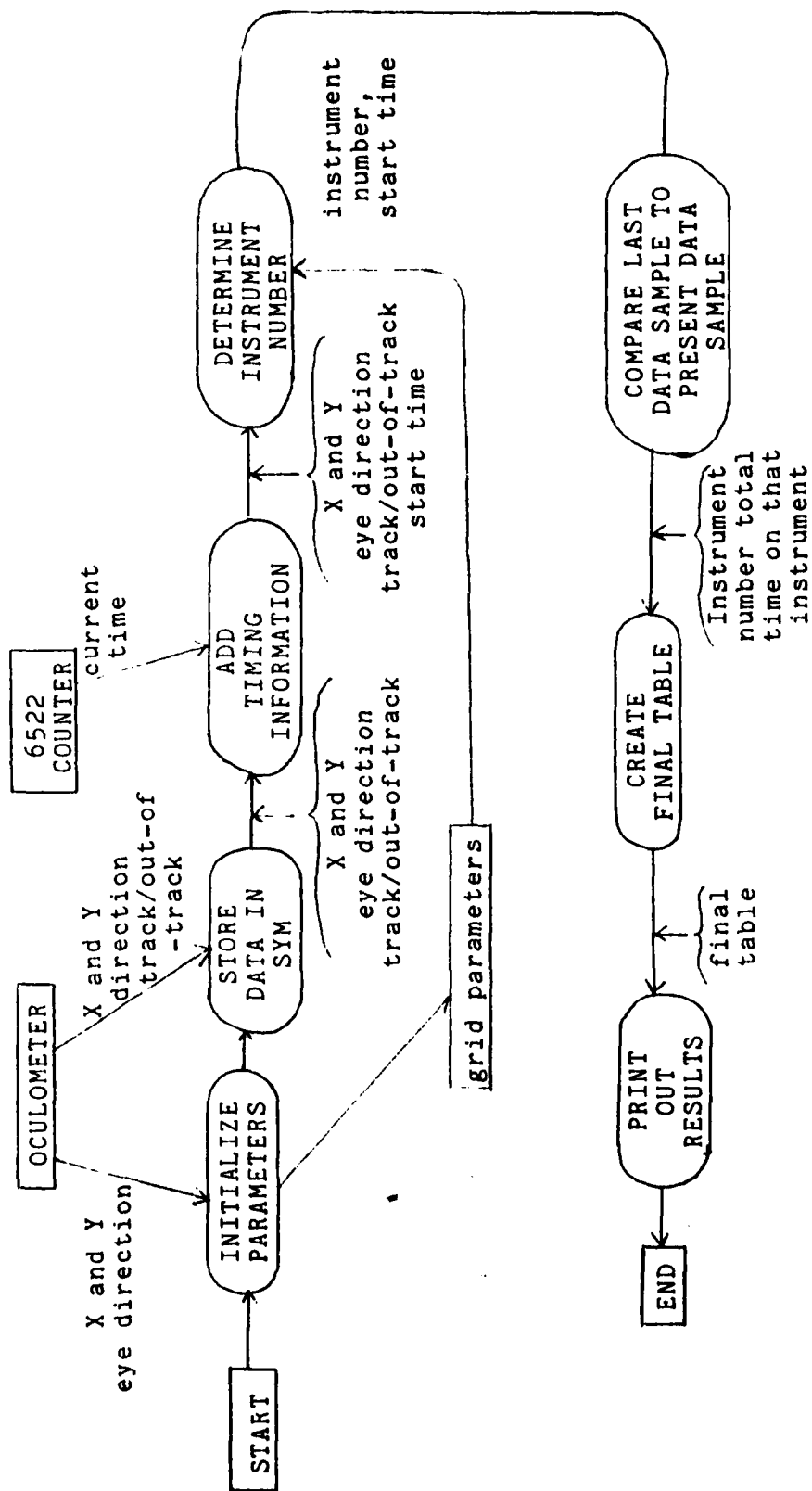


Fig 1. Subsystem Data Flow Diagram

organization of modules, the communication between modules, and the functions of the modules (Ref 8,11).

Each module is represented by a rectangle identified by a name. Modules are connected by arrows. A data element passed between two modules is shown by an arrow with an open circle, and control communication by an arrow with a solid circle. A major decision between two or more modules is designated by a diamond, and a major loop by a semicircle.

Figures 2 and 3 are the structure charts for the Main module and the Print Results module.

A data dictionary is used, among other things to establish a glossary of terms, to provide standard terminology and define all terms, to provide cross-reference capability, and to resolve problems associated with aliases and acronyms (Ref 11). The Initialization module defines the address for all variables used by two or more modules called by the Main module. The remaining variables are declared locally in the declaration section of each module. Therefore, the declaration section of each module serves as a data dictionary.

Each module is discussed, with a Pascal-like description in Appendix C and the assembly language implementation in Appendix D.

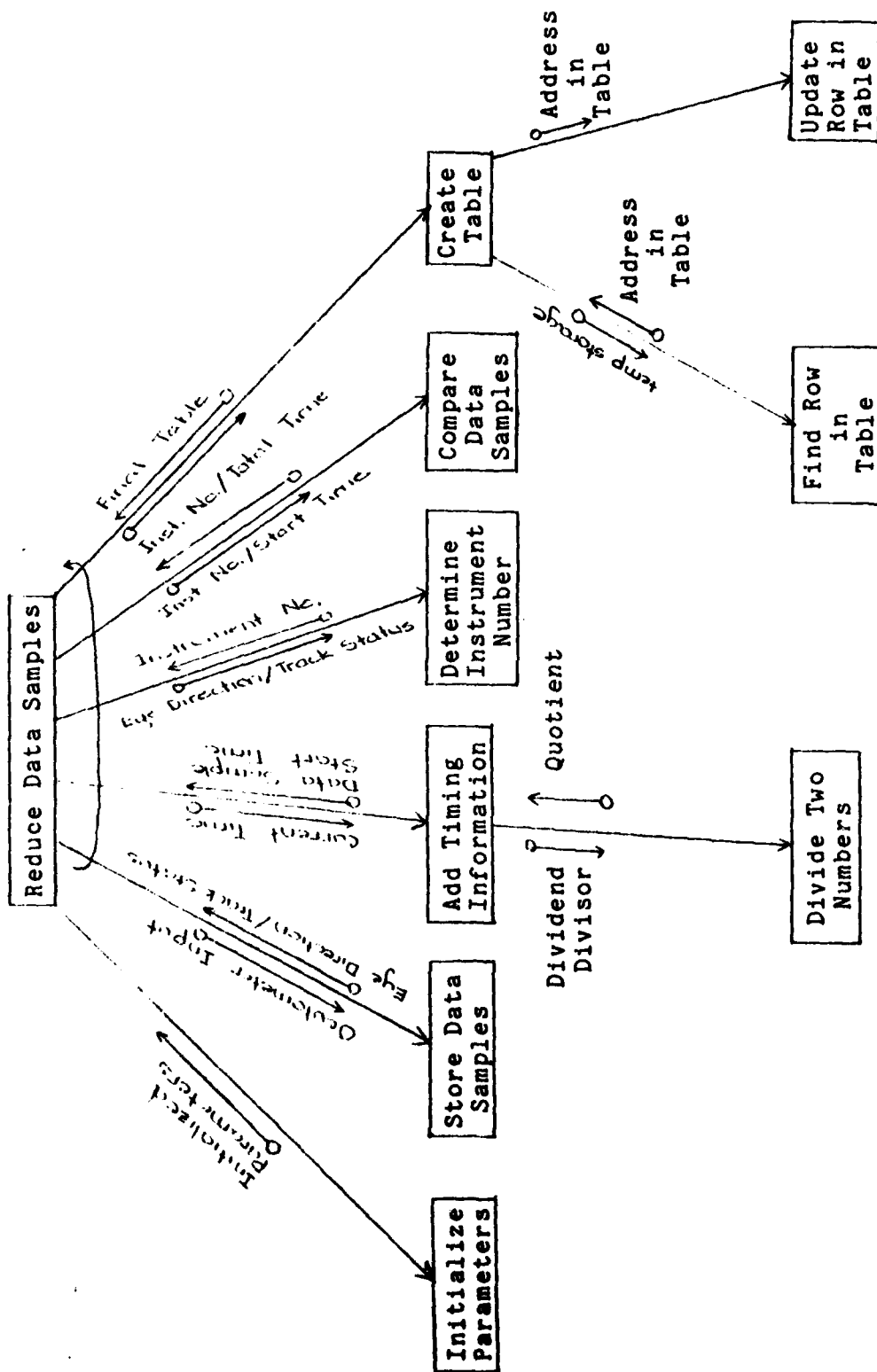


Fig 2. Structure Chart For Main Module

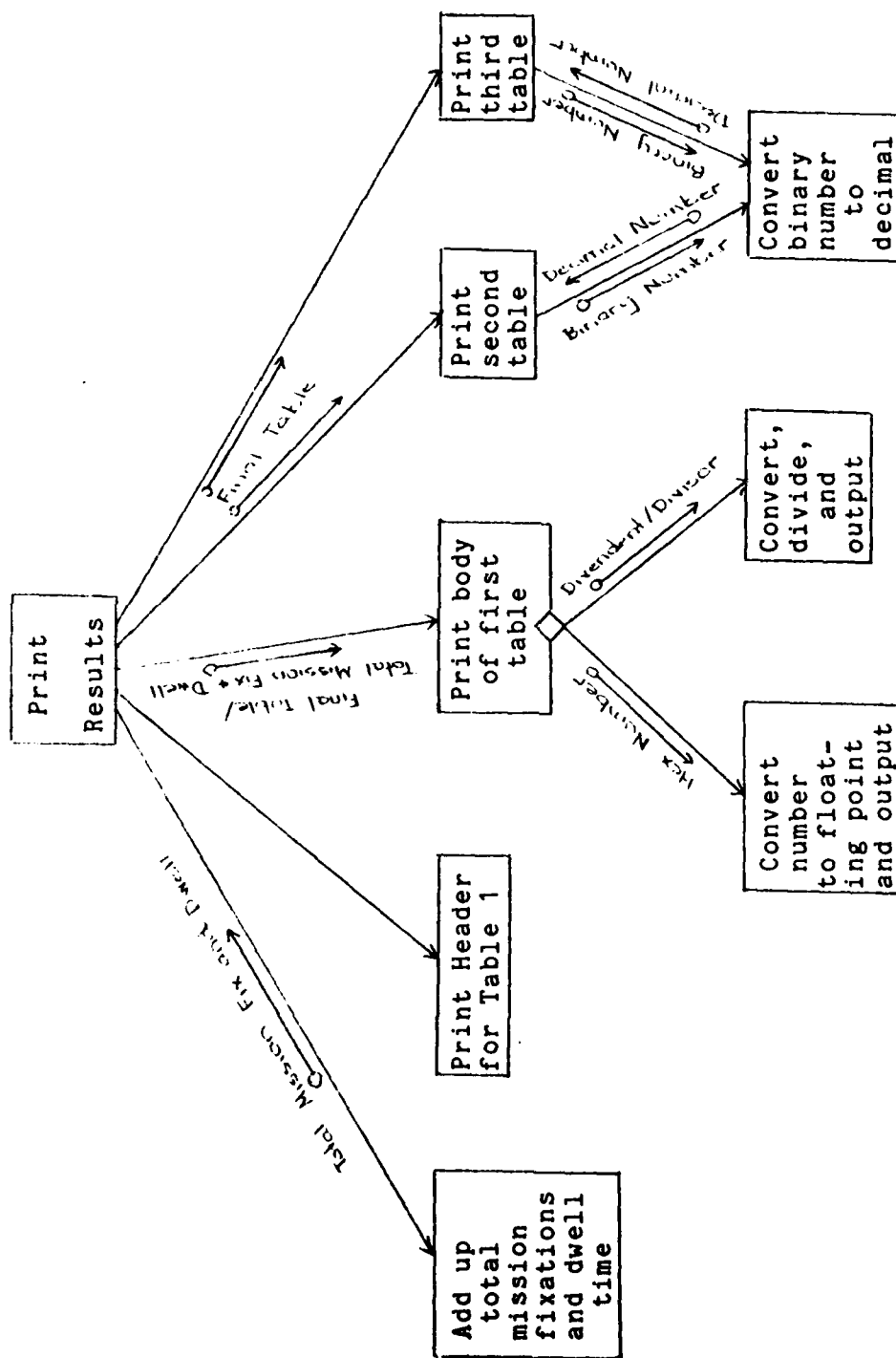


Fig 3. Structure Chart for Print Results Module

Main Subsystem Module

The Initiation module is called by the Main module once at the start of the data mission. Modules Store Data, Add Timing, Determine Instrument Number, Compare Data Samples, and Create Table are called by the Main module each time a data sample comes from the oculometer. Module Print Results is run separately when the data mission is complete. The Main module also has a timing loop to simulate the time between oculometer data samples. The timing loop is approximately 20 milliseconds long. At the end of the loop, the Store Data (Simulated) module is called, followed by Add Timing, Determine Instrument Number, Compare Data Samples and Create Table. Then the Main module checks to see if it is at the end of the array containing the simulated oculometer inputs (see module Store Data). If not, the timing loop is executed again and the process repeated.

Appendix C describes the module in Pascal-like statements. Appendix D lists the assembly language implementation of the module.

Module: Initialize Parameters

This module is called by the Main module before the actual data mission begins. It initializes parameters used in various modules and creates a "grid" of values that defines the boundaries of the instruments, or segments of instruments. The grid parameters were simulated because

actual oculometer data were not available. If the oculometer was available, the grid parameters would be obtained by having the subject look at the four corners of each instrument, and storing the parameters with the associated instrument number.

Table 1 shows each of the parameters, or program variables, that are initialized, the module(s) that uses the parameter, and the value the parameter is initialized to. GRID represents the grid parameters that were simulated.

The SY6522 timer is initialized by the module and starts counting the number of time increments that have passed from the beginning of the data mission. The SY6522 is initialized so it interrupts every 65 milliseconds, or every 65 time increments. At each interrupt, 65 is added to LOTIM, MITIM, and HITIM. These are three bytes that represent the number of time increments from the beginning of the data mission to the last interrupt. The interrupt routine is in the module.

The module also establishes the addresses for all variables used by two or more modules. Variables used by only one module are declared inside that module.

Appendix C describes the module in Pascal-like statements. Appendix D lists the assembly language implementation of the module.

Table I

Subsystem Variable Initialization

<u>Variable</u>	<u>Module(s)</u> <u>That Use Variable</u>	<u>Initialization</u> <u>Value</u>	<u>Description</u>
LSTRT1	Compare Data Samples	00	Start time of last data sample
LSTRT2			
LSTRT3			
LASTIN	Compare Data Samples	00	Last data sample instrument number
LOTIM	Add Timing Information	00	Number of time incre- ments since beginning of data run
MITIM			
HITIM			
TABLE1	Create Table	00	Initialize final data table to zero
TABLE2			
TABLE3			
TABLE4			
OLDIN	Create Table	00	Last instrument number through create table

Module: Store Data

This module is called by the Main module every time a data sample comes from the oculometer. Because the oculometer data is simulated, the Main module calls the Store Data module using a timing loop.

The oculometer output is X eye direction (10 bits), Y eye direction (10 bits), and track/out-of-track status (1 bit). This output is called the current data sample.

When the oculometer is used, the X and Y eye direction is obtained from a connector on the back of the oculometer. Appendix B is the pin connections for the connector. The outputs for pupil diameter are not used.

The oculometer is considered to be "in track" if it is tracking the eye. If the oculometer loses track of the light reflection off the back of the eye, either due to the pilot blinking or moving his eyes outside the tracking area of the oculometer, the oculometer is out-of-track. An out-of-track condition is a logic 0 on the track/out-of-track status line. When the oculometer is out-of-track, the X and Y eye direction remains the same as the last output before the oculometer went out-of-track.

The X direction is stored as 2 bytes, Y direction as 2 bytes, and track/out-of-track status as 1 byte. The oculometer represents negative numbers in two's complement form. Therefore, if the 10th bit from the oculometer is a

zero, the number is positive, and if the 10th bit is one, the number is negative.

Two Store Data modules were developed. Store Data (Oculometer) was designed to be used with the actual oculometer. This module stores X and Y eye direction in 4 bytes by filling in bits 11 through 16 of the high byte of each with either one's or zero's, depending on whether the number is positive or negative. It also stores track/out-of-track status as a one or a zero. These 5 bytes of information are passed to the next module, Add Timing.

The second module is Store Data (Simulated). This module reads simulated X and Y eye direction data and track/out-of-track status from memory and stores them in the same 5 bytes as the Store Data (Oculometer) module.

Appendix C describes both modules in Pascal-like statements. Appendix D lists the assembly language implementation of Store Data (Simulated).

Module: Add Timing

This module adds the necessary timing information to the X and Y eye direction and track/out-of-track status from the Store Data module. The timing information which is added is three bytes that represent the number of time intervals that have elapsed from the beginning of the data mission to the time of the data sample. One time interval is equal to 1 millisecond.

A SY6522, Versatile Interface Adapter, is used for the timing of the data mission. The device is programmed to interrupt the SYM every 65 milliseconds. The SY6522 is initialized in the Initialization module and started at the beginning of the data mission. Every time the SY6522 interrupts, 65 is added to three bytes that represent the number of time intervals that have elapsed since the start of the data mission. These three bytes are named LOTIM, MITIM, and HITIM.

When Add Timing is called, the current 2-byte value of the SY6522 counter is read. This counter counts down from 65000 to zero, interrupts at zero, and automatically reloads 65000 back into the counter and repeats the process. Each count down takes 1 microsecond, therefore the counter will interrupt every 65 milliseconds. Add Timing reads the 2-byte counter, divides the counter value by 1000, subtracts from 65 to get the number of milliseconds that have passed since the last SY6522 interrupt, adds this value to the three bytes LOTIM, MITIM, and HITIM, and stores the resultant 3 bytes with the X and Y direction information of the current data sample.

Therefore, at the end of the Add Timing module, the current data sample consists of X and Y eye direction, track/out-of-track status, and three bytes that represent the number of time intervals from the start of the mission to the current data sample.

Appendix C describes the module in Pascal-like statements. Appendix D lists the assembly language implementation of the module.

Module: Determine Instrument Number

This module takes the X and Y eye direction and track/out-of-track status of the current data sample and defines an instrument number for the sample. The timing information of the data sample remains unchanged.

A grid is created in the Initiation module that defines the boundaries for the cockpit instruments, or segments of instruments. Each instrument is defined by the coordinates of the four corners of the instrument. The grid has a format of upper X, lower X, upper Y, lower Y, and instrument number for each instrument (Fig 2).

The coordinates are 2 bytes each. Each instrument's boundaries are defined in the grid. The order in which the instrument boundaries are stored in the grid is not important. If both the upper and lower values of X or Y are negative, the "lower" value is defined as the lower of the absolute values. For example, if the X coordinates of the instrument are -5 and -7, -5 is the upper X value and -7 is the lower X value. An instrument boundary cannot cross the Y axis. If an instrument does, the instrument must be broken into two sections.

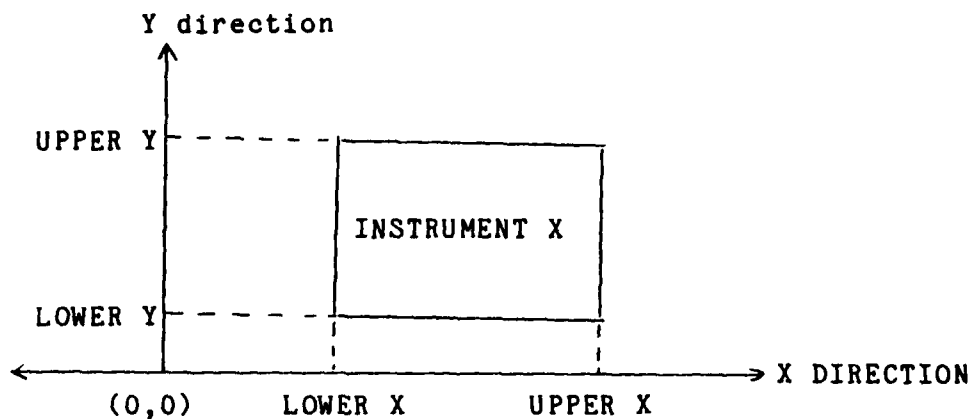


Fig 4. Instrument Boundaries

The instrument boundary coordinates are read and compared to the X and Y eye direction values to see if they are within that instrument. If they are, that instrument number is assigned to the current data sample. If not, the next instrument boundary coordinates in the grid are compared to the X and Y eye direction value. If the X and Y eye direction values are not within any of the defined instruments, the instrument number defined as no instrument is assigned to the present data sample.

If the present data sample is out-of-track, no instrument boundary comparisons are made and the instrument number for the present data sample is zero.

At the completion of the Determine Instrument Number module, the present data sample consists of an instrument

number (1 byte) and the timing information that represents the number of time intervals since the beginning of the data mission (3 bytes).

Appendix C describes the module in Pascal-like statements. Appendix D lists the assembly language implementation of the module.

Module: Compare Data Samples

This module compares the last data sample's instrument number to the present data sample's instrument number.

If the instrument numbers are the same, the current data sample is disregarded because the pilot is still on the same instrument.

If the two instrument numbers are different, the start time (3 bytes) of the last data sample is subtracted from the start time of the current data sample (3 bytes). The result is the total number of time increments spent on the instrument of the last data sample (3 bytes). This total number of time increments and the associated instrument number is sent to the Create Table module. Then the current data sample becomes the last data sample in preparation for the next data sample to go through the module.

Appendix C describes the module in Pascal-like statements. Appendix D lists the assembly language implementation of the module.

Module: Create Table

This module takes each data sample and places the information in a final data table. The data sample consists of an instrument number (INSTNM) and the number of time increments spent on that instrument (TOTIM1, TOTIM2, TOTIM3).

The purpose of the table is to reduce the data to a form that enables the calculation of the performance parameters.

The table is divided into four sections: TABLE1, TABLE2, TABLE3, and TABLE4. Each table contains seven rows. Each row contains information for an instrument, blink, glitch, or data gone. A "glitch" is any data sample where the number of time increments is less than or equal to 50. A "blink" is a data sample with an instrument number of zero (an out-of-track condition) and the number of time increments between 50 and 183. "Data gone" is a data sample with an instrument number of zero and the total number of time increments greater than 183.

Table 2 shows the organization of the information in the data tables. Table 3 shows the format of each of the rows. A '\$' in front of a number means the number is in base 16. A "no instrument" condition is defined as instrument number 25. Therefore, 24 instruments can be defined. The total number of fixations is the number of times the instrument was looked at. The accumulated time is the total amount of time spent on that instrument.

Table II

Final Data Table Row Organization

<u>Starting Memory Location</u>	<u>Contents</u>	<u>Starting Memory Location</u>	<u>Contents</u>
TABLE1 + 00	Glitch	TABLE3 + 00	Inst 12
TABLE1 + \$22	Blink	TABLE3 + \$22	Inst 13
TABLE1 + \$44	Data Gone	TABLE3 + \$44	Inst 14
TABLE1 + \$66	Inst 1	TABLE3 + \$66	Inst 15
TABLE1 + \$88	Inst 2	TABLE3 + \$88	Inst 16
TABLE1 + \$AA	Inst 3	TABLE3 + \$AA	Inst 17
TABLE1 + \$CC	Inst 4	TABLE3 + \$CC	Inst 18
TABLE2 + 00	Inst 5	TABLE4 + 00	Inst 19
TABLE2 + \$22	Inst 6	TABLE4 + \$22	Inst 20
TABLE2 + \$44	Inst 7	TABLE4 + \$44	Inst 21
TABLE2 + \$66	Inst 8	TABLE5 + \$66	Inst 22
TABLE2 + \$88	Inst 9	TABLE4 + \$88	Inst 23
TABLE2 + \$AA	Inst 10	TABLE4 + \$AA	Inst 24
TABLE2 + \$CC	Inst 11	TABLE4 + \$CC	Inst 25

Table III

Final Data Table Column Organization

<u>Byte Number</u>	<u>Contents</u>
0	data gone
1 - 25	instruments 1 through 25
26, 27	unused
28	low byte of accumulated time
29	middle byte of accumulated time
30	high byte of accumulated time
31	low byte of total number of fixations
32	high byte of total number of fixations
33	unused

Appendix C describes the module in Pascal-like statements. The module first determines if the current data sample is a glitch. If it is, the glitch row is updated. If it is not a glitch, it next looks at the instrument number. If the instrument number is a zero, either the blink or data gone row is updated. If it was data gone, the instrument number of the current data sample becomes the "last instrument number" (OLDIN). OLDIN was the last instrument looked at. This is used so the probability of transitioning from one instrument to another can be calculated in the final module. If the instrument number was not zero (and it is not a glitch), the appropriate row for that instrument is updated and the instrument number of the current data sample becomes the last instrument number.

Updating a row consists of incrementing the number of fixations, incrementing the column that corresponds to the current last instrument, and adding the number of time increments of the current data sample to the accumulated time.

Appendix D lists the assembly language implementation of the module.

Module: Print Results

This module prints out the performance measures using the data in the Final Data Table. In the following paragraphs, each of the performance measures are discussed.

1. Total dwell time on each instrument. The three byte accumulated time (byte numbers 28, 29 and 30, see Table 3) is printed out for instruments 1 through 25, data gone, blink and glitch.

2. Mean dwell time on each instrument. The accumulated time for an instrument is divided by the total number of fixations for that instrument. This value is the mean dwell time for that instrument. The mean dwell time is printed out for instruments 1 through 25, data gone, blink and glitch.

3. Proportion of dwell time on each instrument. The accumulated times for instruments 1 through 25 is added up and called total time (TOTIM1, 2 and 3). The accumulated time for an instrument is divided by total time to get the proportion of dwell time on an instrument.

4. Proportion of fixations on each instrument. The total number of fixations for instruments 1 through 25 is added up and called total fixations (TOTFIX1 and 2). The total number of fixations for an instrument is divided by

total fixations to get the proportion of fixations on an instrument.

5. Transitions from one instrument to another. For each instrument 1 through 25, data gone, glitch and blink, the number in the column for each instrument (columns 1 through 25, see Table 3) is printed.

6. Number of fixes per minute for each instrument. The total number of fixations for each instrument is divided by the total number of minutes in the mission.

Floating point routines are used for division and decimal output operations in performance measures 1,2,3,4 and 6. The routines used were derived from floating points routines found in Reference 12. Performance measure 5 requires the data in the table to be printed out in decimal format. The routine to convert from binary to decimal was found in Reference 13.

A description of the module in Pascal-like statements is in Appendix C with the assembly language implementation in Appendix E. Examples of the final printout are in Section V.

IV. Module Validation

The validation of each individual module is discussed in this section. A few of the modules were tested together, but most of the modules were developed and validated independently. The validation of the total subsystem is discussed in Section V.

Testing is an important part of any software development program. Testing is done with the intent of finding errors. There are two major categories of testing: black-box testing and white-box testing. In black-box testing, the tester views the program as a black box and is unconcerned with the internal structure of the program. It is also known as input/output driven testing because the tests are derived solely from the specifications. White-box testing uses knowledge of the internal structure of the program. The test cases examine the program's logic.

Because it is impossible to test every possible input case, a test of any program must be necessarily incomplete. Therefore, the test cases should try and detect the most errors possible. There are various methodologies used to intelligently select test cases. The methodologistics that were used in the thesis are described briefly below.

White-box testing is also called logic coverage testing. The logic coverage testing used is called decision/condition coverage. This type of testing ensures that:

1. every statement is executed at least once
2. each decision has a "true" and a "false" outcome at least once
3. each condition in a decision takes on all possible outcomes at least once.

Three types of black box testing were used: equivalence partitioning, boundary-value analysis and error guessing.

In equivalence partitioning, a subset of all inputs is chosen that has the highest probability of finding all errors. This is accomplished by picking test cases that reduce, by more than one, the number of test cases that must be developed, and that cover a large set of other possible test cases.

Boundary-value analysis examines input and output boundary conditions. The boundaries are near the edges of the input and output equivalence classes.

Error guessing is largely an intuitive process. Test cases are designed using intuition and experience to try and expose certain probable types of errors (Ref 14).

Decision/condition coverage was used to test all the modules. Other tests used are stated in the test descriptions of each module.

Module: Main

The Main module was tested with the Initialization and Store Data modules. The Main module prints out a prompt to start the data mission, calls all other modules (except Print Results) and executes a timing loop to simulate the time between data samples. Therefore, the validation of this module consisted of the following:

1. The proper response to the prompt is the letter 's'. Other letters were input to determine if only an 's' would start the data mission (boundary value testing and equivalence partitioning).
2. The timing loop was tested during the Timing module. The output of the Timing module was examined to determine if the timing loop was operating correctly.
3. The subroutine calls to each subroutine were added to Main one at a time. As each module was added, the output of that module was examined to determine if the module was operating correctly. The modules were added in the order they are called.

Module: Initialization

The Initialization module establishes addresses for variables used by two or more modules, initializes the SY6522 counter and initializes various variables to zero.

Equivalence partitioning and error guessing techniques were used.

For all variables initialized, the memory locations were examined to ensure their contents were correct.

Correct operation of the SY6522 Counter was examined in the Add Timing module.

Module: Store Data

This module merely stores the X and Y eye direction and track/out-of-track status. For the oculometer simulation, simulated data was stored at specified memory locations and read by the Store Data module. Equivalence partitioning was used. Verification of this module consisted of examining the memory locations for X and Y eye direction and track/out-of-track status to verify that the simulated data was being read correctly.

Module: Add Timing

The Add Timing module uses a division subroutine that divides a two-byte dividend by a two-byte divisor (which was 1000 for this application) to get a two-byte quotient and a two-byte remainder. Only the quotient is used by Add Timing. The divisor is 1000 because the timing information from the SY6522 is in microseconds, and the time increment desired is milliseconds.

The test cases for the division subroutine are shown in Appendix E. These cases use the equivalence partitioning technique. The division subroutine was developed and tested separately and then used in Add Timing.

Appendix E also shows the test data from Add Timing. An 18 msec loop was executed and at the end of the loop, the number of time increments from the beginning of the data mission was determined and stored and the timing loop was executed again. The module also tested the SY6522 counter interrupt routine. The SY6522 was started at the beginning of the module and kept track of the elapsed time intervals. These test cases were acquired from equivalence partitioning and error guessing techniques.

Module: Determine Instrument Number

This module determines an instrument number for each data sample from the X and Y eye direction data and the track/out-of-track status. Data samples were simulated and instrument numbers determined. Boundary value and equivalence partitioning tests were conducted. Appendix E contains the test data samples examined and the test instrument boundaries used.

Module: Compare Data Samples

This module compares the present data sample to the last data sample. If the two instrument numbers are the

same, nothing is done. If they are different, the total time on the last instrument is determined and this time and the last instrument number is sent to the Create Table module.

The module was tested by reading in simulated data samples consisting of an instrument number and the time of the data sample, and outputting instrument numbers and the total time on each instrument. Large and small time values were examined (boundary value testing) and random values of time were tested (equivalence partitioning). The test data is shown in Appendix E.

Module: Create Table

The Create Table module takes the information from the Compare Data Sample module and places the information in a table. The table is from address 6000 through 63FF in the SYM memory. Appendix E shows the data used to test the module. The simulated input data consisted of an instrument number and the time on that instrument. The output was the data in the table. The format of the table was described in Section III. Error guessing and equivalence partitioning were used.

Module: Print Results

This module takes the data from the final table and prints the results out in three tables. The information

printed out is described in Section III. Examples of the final data table are in Appendix G.

The module was tested by using the random numbers in memory locations 6000 (hex) to 6400 (hex). This is the location of the final data table. The values computed and printed were compared to the contents of the memory locations to verify correct module operation. This is equivalence partitioning testing.

V. Subsystem Validation

The subsystem was validated using simulated oculometer input data and simulated parameters for the instrument boundaries. Figure 5 shows the simulated instrument boundaries used. The testing methodology used was equivalence partitioning. All instruments and out-of-track conditions were simulated. Various time increments on the instruments were examined. The final data table was examined for validity, and the three output tables were compared to the final data table. Appendix G is the actual SYM-1 output of the final data tables and the corresponding printed output.

One value in the three tables is incorrect. This is the number of fixations for glitch in Table 1. There is an error in the floating point routines. This error remained at thesis completion.

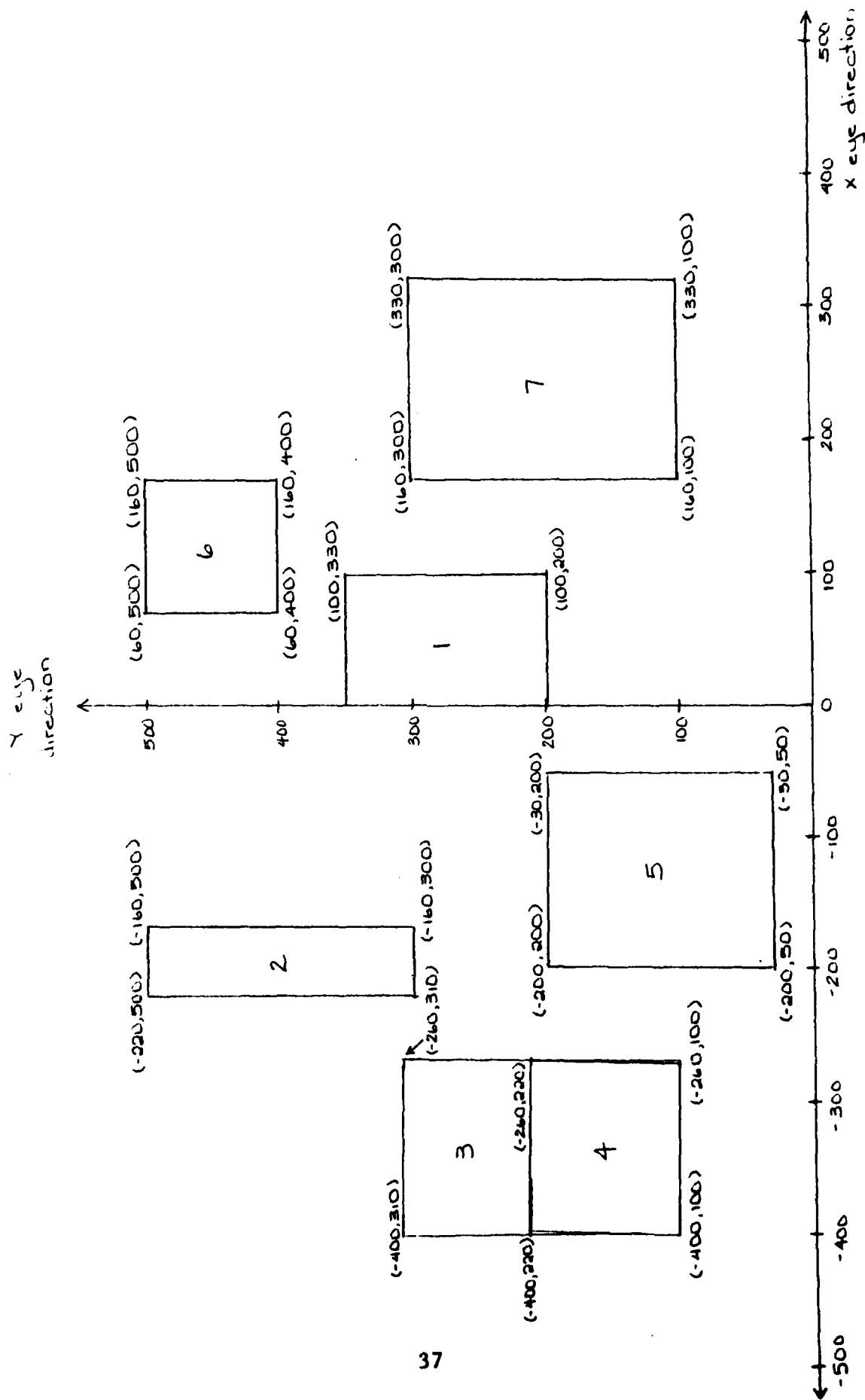


Fig 5. Simulated Instrument Boundaries for Subsystem Validation

VI. Subsystem Timing

The timing of the five modules called by the Main module each time a data sample is input from the SYM is critical. The five modules are Store Data, Add Timing, Determine Instrument Number, Compare Data Samples, and Create Table. These five modules have to be completed in the amount of time between oculometer data sample. The minimum amount of time between data samples is 16.67 milliseconds.

Each module was examined for worst case timing. The worst case timing for each is shown in Table 5.

Table 4
Module Worst Case Timing

<u>Module</u>	<u>Number of Machine Cycles</u>
Store	61
Interrupt	51
Add Timing	2561
Determine Instrument Number	7195
Compare Data Samples	135
Create Table	437
Main	49

The total number of machine cycles is 10489, or 10.489 msec. Therefore, there is more than adequate time between data samples for all six modules.

VII. User's Guide

The following is a guide to using the oculometer data collection subsystem. Items underlined are responses by the SYM-1. Items capitalized are those input by the user. <cr> is a carriage return.

1. Power the SYM-1

When power is applied, the SYM-1 responds with a "beep"

2. Load disks

Put the diskette marked "System Disk" into the right hand disk drive and the diskette marked "oculometer subsystem" into the left hand disk drive

3. Assemble subsystem modules

Hit "Q" on the keyboard. The SYM-1 will respond with a period. Then key in the following sequence of events:

└ G 5000 (Note: the space between "G" and "5000" is not
keyed in by the user. It is automatically
inserted by the SYM-1)

RAE V1.0

FODS COPYRIGHT (a) 1980 HDE INC

0200-0BFC 0C00-0EFC 0F00

0200 0C00

└ SET \$0200 \$1100 \$1101 \$1300 <cr>

0200-1100 1101-1300 0F00

0200 0C00

≥LOAD INIT <cr>

≥AS <cr>

//0000,3028,3028

≥ CL <cr>

≥ LOAD STORE <cr>

≥ AS <cr>

//0000,1A94,1A94

≥ CL <cr>

≥ LOAD TIMDIV <cr>

≥ AS <cr>

//0000,1B44,1B44

≥ CL <cr>

≥LOAD TIMING <cr>

≥ AS <cr>

//0000,1B8E,1B8E

≥ CL <cr>

≥ LOAD INSTNM <cr>

≥ AS <cr>

//0000,1C63,1C63

≥ CL <cr>

≥ LOAD COMPAR <cr>

≥ AS <cr>

//0000,1CDD,1CDD

≥ CL <cr>

≥ LOAD UPDAT <cr>

≥ AS <cr>

//0000,2115,2115

≥ CL <cr>

≥ LOAD INCR <cr>

≥ AS <cr>

//0000,2149,2149

≥ CL <cr>

≥ LOAD MASTER <cr>

≥ AS <cr>

//0000,224B,224B

≥ CL <cr>

≥ LOAD MAIN <cr>

≥ AS <cr>

//0000,647A,647A

≥

4. Start data mission

≥ RUN MAIN <cr>

HIT S to START MISSION = S

Any other character than "S" will not start the simulated data mission and another prompt will be printed out. No other user intervention is required until the program is completed and a ">" is printed.

The program using the actual oculometer instead of simulated data will require a set-up program before the data mission starts. This set-up program will establish the instrument boundaries. The boundaries were hand entered for the simulated data runs. The data mission using the oculometer would be started the same way as the simulated data mission (by running the Main module). The actual data mission would end by hitting reset (RST) on the SYM-1 on-board keyboard. The remainder of the User's Guide is the

same for both the simulated and actual data missions.

5. Print out results.

≥ SET \$0200 \$1300 \$1301 \$1500 <cr>

0200-1300 1301-1500 0F00

0200 0C00

≥ CL <cr>

≥ LOAD THREE <cr>

≥ AS <cr>

//0000,2028,2028

≥ CL <cr>

≥ LOAD SEVEN <cr>

≥ AS <cr>

//0000,20BB,20BB

≥ CL <cr>

≥ LOAD FPMUL <cr>

≥ AS <cr>

//0000,21CF,21CF

≥ CL <cr>

≥ LOAD CONVE <cr>

≥ AS <cr>

// 0000,223F,223F

≥ CL <cr>

≥ LOAD FPDIV <cr>

≥ AS <cr>

//0000,2324,2324

≥ CL <cr>

≥ LOAD FPOUT <cr>

≥ AS <cr>

//0000,241F,241F

≥ CL <cr>

≥ LOAD PRINT <cr>

≥ AS <cr>

//0000,2487,2487

≥ CL <cr>

≥ LOAD TOVAL <cr>

≥ AS <cr>

//000,283F,283F

≥ CL <cr>

≥ LOAD OUTPT <cr>

≥ AS

//0000,2A1E,2A1E

≥ CL <cr>

>LOAD OUTTB <cr>

≥ AS <cr>

//0000,2BB0,2BB0

≥ CL <cr>

≥ LOAD RESTB <cr>

≥ AS <cr>

//0000,2D4F,2D4F

≥ CL <cr>

≥ LOAD OUTTC <cr>

≥ AS <cr>

//0000,3730,3730

≥ CL <cr>

≥ LOAD OUTTD <cr>

≥ AS <cr>

//0000,3929,3929

≥ CL <cr>

≥ LOAD BINDEC <cr>

≥ AS <cr>

//0000,39D7,39D7

≥ CL <cr>

≥ LOAD FINAL <cr>

≥ AS <cr>

//0000,3A01,3A01

≥ RUN FINAL <cr>

All three final tables will print out.

VIII. Recommendations

The oculometer data collection subsystem needs further development to be used with an oculometer. The following additional software is required:

1. Interface with the oculometer. The data from the oculometer needs to be accessed and stored in the SYM-1.
2. Establishment of instrument boundaries. Before the data mission is started, the pilot will look at two diagonal corners of each desired instrument. The X and Y eye direction values of the two corners are used to establish and store the instrument boundaries.
3. Generation of oculometer interrupt. Every time the oculometer generates a new data sample, the SYM-1 needs to be interrupted so the data can be stored by the SYM-1 and used.
4. Correction of the floating point error in Table 1.

Bibliography

1. Merchant, J. and Morrisette, R. A Remote Oculometer Permitting Head Movement. Aerospace Medical Research Laboratory, Aerospace Medical Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, November 1973 (AD 776075).
2. Middleton, David B., George J. Hurt, Marlon A. Wise, and James D. Holt. Description and Flight Tests of an Oculometer. Langley Research Center, Hampton, Virginia, June 1977 (NASA TN D-8419).
3. Lambert, Robert H., Richard A. Monty, and Robert J. Hall. High-Speed Data Processing and Unobtrusive Monitoring of Eye Movements. US Army Human Engineering Laboratory, Aberdeen Proving Ground, Maryland, February 1975 (AD A006162).
4. Waller, Marvin C. Application of Pilot Scanning Behavior to Integrated Display Research. NASA Langley Research Center, Hampton, Virginia, August 1977.
5. Spady, Amos A. Airline Pilot Scanning Behavior During Approaches and Landing in a Boeing 737 Simulator. NASA Langley Research Center, Hampton, Virginia, October 1977.

6. Harris, R.L., M.C. Waller and S. Salmirs. Runway Texturing Requirements for a Head-Down Cathode Ray Tube Approach and Landing Display. NASA Langley Research Center, Hampton, Virginia, September 1978.
7. Waller, M.C., R.L. Harris, and S. Salmirs. An Evaluation of Some Display Parameters for an Advanced Landing Display. NASA Langley Research Center, Hampton, Virginia, August 1979.
8. Page-Jone, Meilir. The Practical Guide To Structured Systems Design. New York: Yourdon Press, 1980.
9. Yourdon, Edward and Larry L. Constantine. Structured Design. New York: Yourdon Press, 1978.
10. Weinberg, Victor. Structured Analysis. New York: Yourdon Press, 1980.
11. Peters, Lawrence J. Software Design: Methods and Techniques. New York: Yourdon Press, 1981.
12. Findley, Robert. 6502 Software Gourmet Guide and Cookbook. Connecticut: Scelbi Computer Consulting, Inc, 1979.
13. Leventhal, Lance A., Winthrop Saville. 6502 Assembly Language Subroutines. California: OSBORNE/McGraw-Hill, 1982.
14. Myers, Glenford J. The Art of Software Testing. New York: John Wiley & Sons, 1979.

APPENDIX A
SYM-1 Memory Map

Memory
Location

FFFF	_____	
-		SYM-1 Operating System Locations
	FF80	
EFFF	_____	
-		Resident Assembler/Editor
DFFF	_____	
-		
CFFF	_____	
-		
BFFF	_____	
-		Resident Assembler/Editor
AFFF	_____	
-		Disk controller
9FFF	_____	
-		4K RAM
8FFF	_____	
-		SUPERMON (SYN-1 Operating System)
7FFF	_____	
-		
6FFF	_____	8K RAM
-		
5FFF	_____	
-		
4FFF	_____	8K EPROM
-		
3FFF	_____	
-		
2FFF	_____	RAM
-		
1FFF	_____	
	OF00	
		Resident Assembler/Editor, Stack
	OA00	
-		RAM
0000	_____	

APPENDIX B

Oculometer Connector for Digital Data Output

Connector type: ITT # KPT02E20 - 41S

<u>Pin Number</u>	<u>Pin Output</u>
1	bit 1 - MSB of X eye direction
2	bit 2
3	bit 3
4	bit 4
5	bit 5
6	bit 6
7	bit 7
8	ground
9	bit 8
10	bit 9
11	bit 10 - LSB of X eye direction
12	clock
13	+5 volts DC
14	bit 1 - MSB of Y eye direction
15	bit 2
16	bit 3
17	bit 4
18	bit 5
19	bit 6
20	bit 7
21	ground
22	bit 8
23	bit 9
24	bit 10 - LSB of Y eye direction
25	clock
26	+5 volts DC
27	bit 1 - LSB of pupil diameter
28	bit 2
29	bit 3
30	bit 4
31	bit 5
32	bit 6
33	bit 7
34	ground
35	bit 8
36	bit 9
37	bit 10 - LSB of pupil diameter
38	clock
39	+5 volts DC
40	not connected
41	not connected

APPENDIX C

Pascal-like Description of Modules

1. Module Main Subsystem

Variable definitions:

INIT : location of Initialization module

STORE : location of Store Data module

TIMING : location of Add Timing module

DETNUM : location of Determine Instrument Number module

COMPAR : location of Compare Data Samples module

TABLE : location of Create Table module

TIMING : a timing loop 20 msec long

WRITELN ('HIT S TO START MISSION')

READ(CHAR)

IF CHAR = S THEN (* start the mission *)

BEGIN

INIT (* call INIT module *)

COUNT : = 0 (* look at beginning of array
that holds simulated
oculometer output *)

TIMLOOP (* execute timing loop that is 20 msec
long *)

STORE (* call STORE module *)

TIMING (* call TIMING module *)

DETNUM (* call DETNUM module *)

COMPAR (*call COMPAR module *)

```

TABLE (* call TABLE module *)

IF we are at the end of simulated oculometer
  output THEN Main module is done

ELSE go back to TIMING

END

ELSE (*character is not an S*) print out another prompt
  to start the mission

```

2. Module Initialization

Variable definitions:

LSTRT1, LSTRT2, LSTRT3 : start time of last data sample

LASTIN : instrument number of last data sample

LOTIM, MITIM, HITIM : number of time increments since beginning of data run

OLDIN : last instrument through create table

Declare all variables in two or more modules

LSTRT1 : = 0

LSTRT2 : = 0

LSTRT3 : = 0

LASTIN : = 0

LOTIM : = 0

MITIM : = 0

HITIM : = 0

OLDIN : = 0

Final Data Table : = 0

Initialize 6522 Counter

Store interrupt routine

3. Module Store Data (Simulated)

Variable definitions:

DATA[I] : array that contains simulated oculometer output in the format of XDIRH, XDIRE, YDIRH, YDIRE, TRACK. The end of the array is a 'D'.

XDIRH : high byte of X eye direction

XDIRE : low byte of X eye direction

YDIRH : high byte of Y eye direction

YDIRE : low byte of Y eye direction

TRACK : track/out-of-track status; = 0 if out-of-track

COUNT : pointer to keep track of where you are in the DATA array. It is initialized to zero in the Main module.

XDIRH : = DATA [COUNT]

COUNT : = COUNT + 1

XDIRE : = DATA [COUNT]

COUNT : = COUNT + 1

YDIRH : = DATA [COUNT]

COUNT : = COUNT + 1

YDIRE : = DATA [COUNT]

COUNT : = COUNT + 1

TRACK : = DATA [COUNT]

COUNT : = COUNT + 1

Module Store Data (Oculometer)

Variable definitions:

XDIRE : low 8 bits of X direction

XDIRH : hi 8 bits of X direction

YDIRL : low 8 bits of Y direction
 YDIRH : hi 8 bits of Y direction
 TRACK : track/out-of-track status
 READ bits 1-8 of X direction into bits 0-7 of XDIRL
 READ bits 9-10 of X direction into bits 0-1 of XDIRH
 READ bits 1-8 of Y direction into bits 0-7 of YDIRL
 READ bits 9-10 of Y direction into bits 0-1 of YDIRH
 IF bit 1 of XDIR = 1 THEN (* it was a negative
 number *) set bits 2-7 of XDIRH to 1
 ELSE (* it was a positive number *) set bits 2-7 of
 XDIRH to 0
 IF bit 1 of YDIRH = 1 THEN (* it was a negative
 number *) set bits 2-7 of YDIRH to 1
 ELSE (*it was a positive number *) set bits 2-7 of
 YDIRH to 0
 IF oculometer out-of-track THEN TRACK : = 0
 ELSE (* oculometer is in track *) TRACK : = 1

4. Module Add Timing

Variable definitions:

LOTIME : low byte of time intervals elapsed to the
 last 6522 interrupt
 MITIME : middle byte of time intervals elapsed to the
 last 6522 intervals
 HITIME : high byte of time intervals elapsed to the
 last 6522 interrupt
 LCOUNT : low byte of 6522 counter
 HCOUNT : high byte of 6522 counter
 START1 : low byte of number of time intervals elapsed
 to the current data sample

START2 : middle byte of number of time intervals
 elapsed to the current data sample

START3 : high byte of number of time intervals elapsed
 to the current data sample

READ HCOUNT and LCOUNT

Divide HCOUNT, LCOUNT by 1000 and subtract from 65

Add this to LOTIME, MITIME, HITIME and store result in
START1, START2, START3

5. Module Determine Instrument Number

Variable definitions:

XDIR : X eye direction of current data sample

YDIR : Y eye direction of current data sample

GRID[I] : array with instrument boundaries followed
 by the instrument number with a format of
 upper X, lower X, upper Y, lower Y,
 instrument number. There is an 'E' at the
 end of the array

NOINST : instrument number for no instrument

INST : the data sample's instrument number

TRACK : track/out-of-track status of current data
 sample. It is equal to zero if data sample
 is out-of-track

XH : storage for upper X boundary from GRID

XL : storage for lower X boundary from GRID

YH : storage for upper Y boundary from GRID

YL : storage for lower Y boundary from GRID

CURINS : storage for instrument number from GRID

```

I := 1
IF TRACK = 0 THEN
    INST := 0
ELSE
    WHILE not at the end of GRID DO
        BEGIN
            XH := GRID [I]
            I := I + 1
            XL := GRID [I]
            I := I + 1
            YH := GRID [I]
            I := I + 1
            YL := GRID [I]
            I := I + 1
            CURINS := GRID [I]
            I := I + 1
            IF XH - XDIR < 0 THEN
                it cannot be in that instrument so
                go back to WHILE and get another
                set of instrument boundaries
            IF XL - XDIR > 0 THEN
                go back to WHILE and get another
                set of instrument boundaries
            IF YH - YDIR < 0 THEN
                go back to WHILE and get another
                set of instrument boundaries
            IF YL - YDIR > 0 THEN
                go back to WHILE and get another
                set of instrument boundaries
            (* If the data sample made it through
            all the tests, it was within that
            instrument's boundaries *)

```

INST : = CURINS

END (* WHILE *)

(* If reached the end of GRID and no instrument number
was found, define the instrument as no instrument *)

IF reached the end of GRID THEN INST : = NOINST

6. Module Compare Data Samples

Variable definitions:

INST : current data sample's instrument number

LASTIN : last data sample's instrument number

START : start time of current data sample

LSTRT : start time of last data sample

TOTIM : total number of time increments on LASTIN.
This is sent to the next module

INSTNM : instrument number sent to the next module
with TOTIM

IF LASTIN $\neq$ INST THEN

BEGIN

TOTIM : = START - LSTRT

INSTNM : = LASTIN

LASTIN : = INST

LSTRT : = START

END

(* ELSE if they are not equal, disregard the data
sample *)

7. Module Create Table

Variable definitions:

INSTNM : instrument number of the current data sample

TOTIM : total number of time increments of the current
current data sample

OLDIN : instrument number of the last data sample
through the module if the sample was not a
glitch or a blink.

GLITCH, BLINK, DATAGN, INS1..25 : each represents the
starting address of a row in the table for that
instrument. Each row is 33 bytes long.

IF TOTIM < 50 THEN (* the data sample is a glitch *)

BEGIN

increment GLITCH + 31, 32 (* 31 and 32 are
bytes for number of fixations *)

increment GLITCH + OLDIN (* determine which
instrument was transitioned from *)

add TOTIM to GLITCH + 28, 29, 30 (* bytes
28, 29, and 30 represent the amount
of time spent on that instrument *)

END

ELSE (*it is not a glitch*)

IF INSTNM = 0 THEN (* it is out-of-track *)

IF TOTIM ≤ 183 THEN

(* it is a blink *)

BEGIN

increment BLINK + 31, 32

increment BLINK + OLDIN

add TOTIM to BLINK + 28, 29, 30

END

```

ELSE (*it is data gone *)
    BEGIN
        increment DATAGN + 31, 32
        increment DATAGN + OLDIN
        add TOTIM to BLINK + 28, 29, 30
        OLDIN : = INSTNM
    END
ELSE (*it is not out-of-track so it is an
    instrument*)
    BEGIN
        X : = INSTNM
        increment INSX + 31, 32
        increment INSX + OLDIN
        add TOTIM to INSX + 28, 29, 30
        OLDIN : = INSTNM
    END

```

8. Module: Print Results

Variable definitions:

TABLE[I] : location of final data table

NEXTINST : difference in location between the same data
in two instruments

ACCTIME : location of accumulated time

FIX : location of the total number of fixations for an
instrument

TOTIME : total accumulated time for all instruments

TOTFIX : total number of fixations for all instruments

```

MINUTES : total number of minutes in a mission
FIXMIN : number of fixes per minute for an instrument
TRANS[I] : number of transitions from instrument I to
            the instrument you are examining
MEANDWL : mean dwell time for an instrument
PROPDWL : proportion of dwell time on an instrument
PROPFIX : proportion of fixations for an instrument
(*print out total dwell time on each instrument*)
DO WHILE there are instruments
    BEGIN
        WRITELN (TABLE[ACCTIME])
        (*go to the next instrument*)
        ACCTIME := ACCTIME + NEXTIN
    END
(*print out mean dwell time on each instrument*)
DO WHILE there are instruments
    BEGIN
        MEANDWL := TABLE[ACCTIME]/TABLE[FIX]
        WRITELN (MEANDWL)
        (*go to next instrument*)
        ACCTIME := ACCTIME + NEXTINST
        FIX := FIX + NEXTINST
    END
(*print proportion of dwell time on each instrument*)
DO WHILE there are instruments

```

```

BEGIN
    TOTIME := TOTIME + TABLE[ACCTIME]
    ACCTIME := ACCTIME + NEXTINST
END
DO WHILE there are instruments
    BEGIN
        PROPDWL := TABLE[ACCTIME]/TOTIME
        WRITELN(PROPDWL)
        (*go to the next instrument*)
        ACCTIME := ACCTIME + NEXTINST
    END
    (*print proportion of fixations on each instrument*)
    DO WHILE there are instruments
        BEGIN
            TOTFIX := TOTFIX + TABLE[FIX]
            FIX := FIX + NEXTINST
        END
        DO WHILE there are instruments
            BEGIN
                PROPFIX := TABLE[FIX]/TOTFIX
                WRITELN (PROPFIX)
                (*go to next instrument*)
                FIX := FIX + NEXTINST
            END
            (*print transition probabilities*)
            DO WHILE there are instruments

```

```

BEGIN
    FOR I := 1 to 25 DO
        WRITELN(TRANS[I]
        (*go to the next instrument*)
        FIX := FIX + NEXTIN
    END
    (*print number of fixes per minute for each instrument*)
    DO WHILE there are instruments
        BEGIN
            FIXMIN := TABLE[FIX]/MINUTES
            WRITELN(FIXMIN)
            (*go to next instrument*)
            FIX := FIX + NEXTINST
        END
    END

```

APPENDIX D

Assembly Language Implementation of Modules

1. Module: Initialization

MODULE INIT

```

$FF
0010          .BA $1823
0020          .DS
0040 ;DEFINE ADDRESSES FOR ALL VARIABLES USED BY TWO MORE
0050 ;MODULES. THE FOLLOWING ARE USED BY STORE DATA AND
0060 ;DETERMINE INST NUM.
0080 XDIRH      .DS 1          ;HI BYTE OF X EYE DIRECTION
0070 XDIRL      .DS 1          ;LO BYTE OF X EYE DIRECTION
0080 YDIRH      .DS 1          ;HI BYTE OF Y EYE DIRECTION
0090 YDIRL      .DS 1          ;LO BYTE OF Y EYE DIRECTION
0100 TRACK      .DS 1          ;TRACK/OUT-OF-TRACK STATUS
0110 ;THE FOLLOWING ARE USED BY ADD TIMING AND COMPARE
0120 TART1       .DS 1          ;3 BYTES OF TIMING INFORMATION
0130 TART2       .DS 1          ; FOR CURRENT DATA SAMPLE
0140 TART3       .DS 1 ;
0150 ;THE FOLLOWING ARE USED BY INST NUM AND COMPARE
0160 INSTNM      .DS 1          ;INST NUM SENT TO CREATE TABLE
0181 ;THE FOLLOWING ARE USED BY TIMING AND CREATE TABLE
0170 TOTIM1      .DS 1          ;TOTAL TIME SPENT
0180 TOTIM2      .DS 1          ; ON INSTNM
0190 TOTIM3      .DS 1 ;
0200 ;THE FOLLOWING IS USED BY COMPARE AND CREATE TABLE
0210 INST        .DS 1          ;INST NUM OF CURRENT DATA SAMPLE
0220 ;THE FOLLOWING ARE USED BY INITIALIZATION AND COMPARE
0231 ;AND CREATE TABLE
0230 LTRT1       .DS 1          ;START TIME OF
0240 LTRT2       .DS 1          ; LAST DATA SAMPLE
0250 LTRT3       .DS 1 ;
0260 LASTIN      .DS 1          ;INST NUM OF LAST DATA SAMPLE
0270 ;THE FOLLOWING ARE USED BY INITIALIZATION AND ADD TIMING
0280 LOTIM       .DS 1          ;NUMBER OF TIME INCREMENTS
0290 MITIM       .DS 1          ; SINCE BEGINNING OF
0300 HITIM       .DS 1          ; DATA RUN
0310 ;THE FOLLOWING ARE USED BY INITIALIZATION AND CREATE TABLE
0320 LAST        .DS 1          ;LAST INST THROUGH CREATE TABLE
0330 TABLE1     .DE $8000      ;STORAGE LOCATIONS
0340 TABLE2     .DE $8100      ; FOR FINAL
0350 TABLE3     .DE $8200      ; DATA TABLE
0360 TABLE4     .DE $8300 ;
0370 ;SIMULATED GRID WITH A FORMAT OF UPPER X, LOWER X, UPPER Y,
0371 ;LOWER Y, INSTRUMENT NUMBER. AN INSTRUMENT CANNOT CROSS THE Y
0372 ;AXIS. IF BOTH LOWER AND UPPER X ARE NEG, THE UPPER X IS THE
0373 ;ONE CLOSEST TO THE Y AXIS.
0380 GRID        .BY 00 $64 $00 $00 $01 $5E $00 $08 01
0390 GP11        .BY $FF $60 $FF $24 $01 $F4 $01 $2C 02

```

```

0400 GRI2      .BY $FE $EC $FE $70 $01 $36 $00 $DC 03
0410 GRI3      .BY $FE $EC $FE $70 $00 $DC $00 $64 04
0420 GRI4      .BY $FF $E2 $FF $38 $00 $C8 $00 $32 05
0430 GRI5      .BY $00 $A0 $00 $3C $01 $F4 $01 $90 06
0440 GRI6      .BY $01 $4A $00 $A0 $01 $2C $00 $64 07 'E'
0449          .BA $19D6
0450 ;THE FOLLOWING ARE USED FOR INITIALIZATION      THE 6522
0460 ACR        .DE $A00B      ;6522 AUXILIARY CONTROL REGISTER
0470 LLATCH     .DE $A006      ;6522 LD LATCH
0480 HLATCH     .DE $A005      ;6522 HI LATCH AND COUNTER
0490 LCOUNT     .DE $A004      ;6522 LD COUNTER
0500 IER        .DE $A00E      ;6522 INTERRUPT ENABLE REG
0510 IPVEC      .DE $A67E      ;SYSTEM INTERRUPT VECTOR
0520 ACCESS     .DE $8B86      ;UNPROTECT SYSTEM RAM
0530 LINT       .BY $00        ;LD BYTE OF INTERRUPT LOCATION
0540 HINT       .BY $30        ;HI BYTE OF INTERRUPT LOCATION
0550 ;
0560 INIT       LDA #00        ;INITIALIZE THE
0570             STA LSTR1      ; NECESSARY
0580             STA LSTR2      ; PARAMETERS TO
0590             STA LSTR3      ; ZERO
0600             STA LASTIN
0610             STA LOTIM
0620             STA MITIM
0630             STA HITIM
0640             STA LAST
0650 ;INITIALIZE FINAL DATA TABLE TO ZERO
0660             LDY #00
0670 G01          LDA #00
0680             STA TABLE1,Y
0690             INY
0700             CPY #$FF
0710             BNE G01
0720             LDY #00
0730 G02          LDA #00
0740             STA TABLE2,Y
0750             INY
0760             CPY #$FF
0770             BNE G02
0780             LDY #00
0790 G03          LDA #00
0800             STA TABLE3,Y
0810             INY
0820             CPY #$FF
0830             BNE G03
0840             LDY #00
0850 G04          LDA #00
0860             STA TABLE4,Y
0870             INY
0880             CPY #$FF
0890             BNE G04
0900             LDY #$FF
0910             LDA #00
0920             STA TABLE1,Y
0930             STA TABLE2,Y
0940             STA TABLE3,Y

```

```

1050          STA TABLE4,Y
1060      ; INITIALIZE 6522 COUNTER
1070          JSR ACCESS
1080          SEI
1090          LDA LINT
1100          STA IROVEC      ; LOAD ADDRESS OF
1110          LDA HINT
1120          STA IROVEC+1    ; INTERRUPT ROUTINE
1130          LDA #$C0
1140          STA IER        ; INITIALIZE IER
1150          LDA #$40
1160          STA ACR        ; INITIALIZE ACR
1170          LDA #$E8      ; LOAD 65000 INTO
1180          STA LLATCH    ; TIMER. TIMER
1190          LDA #$FD      ; IS NOW RUNNING
1200          STA HLATCH
1210          CLI           ; ENABLE INTERRUPTS
1220          RTS
1230      ; INTERRUPT ROUTINE TO ADD 65 TO THE CURRENT
1240      ; NUMBER OF TIME INTERVALS ELAPSED
1250          .BA $3000
1260          PHA
1270          LDA #$41
1280          CLC
1290          ADC LOTIM
1300          STA LOTIM
1310          LDA #$00
1320          ADC MITIM
1330          STA MITIM
1340          LDA #00
1350          ADC HITIM
1360          STA HITIM
1370          LDA LDCOUNT    ; CLEAR 6522 INTERRUPTS
1380          CLI           ; ENABLE SYM INTERRUPTS
1390          PLA
1400          RTI
1410          .EN

```

2. Module: Store Data

LOAD STORE

```

0000 .BA $6490
0020 .DS
0040 ;SIMULATED OCULOMETER INPUT DATA IN FORM OF TRACK,X,Y
0050 DATA .BY 01 00 $3C 01 $2C 01 00 $3C 01 $2C 01 00 04 01 $2C
0060 DATA1 .BY 01 00 04 01 $2C 01 00 04 01 $2C 01 00 04 01 $2C
0070 DATA2 .BY 01 00 04 01 $2C 01 00 04 01 $2C 01 00 04 01 $90
0080 DATA3 .BY 01 00 00 01 $90 01 00 00 01 $90 01 00 00 01 $90
0090 DATA4 .BY 01 $FF $9C 00 $3C 01 $FF $9C 00 $3C 01 $FF $9C 00 $3C
0100 DATA5 .BY 01 $FF $9C 00 $3C 01 $FF $9C 00 $3C 01 $FF $9C 00 $3C
0110 DATA6 .BY 01 $FF $9C 00 $3C 00 $FF $9C 00 $3C 00 $FF $9C 00 $3C
0120 DATA7 .BY 00 $FF $9C 00 $3C 01 $FF $9C 00 $64 01 $FF $9C 00 $64
0130 DATA8 .BY 01 $FF $9C 00 $64 01 $FF $9C 00 $64 01 $FE $D4 00 $C8
0140 DATA9 .BY 01 $FE $D4 00 $C8 01 $FE $D4 00 $C8 01 $FE $D4 $01 $2C
0150 DATA10 .BY 01 $FE $D4 $01 $2C 01 $FE $D4 $01 $2C 01 $FE $D4 $01 $2C
0160 DATA11 .BY 01 $FE $D4 $01 $2C 01 $FE $D4 $01 $2C 01 00 00 01 $90
0170 DATA12 .BY 01 00 00 01 $90 01 00 00 01 $90 01 00 00 01 $90
0180 DATA13 .BY 01 00 00 01 $90 01 00 $64 00 $64 01 00 $64 00 $64
0190 DATA14 .BY 01 00 $64 00 $64 01 00 $64 00 $64 00 00 01 $90 01 'D'
0200 .BA $1A70
0210 XDIRH .DE $1823
0220 XDIRL .DE $1824
0230 YDIRH .DE $1825
0240 YDIRL .DE $1826
0250 TRACK .DE $1827
0260 STORE LDA DATA,Y
0270 STA TRACK
0280 INY
0290 LDA DATA,Y
0300 STA XDIRH
0310 INY
0320 LDA DATA,Y
0330 STA XDIRL
0340 INY
0350 LDA DATA,Y
0360 STA YDIRH
0370 INY
0380 LDA DATA,Y
0390 STA YDIRL
0400 INY
0410 RTS
0420 .EN

```

3. Division routine called by Add Timing module

>LOAD TIMDIV

>FP

```

0010      .BA $1AA4
0020      .DS
0040  DIVHI      .DS 1      ;HIGH BYTE OF DIVISOR
0050  DIVLO      .DS 1      ;LOW BYTE OF DIVISOR
0060  REMHI      .DS 1      ;HI BYTE OF REMAINDER
0070  REMLO      .DS 1      ;LO BYTE OF REMAINDER
0080  QUDHI      .DS 1      ;HI BYTE OF DIVIDEND/QUOTIENT
0090  QUDLO      .DS 1      ;LO BYTE OF DIVIDEND/QUOTIENT
0091  COUNT      .DS 1
0100  ; THIS ROUTINE DIVIDES A TWO-BYTE DIVIDEND BY A
0110  ;2-BYTE DIVISOR TO GET A 2-BYTE QUOTIENT AND A 2-BYTE
0120  ;REMAINDER
0130  START      LDA #00
0140              STA REMHI      ;INITIALIZE ALL VARIABLES
0150              STA REMLO
0151              STA COUNT
0200      LDA #$03
0210              STA DIVHI
0220      LDA #$E8
0230              STA DIVLO
0250              JSR SHIFT
0260  LOOP      SEC
0261              LDA REMLO
0270              SBC DIVLO
0280              STA REMLO
0290              LDA REMHI      ;REM-DIV
0300              SBC DIVHI
0310              STA REMHI
0320              BCS POS      ;IF CARRY=1, RESULT WAS POS
0330              LDA REMLO
0340              ADC DIVLO
0350              STA REMLO
0360              LDA REMHI
0370              ADC DIVHI
0380              STA REMHI
0390              JSR SHIFT
0400              JMP GO
0410  POS      LDA #01
0420              ORA QUDLO
0440              STA QUDLO
0441              JSR SHIFT
0450  GO      INC COUNT
0460              LDA #15
0461              CMP COUNT
0470              BNE LOOP
0480              RTS
0490  ; SUBROUTINE TO SHIFT THREE BYTES LEFT
0500  SHIFT      ASL QUDLO      ;SHIFT LEFT-0 INTO LSB

```

0510		BCC ONE	; IF CARRY=0, DON'T NEED
0520		ASL QUQHI	; TO BRING BIT OVER
0530		LDA #01	
0540		ORA QUQHI	
0550		STA QUQHI	; BRING CARRY OVER
0560		JMP TWO	
0570	ONE	ASL QUQHI	
0580	TWO	BCC THREE	
0590		ASL REMLO	
0600		LDA #01	
0610		ORA REMLO	
0620		STA REMLO	
0630		JMP FOUR	
0640	THREE	ASL REMLO	
0650	FOUR	BCC FIVE	
0660		ASL REMHI	
0670		LDA #01	
0680		ORA REMHI	
0690		STA REMHI	
0700		JMP SIX	
0710	FIVE	ASL REMHI	
0720	SIX	DEC	
0721		RTS	
0730		.EN	

4. Module: Add Timing

LOAD TIMING

```

00FF
0010      .BA $1B50
0020      .DS
0040  DIVIS      .DE $1AAB      ;ROUTINE TO DIVIDE
0050  QUQHI      .DE $1AA8      ;HI BYTE OF DIVIDEND/QUOTIENT
0060  QUQLQ      .DE $1AA9      ;LO BYTE OF DIVIDEND/QUOTIENT
0070  LOTIME      .DE $1834      ;LO BYTE OF NUM OF TIME INTERVALS
0080  MITIME      .DE $1835      ;MIDDLE BYTE OF TIME INTERVALS
0090  HITIME      .DE $1836      ;HI BYTE OF TIME INTERVALS
0100  HLATCH      .DE $A005      ;6522 HI LATCH
0110  LCOUNT      .DE $A004      ;6522 LO COUNTER
0120  START1      .DE $1828
0130  START2      .DE $1829
0140  START3      .DE $182A
0150  TIMING      .SEI
0151  ;THIS MODULE DETERMINES THE NUMBER OF TIME INTERVALS THAT HAVE
0152  ;ELAPSED FROM THE BEGINNING OF THE DATA RUN TO THE CURRENT
0153  ;DATA SAMPLE. IT CALLS DIVIS, WHICH DIVIDES TWO BYTES BY TWO
0154  ;BYTES.
0160      LDA HLATCH      ;READ HI BYTE OF COUNTER
0170      STA QUQHI
0180      LDA LCOUNT      ;READ LO BYTE OF COUNTER
0190      STA QUQLQ
0200      JSR DIVIS      ;DIVIDE COUNT BY 1000
0210  ; SUBTRACT RESULT FROM 65 TO FIND
0220  ;NUMBER OF TIME INTERVALS ELAPSED
0230      SEC
0240      LDA #$41      ;LO BYTE OF 65
0250      SBC QUQLQ      ;$41-QUQLQ
0260      STA QUQLQ
0270      LDA #$00      ;HI BYTE OF 65
0280      SBC QUQHI
0290      STA QUQHI
0300  ; NOW ADD TO THE CURRENT NUMBER
0310  ;OF TIME INTERVALS ELAPSED
0320      LDA LOTIME
0330      CLC
0340      ADC QUQLQ
0350      STA START1      ;START TIME OF CURRENT DATA SAMPLE
0360      LDA MITIME
0370      ADC QUQHI
0380      STA START2
0390      LDA #00
0400      ADC HITIME
0410      STA START3
0420      CLI
0430      RTS
0440      .EN

```

5. Module: Determine Instrument Number

LOAD INSTNM

PP

```

0010 ;FIND INSTRUMENT NUMBER
0020 ;
0030 ; THIS SUBROUTINE WILL TAKE THE X AND Y DATA AND TRACK
0040 ;DATA FROM THE CURRENT DATA SAMPLE AND DEFINE THE
0050 ;INSTRUMENT NUMBER OF THE DATA SAMPLE.
0060 ; THE X DATA IS TWO BYTES AND IS CALLED XDIRH (HIGH
0070 ;ORDER BYTE) AND XDIRL (LOW ORDER BYTE). THE Y DATA
0080 ;IS YDIRL AND YDIRH. THE INSTRUMENTS ARE
0090 ;DEFINED BY THE X AND Y VALUES OF THE DIAGONAL CORNERS,
0100 ;CALLED UPPER AND LOWER X VALUES AND UPPER AND LOWER
0110 ;Y VALUES. THESE 4 VALUES ARE TWO BYTES EACH AND ARE
0120 ;STORED IN THE GRID.
0130 ;
0140 .BA $18B0
0160 .DS
0250 COMH .DS 1
0260 COML .DS 1
0270 GRIDH .DS 1
0280 GRIDL .DS 1
0290 RESH .DS 1 ;DEFINED IN SUBTRACT SUBROUTINE
0300 RESL .DS 1
0310 E .BY 'E'
0320 GRID .DE $1838
0330 INST .DE $182B ;INST NUM THAT IS DETERMINED
0340 XDIRH .DE $1823 ;X EYE DIRECTION OF
0350 XDIRL .DE $1824 ; CURRENT DATA SAMPLE
0360 YDIRH .DE $1825 ;Y EYE DIRECTION OF
0370 YDIRL .DE $1826 ; CURRENT DATA SAMPLE
0371 TRACK .DE $1827 ;TRACK/OUT-OF-TRACK STATUS
0380 NOINST .BY 25 ;INST NUM OF NO INSTRUMENT
0390 BUFFER .DE $3500
0430 ;
0430 ; FIND THE INSTRUMENT NUMBER
0440 ; DETERMINE IF IT IS OUT-OF-TRACK
0441 TXA
0442 PHA ;SAVE X REGISTER FOR BUFFER LOCATION
0450 LDX #00 ;INITIALIZE X
0460 LDA TRACK ;PUT TRACK BYTE IN A
0470 BNE FIND ;IF ZERO, SET INST NUM TO ZERO
0480 LDA #00
0490 STA INST
0500 JMP END
0510 FIND LDA GRID,X ;GET HIGH BYTE OF
0520 STA GRIDH ; UPPER X VALUE
0530 INX
0540 LDA GRID,X ;GET LOW BYTE OF
0550 STA GRIDL ; UPPER X VALUE
0560 LDA XDIRH ;COMH=XDIRH

```

```

0070      STA COMH
0080      LDA XDIRL      ;COML=XDIRL
0090      STA COML
0100      JSR SUBTRA      ;XH-X
0110      BCC BACK1      ;NOT IN GRID IF RESULT NEG
0120 ; GET TWO BYTES OF LOWER X VALUE
0130      INX
0140      LDA GRID,X      ;GET HIGH BYTE OF
0150      STA GRIDH      ; LOWER X VALUE
0160      INX
0170      LDA GRID,X      ;GET LOW BYTE OF
0180      STA GRIDL      ;LOWER X VALUE
0190 ; ALREADY HAVE XDIRL IN COML AND XDIRH IN COMH
0200      JSR SUBTRA      ;XL-X
0210      BCS BACK2      ;NOT IN GRID IF RESULT POS
0220 ; GET TWO BYTES OF UPPER Y VALUE
0230      INX
0240      LDA GRID,X      ;HIGH BYTE OF
0250      STA GRIDH      ;UPPER Y VALUE
0260      INX
0270      LDA GRID,X      ;LOW BYTE OF
0280      STA GRIDL      ; UPPER Y VALUE
0290      LDA YDIRH      ;COMH=YDIRH
0300      STA COMH
0310      LDA YDIRL      ;COML=YDIRL
0320      STA COML
0330      JSR SUBTRA      ;YL-Y
0340      BCC BACK3      ;NOT IN GRID IF RESULT NEG
0350 ; GET TWO BYTES OF LOWER Y VALUE
0360      INX
0370      LDA GRID,X      ;HIGH BYTE OF
0380      STA GRIDH      ; LOWER Y VALUE
0390      INX
0400      LDA GRID,X      ;LOW BYTE OF
0410      STA GRIDL      ; LOWER Y VALUE
0420      JSR SUBTRA      ;YL-Y
0430      BCS BACK4      ;NOT IN GRID IF RESULT POS
0440 ; HAVE FOUND THE INSTRUMENT
0450      INX
0460      LDA GRID,X
0470      STA INST
0480      JMP END
0490 BACK1  INX
0500      INX
0510 BACK2  INX
0520      INX
0530 BACK3  INX
0540      INX
0550 ; STILL NEED TO FIND INSTRUMENT NUMBER
0560 BACK4  INX ;LOOK AT INSTR NUM
0570      INX ;LOOK AT NEXT NUMBER
0580      LDA GRID,X
0590      CMP E
0600      BNE FIND      ;BACK TO BEGINNING IF NOT AN E
0610      LDA NOINST
0620      STA INST

```

```

1420 END          PLA
1421              TAX
1440              RTS
1450 ;
1460 ;SUBROUTINE SUBTRACT
1470 ; NEED THE FOLLOWING VARIABLES FOR THIS SUBROUTINE:
1480 ;COMH - HIGH BYTE OF DATA SAMPLE
1490 ;CQML - LOW BYTE OF DATA SAMPLE
1500 ;GRIDH - HIGH BYTE OF GRID PARAMETER
1510 ;GRIDL - LOW BYTE OF GRID PARAMETER
1520 ;
1530 SUBTRA      SEC ;SET CARRY
1540              LDA GRIDL
1550              SBC CQML      ;SUBTRACT LOW ORDER BYTES
1560              STA RESL      ;STORE RESULT
1570              LDA GRIDH
1580              SBC COMH      ;SUBTRACT HIGH ORDER BYTES
1590              STA RESH      ;STORE RESULT
1600              RTS
1610              .EN

```

5. Module: Compare Data Samples

LOAD COMPAR

CTR

```

0010 ; PROGRAM COMPARE
0020 ; THIS PROGRAM WILL COMPARE THE LAST DATA SAMPLE TO THE
0030 ; PRESENT DATA SAMPLE. IF THE INSTRUMENT NUMBER IS THE SAME, DO
0040 ; NOTHING. IF THEY ARE DIFFERENT, SET THE STOP TIME OF THE
0050 ; LAST DATA SAMPLE EQUAL TO THE START TIME OF THE CURRENT DATA
0060 ; SAMPLE AND SEND THE LAST DATA SAMPLE TO THE BUFFER. THE
0070 ; CURRENT DATA SAMPLE BECOMES THE LAST DATA SAMPLE.
0080 ;
0090 .BA $1080
0100 .DS
0110 INSTNM .DE $182B ;INST NUM INTO COMPARE
0120 INST .DE $182F ;INST NUM TO CREATE TABLE
0130 START1 .DE $1828 ;CURRENT DATA SAMPLE
0140 START2 .DE $1829 ;TIMING INFORMATION
0150 START3 .DE $182A
0160 LASTIN .DE $1833 ;LAST DATA SAMPLE INST NUM
0170 LSTRT1 .DE $1830 ;START TIME OF LAST
0180 LSTRT2 .DE $1831 ;DATA SAMPLE
0190 LSTRT3 .DE $1832
0200 TOTIM1 .DE $182C ;TOTAL TIME SPENT
0210 TOTIM2 .DE $182D ;ON INSTNM
0220 TOTIM3 .DE $182E
0230 BUFFER .DE $3500
0240 TEST .DE $641C ;TEST BYTE USED IN MAIN MODULE
0250 ;
0260 ;COMPARE LAST DATA SAMPLE TO PRESENT DATA SAMPLE
0270 ;
0271 LDA #00
0272 STA TEST
0280 LDA INSTNM
0290 CMP LASTIN ;IS INST=LASTIN?
0300 BEQ END
0310 LDA LASTIN ;IF NO, SEND INFO TO BUFFER
0311 STA INST
0320 STA BUFFER,X
0330 INX
0340 ;
0350 ;SUBTRACT TO FIND TOTAL TIME AND SEND INFO TO BUFFER.
0360 ;3-BYTE SUBTRACT: START-LSTRT1, TO GET TOTAL TIME ON
0370 ;THAT INSTRUMENT.
0380 ;
0390 SEC ;SET CARRY
0400 LDA START1
0410 SBC LSTRT1 ;SUBTRACT LOW ORDER BYTES
0420 STA TOTIM1
0430 STA BUFFER,X ;STORE RESULT IN BUFFER
0440 INX

```

```

0450      LDA START2
0460      SBC LSTRT2      ;SUBTRACT MIDDLE BYTES
0470      STA TOTIM2
0480      STA BUFFER,X      ;STORE RESULT IN BUFFER
0490      INX
0500      LDA START3
0510      SBC LSTRT3      ;SUBTRACT HI ORDER BYTES
0520      STA TOTIM3
0530      STA BUFFER,X      ;STORE RESULT IN BUFFER
0540      INX
0560      ;SET NEW LAST DATA SAMPLE
0570      ;
0580      LDA INSTNM      ;LASTIN=INSTNM
0590      STA LASTIN
0600      LDA START1      ;LSTRT1=START1
0610      STA LSTRT1
0620      LDA START2      ;LSTRT2=START2
0630      STA LSTRT2
0640      LDA START3      ;LSTRT3=START3
0650      STA LSTRT3
0651      LDA #01
0652      STA TEST
0660      END            RTS
0670      .EN

```

7. Update-routine called by Create Table module

LOAD UPDAT

>PR

```

0010      .BA $2000
0020      .DS
0040      ;THIS PROGRAM IS CALLED BY MASTER.
0050      ;IT WILL UPDATE THE PROPER COLUMNS IN THE TABLE.
0060      TABLE1      .DE $6000
0070      TABLE2      .DE $6100
0080      TABLE3      .DE $6200
0090      TABLE4      .DE $6300
0100      COLTT1      .BY 28      ;COLUMN FOR HI BYTE OF TIME
0110      COLTT2      .BY 29      ;COLUMN FOR MID BYTE OF TIME
0120      COLTT3      .BY 30      ;COLUMN FOR LO BYTE OF TIME
0130      COLFXL      .BY 31      ;COL FOR NUM OF FIXATIONS
0140      COLFXH      .BY 32
0150      TEMP      .DE $2160
0160      LAST      .DE $1837
0170      TOTIM1      .DE $182C
0180      TOTIM2      .DE $182D
0190      TOTIM3      .DE $182E
0200      UPDAT1      CLC
0210      TXA      ;TRANSFER X TO ACC
0220      ADC COLTT1
0230      TAX      ;TRANSFER ACC TO X
0240      CLC
0250      LDA TOTIM1
0260      ADC TABLE1,X
0270      STA TABLE1,X
0280      INX
0290      LDA TOTIM2
0300      ADC TABLE1,X
0310      STA TABLE1,X
0320      INX
0330      LDA TOTIM3
0340      ADC TABLE1,X
0350      STA TABLE1,X
0360      INX
0370      CLC
0380      LDA #01
0390      ADC TABLE1,X
0400      STA TABLE1,X
0410      INX
0420      LDA #00
0430      ADC TABLE1,X
0440      STA TABLE1,X
0450      LDX TEMP
0460      CLC
0470      TXA
0480      ADC LAST
0490      TAX

```

0500		INC TABLE1,X
0510		RTS
0520	UPDAT2	CLC
0530		TXA ;TRANSFER X TO ACC
0540		ADC COLTT1
0550		TAX ;TRANSFER ACC TO X
0560		CLC
0570		LDA TOTIM1
0580		ADC TABLE2,X
0590		STA TABLE2,X
0600		INX
0610		LDA TOTIM2
0620		ADC TABLE2,X
0630		STA TABLE2,X
0640		INX
0650		LDA TOTIM3
0660		ADC TABLE2,X
0670		STA TABLE2,X
0680		INX
0690		CLC
0700		LDA #01
0710		ADC TABLE2,X
0720		STA TABLE2,X
0730		INX
0740		LDA #00
0750		ADC TABLE2,X
0760		STA TABLE2,X
0770		LDX TEMP
0780		CLC
0790		TXA
0800		ADC LAST
0810		TAX
0820		INC TABLE2,X
0830		RTS
0840	UPDAT3	CLC
0850		TXA ;TRANSFER X TO ACC
0860		ADC COLTT1
0870		TAX ;TRANSFER ACC TO X
0880		CLC
0890		LDA TOTIM1
0900		ADC TABLE3,X
0910		STA TABLE3,X
0920		INX
0930		LDA TOTIM2
0940		ADC TABLE3,X
0950		STA TABLE3,X
0960		INX
0970		LDA TOTIM3
0980		ADC TABLE3,X
0990		STA TABLE3,X
1000		INX
1010		CLC
1020		LDA #01
1030		ADC TABLE3,X
1040		STA TABLE3,X

1050		INX
1060		LDA #00
1070		ADC TABLE3,X
1080		STA TABLE3,X
1090		LDX TEMP
1100		CLC
1110		TXA
1120		ADC LAST
1130		TAX
1140		INC TABLE3,X
1150		RTS
1160	UPDAT4	CLC
1170		TXA ;TRANSFER X TO ACC
1180		ADC COLTT1
1190		TAX ;TRANSFER ACC TO X
1200		CLC
1210		LDA TOTIM1
1220		ADC TABLE4,X
1230		STA TABLE4,X
1240		INX
1250		LDA TOTIM2
1260		ADC TABLE4,X
1270		STA TABLE4,X
1280		INX
1290		LDA TOTIM3
1300		ADC TABLE4,X
1310		STA TABLE4,X
1320		INX
1330		CLC
1340		LDA #01
1350		ADC TABLE4,X
1360		STA TABLE4,X
1370		INX
1380		LDA #00
1390		ADC TABLE4,X
1400		STA TABLE4,X
1410		LDX TEMP
1420		CLC
1430		TXA
1440		ADC LAST
1450		TAX
1460		INC TABLE4,X
1470		RTS
1480		.EN

3. Incr-routine called by Create Table module

>LOAD INCR

>FF

0010		.BA \$2130	
0020		.DS	
0050	TEMP	.DE \$2160	
0060	TEMSTO	.DS 1	
0090	INCR	DEC TEMSTO	; STORAGE FOR
0091		LDA TEMSTO	
0100		BEQ DONE	; INSTRUMENT NUMBER
0110		CLC	
0120		LDA #\$22	
0130		ADC TEMP	; TEMP IS THE
0140		STA TEMP	; PROPER ADDRESS
0150		JMP INCR	; IN THE TABLE
0160	DONE	LDX TEMP	
0170		RTS	
0180		.EN	

9. Module: Create Table

>LOAD MASTER

>EP

```

0010 ;MASTER ALGORITHM. THIS ALGORITHM WILL PUT THE INFORMATION
0020 ;IN THE DATA SAMPLES INTO THE FINAL DATA TABLE. THE DATA
0030 ;COMING IN IS IN THE FORM OF:
0040 ; INSTNM = INSTRUMENT NUMBER
0050 ; TOTIM1,TOTIM2,TOTIM3 = NUMBER OF TIME INTERVALS SPENT
0060 ;ON THAT INSTRUMENT.
0070 ;
0080 .BA $2160
0090 .DS
0110 TEMP .DS 1
0120 ; DEFINE SPACE FOR TABLE
0130 TABLE1 .DE $6000 ;FOR GLICH,BLINK,DATA GONE
0140 ;AND INST 1-4
0150 TABLE2 .DE $6100 ;FOR INSTS 5-11
0160 TABLE3 .DE $6200 ;FOR INSTS 12-18
0170 TABLE4 .DE $6300 ;FOR INSTS 19-25
0180 UPDAT1 .DE $2005
0190 UPDAT2 .DE $2049
0200 UPDAT3 .DE $208D
0210 UPDAT4 .DE $20D1
0220 INCR .DE $2131
0230 TEMSTO .DE $2130
0240 COMMA .DE $833A
0250 INST .DE $182F ;INST NUM THROUGH MASTER
0261 ; THIS IS LASTIN FROM MODULE COMPARE
0270 TOTIM1 .DE $182C ;TOTAL NUMBER OF TIME INCREMENTS
0280 TOTIM2 .DE $182D ; ON INSTNM
0290 TOTIM3 .DE $182E
0291 LAST .DE $1837 ;LAST INST NUM THROUGH CREATE TABLE
0300 ;
0310 MASTER TXA
0320 PHA ;SAVE X REGISTER
0330 LDA TOTIM3
0340 BNE NOGLIT ;IF TOTIM3 NOT 0,CANNOT BE A GLITCH
0350 LDA TOTIM2
0360 BNE NOGLIT ;IF TOTIM2 NOT 0,CANNOT BE A GLITCH
0370 SEC
0380 LDA #132
0390 BCC TOTIM1 ;IF TOTIM1>50 IT IS
0400 BCC TEST1 ; NOT A GLITCH
0410 LDX #00
0420 LDA #00
0430 STA TEMP
0440 JSR UPDAT1 ;UPDATE TABLE
0450 JMP END
0460 ;NOW SEE IF THE INST NUM IS OUT-OF-TRACK
0470 TEST1 LDA INST ;ALREADY KNOW TOTIM2+3 = 0.
0480 BNE FIND ;IF INSTNM NOT ZERO,IT IS AN INST

```

```

0490      SEC
0500      LDA #B7
0510      SBC TOTIM1      ;183-TOTIM1
0520      BCC NOGLIT      ;NO GLICH IF TOTIM1>183
0530      LDX #22        ;IF TOTIM1<183, IT IS A BLINK
0540      LDA #22
0550      STA TEMP
0560      JSR UPDAT1
0570      JMP END
0580 NOGLIT      LDA INST
0590      BNE FIND        ;GO TO FIND IF IT IS AN INST
0600      LDX #44        ;ELSE IT IS OUT-OF-TRACK
0610      LDA #44
0620      STA TEMP
0630      JSR UPDAT1
0640      LDA INST
0650      STA LAST
0660      JMP END
0670 ; FIND THE CORRECT SPOT IN TABLE FOR THAT INST
0680 FIND      SEC
0690      LDA #04
0700      SBC INST        ;IF INSTNM<5, THEN THAT INST
0710      BCC TEST2      ; IS IN TABLE1
0720      LDA #66
0730      STA TEMP
0740      LDA INST
0750      STA TEMSTQ
0760      JSR INCR        ;FIND CORRECT ROW IN TABLE
0770      JSR UPDAT1
0780      LDA INST
0790      STA LAST
0800      JMP END
0810 TEST2     SEC
0820      LDA #08
0830      SBC INST        ;IF INSTNM<12, THEN THAT INST
0840      BCC TEST3      ; IS IN TABLE2
0850      LDA #00
0860      STA TEMP
0870      SEC
0880      LDA INST
0890      SBC #104
0900      STA TEMSTQ
0910      JSR INCR
0920      JSR UPDAT2
0930      LDA INST
0940      STA LAST
0950      JMP END
0960 TEST3     SEC
0970      LDA #12
0980      SBC INST        ;IF INSTNM<18, THAT INST IS
0990      BCC TEST4      ; IN TABLE3
1000      LDA #00
1010      STA TEMP
1020      SEC
1030      LDA INST

```

1040		SBC #\$0B	
1050		STA TEMST0	
1060		JSR INCR	
1070		JSR UPDAT3	
1080		LDA INST	
1090		STA LAST	
1100		JMP END	
1110	TEST4	SEC	
1120		LDA #\$1A	
1130		SBC INST	; IF INSTNM<26, IT IS
1140		BCC TEST5	; IN TABLE4
1150		LDA #\$00	
1160		STA TEMP	
1170		SEC	
1180		LDA INST	
1190		SBC #\$12	
1200		STA TEMST0	
1210		JSR INCR	
1220		JSR UPDAT4	
1230		LDA INST	
1240		STA LAST	
1250		JMP END	
1260	TEST5	JSR COMMA	; HAVE AN ERROR
1270	END	PLA	
1280		TAX	
1290		RTS	
1300		.EN	

10. Module: Main Subsystem

LOAD MAIN

```

PR
0010      .BA $6400
0020      .DS
0040 DATA      .DE $6490      ;LOCATION OF SIMULATED OCULOMETER DATA
0050 INIT      .DE $19D8      ;INITIALIZATION MODULE
0060 STORE      .DE $1A70      ;STORE DATA FROM OCULOMETER
0070 TIMING     .DE $1B50
0080 INSTNM     .DE $1BB8
0090 COMPAR     .DE $1C80
0100 MASTER     .DE $2161
0110 OUTCHR     .DE $8A47      ;OUTPUT A CHARACTER
0120 INCHR      .DE $8A1B      ;INPUT A CHARACTER
0130 CRLF       .DE $834D      ;CARRIAGE RETURN AND LINE FEED
0140 START      .BY 'HIT S TO START MISSION-'
0150 COUNT1     .BY 23
0160 S          .BY 'D'
0170 COUNT2     .DS 1
0180 LD         .DS 1
0190 HI         .DS 1
0191 TEST       .DS 1          ;TEST TO SEE IF MODULE MASTER IS CALLED
0200 D          .BY 'D'
0210 ;PRINT OUT PROMPT TO START PROGRAM
0220 MAIN       JSR CRLF
0230           LDX #00
0240 LOOP1      LDA START,X
0250           JSR OUTCHR
0260           INX
0270           CPX COUNT1
0280           BNE LOOP1
0290           JSR INCHR
0300           CMP S
0310           BNE MAIN
0311           LDX #00
0320 ;DATA MISSION HAS STARTED
0330           JSR INIT        ;TIMER IS RUNNING
0340           LDY #00         ;DON'T USE Y IN ANY OTHER MODULES
0350 ;THE FOLLOWING IS A TIMING LOOP TO SIMULATE
0360 ;OCULOMETER DATA SAMPLES
0370 NEXT       LDA #$40      ;LO BYTE OF 832
0380           STA LD
0390           LDA #$03        ;HI BYTE OF 832
0400           STA HI
0410 LOOP2      SEC ;THIS LOOP IS APPROX
0420           LDA LD          ; APPROX 20 MSEC LONG
0430           SBC #01
0440           STA LD
0450           NOP
0460           LDA HI

```

0470		SBC #00	
0480		STA HI	
0490		BNE LOOP2	
0500	INOW STORE	A DATA SAMPLE	
0510		JER STORE	
0520		JER TIMING	
0530		JER INSTNM	
0540		JER COMPAR	
0541		LDA TEST	
0542		BEO OVER	
0550		JER MASTER	
0560	OVER	LDA DATA.Y	
0570		CMP D	ARE AT END OF DATA?
0580		BNE NEXT	IF NOT, TIME NEXT SAMPLE
0581		SEI	
0590		RTI	
0600		.EN	

The following modules are used by the Print Results module.

11. Module: Three

LOAD THREE

```

RF
0010 ;THREE
0020 ; THIS IS A GROUPING OF THREE ROUTINES USED BY
0030 ;THE PRINT RESULTS ROUTINE.
0040 .BA $2000
0050 .DB
0070 VALUE .DE $0000
0080 ;ROTATE RIGHT
0090 ; THIS ROUTINE ROTATES Y (INDEX REGISTER) BYTES TO
0100 ;THE RIGHT. THE X REGISTER IS THE LOCATION OF THE
0110 ;LS BYTE OF THE DATA TO BE ROTATED.
0120 ROTATL CLC ;CLEAR CARRY
0130 ROTL ROL VALUE,X ;ROTATE THE BYTE LEFT
0140 DEY ;DECREMENT BYTE COUNTER
0150 BNE MORRTL ;NOT ZERO, CONTINUE ROTATE
0160 RTS ;DONE, RETURN
0170 MORRTL INC ;ADVANCE MEMORY POINTER
0180 JMP ROTL ;CONTINUE TO ROTATE LEFT
0190 ;
0200 ;ROTATE RIGHT
0210 ; THE SAME PARAMETERS ARE USED AS IN ROTATE LEFT.
0220 ROTATR CLC ;CLEAR CARRY
0230 ROTR ROR VALUE,X ;ROTATE THE BYTE RIGHT
0240 DEY ;DECREMENT BYTE COUNTER
0250 BNE MORRTR ;NOT ZERO, CONTINUE ROTATE
0260 RTS ;DONE, RETURN
0270 MORRTR DEC ;DECREMENT MEMORY POINTER
0280 JMP ROTR ;CONTINUE ROTATE RIGHT
0290 ;
0300 ;COMPLEMENT
0310 ; THIS PERFORMS THE TWO'S COMPLEMENT OF Y NUMBER
0320 ;OF BYTES. THE X REGISTER CONTAINS THE LS BYTE OF THE
0330 ;VALUE TO BE COMPLEMENTED
0340 COMPLM SEC ;SET CARRY
0350 COMPL LDA #$FF ;LOAD $FF FOR COMPLEMENT OP
0360 EOR VALUE,X ;COMPLEMENT BYTE
0370 ADC #$00 ;IF CARRY=1, TWO'S COMPLEMENT
0380 STA VALUE,X ;STORE BYTE IN MEMORY
0390 INC ;ADVANCE MEMORY POINTER
0400 DEY ;DECREMENT BYTE COUNTER
0410 BNE COMPL ;NOT ZERO, CONTINUE
0420 RTS ;RETURN TO CALLING PROGRAM
0430 .EH

```

AD-A124 700

DEVELOPMENT OF AN OCULOMETER DATA COLLECTION SUBSYSTEM

2/2

(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH

SCHOOL OF ENGINEERING N L WOOD DEC 82

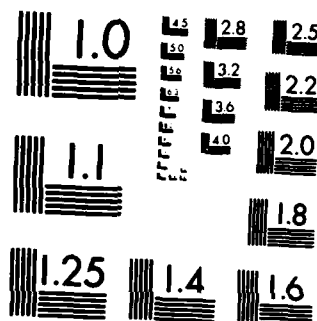
UNCLASSIFIED

AFIT/GE/EE/82D-72

F/G 9/2

NL

END
DATE
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2. Module: Seven

LOAD SEVEN

```

PP
0010          .BA $2030
0030          .DI
0040      :SUBROUTINE SEVEN
0050      :  THESE ARE SEVEN ROUTINES NEEDED FOR THE FLOATING
0060      :POINT ROUTINES.
0070      :
0080      FMPNT      .DE $0000      :FROM POINTER
0090      TOPNT      .DE $0002      :TO POINTER
0100      IIGN      .DE $0006      :SIGN INDICATOR
0110      FPLSW      .DE $0008      :FPACC LS BYTE
0120      FPNEN      .DE $0009      :FPACC NEXT SIGNIFICANT BYTE
0130      FPMEN      .DE $000A      :FPACC MOST SIGNIFICANT BYTE
0140      MCANDW      .DE $000C      :MULTIPLICATION WORK AREA
0150      FOPLSW      .DE $0010      :FPDP LEAST SIGNIFICANT BYTE
0160      FOPNSW      .DE $0011      :FPDP NEXT SIGNIFICANT BYTE
0170      FOPMSW      .DE $0012      :FPDP MOST SIGNIFICANT BYTE
0180      FOPEXP      .DE $0013      :FPDP EXPONENT
0190      WORKO      .DE $0014      :WORK AREA
0200      IOLEW      .DE $001F      :I/O WORK AREA LS BYTE
0210      IOSTR      .DE $0023      :I/O STORAGE
0220      IOSTR3      .DE $0026      :I/O STORAGE
0230      IOEXPD      .DE $0027      :I/O EXPONENT STORAGE
0240      ROTATL      .DE $2000      :ROTATE LEFT
0241      COMPLM      .DE $2018      :COMPLEMENT
0250      FPMULT      .DE $2110      :FLOATING POINT MULTIPLY
0260      :
0270      :CLEAR A SECTION OF MEMORY
0280      CLRMEM      LDA #$00      :SET UP ZERO VALUE
0290                  TAY          :INITIALIZE INDEX POINTER
0300      CLRM1       STA (TOPNT),Y      :CLEAR MEMORY LOCATION
0310                  INY          :ADVANCE INDEX POINTER
0320                  DEX          :DECREMENT COUNTER
0330                  BNE CLRM1      :NOT ZERO,CONTINUE CLEARING
0340                  RTS          :RETURN
0350      :
0360      :TRANSFER A SECTION OF MEMORY
0370      MOVIND      LDY #$00      :INITIALIZE INDEX POINTER
0380      MOVIN1       LDA (FMPNT),Y      :FETCH BYTE TO TRANSFER
0390                  STA (TOPNT),Y      :STORE BYTE IN NEW LOCATION
0400                  INY          :ADVANCE INDEX POINTER
0410                  DEX          :DECREMENT BYTE COUNTER
0420                  BNE MOVIN1      :NOT ZERO, CONTINUE
0430                  RTS          :RETURN
0440      :
0450      :MULTIPLE PRECISION ADDITION
0460      ADDER        CLC          :CLEAR CARRY FLAG
0470      ADDR1        LDA (TOPNT),Y      :FETCH BYTE FROM ONE VALUE
0480                  ADC (FMPNT),Y      :ADD BYTE OF OTHER VALUE

```

```

0440      STA (TOPNT),Y      ;STORE RESULT
0500      INY      ;INCREMENT INDEX POINTER
0510      DEX      ;DECREMENT COUNTER
0520      BNE ADDR1      ;NOT ZERO, CONTINUE ADDITION
0530      RTS      ;RETURN
0540      ;
0550      ;MULTIPLY FPACC BY TEN
0560      FPX10      LDA #$04
0570      STA FOPEXP      ;LOAD FPOP WITH A VALUE OF TEN
0580      LDA #$50      ; BY SETTING THE EXPONENT TO FOUR
0590      STA FOPMSW      ;AND THE MANTISSA TO $50,$00,$00
0600      LDA #$00
0610      STA FOPMSW
0620      STA FOPLSW
0630      JSR FPMULT      ;MULTIPLY FPACC BY FPOP
0640      DEC IOEXPD      ;DECREMENT DECIMAL EXPONENT
0650      RTS      ;RETURN TO TEST FOR COMPLETION
0660      ;
0670      ;MULTIPLY FPACC BY 0.1
0680      FPD10      LDA #$FD      ;PLACE 0.1 IN FPOP BY
0690      STA FOPEXP      ; SETTING FPOP EXPONENT TO -3
0700      LDA #$66      ;AND LOADING MANTISSA
0710      STA FOPMSW      ;WITH $66,$66,$67
0720      LDA #$67
0730      STA FOPLSW
0740      JSR FPMULT      ;MULTIPLY FPACC BY FPOP
0750      INC IOEXPD      ;INCREMENT DECIMAL EXPONENT
0760      RTS      ;RETURN
0770      ;
0780      DECBIN      LDA #$00
0790      STA IOCTR3      ;CLEAR MS BYTE+1 OF RESULT
0800      LDX #IOLSW      ;SET POINTER TO I/O WORK AREA
0810      STX TOPNT      ;STORE IN TOPNT
0820      LDX #IOSTR      ;SET POINTER TO I/O STORAGE
0830      STX FMPNT      ;STORE IN FMPNT
0840      LDX #$04      ;SET PRECISION COUNTER
0850      JSR MOVIND      ;MOVE I/O STORAGE TO WORK AREA
0860      LDX #IOSTR      ;SET POINTER TO ORIGINAL VALUE
0870      LDY #$04      ;SET PRECISION COUNTER
0880      JSR ROTATL      ;START X 10 ROUTINE (TOTAL=X2)
0890      LDX #IOSTR      ;RESET POINTER
0900      LDY #$04      ;SET PRECISION COUNTER
0910      JSR ROTATL      ;MULTIPLY BY TWO AGAIN (TOTAL=X4)
0920      LDX #IOLSW      ;SET POINTER TO I/O WORK AREA
0930      STX FMPNT      ;STORE IN FMPNT
0940      LDX #IOSTR      ;SET POINTER TO I/O STORAGE
0950      STX TOPNT      ;STORE IN TOPNT
0960      LDX #$04      ;SET PRECISION COUNTER
0970      JSR ADDER      ;ADD ORIGINAL TO ROTATED (TOTAL=X5)
0980      LDX #IOSTR      ;RESET POINTER
0990      LDY #$04      ;SET PRECISION COUNTER
1000      JMP ROTATL      ;X2 AGAIN (TOTAL=X10) AND RETURN
1010      .EN

```

1 Module: Floating Point Multiply

>LOAD FPMUL

>PR

```

0010 :FLOATING POINT MULTIPLY
0020 .BA $2110
0030 .DS
0050 FMPNT .DE $0000 :FPM POINTER
0060 TOPNT .DE $0002 :TO POINTER
0070 CNTR .DE $0004 :COUNTER STORAGE
0080 SIGNS .DE $0006 :SIGNS INDICATOR
0090 FPLSWE .DE $0007 :FPACC EXTENSION
0100 FPLSW .DE $0008 :FPACC LEAST SIGNIFICANT BYTE
0110 FPNLW .DE $0009 :FPACC NEXT SIGNIFICANT BYTE
0120 FPMWL .DE $000A :FPACC MOST SIGNIFICANT BYTE
0130 FPACCE .DE $000B :FPACC EXPONENT
0140 MCAND0 .DE $000C :MULTIPLICATION WORK AREA
0150 MCAND1 .DE $000D :MULTIPLICATION WORK AREA
0160 FOPLSW .DE $0010 :FPOP LEAST SIGNIFICANT BYTE
0170 FOPMSW .DE $0012 :FPOP MOST SIGNIFICANT BYTE
0180 FOPEXP .DE $0013 :FPOP EXPONENT
0190 WORK0 .DE $0014 :WORK AREA
0200 WORK1 .DE $0015 :WORK AREA
0210 WORK3 .DE $0017 :WORK AREA
0220 WORK6 .DE $001A :WORK AREA
0230 PAGE0 .DE $0000
0250 ROTATR .DE $200C :ROTATE RIGHT
0260 ADDER .DE $2045 :ADDITION ROUTINE
0270 MOVIND .DE $203A :ROUTINE TO MOVE BYTES

0280 COMPLM .DE $2018 :COMPLEMENT BYTES

0290 CLRMEM .DE $2030 :CLEAR AN AREA OF MEMORY
0300 FPNORM .DE $21FA :FLOATING POINT NORMALIZATION
0310 FPMULT JSP CFSIGN :SET UP AND CHECK SIGN OF MANTISSAS
0320 LDA FOPEXP :GET FPOP EXPONENT
0330 CLC :ADD FPACC EXPONENT
0340 ADC FPACCE :TO FPOP EXPONENT
0350 STA FPACCE :SAVE IN FPACC EXPONENT
0360 INC FPACCE :ADD ONE FOR ALGORITHM COMPENSATION
0370 SETMCT LDA #$17 :SET BIT COUNTER
0380 STA CNTR :STORE BIT COUNTER
0390 MULTIP LDX #FPMWL :SET POINTER TO FPACC MS BYTE
0400 LDY #$3 :SET PRECISION COUNTER
0410 JSP ROTATR :ROTATE FPACC RIGHT
0420 BCC NADOPP :CARRY=0, DON'T ADD PARTIAL PRODUCT
0430 ADOPP LDX #MCAND1 :POINTER TO LS BYTE OF MULTIPLICAND
0440 STX FMPNT :STORE POINTER
0450 LDX #WORK1 :POINTER TO LS BYTE OF PARTIAL PRODUCT
0460 STX TOPNT :STORE POINTER
0470 LDX #$6 :SET PRECISION COUNTER
0480 JSP ADDER :ADD MULTIPLICAND TO PARTIAL PRODUCT

```

```

0490  NADOPP      LDX #WORK6      ;SET POINTER TO MS BYTE OF PARTIAL
0500  : PRODUCT
0510      LDY #16      ;SET PRECISION COUNTER
0520      JSR ROTATR    ;ROTATE PARTIAL PRODUCT RIGHT
0530      DEC CNTR      ;DECREMENT BIT COUNTER
0540      BNE MULTIP    ;NOT ZERO, CONTINUE MULTIPLYING
0550      LDX #WORK6    ;ELSE, SET POINTER TO PARTIAL PRODUCT
0560      LDY #16      ;SET PRECISION COUNTER
0570      JSR ROTATR    ;MAKE ROOM FOR POSSIBLE ROUNDING
0580      LDX WORK3     ;SET POINTER TO 24TH BIT OF
0590  : PARTIAL PRODUCT
0600      LDA PAGE0,X    ;FETCH LS BYTE-1 OF RESULT
0610      ROL A          ;ROTATE 24TH BIT TO SIGN
0620      BPL PPREXFR    ;IF 24TH BIT=0, BRANCH AHEAD
0630      CLC           ;CLEAR CARRY FOR ADDITION
0640      LDY #13      ;SET PRECISION COUNTER
0650      LDA #140      ;ADD ONE TO 23RD BIT OF PARTIAL-PRODUCT
0660      ADC PAGE0,X    ;TO ROUND OFF RESULT
0670      STA WORK3     ;STORE SUM IN MEMORY
0680  CROUND      LDA #0      ;CLEAR A WITHOUT CHANGING CARRY
0690      ADC PAGE0,X    ;ADD WITH CARRY TO PROPAGATE
0700      STA PAGE0,X    ;STORE IN PARTIAL-PRODUCT
0710      INX           ;INCREMENT INDEX POINTER
0720      DEY           ;DECREMENT COUNTER
0730      BNE CROUND    ;NOT ZERO, ADD NEXT BYTE
0740  PPREXFR     LDX #FPPLSW    ;SET POINTER TO FPACC LSW-1
0750      STX TOPNT      ;STORE IN TOPNT
0760      LDX #WORK3     ;SET POINTER TO PARTIAL PRODUCT
0770  : LSW-1
0780      STX FMPNT      ;STORE IN FMPNT
0790      LDX #14      ;SET PRECISION COUNTER
0800  EXMLDV      JSR MOVIND    ;MOVE PARTIAL PRODUCT TO FPACC
0810      JSR FPNORM     ;NORMALIZE RESULT
0820      LDA SIGNS      ;GET SIGN STORAGE
0830      BNE MULTEX     ;IF NOT ZERO, SIGN IS POSITIVE
0840      LDX #FPPLSW    ;ELSE, SET POINTER TO FPACC LS BYTE
0850      LDY #13      ;SET PRECISION COUNTER
0860      JSR COMPLM     ;COMPLEMENT RESULT
0870  MULTEX      RTS          ;EXIT FPMUL1
0880  CKSIGN      LDA #0      ;SET PAGE PORTION OF POINTERS
0890      STA TOPNT+1    ;STORE IN TOPNT
0900      STA FMPNT+1    ;STORE IN FMPNT
0910      LDA #WORK0     ;SET POINTER TO WORK AREA
0920      STA TOPNT      ;STORE IN TOPNT
0930      LDX #18      ;SET PRECISION COUNTER
0940      JSR CLMEM      ;CLEAR WORK AREA
0950      LDA #MCAND0    ;SET POINTER TO MULTIPLICAND STORAGE
0960      STA TOPNT      ;STORE IN TOPNT
0970      LDX #14      ;SET PRECISION COUNTER
0980      JSR CLMEM      ;CLEAR MULTIPLICAND STORAGE
0990      LDA #11      ;INITIALIZE SIGN INDICATOR
1000      STA SIGNS      ; BY STORING ONE IN SIGNS
1010      LDA FPM5W      ;FETCH FPACC MS BYTE
1020      BPL DP8IGN     ;POSITIVE, CHECK FPOP

```

1030	NEGFPB	DEC SIGNS	;IF NEGATIVE, DECREMENT SIGNS
1040		LDX #FPLSW	;SET POINTER TO FPACC LS BYTE
1050		LDY #13	;SET PRECISION COUNTER
1060		JSR COMPLM	;MAKE POSITIVE FOR MULTIPLICATION
1070	OPSGNT	LDA FOPMSW	;IS FPOP NEGATIVE?
1080		BMI NEGOP	;YES, COMPLEMENT VALUE
1090		RTS	;ELSE, RETURN
1100	NEGOP	DEC SIGNS	;DECREMENT SIGNS INDICATOR
1110		LDX #FOPLSW	;SET POINTER TO FPOP LS BYTE
1120		LDY #13	;SET PRECISION COUNTER
1130		JMP COMPLM	;COMPLEMENT FPOP AND RETURN
1140		.EN	

Module: Convert to Floating Point

>LOAD CONVE

```

>PR
0010 ;CONVERT
0020 ; THIS CONVERTS A THREE BYTE NUMBER TO FLOATING POINT
0030 ;FORMAT. THE NUMBER TO BE CONVERTED IS IN NUM1, 2, AND 3
0040 ;WITH NUM1 THE LSB. THE RESULT IS IN THE FPLSW, FPNSW, FPACCE.
0050 .BA $21E0
0060 .DS
0070 TSIGN .DE $0005 ;SIGN INDICATOR
0080 FPLSW .DE $0008 ;FPACC LEAST SIG BYTE
0090 FPNSW .DE $0009 ;FPACC NEXT SIG BYTE
0100 FPMSW .DE $000A ;FPACC MOST SIG BYTE
0110 FPACCE .DE $000B ;FPACC EXPONENT
0120 NUM1 .DS 1 ;LO BYTE OF NUMBER TO BE CONVERTED
0130 NUM2 .DS 1 ;MIDDLE BYTE OF NUMBER TO BE CONVERTED
0140 NUM3 .DS 1 ;HIGH BYTE OF NUMBER TO BE CONVERTED
0150 PAGE0 .DE $0000
0160 FPLSWE .DE $0007
0170 COMPLM .DE $2018 ;COMPLEMENT SUBROUTINE
0180 ROTATL .DE $2000 ;ROTATE LEFT SUBROUTINE
0190 ROTATR .DE $200C ;ROTATE RIGHT SUBROUTINE
0200 ;
0210 CONVER LDA NUM1 ;LOAD THE NUMBER TO
0220 STA FPLSW ; BE CONVERTED INTO
0230 LDA NUM2 ; THE FLOATING POINT
0240 STA FPNSW ; ACCUMULATOR
0250 LDA NUM3
0260 STA FPMSW
0270 LDA #23
0280 STA FPACCE
0290 ;
0300 FPNORM LDX #TSIGN ;SET POINTER TO SIGN REGISTER
0310 LDA FPMSW ;FETCH FPACC MS BYTE
0320 BMI ACCMIN ;IF NEGATIVE- BRANCH
0330 LDY #300 ;IF POS, CLEAR SIGN REGISTER
0340 TYA
0350 STA PAGE0,X ; BY STORING ZERO
0360 JMP ACZERT ;THEN TEST IF FPACC=0
0370 ACCMIN STA PAGE0,X ;SET SUGN INDICATOR IF MINUS
0380 LDY #304 ;SET PRECISION COUNTER
0390 LDX #FPLSWE ;SET POINTER TO FPACC LS BYTE-1
0400 JSR COMPLM ;TWO'S COMPLEMENT FPACC
0410 ACZERT LDX #FPMSW ;SET POINTER TO FPACC MS BYTE
0420 LDY #304 ;SET PRECISION COUNTER
0430 LOOK0 LDA PAGE0,X ;SEE IF FPACC=0
0440 BNE ACNONZ ;BRANCH IF NONZERO
0450 DEX ;DECREMENT INDEX POINTER
0460 DEY ;DECEMENT BYTE COUNTER
0470 BNE LOOK0 ;IF COUNTER NOT ZERO, CONTINUE

```

```

0480      STY FPACCE      ;FPACC=0, CLEAR EXPONENT TOO
0490  NORMEX      PTS      ;EXIT NORMALIZATION ROUTINE
0500  ACNONZ      LDX #FPLSWE      ;SET POINTER TO FPACC LS BYTE-1
0510      LDY #04          ;SET PRECISION COUNTER
0520      JSR ROTATL      ;ROTATE FPACC LEFT
0530      LDA PAGE0,X      ;SEE IF ONE IN MS BIT
0540      BMI ACCSET      ;IF MINUS, PROPERLY JUSTIFIED
0550      DEC FPACCE      ;IF POSITIVE, DECREMENT FPACC EXPONENT
0560      JMP ACNONZ      ;CONTINUE ROTATING
0570  ACCSET      LDX #FPMEM      ;SET POINTER TO FPACC MS BYTE
0580      LDY #103          ;SET PRECISION COUNTER
0590      JSR ROTATR      ;COMPENSATING ROTATE RIGHT FPACC
0600      LDA TSIGN      ;IS ORIGINAL SIGN POSITIVE
0610      BEQ NORMEX      ;YES, SIMPLY RETURN
0620      LDY #103          ;WITH POINTER AT LS BYTE
0630      ;SET PRECISION COUNTER
0640      JMP COMPLEM      ;RESTORE FPACC TO NEG AND RETURN
0650      .EN

```

Module: Floating Point Divide

>LOAD FPDIV

>PR

```

0010  ;FLOATING POINT DIVIDE
0020  ; DIVIDENT=FPACC
0030  ; DIVISOR=FFOP
0040  ; RESULT=FPACC
0050      .BA $2260
0060      .DS
0080  OUTCHR      .DE $8A47      ;OUTPUT A CHARACTER
0090  FMPNT      .DE $0000      ;FROM POINTER
0100  TOPNT      .DE $0002      ;TO POINTER
0110  CNTR      .DE $0004      ;COUNTER STORAGE
0120  SIGNS      .DE $0006      ;SIGN INDICATOR
0130  FPLSWE      .DE $0007      ;FPACC EXTENSION
0140  FPLSW      .DE $0008      ;FPACC LEAST SIG BYTE
0150  FPNBW      .DE $0009      ;FPACC NEXT SIG BYTE
0160  FPMBW      .DE $000A      ;FPACC MOST SIG BYTE
0170  FPACCE      .DE $000B      ;FPACC EXPONENT
0180  FOLSWE      .DE $000F      ;FFOP EXTENSION
0190  FOPLSW      .DE $0010      ;FFOP LEAST SIG BYTE
0200  FOPNSW      .DE $0011      ;FFOP NEXT SIG BYTE
0210  FOPMSW      .DE $0012      ;FFOP MOST SIG BYTE
0220  FOPEXP      .DE $0013      ;FFOP EXPONENT
0230  WORK0      .DE $0014      ;WORK AREA
0240  WORK1      .DE $0015
0250  WORK2      .DE $0016
0260  WORK3      .DE $0017
0270  WORK4      .DE $0018
0280  WORK5      .DE $0019
0290  WORK6      .DE $001A
0300  CKSIGN      .DE $218F      ;CHECK SIGN
0310  MOVIND      .DE $203A      ;MOVE BYTES
0320  ROTL      .DE $2001      ;ROTATE LEFT
0330  ROTATL      .DE $2000      ;ROTATE LEFT
0340  ROTATR      .DE $200C      ;ROTATE RIGHT
0341  COMPLM      .DE $2018      ;TWO'S COMPLEMENT BYTES
0350  FPNORM      .DE $21FA      ;FLOATING POINT NORMALIZATION
0360  ;
0370  FPDIV      JCP CKSIGN      ;CLEAR WORK AREA AND SET SIGNS
0380      LDA FPMBW      ;CHECK FOR DIVIDE BY ZERO
0390      BEQ DERPOP      ;DIVISOR=0, DIVIDE BY ZERO ERROR
0400  SUBEXP      LDA FOPEXP      ;GET DIVIDEND EXPONENT
0410      SEC      ;SET CARRY FOR SUBTRACTION
0420      SBC FPACCE      ;SUBTRACT DIVISOR EXPONENT
0430      STA FPACCE      ;STORE RESULT IN FPACC EXPONENT
0440      INC FPACCE      ;COMPENSATE FOR DIVIDE ALGORITHM
0450  SETDCT      LDA #$17      ;SET BIT COUNTER STORAGE
0460      STA CNTR      ; TO 17 HEXADEGIMAL
0470  DIVIDE      JSR SETSUB      ;SUBTRACT DIVISOR FROM DIVIDEND

```

```

0480      BMI NOGO      ;IF RESULT MINUS, ROTATE ZERO
0490      ; IN QUOTIENT
0500      LDX #FOPLSM   ;SET POINTER TO DIVIDEND
0510      STX TOPNT     ;STORE IN TOPNT
0520      LDX #WORK0    ;SET POINTER TO QUOTIENT
0530      STX FMPNT     ;STORE IN FMPNT
0540      LDX #S03      ;SET PRECISION COUNTER
0550      JIR MOVIND     ;MOVE QUOTIENT TO DIVIDEND
0560      SEC           ;SET CARRY FOR POSITIVE RESULTS
0570      JMP QUOPOT     ;ROTATE INTO QUOTIENT
0580  DERROR  LDA #BF     ;SET ASCII FOR ?
0590      JIR OUTCHR
0600      RTS
0610  NOGO    CLC        ;NEGATIVE RESULT, CLEAR CARRY
0620  QUOPOT  LDX #WORK4   ;SET POINTER TO QUOTIENT LS BYTE
0630      LDY #S3        ;SET PRECISION COUNT
0640      JSR ROTL        ;ROTATE CARRY INTO LSB OF QUOTIENT
0650      LDX #FOPLSM    ;SET POINTER TO DIVIDEND LS BYTE
0660      LDY #S3        ;SET PRECISION COUNTER
0670      JSR ROTATE     ;ROTATE DIVIDEND LEFT
0680      DEC CNTR       ;DECREMENT BIT COUNTER
0690      BNE DIVIDE     ;IF NOT ZERO, CONTINUE
0700      JIR SETSUB     ;DO ONE MORE FOR ROUNDING
0710      BMI DVEXIT     ;IF MINUS, NO ROUNDING
0720      LDA #S1        ;IF 0 OR +, ADD 1 TO 23RD BIT
0730      CLC           ;CLEAR CARRY FOR ADDITION
0740      ADC WORK4       ;ROUND LS BYTE OF QUOTIENT
0750      STA WORK4       ;RESTORE BYTE IN WORK AREA
0760      LDA #S0        ;CLEAR A, NOT THE CARRY
0770      ADC WORK5       ;ADD CARRY TO 2ND BYTE OF QUOTIENT
0780      STA WORK5       ;STORE RESULT
0790      LDA #S0        ;CLEAR A, NOT THE CARRY
0800      ADC WORK6       ;ADD CARRY TO MS BYTE
0810      ; OF QUOTIENT
0820      STA WORK6       ;STORE RESULT
0830      BPL DVEXIT     ;IF MSB=0, EXIT
0840      LDX #WORK6     ;ELSE PREPARE TO ROTATE RIGHT
0850      LDY #S3        ;SET PRECISION COUNTER
0860      JSR ROTATE     ;CLEAR SIGN BIT COUNTER
0870      INC FPACCE     ;COMPENSATE EXP FOR ROTATE
0880  DVEXIT  LDX #FPLSWE  ;SET POINTER TO FPACC
0890      STX TOPNT     ;STORE IN TOPNT
0900      LDX #WORK3     ;SET POINTER TO QUOTIENT
0910      STX FMPNT     ;STORE IN FMPNT
0920      LDX #S4        ;SET PRECISION COUNTER
0930      JMP EXMLDV     ;MOVE QUOTIENT TO FPACC
0940  SETSUB  LDX #WORK0    ;SET POINTER TO WORK AREA
0950      STX TOPNT     ;STORE IN TOPNT
0960      LDX #FPLSM     ;SET POINTER TO FPACC
0970      STX FMPNT     ;STORE IN FMPNT
0980      LDX #S3        ;SET PRECISION COUNTER
0990      JSR MOVIND     ;MOVE FPACC TO WORK AREA
1000      LDX #WORK0    ;PREPARE FOR SUBTRACTION
1010      STX TOPNT     ;STORE POINTER TO DIVISOR
1020      LDX #FOPLSM   ;SET POINTER TO FPOP LS BYTE-1

```

```

1030      STX FMPNT      ;STORE POINTER TO DIVIDEND
1040      LDY #50        ;INITIALIZE INDEX POINTER
1050      LDX #3         ;SET PRECISION COUNTER
1060      SEC           ;SET CARRY FOR SUBTRACTION
1070 SUBR1  LDA (FMPNT),Y      ;FETCH FPOP BYTE (DIVIDEND)
1080      SBC (TOPNT),Y      ;SUBTRACT FPACC BYTE (DIVISOR)
1090      STA (TOPNT),Y      ;STORE IN PLACE OF DIVISOR
1100      INY               ;ADVANCE INDEX POINTER
1110      DEX               ;DECREMENT PRECISION COUNTER
1120      BNE SUBR1        ;NOT ZERO, CONTINUE SUBTR
1130      LDA WORK2        ;SET SIGN BIT RESULT IN N FLAG
1140      RTS              ;RETURN WITH FLAG CONDITIONED
1150 EXMLDV  JSR MOVIND     ;MOVE PARTIAL PRODUCT TO FPACC
1160      JSR FPNORM       ;NORMALIZE RESULT
1170      LDA SIGN5        ;GET SIGN STORAGE
1180      BNE MULTEX       ;IF NOT ZERO, SIGN IS POSITIVE
1190      LDX #FPLSW       ;ELSE, SET POINTER TO FPACC LS BYTE
1200      LDY #53         ;SET PRECISION COUNTER
1210      JSR COMPLM       ;COMPLEMENT RESULT
1220 MULTEX  RTS           ;EXIT FPDIV
1230      .EN

```

2 Module: Floating Point Output

LOAD FPOUT

SPP

```

0010  $FLOATING POINT OUTPUT ROUTINE
0020  $ THIS ROUTINE PRINTS OUT A FLOATING POINT BINARY NUMBER
0030  $ IN DECIMAL.
0040      .BA $2340
0050      .DS
0070  IOEXPD      .DE $0027      $I/O EXPONENT STORAGE
0080  FPMEM      .DE $000A      $FPACC MOST SIGNIFICANT BYTE
0090  FPLSM      .DE $0008      $FPACC LEAST SIGNIFICANT BYTE
0100  FMPNT      .DE $0000      $FPOM POINTER
0110  FPACCE      .DE $000B      $FPACC EXPONENT
0120  CNTR      .DE $0004      $COUNTER STORAGE
0130  TOPNT      .DE $0002      $TO POINTER
0140  IDSTR      .DE $0023      $I/O STORAGE
0150  IDSTR1      .DE $0024      $I/O STORAGE
0160  IDSTR2      .DE $0025      $I/O STORAGE
0170  IDSTR3      .DE $0026      $I/O STORAGE
0180  COMPLM      .DE $2018      $COMPLEMENT ROUTINE
0190  ROTATL      .DE $2000      $ROTATE LEFT
0200  ROTATR      .DE $2000      $ROTATE RIGHT
0210  FFX10      .DE $2051      $MULTIPLY FPACC BY 10
0220  FPD10      .DE $206A      $MULTIPLY FPACC BY 0.1
0230  MOVIND      .DE $203A      $TRANSFER A SECTION OF MEMORY
0240  DECBIN      .DE $2083
0250  OUTCHR      .DE $8A47      $OUTPUT A CHARACTER
0260  FPOUT      LDA #$0
0270      STA IOEXPD      $CLEAR DECIMAL EXPONENT STORAGE
0280      LDA FPMEM      $IS VALUE TO BE OUTPUT NEG?
0290      BMI OUTNEG      $YES, MAKE POS AND OUTPUT '-'
0300      LDA #$AB      $ELSE, SET ASCII CODE FOR '+'
0310      BNE AHEAD1      $GO DISPLAY + SIGN
0320  OUTNEG      LDY $FPLSM      $SET POINTER TO LS BYTE OF FPACC
0330      LDY #$3      $SET PRECISION COUNTER
0340      JSR COMPLM      $MAKE FPACC POSITIVE
0350      LDA #$AD      $SET ASCII CODE FOR '-'
0360  AHEAD1      JSR OUTCHR      $OUTPUT SIGN OF RESULT
0370      LDA #$B0      $SET UP ASCII ZERO
0380      JSR OUTCHR      $OUTPUT ZERO TO DISPLAY
0390      LDA $AE      $SET UP ASCII DECIMAL POINT
0400      JSR OUTCHR      $OUTPUT DECIMAL POINT
0410      DEC FPACCE      $DECREMENT FPACC EXPONENT
0420  DECEXT      BPL DECEXD      $IF COMPENSATED, EXP=0
0430      LDA #$4      $EXPONENT NEG, ADD FOUR TO FPACCE
0440      CLC      $CLEAR CARRY FOR ADDITION
0450      ADC FPACCE      $ADD FOUR TO FPACC EXPONENT
0460      BPL DECDUT      $IF EXP=0, OUTPUT MANTISSA
0470      JSR FFX10      $ELSE MULTIPLY MANISSA BY TEN
0480  DECEP      LDA FPACCE      $GET EXPONENT

```

0490		JMP DECEXT	REPEAT TEST FOR X=0
0500	DECEXD	JSR FPD10	MULTIPLY FPACC BY 0.1
0510		JMP DECREP	CHECK STATUS OF FPACC EXPONENT
0520	DECOU	LDX #10CTR	SET UP FOR MOVE OPERATION
0530		STX TOPNT	SET TOPNT TO WORKING REGISTER
0540		LDX #FPLSM	SET POINTER TO FPACC LS BYTE
0550		STX FMPNT	STORE IN FMPNT
0560		LDX #13	SET PRECISION COUNTER
0570		JSR MOVIND	MOVE FPACC TO OUTPUT REGISTERS
0580		LDA #10	
0590		STA 10CTR3	CLEAR OUTPUT REGISTER MS BYTE+1
0600		LDX #10CTR	SET POINTER TO OUTPUT LS BY
0610		LDY #13	SET PRECISION COUNTER
0620		JSR ROTATL	ROTATE TO COMPENSATE FOR SIGN BIT
0630		JSR DECBIN	OUTPUT REGISTER X 10, OVERFLOW
0640	IN MS BYTE +1		
0650	COMPEN	INC FPACCE	INCREMENT FPACC EXPONENT
0660		BEO OUTDIG	OUTPUT A DIGIT WHEN COMPENSATION DONE
0670		LDX #10CTR3	ELSE ROTATE RIGHT TO COMPENSATE
0680		LDY #14	FOR ANY REMAINDER IN BINARY EXP
0690		JSR ROTATR	PERFORM ROTATE RIGHT OPERATION
0700		JMP COMPEN	REPEAT LOOP UNTIL EXP=0
0710	OUTDIG	LDA #17	SET DIGIT COUNTER TO SEVEN
0720		STA CNTR	FOR OUTPUT OPERATION
0730		LDA 10CTR3	FETCH BCD, SEE IF FIRST DIGIT = 0
0740		BEO ZERPDG	YES, CHECK REMAINDER OF DIGITS
0750	OUTDGS	LDA 10CTR3	GET BCD FROM OUTPUT REGISTER
0760		ORA #180	FORM ASCII CODE FOR NUMBERS
0770		JSR OUTCHR	AND OUTPUT BIT
0780	DECRDG	DEC CNTR	DECREMENT DIGIT COUNTER
0790		BEO EXPDUT	=0, DONE OUTPUT EXPONENT
0800		JSR DECBIN	ELSE GET NEXT DIGIT
0810		JMP OUTDGS	FORM ASCII AND OUTPUT
0820	ZERPDG	DEC 10EXPD	DECREMENT EXP FOR SKIPPING DISPLAY
0830		LDA 10CTR2	CHECK IF MANTISSA = 0
0840		BNE DECRDG	IF NOT ZERO, CONTINUE OUTPUT
0850		LDA 10CTR1	
0860		BNE DECRDG	
0870		LDA 10CTR	
0880		BNE DECRDG	
0890		LDA #10	MANTISSA ZERO, CLEAR EXPONENT
0900		STA 10EXPD	
0910		BEO DECRDG	BEFORE FINISHING DISPLAY
0920	EXPDUT	LDA #5C5	SET UP ASCII CODE FOR E
0930		JSR OUTCHR	DISPLAY E FOR EXPONENT
0940		LDA 10EXPD	TEST IF NEGATIVE
0950		BMI EXDUTN	YES, DISPLAY '-' AND NEGATE
0960		LDA #5AB	NO, SET ASCII CODE FOR '+'
0970		JMP AHEAD2	DISPLAY EXPONENT VALUE
0980	EXDUTN	EOB #FF	TWO'S COMPLEMENT EXPONENT
0990		STA 10EXPD	TO MAKE NEG VALUE POS
1000		INC 10EXPD	FOR OUTPUT OF EXPONENT VALUE
1010		LDA #5AD	SET ASCII CODE FOR '-'
1020	AHEAD2	JSR OUTCHR	OUTPUT SIGN OF EXPONENT
1030		LDY #50	CLEAR TEN'S COUNTER

1040		LDA IDEXPD	:FETCH EXPONENT
50	SUB12	SEC	:SET CARRY FOR SUBTRACTION
1060		SBC #10A	:SUBTRACT TEN'S FROM EXPONENT
1070		BMI TOMUCH	:IF MINUS, READY FOR OUTPUT
1080		STA IDEXPD	:STORE POSITIVE RESULT
1090		INY	:ADVANCE TEN'S COUNTER
1100		JMP SUB12	:CONTINUE SUBTRACTION
1110	TOMUCH	TYA	:PUT MS. DIGIT INTO A
1120		ORA #80	:FORM ASCII CODE
1130		OUT OUTCHP	:OUTPUT TEN'S DIGIT TO DISPLAY
1140		LDA IDEXPD	:FETCH UNIT'S DIGIT
1150		ORA #80	:FORM ASCII CODE
1160		OUT OUTCHP	:OUTPUT DIGIT
1170		RTS	
1180		.EN	

Module: Convert, Divide and Output Floating Point.

LOAD PRINT

```

SPP
0010      .BR $2430
0020      .DS
0030      :THIS MODULE WILL CONVERT A THREE BYTE DIVISOR AND DIVIDEND
0040      :TO FLOATING POINT FORMAT. DO A FLOATING POINT DIVIDE, AND
0050      :OUTPUT THE RESULT IN DECIMAL FLOATING POINT FORMAT.
0060  DIVIS1  .DS 1      :LO BYTE OF DIVISOR
0070  DIVIS2  .DS 1      :MIDDLE BYTE OF DIVISOR
0080  DIVIS3  .DS 1      :HIGH BYTE OF DIVISOR
0090  DVDND1  .DS 1      :LO BYTE OF DIVIDEND
0100  DVDND2  .DS 1      :MIDDLE BYTE OF DIVIDEND
0110  DVDND3  .DS 1      :HIGH BYTE OF DIVIDEND
0120  CONVE   .DE $21E3   :CONVERT TO FLOATING POINT
0130  FPDIV   .DE $2260   :FLOATING POINT DIVISION
0140  FPQUT   .DE $2340   :OUTPUT FP IN DECIMAL
0150  FPLSW   .DE $0008   :FPACC LEAST SIGNIFICANT BYTE
0160  FPNSW   .DE $0009   :FPACC NEXT SIGNIFICANT BYTE
0170  FPMSW   .DE $000A   :FPACC MOST SIGNIFICANT BYTE
0180  FPACCE  .DE $000B   :FPACC EXPONENT
0190  FOPLSW  .DE $0010   :FPDP LEAST SIGNIFICANT BYTE
0200  FOPNSW  .DE $0011   :FPDP NEXT SIGNIFICANT BYTE
0210  FOPMSW  .DE $0012   :FPDP MOST SIGNIFICANT BYTE
0220  FOPEXP  .DE $0013   :FPDP EXPONENT
0230  NUM1    .DE $21E0
0240  NUM2    .DE $21E1
0250  NUM3    .DE $21E2
0260  :
0270  PRINT   TXA
0280          PHA
0290          TYA
0300          PHA
0310  LDA DVDND1  :PUT DVDND1, 2, AND 3
0320  STA NUM1    : INTO NUM1, 2, AND 3
0330  LDA DVDND2
0340  STA NUM2
0350  LDA DVDND3
0360  STA NUM3
0370  JSR CONVE   :CONVERT DIVIDEND TO FP
0380  LDA FPLSW   :PUT FP RESULT INTO
0390  STA FOPLSW   : FPOP
0400  LDA FPNSW
0410  STA FOPNSW
0420  LDA FPMSW
0430  STA FOPMSW
0440  LDA FPACCE
0450  STA FOPEXP
0460  LDA DIVIS1  :PUT DIVIS1, 2, AND 3
0470  STA NUM1    : INTO NUM1, 2 AND 3

```

0480	LDA DIVIS2	
40	STA NUM2	
0500	LDA DIVIS3	
0510	STA NUM3	
0520	JCR CONVE	:CONVERT DIVISOR TO FP
0530	JCR FPDIV	:FP DIVISION
0540	JCR FPOUT	:PRINT OUT RESULT OF DIVISION
0550	PLA	
0560	TAY	
0570	PLA	
0580	TAX	
0590	RTS	
0600	.EN	

Module: Total Mission Parameters.

>LOAD TOVAL

```

>PP
0010      .BA $2700
0020      .DC
0040  TOTFX1  .DE 1      ;LO BYTE OF TOTAL MISSION FIXATIONS
0050  TOTFX2  .DE 1      ;MID BYTE OF TOTAL MISSION FIXATIONS
0060  TOTFX3  .DE 1      ;HI BYTE OF TOTAL MISSION FIXATIONS
0070  TODWL1  .DE 1      ;LO BYTE OF TOTAL MISSION DWELL TIME
0080  TODWL2  .DE 1      ;MID BYTE OF TOTAL MISSION DWELL TIME
0090  TODWL3  .DE 1      ;HI BYTE OF TOTAL MISSION DWELL TIME
0100  TABLE1 .DE $6000   ;LOCATION OF
0110  TABLE2 .DE $6100   ; FINAL DATA
0120  TABLE3 .DE $6200   ; TABLE
0130  TABLE4 .DE $6300
0140  ;
0150  ;ADD UP THE TOTAL NUMBER OF FIXATIONS AND THE TOTAL
0160  ;MISSION DWELL TIME FOR THE DATA MISSION. THIS IS
0170  ;DONE FOR INSTS 0-25. NOT FOR BLINK, GLITCH AND
0180  ;DATA GONE.
0190  TOVAL   LDA #00      ;INITIALIZE TOTAL FIXATIONS
0200          STA TOTFX1    ;AND TOTAL DWELL TIME
0210          STA TOTFX2
0220          STA TOTFX3
0230          STA TODWL1
0240          STA TODWL2
0250          STA TODWL3
0260          LDA #182      ;LOCATION OF THE LOW BYTE OF THE
0270          ; DWELL TIME FOR INST 1
0280  LOOP1   LDA TABLE1,X  ;ADD THE DWELL TIME
0290          CLC ; FOR AN INSTRUMENT
0300          ADC TODWL1     ; TO THE TOTAL
0310          STA TODWL1     ; MISSION DWELL TIME
0320          INX
0330          LDA TABLE1,X
0340          ADC TODWL2
0350          STA TODWL2
0360          INX
0370          LDA TABLE1,X
0380          ADC TODWL3
0390          STA TODWL3
0400          INX
0410          CLC
0420          LDA TABLE1,X  ;ADD THE NUMBER OF
0430          ADC TOTFX1     ; FIXATIONS FOR AN
0440          STA TOTFX1     ; INSTRUMENT TO THE
0450          INX ; TOTAL MISSION DWELL
0460          LDA TABLE1,X  ; TIME
0470          ADC TOTFX2
0480          STA TOTFX2

```

```

0490      LDA #00
0500      ADC TOTFX3
0510      STA TOTFX3
0520      CPX #1EC      ;APE AT THE END OF TABLE 1?
0530      BEQ TAB2      ;IF 10, GO TO TABLE2
0540      CLC
0550      TXA      ;TRANSFER X TO ACC
0560      ADC #1E      ;GO TO NEXT INST IN TABLE 1
0570      TAX
0580      JMP LOOP1
0590      TAB2      LDX #1C      ;LOOK AT FIRST INST IN TABLE 2
0600      LOOP2      LDA TABLE2,X      ;ADD THE DWELL TIME
0610      CLC      ; FOR AN INSTRUMENT
0620      ADC TODWL1      ; TO THE TOTAL
0630      STA TODWL1      ; MISSION DWELL TIME
0640      INX
0650      LDA TABLE2,X
0660      ADC TODWL2
0670      STA TODWL2
0680      INX
0690      LDA TABLE2,X
0700      ADC TODWL3
0710      STA TODWL3
0720      INX
0730      CLC
0740      LDA TABLE2,X      ;ADD THE NUMBER OF
0750      ADC TOTFX1      ; FIXATIONS FOR AN
0760      STA TOTFX1      ; INSTRUMENT TO THE
0770      INX      ; TOTAL MISSION DWELL
0780      LDA TABLE2,X      ; TIME
0790      ADC TOTFX2
0800      STA TOTFX2
0810      LDA #00
0820      ADC TOTFX3
0830      STA TOTFX3
0840      CPX #1EC      ;APE AT THE END OF TABLE 2?
0850      BEQ TAB3      ;IF 50, GO TO TABLE2
0860      CLC
0870      TXA      ;TRANSFER X TO ACC
0880      ADC #1E      ;GO TO NEXT INST IN TABLE 2
0890      TAX
0900      JMP LOOP2
0910      TAB3      LDX #1C      ;LOOK AT FIRST INST IN TABLE 2
0920      LOOP3      LDA TABLE3,X      ;ADD THE DWELL TIME
0930      CLC      ; FOR AN INSTRUMENT
0940      ADC TODWL1      ; TO THE TOTAL
0950      STA TODWL1      ; MISSION DWELL TIME
0960      INX
0970      LDA TABLE3,X
0980      ADC TODWL2
0990      STA TODWL2
1000      INX
1010      LDA TABLE3,X
1020      ADC TODWL3
1030      STA TODWL3

```

```

1040      INX
1050      CLC
1060      LDA TABLE3,X      ;ADD THE NUMBER OF
1070      ADC TOTFX1      ; FIXATIONS FOR AN
1080      STA TOTFX1      ; INSTRUMENT TO THE
1090      INX      ; TOTAL MISSION DWELL
1100      LDA TABLE3,X      ; TIME
1110      ADC TOTFX2
1120      STA TOTFX2
1130      LDA #00
1140      ADC TOTFX3
1150      STA TOTFX3
1160      CPX #1EC      ;ARE AT THE END OF TABLE 3?
1170      BEQ TAB4      ;IF SO, GO TO TABLE4
1180      CLC
1190      TXA      ;TRANSFER X TO ACC
1200      ADC #1E      ;GO TO NEXT INST IN TABLE 3
1210      TAX
1220      JMP LOOP3
1230  TAB4      LDX #1C      ;LOOK AT FIRST INST IN TABLE 4
1240  LOOP4      LDA TABLE4,X      ;ADD THE DWELL TIME
1250      CLC      ; FOR AN INSTRUMENT
1260      ADC TODML1      ; TO THE TOTAL
1270      STA TODML1      ; MISSION DWELL TIME
1280      INX
1290      LDA TABLE4,X
1300      ADC TODML2
1310      STA TODML2
1320      INX
1330      LDA TABLE4,X
1340      ADC TODML3
1350      STA TODML3
1360      INX
1370      CLC
1380      LDA TABLE4,X      ;ADD THE NUMBER OF
1390      ADC TOTFX1      ; FIXATIONS FOR AN
1400      STA TOTFX1      ; INSTRUMENT TO THE
1410      INX      ; TOTAL MISSION DWELL
1420      LDA TABLE4,X      ; TIME
1430      ADC TOTFX2
1440      STA TOTFX2
1450      LDA #00
1460      ADC TOTFX3
1470      STA TOTFX3
1480      CPX #1EC      ;ARE AT THE END OF TABLE 4?
1490      BEQ DONE      ;IF SO, DONE
1500      CLC
1510      TXA      ;TRANSFER X TO ACC
1520      ADC #1E      ;GO TO NEXT INST IN TABLE 4
1530      TAX
1540      JMP LOOP4
1550  DONE      RTS
1560      .EN

```

. Module: First Table Header.

>LOAD OUTPT

```

$FP
0010      .BA $2850
0020      .OS
0030      ; THIS ROUTINE PRINTS OUT THE FIRST TABLE
0040      ; COLUMN HEADERS, THE TOTAL NUMBER OF FIXATIONS
0050      ; IN THE DATA MISSION AND THE TOTAL MISSION DWELL TIME.
0060      SPACE      .BY ' '
0070      HEAD1      .BY 'INST'
0080      HEAD2      .BY 'TOTAL NUMBER'
0090      HEAD3      .BY 'TOTAL DWELL'
0100      HEAD4      .BY 'MEAN DWELL'
0110      HEAD5      .BY 'PROPORTION OF'
0120      HEAD6      .BY 'PROPORTION OF'
0130      HEAD7      .BY 'NUM'
0140      HEAD8      .BY 'OF FIXATIONS'
0150      HEAD9      .BY 'TIME'
0160      HEAD10     .BY 'TIME'
0170      HEAD11     .BY 'TOTAL TIME'
0180      HEAD12     .BY 'TOTAL FIXATION'
0190      UNDER     .BY ' '
0200      NUM1      .DE $21E0      ;LO BYTE OF NUMBER TO BE CONVERTED TO FP
0210      NUM2      .DE $21E1      ;MIDDLE BYTE OF NUMBER TO BE CONVERTED
0220      NUM3      .DE $21E2      ;HI BYTE OF NUMBER TO BE CONVERTED
0230      TOTFX1     .DE $2700      ;LO BYTE OF TOTAL MISSION FIXATIONS
0240      TOTFX2     .DE $2701      ;MIDDLE BYTE OF TOTAL MISSION FIXATIONS
0250      TOTFX3     .DE $2702      ;HI BYTE OF TOTAL MISSION FIXATIONS
0260      TODWL1     .DE $2703      ;LO BYTE OF TOTAL MISSION DWELL TIME
0270      TODWL2     .DE $2704      ;MIDDLE BYTE OF TOTAL MISSION DWELL TIME
0280      TODWL3     .DE $2705      ;HI BYTE OF TOTAL MISSION DWELL TIME
0290      TOTFX      .BY 'TOTAL NUMBER OF FIXATIONS IN THE DATA MISSION= '
0300      TODWL      .BY 'TOTAL DWELL TIME IN THE DATA MISSION= '
0310      CPLF      .DE $834D      ;CARRIAGE RETURN AND LINE FEED
0311      CONVE      .DE $21E3      ;CONVERT TO FP
0312      FPOUT      .DE $2340      ;PRINT FP RESULT
0320      OUTCHP      .DE $8A47      ;OUTPUT A CHARACTER
0330      OUTPT1     LDY #47
0340      LDX #00
0350      JSR CPLF      ;GO TO NEXT LINE
0360      LOOPA      LDA TOTFX,X      ;PRINT OUT THE
0370      JSR OUTCHP      ; TOTAL NUMBER OF
0380      INX      ; FIXATIONS IN THE
0390      DEY      ;DATA MISSION
0400      BNE LOOPA
0410      LDA SPACE
0420      JSR OUTCHP
0430      LDA TOTFX1
0440      STA NUM1
0450      LDA TOTFX2

```

```

0460      STA NUM2
      70      LDA TOTFX3
0480      STA NUM3
0490      JSR CONVE      ;CONVERT FIXATIONS TO FP
0500      JSR FPOUT      ;PRINT FLOATING POINT RESULT
0510      JSR CPLF      ;GO TO NEXT LINE
0520      LDY #38
0530      LDX #00
0540  LOOPB  LDA TOTDML,X      ;PRINT OUT THE
0550      JSR OUTCHP      ; TOTAL MISSION DWELL
0560      INX      ;TIME
0570      DEY
0580      BNE LOOPB
0590      LDA SPACE
0600      JSR OUTCHP
0610      LDA TODML1
0620      STA NUM1
0630      LDA TODML2
0640      STA NUM2
0650      LDA TODML3
0660      STA NUM3
0670      JSR CONVE      ;CONVERT DWELL TIME TO FP
0680      JSR FPOUT      ;OUTPUT THE RESULT
0690      JSR CPLF
0700      JSR CPLF      ;SKIP A LINE
0710      ;
0720  ;PRINT OUT HEADER
      730      JSR CPLF
0740      LDX #00      ;X IS THE POSITION IN HEAD1-12
0750      LDY #4      ;Y IS THE NUM OF CHARS OR SPACES TO PRINT
0760      JSR HOUT      ;PRINT OUT PORTION OF HEADER
0770      LDY #$02
0780      JSR PRSP      ;PPRINT OUT 2 SPACES
0790      LDY #12
0800      JSR HOUT      ;PPRINT 12 CHARS OF HEADER
0810      LDY #4
0820      JSR PRSP      ;PPRINT OUT 4 SPACES
0830      LDY #11
0840      JSR HOUT      ;PPRINT 11 CHARS OF HEADER
0850      LDY #4
0860      JSR PRSP      ;PRINT 4 SPACES
0870      LDY #10
0880      JSR HOUT      ;PRINT 10 CHARS OF HEADER
0890      LDY #4
0900      JSR PRSP      ;PRINT 4 SPACES
0910      LDY #13
0920      JSR HOUT      ;PRINT 13 CHARS OF HEADER
0930      LDY #2
0940      JSR PRSP      ;PPRINT 2 SPACES
0950      LDY #13
0960      JSR HOUT      ;PRINT 13 CHARS OF HEADER
0970      JSR CPLF      ;GO TO NEXT LINE
0980      LDY #1
0990      JSR PRSP      ;PPRINT 1 SPACE
1000      LDY #3

```

```

1010      JSR HOUT      ;PRINT 3 CHARS OF HEADER
      20      LDY #2
1030      JSR PRSP      ;PRINT 2 SPACES
1040      LDY #12
1050      JSR HOUT      ;PRINT 3 CHARS OF HEADER
1060      LDY #7
1070      JSR PRSP      ;PRINT 7 SPACES
1080      LDY #4
1090      JSR HOUT      ;PRINT 4 CHARS OF HEADER
1100      LDY #11
1110      JSR PRSP      ;PRINT 11 SPACES
1120      LDY #4
1130      JSR HOUT      ;PRINT 4 CHARS OF HEADER
1140      LDY #8
1150      JSR PRSP      ;PRINT 8 SPACES
1160      LDY #10
1170      JSR HOUT      ;PRINT 10 CHARS OF HEADER
1180      LDY #3
1190      JSR PRSP      ;PRINT 3 SPACES
1200      LDY #14
1210      JSR HOUT      ;PRINT 14 CHARS OF HEADER
1220      JSR CRLF
1240      LDY #79
1250  LOOP1  LDA UNDER  ;UNDERLINE THE HEADER
1260      JSR OUTCHR
1270      DEY
1280      BNE LOOP1
      90      JSR CRLF      ;TO NEXT LINE
1300      JSR CRLF      ;TO NEXT LINE
1301      RTS
1310      ; SUBROUTINE TO PRINT Y CHARS FROM HEAD
1320  HOUT      LDA HEAD1,X
1330      JSR OUTCHR
1340      INX
1350      DEY
1360      BNE HOUT
1370      RTS
1380      ; SUBROUTINE TP PRINT OUT Y SPACES
1390  PRSP      LDA SPACE
1400      JSR OUTCHR
1410      DEY
1420      BNE PRSP
1430      RTS
1440      .EN

```

. Module: Output First Table.

>LOAD OUTTB

>PP

```

0010      .BA $2A40
0020      .DE
0030      CRLF      .DE $834D      ;CARRIAGE RETURN AND LINE FEED
0040      OUTCHR    .DE $8A47      ;OUTPUT A CHARACTER
0050      NUMBER    .BY 01 02 03 04 05 06 07 08 09 10 11 12 13 14
0060      NUMB1     .BY 15 16 17 18 19 20 21 22 23 24 25
0070      SPACE     .BY
0080      GLITCH    .BY 'GLIT'
0090      BLINK     .BY 'BLNK'
0100      DATAGN    .BY 'DAGN'
0110      RESTB     .DE $2BF2
0120      LOTAB     .DE $002D
0130      HITAB     .DE $002E
0140      ;
0150      ; PRINT OUT THE INFO FOR EACH INST, GLITCH, BLINK,
0160      ;AND DATA GONE ROW BY ROW
0170      OUTTB     LDY #4
0180              LDX #00
0190      LOOP2     LDA GLITCH,X
0200              JSR OUTCHR
0210              INC
0220              DEY
0230              BNE LOOP2
0240              LDA SPACE      ;PRINT 1 SPACE
0250              JSR OUTCHR
0260              LDX #00
0270              LDA #00
0280              STA LOTAB
0290              LDA #$60
0300              STA HITAB      ;POINT TO TABLE1
0310              JSR RESTB      ;PRINT OUT ROW FOR BLINK
0320              JSR CRLF      ;GO TO NEXT LINE
0330              LDX #00
0340              LDY #4
0350      LOOP3     LDA BLINK,X
0360              JSR OUTCHR      ;PRINT HEADER FOR BLINK
0370              INC
0380              DEY
0390              BNE LOOP3
0400              LDA SPACE
0410              JSR OUTCHR      ;PRINT 1 SPACE
0420              LDX #$22
0430              LDA #00
0440              STA LOTAB
0450              LDA #$60
0460              STA HITAB
0470              JSR RESTB      ;PRINT OUT ROW FOR BLINK

```

```

0480      JSR CRLF      ;GO TO NEXT LINE
0490      LDX #00
0500      LDY #4
0510 LOOP4  LDA DATAGN+X
0520      JSR OUTCHR    ;PRINT HEADER FOR DATA GONE
0530      INX
0540      DEY
0550      BNE LOOP4
0560      LDA SPACE
0570      JSR OUTCHR
0580      LDX #$44
0590      LDA #00
0600      STA LOTAB
0610      LDA #$60
0620      STA HITAB      ;POINT AT TABLE1
0630      JSR RESTB      ;PRINT ROW FOR DATA GONE
0640      JSR CRLF      ;GO TO NEXT LINE
0650      LDX #$66
0660      LDY #00
0670 BACK1  JSR FORMAT    ;PRINT HEADER FOR THE INST
0680      LDA #00
0690      STA LOTAB
0700      LDA #$60
0710      STA HITAB      ;POINT AT TABLE1
0720      JSR RESTB      ;PRINT OUT ROW FOR THAT INST
0730      JSR CRLF
0740      CPX #100
0750      BEQ NEXT2      ;GO TO TABLE2 IF DONE WITH TABLE1
0760      TXA
0770      CLC
0780      ADC #$22      ;GO TO NEXT INST IN TABLE1
0790      TAX
0800      JMP BACK1
0810 NEXT2  LDX #00
0820 BACK2  JSR FORMAT    ;PRINT HEADER FOR THE INST
0830      LDA #00
0840      STA LOTAB
0850      LDA #$61
0860      STA HITAB      ;POINT TO TABLE2
0870      JSR RESTB      ;PRINT OUT ROW FOR AN INSTRUMENT
0880      JSR CRLF
0890      CPX #100
0900      BEQ NEXT3      ;IF DONE WITH TABLE2 GO TO TABLE3
0910      TXA
0920      CLC
0930      ADC #$22      ;GO TO NEXT INST IN TABLE2
0940      TAX
0950      JMP BACK2
0960 NEXT3  LDX #00
0970 BACK3  JSR FORMAT    ;PRINT HEADER FOR THE INST
0980      LDA #00
0990      STA LOTAB
1000      LDA #$62
1010      STA HITAB
1020      JSR RESTB      ;PRINT OUT ROW FOR AN INST

```

```

1030      JSR CPLF
1040      CPX #$00
1050      BEQ NEXT4      ;GO TO TABLE4 IF DONE WITH TABLE3
1060      TXA
1070      CLC
1080      ADC #$22      ;GO TO NEXT INST IN TABLE3
1090      TAX
1100      JMP BACK3
1110  NEXT4      LDY #00
1120  BACK4      JSR FORMAT      ;PRINT OUT HEADER FOR THE INST
1130      LDA #00
1140      STA LOTAB
1150      LDA #$63
1160      STA HITAB
1170      JSR FEETB      ;PRINT OUT ROW FOR AN INST
1180      JSR CPLF
1190      CPX #$00
1200      BEQ DONE      ;DONE WITH ALL FOUR TABLES
1210      TXA
1220      CLC
1230      ADC #$22      ;GO TO NEXT INST IN TABLE4
1240      TAX
1250      JMP BACK4
1260  DONE      RTS
1270  ; ROUTINE TO PRINT THE HEADER FOR EACH INST NUMBER
1280  FORMAT      LDA SPACE
1290      JSR OUTCHR
1300      LDA NUMBER,Y
1310      JSR OUTCHR
1320      INY
1330      LDA NUMBER,Y
1340      JSR OUTCHR
1350      INY
1360      LDA NUMBER,Y
1370      JSR OUTCHR
1380      INY
1390      JSR OUTCHR
1400      RTS
1410      .EN

```

. Module: Output Row of First Table - called by Output First Table module.

>LOAD RESTB

>FP

```

0010      .BA 128E0
0020      .DS
0030  NUM1      .DE 121E0      ;NUMBER TO BE CONVERTED TO FLOATING PT
0040  NUM2      .DE 121E1
0050  NUM3      .DE 121E2
0060  PRINT     .DE 12436      ;ROUTINE TO CONVERT TO FP, DIVIDE, AND PRINT
0070  DIVIS1     .DE 12430      ;LO BYTE OF DIVISOR
0080  DIVIS2     .DE 12431      ;MID BYTE OF DIVISOR
0090  DIVIS3     .DE 12432      ;HI BYET OF DIVISOR
0100  DVDND1     .DE 12433      ;LO BYTE OF DIVIDEND
0110  DVDND2     .DE 12434      ;MID BYTE OF DIVIDEND
0120  DVDND3     .DE 12435      ;HI BYTE OF DIVIDEND
0130  TOTFX1     .DE 12700      ;LO BYTE OF TOTAL MISSION FIXATIONS
0140  TOTFX2     .DE 12701      ;MID BYTE OF TOTAL MISSION FIXATIONS
0150  TOTFX3     .DE 12702      ;HI BYTE OF TOTAL MISSION FIXATIONS
0160  TODML1     .DE 12703      ;LO BYTE OF TOTAL MISSION DWELL TIME
0170  TODML2     .DE 12704      ;MID BYTE OF TOTAL MISSION DWELL TIME
0180  TODML3     .DE 12705      ;HI BYTE OF TOTAL MISSION DWELL TIME
0190  CONVE      .DE 121E3      ;CONVERT TO FP
0200  FPOUT      .DE 12340      ;PRINT FP RESULT
0210  SPACE      .BY
0220  OUTCHR      .DE 12847      ;OUTPUT A CHARACTER
0230  ZEROFX      .DS 1          ;SET TO ZERO IF FIXATIONS=0
0240  ZERODDL      .DS 1          ;SET TO ZERO IF DWELL TIME=0
0250  ZEPDUT      .BY 1+0.0000000E+00
0251  COUNT      .DS 1
0260  ;
0270  ;ROUTINE TO PRINT OUT RESULTS FOR THE INSTS IN TABLE
0280  RESTB      TYA
0290      PHA      ;SAVE Y ON STACK
0300      LDA #1FF
0310      STA ZEROFX
0320      STA ZERODDL
0330      TXA      ;PRINT OUT THE TOTAL
0340      CLC      ;NUMBER OF FIXATIONS
0350      ADC #31
0360      TAY
0370      LDA ($2D),Y      ;LO BYTE OF FIXATIONS
0380      STA NUM1
0390      INY
0400      LDA ($2D),Y
0410      STA NUM2
0420      LDA #00
0430      STA NUM3
0440      LDA NUM1      ;SEE IF NUMBER OF
0450      BNE NOZER1      ; FIXATIONS = 0
0460      LDA NUM2
0470      BNE NOZER1

```

```

0480      LDA NUM3
0490      BNE NOZER1
0500      LDA #00      ;IF SO, PRINT OUT ZEROS
0510      STA ZEROFX
0520      JSR PRZERO
0530      JMP NEXT1
0540 NOZER1    TXA
0550      PHA
0560      TYA
0570      PHA
0580      JSR CONVE
0590      JSR FROUT      ;PRINT RESULT
0600      PLA
0610      TAY
0620      PLA
0630      TAX
0640      LDA SPACE
0650      JSR OUTCHP      ;PRINT A SPACE
0660      TXA      ;PRINT OUT TOTAL
0670      CLC      ; DWELL TIME
0680      ADC #28
0690      TAY
0700      LDA #120*Y      ;DWELL TIME FOR INST
0710      STA NUM1
0720      INY
0730      LDA #120*Y      ;MID BYTE OF DWELL TIME
0740      STA NUM2
0750      INY
0760      LDA #120*Y      ;HI BYTE OF DWELL TIME
0770      STA NUM3
0780      LDA NUM1      ;SEE IF DWELL TIME
0790      BNE NOZER2      ; IS ZERO
0800      LDA NUM2
0810      BNE NOZER2
0820      LDA NUM3
0830      BNE NOZER2
0840      LDA #00      ;IF SO, PRINT OUT ZEROS
0850      STA ZEROFL
0860      JSR PRZERO
0870      JMP NEXT2
0880 NOZER2    TXA
0890      PHA
0900      TYA
0910      PHA
0920      JSR CONVE
0930      JSR FROUT      ;PRINT RESULT
0940      PLA
0950      TAY
0960      PLA
0970      TAX
0980      LDA SPACE
0990      JSR OUTCHP      ;PRINT A SPACE
1000      ;PRINT MEAN DWELL TIME=DWELL TIME/NUM OF FIXATIONS
1010      TXA
1020      CLC

```

```

0050      ADC #28
0060      TAY
0070      LDA ZERO0L
0080      BNE NOZER3    ;JUMP IF RESULT NOT ZERO
0090      LDA ZERO0H
0100      BNE NOZER3
0110      JIP PRZERO    ;PRINT OUT ZEROS
0120      JMP NEXT3
0130      NOZER3      LDA #120.47    ;LOW BYTE OF DWELL TIME
0140      STA DVND01
0150      INY
0160      LDA #120.47    ;MID BYTE OF DWELL TIME
0170      STA DVND02
0180      INY
0190      LDA #120.47    ;HI BYTE OF DWELL TIME
0200      STA DVND03
0210      INY
0220      LDA #120.47    ;LO BYTE OF NUMBER OF FIXATIONS
0230      STA DIV101
0240      INY
0250      LDA #120.47    ;HI BYTE OF NUMBER OF FIXATIONS
0260      STA DIV102
0270      LDA #00
0280      STA DIV103
0290      JIP PRINT    ;DWELL TIME/NUM OF FIXATIONS-
0300      ;OUTPUT RESULT IN FP
0310      NEXT3      LDA SPACE
0320      JIP OUTCHR    ;PRINT A SPACE
0330      ;PRINT PROPORTION OF TOTAL TIME=DWELL TIME/TOTAL MISSION
0340      ;DWELL TIME
0350      TXA
0360      CLC
0370      ADC #28
0380      TAY
0390      LDA ZERO0L
0400      BNE NOZER4
0410      JIP PRZERO
0420      JMP NEXT4
0430      NOZER4      LDA #120.47    ;LO BYTE OF DWELL TIME
0440      STA DVND01
0450      INY
0460      LDA #120.47    ;MID BYTE OF DWELL TIME
0470      STA DVND02
0480      INY
0490      LDA #120.47    ;HI BYTE OF DWELL TIME
0500      STA DVND03
0510      LDA TOTML1    ;PUT TOTAL MISSION
0520      STA DIV101    ; DWELL TIME INTO
0530      LDA TOTML2    ; THE DIVISOR
0540      STA DIV102
0550      LDA TOTML3
0560      STA DIV103
0570      JIP PRINT    ;DWELL TIME/TOTAL MISSION DWELL TIME-
0580      ;RESULT PRINT IN FP FORMAT
0590      NEXT4      LDA SPACE

```

```

1480      JCP OUTCHR      ;PRINT A SPACE
1490      ;PRINT PERCENTAGE OF TOTAL FIXATIONS=NUM OF FIXATIONS/
1500      ;TOTAL MISSION FIXATIONS
1510      TXA
1520      CLC
1530      ADC #31
1540      TAY
1550      LDA ZEROFX
1560      BNE NOZER5
1570      JSP PRZERO
1580      JMP NEXT5
1590  NOZER5      LDA #20,Y      ;LOW BYTE OF NUMBER OF FIXATIONS
1600      STA DVND1
1610      INY
1620      LDA #20,Y      ;HIGH BYTE OF NUMBER OF FIXATIONS
1630      STA DVND2
1640      LDA #00
1650      STA DVND3
1660      LDA TOTFX1      ;PUT THE TOTAL NUMBER OF
1670      STA DIV11      ; FIXATIONS FOR THE
1680      LDA TOTFX2      ; MISSION INTO
1690      STA DIV12      ; THE DIVISOR
1700      LDA TOTFX3
1710      STA DIV13
1720      JCP PRINT      ;FIXATIONS/TOTAL MISSION FIXATION-
1730      ;PRINT RESULT IN FP FORMAT
1740  NEXT5      PLA
1750      TAY      ;RESTORE Y FROM STACK
1760      RTS
1770      ;SUBROUTINE TO PRINT OUT A ZERO RESULT
1780  PRZERO      TXA
1790      PHA      ;SAVE X REGISTER
1800      LDA #14
1810      STA COUNT
1820      LDX #00
1830  LOOP      LDA ZEROUT,X
1840      JCP OUTCHR      ;PRINT A CHAR
1850      INX
1860      DEC COUNT
1870      BNE LOOP
1880      PLA
1890      TAX      ;RESTORE X
1900      RTS
1910      .EN

```

Module: Output Second Table

>LOAD OUTTC

>PP

```

0010      .BA $3500
0020      .DS
0030 TRANS      .BY 'TRANSITIONS BETWEEN TWO INSTRUMENTS'
0040 SOURCE      .BY 'SOURCE'
0050 BLANK       .BY ' '
0060 UNDER      .BY ' '
0070 HEAD1      .BY 'NO DG 01 02 03 04 05 06 07 08 09 10 '
0080 HEAD2      .BY '11 12 '
0090 COL        .BY 'GLBLD0102030405060708091011121314151617181920'
0100 COL1       .BY '2122232425E'
0110 TABLE1    .DE $6000
0120 TABLE2    .DE $6100
0130 TABLE3    .DE $6200
0140 TABLE4    .DE $6300
0150 OUTBYT      .DE $32FA      ;OUTPUT A BYTE
0160 OUTCHR      .DE $8H47      ;OUTPUT A CHARACTER
0170 CRLF        .DE $834D      ;CARRIAGE RETURN AND LINE FEED
0180 COUNT1      .DS 1
0190 COUNT2      .BY 6
0200 BINDEC      .DE $39B1
0210 TEMP        .DS 1
0220 ;
0230 ;PPINT OUT  HEADER FOR THE SECOND TABLE OF RESULTS
0240 OUTTC        LDX #00
0250              LDY #35
0260              JIR CRLF
0270 LOOPC        LDA TRANS,X
0280              JIR OUTCHR      ;PPINT HEADER
0290              INX
0300              DEY
0310              BNE LOOPC
0320              JIR CRLF
0330              JIR CRLF
0340              LDX #00
0350              LDY #32
0360 LOOPA        LDA BLANK
0370              JIR OUTCHR      ;PRINT A SPACE
0380              DEY
0390              BNE LOOPA
0400 LOOPB        LDA SOURCE,X
0410              JIR OUTCHR      ;PRINT 'SOURCE'
0420              INX
0430              DEC COUNT2
0440              BNE LOOPB
0450              JIR CRLF
0460              JIR CRLF      ;SKIP A LINE
0470              LDX #00
0480              LDY #00

```

```

0490 LOOP1      LDA BLANK
0500           JSR OUTCHR      ;PRINT A BLANK
0510           JSR OUTCHR      ;PRINT A BLANK
0520           LDA HEAD1,X
0530           JSR OUTCHR      ;PRINT ONE CHAR OF HEADER
0540           INC
0550           LDA HEAD1,X
0560           JSR OUTCHR      ;PRINT ONE CHAR OF HEADER
0570           INC
0580           LDA HEAD1,X
0590           JSR OUTCHR      ;PRINT OUT ONE CHAR OF HEADER
0600           INC
0610           INC
0620           CPY #14
0630           BNE LOOP1
0640           LDY #00         ;DEFINES LOCATION ON COL
0650           LDX #70
0660           JSR CLRF
0670 LOOP0      LDA UNDER
0680           JSR OUTCHR      ;UNDERLINE THE HEADER
0690           DEX
0700           BNE LOOP0
0710           JSR CLRF
0720           LDX #00
0730 OUTLO1     LDA #13
0740           STA COUNT1
0750           JSR CLRF        ;GO TO NEXT LINE
0760           JSR OUTBG       ;OUTPUT BEGINNING OF ROW
0770 VLOOP1      JSR VOUT1      ;PRINT DATA FOR AN INST
0780           DEC COUNT1
0790           BNE VLOOP1
0800           CPX #109
0810           BEQ NEXT2       ;GO TO NEXT TABLE
0820           TXA
0830           CLC
0840           ADC #21          ;GO TO NEXT INST IN TABLE
0850           TAX             ;GO TO NEXT INST
0860           JMP OUTLO1
0870 NEXT2      LDX #00
0880 OUTLO2     LDA #13
0890           STA COUNT1      ;REINITIALIZE COUNT1
0900           JSR CLRF        ;GO TO NEXT LINE
0910           JSR OUTBG       ;PRINT OUT COLUMN HEADER
0920 VLOOP2      JSR VOUT2      ;PRINT ROW FOR AN INST
0930           DEC COUNT1
0940           BNE VLOOP2
0950           CPX #109
0960           BEQ NEXT3       ;GO TO NEXT TABLE IF DONE WITH THIS ONE
0970           TXA
0980           CLC
0990           ADC #21          ;GO TO NEXT INST IN TABLE
1000           TAX
1010           JMP OUTLO2
1020 NEXT3      LDX #00
1030 OUTLO3     LDA #13

```

```

1040      STA COUNT1      ;REINITIALIZE COUNT1
1050      JSP CRLF        ;GO TO NEXT LINE
1060      JSP OUTB6       ;PRINT OUT COLUMN HEADER
1070  VLOOP3      JSP VOUT3      ;PRINT OUT A ROW FOR AN INST
1080      DEC COUNT1
1090      BNE VLOOP3
1100      CPX #109
1110      BEQ NEXT4      ;GO TO NEXT TABLE IF DONE WITH THIS ONE
1120      TXA
1130      CLC
1140      ADC #21      ;GO TO NEXT INST IN TABLE
1150      TAX
1160      JMP OUTLO3
1170  NEXT4      LDA #00
1180  OUTLO4      LDA #13
1190      STA COUNT1      ;REINITIALIZE COUNT1
1200      JSP CRLF        ;GO TO NEXT LINE
1210      JSP OUTB6       ;PRINT OUT COLUMN HEADER
1220  VLOOP4      JSP VOUT4      ;PRINT ROW FOR AN INST
1230      DEC COUNT1
1240      BNE VLOOP4
1250      CPX #109
1260      BEQ NEXT5      ;DONE WITH THE FOUR TABLES IF DONE WITH THIS ONE
1270      TXA
1280      CLC
1290      ADC #21      ;GO TO NEXT INST IN TABLE
1300      TAX
1310      JMP OUTLO4
1320  NEXT5      RTS
1330      ;
1340  OUTB6      LDA BLANK
1350      JSP OUTCHR      ;PRINT A BLANK
1360      JSP OUTCHR      ;PRINT A BLANK
1370      LDA COL.Y
1380      JSP OUTCHR      ;PRINT OUT COLUMN HEADER
1390      INY
1400      LDA COL.Y
1410      JSP OUTCHR
1420      INY
1430      LDA BLANK
1440      JSP OUTCHR
1450      RTS
1460      ;
1470  VOUT1      LDA TABLE1,X
1480      STA TEMP
1490      TYA
1500      PHA      ;PUT INDEX Y ON STACK
1510      TXA
1520      PHA
1530      LDA TEMP
1540      JSP BINDEC      ;CONVERT NUMBER TO DECIMAL
1550      JSP OUTBYT      ;PRINT HI BYTE OF DECIMAL NO
1560      TYA
1570      JSP OUTBYT      ;PRINT LO BYTE OF DECIMAL NO
1580      LDA BLANK

```

1590		JER OUTCHR	:PRINT A BLANK
1600		PLA	
1610		TAX	
1620		PLA	
1630		TAY	:RESTORE INDEX Y FROM STACK
1640		INX	
1650		RTS	
1660	:		
1670	VOUT2	LDA TABLE2,X	
1680		STA TEMP	
1690		TYA	
1700		PHA	:PUT INDEX Y ON STACK
1710		TXA	
1720		PHA	
1730		LDA TEMP	
1740		JER BINDEC	:CONVERT NUMBER TO DECIMAL
1750		JER OUTBYT	:PRINT HI BYTE OF DECIMAL NO
1760		TYA	
1770		JER OUTBYT	:PRINT LO BYTE OF DECIMAL NO
1780		LDA BLANK	
1790		JER OUTCHR	:PRINT A BLANK
1800		PLA	
1810		TAX	
1820		PLA	
1830		TAY	:RESTORE INDEX Y FROM STACK
1840		INX	
1850		RTS	
1860	:		
1870	VOUT3	LDA TABLE3,X	
1880		STA TEMP	
1890		TYA	
1900		PHA	:PUT INDEX Y ON STACK
1910		TXA	
1920		PHA	
1930		LDA TEMP	
1940		JER BINDEC	:CONVERT NUMBER TO DECIMAL
1950		JER OUTBYT	:PRINT HI BYTE OF DECIMAL NO
1960		TYA	
1970		JER OUTBYT	:PRINT LO BYTE OF DECIMAL NO
1980		LDA BLANK	
1990		JER OUTCHR	:PRINT A BLANK
2000		PLA	
2010		TAX	
2020		PLA	
2030		TAY	:RESTORE INDEX Y FROM STACK
2040		INX	
2050		RTS	
2060	:		
2070	VOUT4	LDA TABLE4,X	
2080		STA TEMP	
2090		TYA	
2100		PHA	:PUT INDEX Y ON STACK
2110		TXA	
2120		PHA	
2130		LDA TEMP	

2140	JSP BINDEC	:CONVERT NUMBER TO DECIMAL
2150	JSP OUTBYT	:PRINT HI BYTE OF DECIMAL NO
2160	TYA	
2170	JSP OUTBYT	:PRINT LO BYTE OF DECIMAL NO
2180	LDA BLANK	
2190	JSP OUTCHP	:PRINT A BLANK
2200	PLA	
2210	TAX	
2220	PLA	
2230	TAY	:PESTORE INDEX Y FROM STACK
2240	INX	
2250	RTS	
2260	.EN	

22. Module: Output Third Table.

>LOAD OUTTD

SPP

```

0010      .BA $3750
0020      .DS
0030      BLANK      .BY
0040      UNDER      .BY
0050      HEAD1      .BY 13 14 15 16 17 18 19 20 21 22 23
0060      HEAD2      .BY 24 25
0070      COL      .BY 16161650102030405060708091011121314151617181920
0080      COL1      .BY 1212232425
0090      TABLE1    .DE $6000
0100      TABLE2    .DE $6100
0110      TABLE3    .DE $6200
0120      TABLE4    .DE $6300
0130      OUTBYT      .DE $82FA      ;OUTPUT A BYTE
0140      OUTCHP      .DE $8A47      ;OUTPUT A CHARACTER
0150      CPLF        .DE $834D      ;CARRIAGE RETURN AND LINE FEED
0160      COUNT1      .DS 1
0170      BINDEC      .DE $39B1
0180      TEMP        .DS 1
0190      ;
0200      ;PRINT OUT HEADER FOR THE SECOND TABLE OF RESULTS
0210      OUTTD        JSR CPLF
0220                  JSR CPLF
0230                  JSR CPLF      ;SKIP A LINE
0240                  LDY #00
0250                  LDY #00
0260      LOOP1        LDA BLANK
0270                  JSR OUTCHP      ;PRINT A BLANK
0280                  JSR OUTCHP      ;PRINT A BLANK
0290                  LDA HEAD1,X
0300                  JSR OUTCHP      ;PRINT ONE CHAR OF HEADER
0310                  INX
0320                  LDA HEAD1,X
0330                  JSR OUTCHP      ;PRINT ONE CHAR OF HEADER
0340                  INX
0350                  LDA HEAD1,X
0360                  JSR OUTCHP      ;PRINT OUT ONE CHAR OF HEADER
0370                  INX
0380                  INY
0390                  CPY #14
0400                  BNE LOOP1
0410                  LDY #00      ;DEFINES LOCATION ON COL
0420                  LDY #70
0430                  JSR CPLF
0440      LOOPU        LDA UNDER
0450                  JSR OUTCHP      ;UNDEPLINE THE HEADER
0460                  DEX
0470                  BNE LOOPU

```

```

0480      JSR CPLF
0490      LDX #100
      00 OUTL01 LDA #13
0510      STA COUNT1
0520      JSR CPLF      ;GO TO NEXT LINE
0530      JSR OUTB6     ;OUTPUT BEGINNING OF ROW
0540 VLOOP1 JSR VOUT1    ;PRINT DATA FOR AN INST
0550      DEC COUNT1
0560      BNE VLOOP1
0570      CPX #1E6
0580      BEQ NEXT2     ;GO TO NEXT TABLE
0590      TXA
0600      CLC
0610      ADC #21        ;GO TO NEXT INST IN TABLE
0620      TAX      ;GO TO NEXT INST
0630      JMP OUTL01
0640 NEXT2  LDX #100
0650 OUTL02 LDA #13
0660      STA COUNT1     ;REINITIALIZE COUNT1
0670      JSR CPLF      ;GO TO NEXT LINE
0680      JSR OUTB6     ;PRINT OUT COLUMN HEADER
0690 VLOOP2 JSR VOUT2    ;PRINT ROW FOR AN INST
0700      DEC COUNT1
0710      BNE VLOOP2
0720      CPX #1F6
0730      BEQ NEXT3     ;GO TO NEXT TABLE IF DONE WITH THIS ONE
0740      TXA
0750      CLC
0760      ADC #21        ;GO TO NEXT INST IN TABLE
0770      TAX
0780      JMP OUTL02
0790 NEXT3  LDX #100
0800 OUTL03 LDA #13
0810      STA COUNT1     ;REINITIALIZE COUNT1
0820      JSR CPLF      ;GO TO NEXT LINE
0830      JSR OUTB6     ;PRINT OUT COLUMN HEADER
0840 VLOOP3 JSR VOUT3    ;PRINT OUT A ROW FOR AN INST
0850      DEC COUNT1
0860      BNE VLOOP3
0870      CPX #1E6
0880      BEQ NEXT4     ;GO TO NEXT TABLE IF DONE WITH THIS ONE
0890      TXA
0900      CLC
0910      ADC #21        ;GO TO NEXT INST IN TABLE
0920      TAX
0930      JMP OUTL03
0940 NEXT4  LDX #100
0950 OUTL04 LDA #13
0960      STA COUNT1     ;REINITIALIZE COUNT1
0970      JSR CPLF      ;GO TO NEXT LINE
0980      JSR OUTB6     ;PRINT OUT COLUMN HEADER
0990 VLOOP4 JSR VOUT4    ;PRINT ROW FOR AN INST
1000      DEC COUNT1
1010      BNE VLOOP4
1020      CPX #1E6

```

```

1030      BEQ NEXT5      ;DONE WITH THE FOUR TABLES IF DONE WITH THIS ONE
1040      TXA
1050      CLC
1060      ADC #21        ;GO TO NEXT INST IN TABLE
1070      TAX
1080      JMP OUTLO4
1090      RTS
1100      ;
1110      OUTB6          LDA BLANK
1120                      JSR OUTCHR      ;PRINT A BLANK
1130                      JSR OUTCHR      ;PRINT A BLANK
1140                      LDA COL,Y
1150                      JSR OUTCHR      ;PRINT OUT COLUMN HEADER
1160                      INY
1170                      LDA COL,Y
1180                      JSR OUTCHR
1190                      INY
1200                      LDA BLANK
1210                      JSR OUTCHR
1220                      RTS
1230      ;
1240      VOUT1          LDA TABLE1,X
1250                      STA TEMP
1260                      TYA
1270                      PHA      ;PUT INDEX Y ON STACK
1280                      TXA
1290                      PHA
1300                      LDA TEMP
1310                      JSR BINDEC      ;CONVERT NUMBER TO DECIMAL
1320                      JSR OUTBYT      ;PRINT HI BYTE OF DECIMAL NO
1330                      TYA
1340                      JSR OUTBYT      ;PRINT LO BYTE OF DECIMAL NO
1350                      LDA BLANK
1360                      JSR OUTCHR      ;PRINT A BLANK
1370                      PLA
1380                      TAX
1390                      PLA
1400                      TAY      ;RESTORE INDEX Y FROM STACK
1410                      INX
1420                      RTS
1430      ;
1440      VOUT2          LDA TABLE2,X
1450                      STA TEMP
1460                      TYA
1470                      PHA      ;PUT INDEX Y ON STACK
1480                      TXA
1490                      PHA
1500                      LDA TEMP
1510                      JSR BINDEC      ;CONVERT NUMBER TO DECIMAL
1520                      JSR OUTBYT      ;PRINT HI BYTE OF DECIMAL NO
1530                      TYA
1540                      JSR OUTBYT      ;PRINT LO BYTE OF DECIMAL NO
1550                      LDA BLANK
1560                      JSR OUTCHR      ;PRINT A BLANK
1570                      PLA

```

```

1580      TAX
1590      PLA
1600      TAY  :PESTORE INDEX Y FROM STACK
1610      INX
1620      RTS
1630  :
1640  VOUT3  LDA TABLE3,X
1650      STA TEMP
1660      TYA
1670      PHA  :PUT INDEX Y ON STACK
1680      TXA
1690      PHA
1700      LDA TEMP
1710      JSP BINDEC  :CONVERT NUMBER TO DECIMAL
1720      JSP OUTBYT  :PRINT HI BYTE OF DECIMAL NO
1730      TYA
1740      JSP OUTBYT  :PRINT LO BYTE OF DECIMAL NO
1750      LDA BLANK
1760      JSP OUTCHR  :PRINT A BLANK
1770      PLA
1780      TAX
1790      PLA
1800      TAY  :PESTORE INDEX Y FROM STACK
1810      INX
1820      RTS
1830  :
1840  VOUT4  LDA TABLE4,X
1850      STA TEMP
1860      TYA
1870      PHA  :PUT INDEX Y ON STACK
1880      TXA
1890      PHA
1900      LDA TEMP
1910      JSP BINDEC  :CONVERT NUMBER TO DECIMAL
1920      JSP OUTBYT  :PRINT HI BYTE OF DECIMAL NO
1930      TYA
1940      JSP OUTBYT  :PRINT LO BYTE OF DECIMAL NO
1950      LDA BLANK
1960      JSP OUTCHR  :PRINT A BLANK
1970      PLA
1980      TAX
1990      PLA
2000      TAY  :PESTORE INDEX Y FROM STACK
2010      INX
2020      RTS
2030      .EN

```

24. Module: Convert From Binary to Decimal - called by Output
Second Table and Output Third Table modules.

>LOAD BINDEC

```

$FF
0010      .BA $39B0
0020      .DE
0030      ; THIS ROUTINE CONVERTS A ONE BYTE HEX NUMBER TO
0040      ; A TWO BYTE DECIMAL NUMBER.  THE HEX NUMBER IS IN THE
0050      ; ACC.  THE HI BYTE OF THE DECIMAL NUMBER IS IN THE
0060      ; ACC AND THE LOW BYTE IN REGISTER Y
0070      TEMP      .DE 1      ;TEMPORARY STORAGE
0080      BINDEC     LDY #$FF      ;START QUOTIENT AT -1
0090      SEC        ;SET CARRY FOR INITIAL SUBTRACTION
0100      D100LP     INY      ;ADD 1 TO QUOTIENT
0110      SBC #100    ;SUBTRACT 100
0120      BCS D100LP  ;BRANCH IF A IS STILL LARGER THAN 100
0130      ADC #100    ;ADD THE LAST 100 BACK
0140      TAX        ;SAVE REMAINDER
0150      TYA
0160      PHA        ;SAVE 100'S DIGIT ON THE STACK
0170      TRA        ;GET REMAINDER
0180      ; CALCULATE 10'S AND 1'S DIGITS.  DIVIDE REMAINDER
0190      ; OF THE 100'S DIGIT BY 10.
0200      ; Y = 10'S DIGIT
0210      ; A = 1'S DIGIT
0220      LDY #$FF      ;START QUOTIENT AT -1
0230      SEC        ;SET CARRY FOR INITIAL SUBTRACTION
0240      D10LP      INY      ;ADD 1 TO QUOTIENT
0250      SBC #10
0260      BCS D10LP    ;BRANCH IF A IS STILL LARGER THAN 10
0270      ADC #10      ;ADD THE LAST 10 BACK
0280      ;COMBINE 1'S AND 10'S DIGITS
0290      STA TEMP
0300      TYA
0310      ASL A
0320      ASL A
0330      ASL A
0340      ASL A      ;MOVE 10'S TO HIGH NIBBLE OF A
0350      ORA TEMP     ;OR IN THE 1'S DIGIT
0360      ;RETURN WITH Y = LOW BYTE A = HIGH BYTE
0370      TAY        ;PLACE IN REG Y
0380      PLA        ;GET 100'S DIGIT
0390      RTS
0400      .EN

```

25. Module: Print Results.

LOAD FINAL

```

0000
0010      .BA $39F0
0020      .DS
0030      ; THIS ROUTINE WILL PRINT OUT THE FINAL THREE TABLES
0040 OUTTC      .DE $3591      ;FIRST TABLE THAT PRINTS OUT TRANSITIONS
0050      ;BETWEEN INSTRUMENTS
0060 OUTTD      .DE $37B7      ;SECOND TABLE THAT PRINTS OUT TRANSITIONS
0070      ;BETWEEN INSTRUMENTS
0080 TOVAL      .DE $2706      ;ADD UP TOTAL MISSION FIXATIONS AND
0090      ;TOTAL MISSION DWELL TIME
0100 OUTPT1     .DE $2915      ;PRINT OUT HEADER FOR FIRST TABLE
0110 OUTTB      .DE $2A98      ;PRINT OUT BODY OF FIRST TABLE
0120 FINAL      JIP TOVAL
0130            JIP OUTPT1
0140            JIP OUTTB
0150            JIP OUTTC
0160            JIP OUTTD
0170
0180      RTS
0190      .EN

```

APPENDIX E
Module Validation Data

This appendix shows the data used to test each module or portion of a module. A "\$" in front of a number means that number is in base 16.

1. Division subroutine for Add Timing

Table 5
Division Validation Data

<u>Dividend</u>	<u>Divisor</u>	<u>Quotient</u>	<u>Remainder</u>
60000	1000	60	0
60000	50	1200	0
50000	300	16	200
60000	90	666	60

2. Module Add Timing

Table 6

Add Timing Validation Data

Number of Sample	Number of Time Intervals		Num of Intervals Since Last Sample
	Hexidecimal	Decimal	
01	00 00 11	17	
02	00 00 23	35	18
03	00 00 35	53	18
04	00 00 48	72	19
05	00 00 5A	90	18
06	00 00 6C	108	18
07	00 00 7E	126	18
08	00 00 91	145	19
09	00 00 A3	163	18
\$0A	00 00 B5	181	18
\$0B	00 00 C6	198	17
\$0C	00 00 DA	218	20
\$0D	00 00 EC	236	18
\$0E	00 00 FE	254	18
\$0F	00 01 0F	271	17
\$10	00 01 21	289	18
\$11	00 01 33	307	18
\$12	00 01 46	326	19
\$13	00 01 58	344	18
\$14	00 01 6A	362	18

3. Module Determine Instrument Number

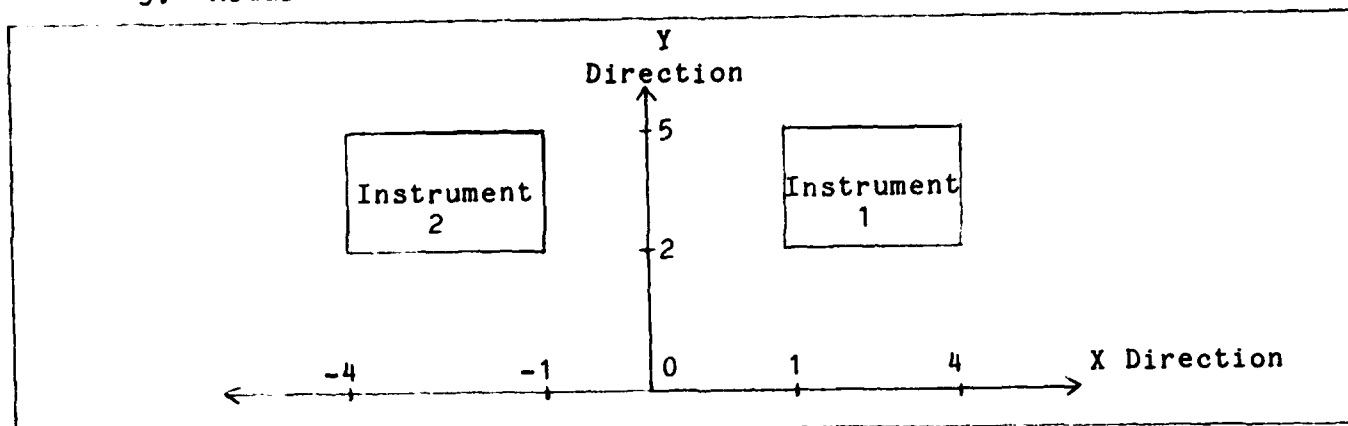


Fig. 6. Test Instrument Boundaries

Table 7

Determine Instrument Number Validation Data

<u>X eye direction</u>	<u>Y eye direction</u>	<u>Instrument Number</u>
2	3	01
5	3	No instrument
6	3	No instrument
-2	3	02
-2	6	No instrument
16383	3	No instrument
1	3	No instrument
4	3	01
-1	3	02
-4	3	No instrument
0	0	No instrument
3	2	No instrument
3	5	01

4. Module Compare Data Samples

Table 8

Compare Data Samples Validation Data

<u>Input Data Samples</u>				<u>Output</u>			
<u>Instrument Number</u>	<u>Data Sample Time(Hexidecimal)</u>			<u>Instrument Number</u>	<u>Total Time On Inst(Hexidecimal)</u>		
1st test:							
01	00	00	11	00	00	00	11
01	00	00	22				
01	00	00	33				
02	00	00	44	01	00	00	33
02	00	00	55				
03	00	00	66	02	00	00	11
01	00	00	77				
01	00	00	88				
03	00	00	99	01	00	00	22
2nd test:							
\$10	00	00	14	00	00	00	14
05	00	01	11	10	00	00	FD
05	00	01	A1				
03	01	CF	00	05	01	CD	EF
03	02	00	00				
03	02	11	01				
\$AA	02	11	A1	03	00	42	A1
\$AA	02	11	C0				
01	0C	D0	54	AA	0A	BE	BC
02	0C	D0	64	01	00	00	10

5. Module Create Table

Table 9

Create Table Validation Data

<u>Inst Number</u>	<u>Total Time On Inst(Hexidecimal)</u>	<u>Last Instrument (Determined by Create Table)</u>
------------------------	--	---

Test 1:

01	00 00 31	00
01	00 00 33	00
00	00 00 B6	01
00	00 00 B8	01
10	00 00 FC	00
13	00 11 00	10
20	00 00 CD	13
10	00 01 00	20

Test 2:

11	00 00 FF	00
11	00 01 11	11
11	00 00 20	11
00	00 00 CC	11
22	00 20 14	00
23	00 20 14	22
24	00 30 14	23
19	00 32 00	24
12	00 A0 00	19

Test 3:

08	00 8C 00	00
03	00 01 00	08
04	00 04 00	03
02	00 02 00	04
05	00 05 01	02
06	00 00 E6	05
07	00 00 F7	06
09	01 01 11	07
14	01 01 00	09

Table 10

Create Table Validation Results

Starting Mem Location	Consecutive Mem contents (if location between 6000 and 63FF is not listed, it is zero)
-----------------------------	---

Test 1:

6000	01	00	00	00	00	00	00
6018	00	00	00	00	31	00	01
6020	00	00	00	01	00	00	00
6038	00	00	00	00	00	00	B6
6040	00	01	00	00	00	01	00
6060	B8	00	00	01	00	00	01
6080	33	00	00	01	00	00	00
61A8	00	00	01	00	00	00	00
61B8	00	00	00	00	00	00	01
61C0	00	00	00	00	00	00	FC
61C8	00	02	00	00	00	00	00
6228	00	00	00	00	01	00	00
6238	00	00	00	00	00	00	CD
6340	00	01	00	00	00	00	00

Test 2:

6008	00	00	00	01	00	00	00
6018	00	00	00	00	20	00	01
6048	00	00	00	00	00	00	01
6060	CC	00	00	01	00	00	00
61C8	00	00	00	00	01	00	00
61D0	00	00	00	00	00	00	01
61E8	10	02	00	02	00	00	00
6210	00	00	00	01	00	00	00
6218	00	00	00	00	00	A0	01
6318	01	00	00	00	00	32	01
6360	00	00	00	00	00	00	01
6380	00	00	14	20	00	01	00
6398	00	00	00	00	00	00	01
63A0	00	00	00	00	14	20	01
63C0	00	01	00	00	00	00	14
63C8	00	01	00	00	00	00	30

Test 3

6088	00	00	00	00	01	00	00	00
60A0	00	00	00	00	00	02	00	01
60B0	00	00	01	00	00	00	00	00
60C0	00	00	00	00	00	00	00	01
60C8	00	01	00	00	00	00	00	01
60E8	00	04	00	01	00	00	00	00
6100	00	00	01	00	00	00	00	00
6118	00	00	00	00	01	05	00	01
6120	00	00	00	00	00	00	00	01
6138	00	00	00	00	00	00	E6	00
6140	00	01	00	00	00	00	00	00
6148	00	00	01	00	00	00	00	00
6160	F7	00	00	01	00	00	01	00
6180	00	00	00	8C	00	01	00	00
6188	00	00	00	00	00	00	00	01
61A0	00	00	00	00	11	01	01	01
6248	00	00	00	00	00	01	00	00
6260	00	01	01	01	00	00	00	00

Appendix F

Simulated Oculometer Input for Subsystem Validation

<u>Track/Out-of Track Status</u>	<u>X Eye Direction (hex)</u>	<u>Y Eye Direction (hex)</u>
Test 1:		
01	00 3C	01 2C
01	00 3C	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 04	01 2C
01	00 00	01 90
01	00 00	01 90
01	00 00	01 90
01	FF 9C	01 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
01	FF 9C	00 3C
00	FF 9C	00 3C
00	FF 9C	00 3C
00	FF 9C	00 3C
01	FF 9C	00 64
01	FF 9C	00 64
01	FF 9C	00 64
01	FF 9C	00 64
01	FE D4	00 C8
01	FE D4	00 C8
01	FE D4	00 C8
01	FE D4	01 2C
01	FE D4	00 2C
01	FE D4	01 2C
01	FE D4	01 2C
01	FE D4	01 2C
01	FE D4	01 2C
01	00 00	01 90
01	00 00	01 90
01	00 00	01 90
01	00 00	01 90

00	90
00	64
00	64
00	64
00	64
00	01

01	90
01	90
01	90
01	90
01	DB
01	DB
01	DB
01	DB
01	DB
00	7D
00	7D
00	7D
00	7D
00	7D
00	7D
00	64
00	64
00	64
00	3C
00	3C
01	2C
01	2C
01	2C
01	2C
01	E0
01	E0
01	E0
01	E0
01	DB
01	DB
01	DB
01	DB

APPENDIX G

Final Data Tables and Outputs

1. The following are the results of Test 1 inputs of Appendix F.

a. Final Data Table in memory locations 6000 (hex) to 6400 (hex).

6000-6400

6000	01	00	00	00	00	00	00	00	00	01
6008	00	00	00	00	00	00	00	00	00	01
6010	00	00	00	00	00	00	00	00	00	01
6018	00	00	00	00	11	00	00	00	01	13
6020	00	00	00	00	00	00	00	00	01	14
6028	00	00	00	00	00	00	00	00	00	14
6030	00	00	00	00	00	00	00	00	00	14
6038	00	00	00	00	00	00	00	37	00	4B
6040	00	01	00	00	00	00	00	00	00	4C
6048	00	00	00	00	00	00	00	00	00	4C
6050	00	00	00	00	00	00	00	00	00	4C
6058	00	00	00	00	00	00	00	00	00	4C
6060	00	00	00	00	00	00	00	01	00	4D
6068	00	00	00	00	00	00	00	00	00	4D
6070	00	00	00	00	00	00	00	00	00	4D
6078	00	00	00	00	00	00	00	00	00	4D
6080	00	00	00	90	00	00	01	00	00	DE
6088	00	00	00	00	00	00	00	00	00	DE
6090	00	00	00	00	00	00	00	00	00	DE
6098	00	00	00	00	00	00	00	00	00	DE
60A0	00	00	00	00	00	00	00	00	00	DE
60A8	00	00	00	00	00	00	00	01	00	DF
60B0	00	00	00	00	00	00	00	00	00	DF
60B8	00	00	00	00	00	00	00	00	00	DF
60C0	00	00	00	00	00	00	00	6E	00	4D
60C8	00	01	00	00	00	00	00	00	00	4E
60D0	00	01	00	00	00	00	00	00	00	4F
60D8	00	00	00	00	00	00	00	00	00	4F
60E0	00	00	00	00	00	00	00	00	00	4F
60E8	39	00	00	01	00	00	00	00	00	89
60F0	00	00	00	00	00	00	00	00	00	89
60F8	00	00	00	00	00	00	00	00	00	89
6100	00	00	00	00	00	00	01	00	00	8A
6108	00	00	00	00	00	00	00	00	00	8A
6110	00	00	00	00	00	00	00	00	00	8A
6118	00	01	00	00	CB	00	00	00	02	58
6120	00	00	00	00	00	00	00	00	00	58
6128	00	00	00	00	00	00	00	00	00	58
6130	00	00	00	00	00	00	00	00	00	58
6138	00	00	00	00	00	00	00	00	00	58
6140	00	00	00	00	00	00	00	00	00	58
6148	00	00	00	00	00	00	00	00	00	58
6150	00	00	00	00	00	00	00	00	00	58
6158	00	00	00	00	00	00	00	00	00	58

135

6:18	00	00	00	00	00	00	00	00.58
6:20	00	00	00	00	00	00	00	00.58
6:28	00	00	00	00	00	00	00	00.58
6:30	00	00	00	00	00	00	00	00.58
6:38	00	00	00	00	00	00	00	00.58
6:40	00	00	00	00	00	00	00	00.58
6:48	00	00	00	00	00	00	00	00.58
6:50	00	00	00	00	00	00	00	00.58
6:58	00	00	00	00	00	00	00	00.58
6:50	00	00	00	00	00	00	00	00.58
6:58	00	00	00	00	00	00	00	00.58
6:70	00	00	00	00	00	00	00	00.58
6:78	00	00	00	00	00	00	00	00.58
6:80	00	00	00	00	00	00	00	00.58
6:88	00	00	00	00	00	00	00	00.58
6:90	00	00	00	00	00	00	00	00.58
6:98	00	00	00	00	00	00	00	00.58
6:00	00	00	00	00	00	00	00	00.58
6:08	00	00	00	00	00	00	00	00.58
6:10	00	00	00	00	00	00	00	00.58
6:18	00	00	00	00	00	00	00	00.58
6:20	00	00	00	00	00	00	00	00.58
6:28	00	00	00	00	00	00	00	00.58
6:30	00	00	00	00	00	00	00	00.58
6:38	00	00	00	00	00	00	00	00.58
6:40	00	00	00	00	00	00	00	00.58
6:48	00	00	00	00	00	00	00	00.58
6:50	00	00	00	00	00	00	00	00.58
6:58	00	00	00	00	00	00	00	00.58
6:00	00	00	00	00	00	00	00	00.58
6:08	00	00	00	00	00	01	00	01.5A
6:10	00	00	00	00	00	00	00	00.5A
6:18	00	00	00	00	00	00	00	00.5A
6:20	00	00	00	00	00	00	00	00.5A
6:28	FB	00	00	02	00	00	00	00.57
6:30	00	00	00	00	00	00	00	00.57
6:38	00	00	00	00	00	00	00	00.57

b. Output of first table.

MAIN FINAL

TOTAL NUMBER OF FIXATIONS IN THE DATA MISSION= +0.7001953E+01
TOTAL DWELL TIME IN THE DATA MISSION= +0.7650545E+03

INIT	TOTAL NUMBER NUM OF FIXATIONS	TOTAL DWELL TIME	MEAN DWELL TIME	PROPORTION OF TOTAL TIME	PROPORTION OF TOTAL FIXATION
GLIT	+0.1945312E+01	+0.1735938E+02	+0.1735938E+02	+0.2269199E-01	+0.2142857E+00
BLNK	+0.1000000E+01	+0.5500001E+02	+0.5521876E+02	+0.724469E-01	+0.1428571E+00
DWGN	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
01	+0.1000000E+01	+0.1440000E+03	+0.1442500E+03	+0.1885621E+00	+0.1428571E+00
02	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
03	+0.1000000E+01	+0.1100000E+03	+0.1103281E+03	+0.1447304E+00	+0.1428571E+00
04	+0.1000000E+01	+0.5699999E+02	+0.5764064E+02	+0.748366E-01	+0.1428571E+00
05	+0.2000000E+01	+0.2029999E+03	+0.1019140E+03	+0.2656454E+00	+0.2857142E+00
06	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
07	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
08	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
09	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
10	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
11	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
12	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
13	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
14	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
15	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
16	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
17	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
18	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
19	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
20	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
21	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
22	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
23	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
24	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
25	+0.2000000E+01	+0.2510000E+03	+0.1256757E+03	+0.3282884E+00	+0.2857142E+00

c. Output of second table.

TRANSITIONS BETWEEN TWO INSTRUMENTS

NO	SOURCE												
	05	01	02	03	04	05	06	07	08	09	10	11	12
GL	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
EL	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000
05	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
02	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
03	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000
04	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000
05	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0000
06	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
07	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
08	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
09	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
11	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
12	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
13	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
4	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
15	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
16	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
17	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
18	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
19	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
21	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
22	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
23	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
24	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
25	0000	0001	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000

d. Output of third table.

[illegible]

2. The following are the results of Test 2 inputs of Appendix F.

a. Final Data Table in memory locations 6000 (hex) to 6400 (hex)

.V 6000,6400

6000	01	00	00	00	00	00	00	00	00.01
6008	00	00	00	00	00	00	00	00	00.01
6010	00	00	00	00	00	00	00	00	00.01
6018	00	01	00	00	36	00	00	02	3A
6020	00	00	00	00	00	00	00	00	00.3A
6028	00	00	00	00	00	00	00	00	00.3A
6030	00	00	00	00	00	00	00	00	00.3A
6038	00	00	00	00	00	00	00	00	00.3A
6040	00	00	00	00	00	00	00	00	00.3A
6048	00	00	00	00	00	00	00	00	00.3A
6050	00	00	00	00	00	00	00	00	00.3A
6058	00	00	00	00	00	00	00	00	00.3A
6060	00	00	00	00	00	00	00	00	00.3A
6068	00	00	00	00	00	00	00	00	00.3A
6070	00	00	00	00	00	00	00	00	00.3A
6078	00	00	00	00	00	00	00	00	00.3A
6080	00	00	00	00	00	00	00	00	00.3A
6088	01	00	00	00	00	00	00	00	00.3B
6090	00	00	00	00	00	00	00	00	00.3B
6098	00	00	00	00	00	00	00	00	00.3B
60A0	00	00	00	00	4A	00	00	01	85
60A8	00	00	00	00	00	00	00	00	00.85
60B0	00	00	00	00	00	00	00	00	00.85
60B8	00	00	00	00	00	00	00	00	00.85
60C0	00	00	00	00	00	00	00	00	00.85
60C8	00	00	00	00	00	00	00	00	00.85
60D0	00	00	00	00	00	00	00	00	00.85
60D8	00	00	00	00	00	00	00	00	00.85
60E0	00	00	00	00	00	00	00	00	00.85
60E8	00	00	00	00	00	00	00	00	00.85
60F0	00	00	00	00	00	00	00	00	00.85
60F8	00	00	00	00	00	00	00	00	00.85
6100	00	00	00	00	00	00	00	00	00.85
6108	00	00	00	00	00	00	00	00	00.85
6110	00	00	00	00	00	00	00	00	00.85
6118	00	00	00	00	00	00	00	00	00.85
6120	00	00	00	00	01	00	00	00	00.86
6128	00	00	00	00	00	00	00	00	00.86
6130	00	00	00	00	00	00	00	00	00.86
6138	00	00	00	01	00	00	F7	00	7E
6140	00	02	00	00	00	00	00	00	00.80
6148	00	00	01	00	00	00	00	00	00.81
6150	00	00	00	00	00	00	00	00	00.81
6158	00	00	00	00	00	00	00	00	00.81

6160 72 00 00 01 00 00 00 00.F4
 6178 00 00 00 00 00 00 00 00.F4
 6180 00 00 00 00 00 00 00 00.F4
 6178 00 00 00 00 00 00 00 00.F4
 6180 00 00 00 00 00 00 00 00.F4
 6188 00 00 00 00 00 00 00 00.F4
 6190 00 00 00 00 00 00 00 00.F4
 6198 00 00 00 00 00 00 00 00.F4
 61A0 00 00 00 00 00 00 00 00.F4
 61A8 00 00 00 00 00 00 00 00.F4
 61B0 00 00 00 00 00 00 00 00.F4
 61B8 00 00 00 00 00 00 00 00.F4
 61C0 00 00 00 00 00 00 00 00.F4
 61C8 00 00 00 00 00 00 00 00.F4
 61D0 00 00 00 00 00 00 00 00.F4
 61D8 00 00 00 00 00 00 00 00.F4
 61E0 00 00 00 00 00 00 00 00.F4
 61E8 00 00 00 00 00 00 00 00.F4
 61F0 00 00 00 00 00 00 00 00.F4
 61F8 00 00 00 00 00 00 00 00.F4
 6200 00 00 00 00 00 00 00 00.F4
 6208 00 00 00 00 00 00 00 00.F4
 6210 00 00 00 00 00 00 00 00.F4
 6218 00 00 00 00 00 00 00 00.F4
 6220 00 00 00 00 00 00 00 00.F4
 6228 00 00 00 00 00 00 00 00.F4
 6230 00 00 00 00 00 00 00 00.F4
 6238 00 00 00 00 00 00 00 00.F4
 6240 00 00 00 00 00 00 00 00.F4
 6248 00 00 00 00 00 00 00 00.F4
 6250 00 00 00 00 00 00 00 00.F4
 6258 00 00 00 00 00 00 00 00.F4
 6260 00 00 00 00 00 00 00 00.F4
 6268 00 00 00 00 00 00 00 00.F4
 6270 00 00 00 00 00 00 00 00.F4
 6278 00 00 00 00 00 00 00 00.F4
 6280 00 00 00 00 00 00 00 00.F4
 6288 00 00 00 00 00 00 00 00.F4
 6290 00 00 00 00 00 00 00 00.F4
 6298 00 00 00 00 00 00 00 00.F4
 62A0 00 00 00 00 00 00 00 00.F4
 62A8 00 00 00 00 00 00 00 00.F4
 62B0 00 00 00 00 00 00 00 00.F4
 62B8 00 00 00 00 00 00 00 00.F4
 62C0 00 00 00 00 00 00 00 00.F4
 62C8 00 00 00 00 00 00 00 00.F4
 62D0 00 00 00 00 00 00 00 00.F4
 62D8 00 00 00 00 00 00 00 00.F4
 62E0 00 00 00 00 00 00 00 00.F4
 62E8 00 00 00 00 00 00 00 00.F4
 62F0 00 00 00 00 00 00 00 00.F4
 62F8 00 00 00 00 00 00 00 00.F4
 6300 00 00 00 00 00 00 00 00.F4
 6308 00 00 00 00 00 00 00 00.F4
 6310 00 00 00 00 00 00 00 00.F4

6	:18	00	00	00	00	00	00	00	.F4
6	:	0	00	00	00	00	00	00	.F4
6	:28	00	00	00	00	00	00	00	.F4
6	:30	00	00	00	00	00	00	00	.F4
6	:38	00	00	00	00	00	00	00	.F4
6	:40	00	00	00	00	00	00	00	.F4
6	:48	00	00	00	00	00	00	00	.F4
6	:50	00	00	00	00	00	00	00	.F4
6	:58	00	00	00	00	00	00	00	.F4
6	:60	00	00	00	00	00	00	00	.F4
6	:68	00	00	00	00	00	00	00	.F4
6	:70	00	00	00	00	00	00	00	.F4
6	:78	00	00	00	00	00	00	00	.F4
6	:80	00	00	00	00	00	00	00	.F4
6	:88	00	00	00	00	00	00	00	.F4
6	:90	00	00	00	00	00	00	00	.F4
6	:98	00	00	00	00	00	00	00	.F4
6	:A0	00	00	00	00	00	00	00	.F4
6	:B8	00	00	00	00	00	00	00	.F4
6	:C0	00	00	00	00	00	00	00	.F4
6	:D8	00	00	00	00	00	00	00	.F4
6	:E0	00	00	00	00	00	00	00	.F4
6	:E8	00	00	00	00	00	00	00	.F4
6	:F0	00	00	00	00	00	00	00	.F4
6	:F8	84	00	00	02	00	00	00	.70
6	:FF	00	00	00	00	00	00	00	.70
:	18	00	00	00	00	00	00	00	.70

b. Output of first table.

AFUN FINAL

TOTAL NUMBER OF FIXATIONS IN THE DATA MISSION= +0.6001952E+01
TOTAL DWELL TIME IN THE DATA MISSION= +0.5667657E+03

INFT	TOTAL NUMBER OF FIXATIONS	TOTAL DWELL TIME	MEAN DWELL TIME	PROPORTION OF TOTAL TIME	PROPORTION OF TOTAL FIXATION
GLIT	+0.2250000E+01	+0.5475000E+02	+0.2710937E+02	+0.964829E-01	+0.3333334E+00
BLNK	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
DATA	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
01	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
02	+0.1000000E+01	+0.7299999E+02	+0.7348437E+02	+0.1291547E+00	+0.1666667E+00
03	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
04	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
05	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
06	+0.2000000E+01	+0.2469999E+03	+0.1237812E+03	+0.4363957E+00	+0.3333334E+00
07	+0.1000000E+01	+0.1139999E+03	+0.1145625E+03	+0.2016894E+00	+0.1666667E+00
08	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
09	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
10	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
11	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
12	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
13	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
14	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
15	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
16	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
17	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
18	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
19	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
20	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
21	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
22	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
23	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
24	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00	+0.0000000E+00
25	+0.2000000E+01	+0.1320000E+03	+0.6649219E+02	+0.2342645E+00	+0.3333334E+00

Output of second table.

TRANSITIONS BETWEEN TWO INSTRUMENTS

NO	SOURCE												
	D5	01	02	03	04	05	06	07	08	09	10	11	12
01	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
02	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
03	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
04	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
05	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
06	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
07	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
08	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
09	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
11	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
12	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
13	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
14	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
15	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
16	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
17	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
18	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
19	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
21	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
22	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
23	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
24	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
25	0000	0000	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000

J. Output of third table.

NO	13	14	15	16	17	18	19	20	21	22	23	24	25
01	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001
02	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
03	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
04	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
05	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
06	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001
07	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
08	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
09	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
11	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
12	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
13	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
14	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
15	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
16	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
17	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
18	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
19	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
21	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
22	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
23	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
24	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
25	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0001

VITA

Nancy Lillian Wood was born on 13 May 1955 in Bronx, New York. She graduated from high school in Theills, New York in 1973 and attended the University of New Hampshire from which she received the degree of Bachelor of Electrical Engineering in December 1977. Upon graduation, she received a commission in the USAF through the ROTC program. She entered active duty in January 1978 and was assigned to the 475th Test Squadron, Tyndall AFB, Florida. There, she was an engineering project officer and a flight test engineer, flying in F-101, F-106, T-33, and CH-3 aircraft. She was assigned to the Test and Evaluation branch of the Air Defense Weapons Center, Tyndall AFB, Florida, Sep 80, as a project Engineer. She was assigned to the school of Engineering, Air Force Institute of Technology, in June 1981. She is a member of IEEE, Tau Beta Phi, and Eta Kappa Nu.

Permanent address: 29 Fay Rd

New City, New York

10923

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/82D-72	2. GOVT ACCESSION NO. AD-A124700	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DEVELOPMENT OF AN OCULOMETER DATA COLLECTION SUBSYSTEM		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Nancy L. Wood Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Flight Dynamics Laboratory (AFWAL/FIGD) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December, 1982
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES 146
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release: 1AW AFR 190-17. <i>Lyne E. Wolaver</i> LYNE E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology (AIG) Wright-Patterson AFB OH 45433 4 JAN 1983		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Oculometer SYM-1 6502 Assembly Language		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A SYM-1 microprocessor with dual 5-1/4 inch disk drives was used to develop software to gather and reduce data from a Cubic-Foot Remote Oculometer built by Honeywell, Inc. The primary function of the oculometer is to measure the look direction of a pilot's eye in a ground cockpit simulator. The output of the oculometer used for this effort is eye		

DD FORM 1473

1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

lookpoint in azimuth and elevation and whether the oculometer is tracking the eye or not. The line-of-sight measurement covers a viewing field of plus and minus 30 degrees in azimuth and zero to plus 30 degrees in elevation. This viewing field is broken into instruments whose boundaries are defined by the data collection subsystem.

The following performance measures are printed out at the end of the data mission.

1. Total dwell time on each instrument.
2. Mean dwell time on each instrument.
3. Proportion of dwell time on each instrument.
4. Proportion of fixations on each instrument.
5. Transition probability from one instrument to another.
6. Number of fixes per minute for each instrument.

The software for the SYM-1 was developed modularly with each module tested separately and then the whole subsystem tested. Simulated oculometer data was used to test the software. The data collection subsystem was designed to run with minimal knowledge and interaction required by the user.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)