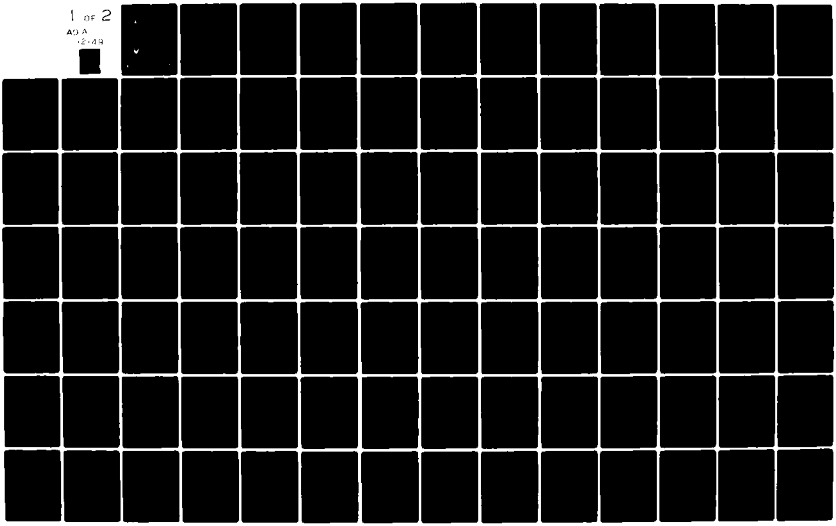
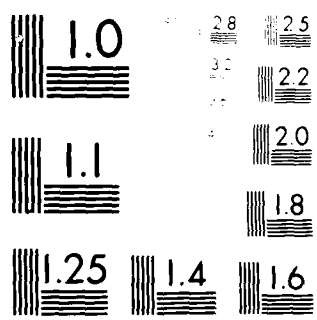
AD-A112 149 ALABAMA UNIV IN HUNTSVILLE SCHOOL OF SCIENCE AND ENG--ETC F/G 14/2
ELECTRONIC TARGET SIGNAL GENERATOR (ETSG) SOFTWARE DEVELOPMENT.(U)
OCT 81 P F PRITCHETT, N A KHEIR DAAH01-81-D-A006
UNCLASSIFIED UAH-296 DRSMI-RD-CR-82-4 NL

1 OF 2
AD A
112149





MICROCOPY RESOLUTION TEST CHART

TECHNICAL REPORT RD-CR-82-4

ELECTRONIC TARGET SIGNAL GENERATOR (ETSG) SOFTWARE DEVELOPMENT

Paul F. Pritchett and N. A. Kheir
The University of Alabama in Huntsville
Huntsville, Alabama

October 1981

Approved for public release; distribution unlimited.

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Prepared for:

Systems Simulation and Development Directorate
US Army Missile Laboratory

Contract DAAH01-81-D-A006

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RD-CR-82-4	2. GOVT ACCESSION NO.: AD-A112 149
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): ELECTRONIC TARGET SIGNAL GENERATOR (ETSG) SOFTWARE DEVELOPMENT	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER: UAH Report No. 296
7. AUTHOR(s): Paul F. Pritchett and N. A. Kheir (Principal Investigator)	8. CONTRACT OR GRANT NUMBER(s): DAAH01-81-D-A006 Delivery Order 0009
9. PERFORMING ORGANIZATION NAME AND ADDRESS: School of Science and Engineering, The University of Alabama in Huntsville, Huntsville, AL 35899	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Commander, US Army Missile Command, ATTN: DRSMI-RPT, Redstone Arsenal, AL 35898	12. REPORT DATE: October 1981
	13. NUMBER OF PAGES: 131
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Commander, US Army Missile Command, ATTN: DRSMI-RD, Redstone Arsenal, AL 35898	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

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Electronic Target Signal Generator
Software Engineering
Simulation
Parameters
Initialization

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

This report documents the study of Electronic Target Signal Generator (ETSG) Software. It is intended to provide a reference for ETSG operation and development.

Chapter one introduces the concept and function of the ETSG. Chapter two outlines the initialization software and chapter three describes the real-time or target CPU firmware. Chapter four contains conclusions and recommendations for future work.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Appendices one through eight contain information about program variables, parameters, subroutines, and algorithms. Appendix nine is a listing of a BASIC program which was developed to aid in doing ETSG-related calculations. Appendices ten through twelve are operating instructions. Appendix thirteen is a listing of ETSG diskette files.

PREFACE

This technical report is prepared by Paul F. Pritchett, Research Associate, under the supervision of Dr. N. A. Kheir, Principal Investigator and Associate Professor of Electrical Engineering, The University of Alabama in Huntsville. The purpose of this report is to provide documentation of Electronic Target Signal Generator (ETSG) software and firmware (software programmed on PROMs).

This documentation effort is in accordance with requirements specified in Delivery Order #0009 of MICOM Contract Number DAAH-01-81-D-A006.

The authors acknowledge with appreciation the assistance and technical support of Don Dublin, contract technical monitor at MICOM, Robert Burt, Research Associate, UAH, Donn Hall, and Don Sprinkle of UAH, and G. R. Loefer, James Randolph, M. J. Sinclair, T. N. Long, and C. E. Barnett of the Georgia Institute of Technology, Engineering Experiment Station, Atlanta, Georgia.

The views and conclusions expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U. S. Army Missile Command.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

APPENDICES (Cont'd)

CHAPTER ONE

ETSG CONCEPT AND FUNCTION

1.0 Introduction

The author's objective in this document is to present a comprehensive survey of the ETSG software and firmware. This study is intended to provide a reference for ETSG operation and to aid in trouble-shooting and the continued development of the ETSG.

Chapter One is a general description of the ETSG with respect to its function, application, attributes, and limitations.

The initialization software is examined in Chapter Two. This chapter contains a glossary of the input data which the system operator supplies to the ETSG. This data consists of seeker parameters for the particular seeker being simulated, target parameters for each target, and field of view background information. Flow-charts and equations, are provided to explain the process by which the Initialization Processor (IP) interface variables are generated from the input data.

After the initialization phase is completed and the ETSG is in "run" mode, the IP interface variables are manipulated by the target Central Processing Unit (CPU). The operations performed by the target CPUs are the topic of Chapter Three. The target CPUs receive dynamic data from the CDC 6600 via the Direct Cell Buffer (DCB). Real-time calculations are performed on the IP interface variables and the Direct Cell (DC) interface variables. Real-time information flow is delineated in the flowcharts at the end of Chapter Three.

Appendices I through VIII contain a glossary of variables, programs, and subroutines. Appendix IX is a listing of the BASIC program used to do number base conversions and to emulate some of the internal processes of the ETSG. A programmed approach to operating the ETSG is found in Appendix X. A summary of commonly used MDOS and 6800 EDITOR commands comprises Appendices XI and XII respectively. Appendix XIII is a list of all ETSG related program files and the discs on which these files reside.

1.1 Overview of the ETSG

The Electronic Target Signal Generator (ETSG) is a specialized hybrid computer which, when given the proper initial and dynamic input data, will generate an analog voltage which simulates the detector output of a variety of electro-optical seekers. Redeye, Stinger, Stinger-POST as well as postulated electro-optical threat seekers may be simulated with the ETSG.

As many as twenty sources of specified shape, size, spatial orientation, spatial position, intensity, and intensity gradient can be created and controlled for the simulation of a particular target/background/countermeasure scenario.

The sources may represent simple targets, complex targets made up of more than one source, infrared flares, and pulsed jammers.

Complex targets such as aircraft can be constructed from five sources, one each for the fuselage, canopy, and plume, and two sources to represent the wings. These five sources are assigned a single control point and a single set of target coordinates, aspect angle, and orientation angle are calculated and transmitted to the ETSG independently for all five sources. The ETSG then uses this data to fly the five sources as one.

Two spectral bands are available for source designation. Band one is unipolar and band two is bipolar.

The ETSG supplies independent outputs for each of the two spectral bands. Output polarity may be reversed by a hardware switch on the final digital to analog converter board which interfaces with the seeker electronics.

Non-expendable pulsed-jammers may be designated as part of a complex target.

Flare sources are controlled independently. Coordinates are calculated by the CDC-6600 with flare initial conditions equal to those of the dispensing aircraft and new positions are calculated from aerodynamic drag equations. Refer to Fig. 1 for a functional block diagram of the simulation subsystems.

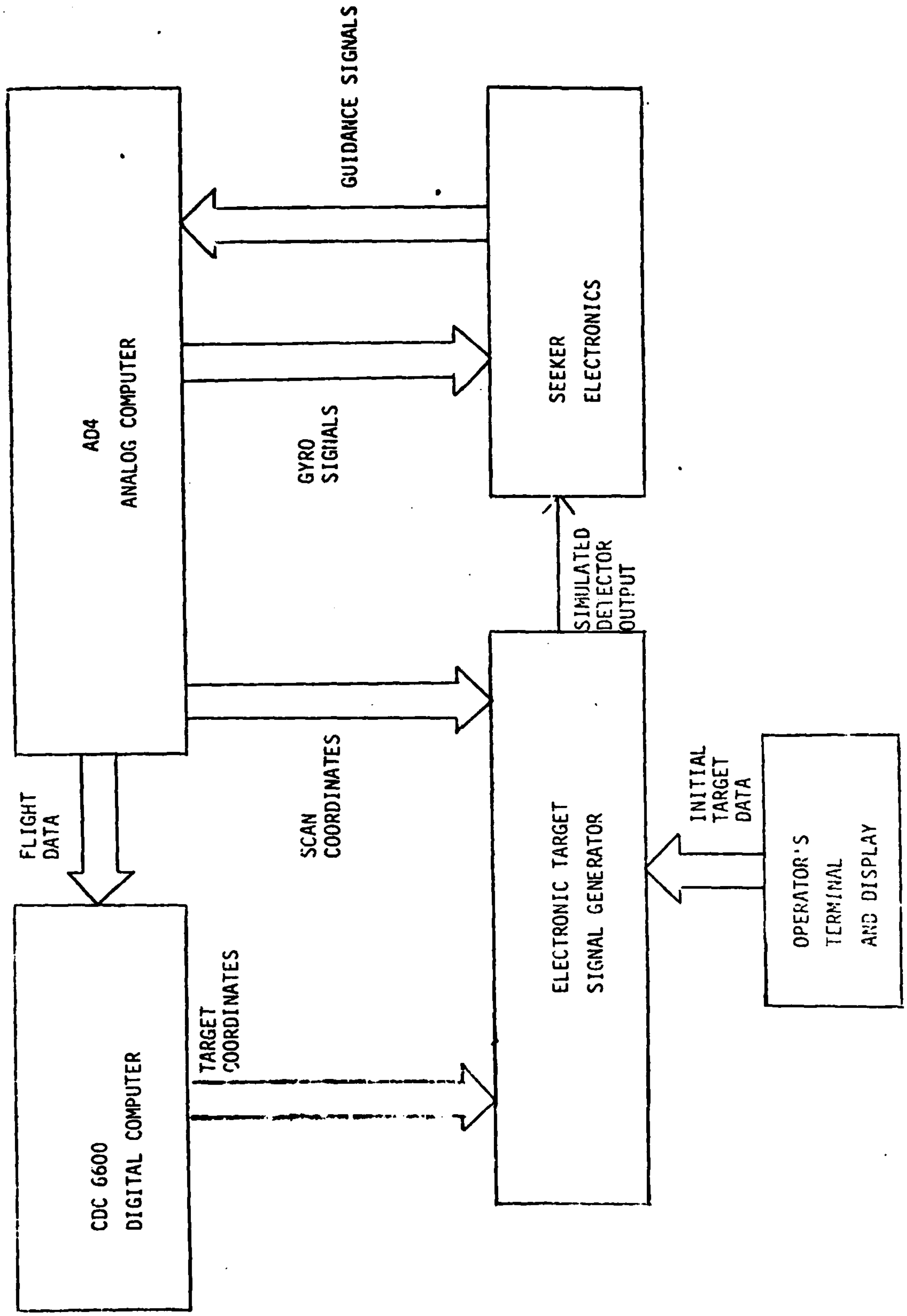


Figure 1: Simulation Subsystems

Each flare is turned on by a command from the CDC-6600 and may be recycled after the flare has dropped beyond the tracking field of view.

The operator's console and display are used to initialize the simulation and to display the dynamic position of the seeker field of view (FOV). The simulation must be initialized for seeker, target, flare, and pulse-jammer parameters. After initialization dynamic target/source data is transferred to the ETSG from the CDC-6600 via the Direct Cell Interface. With this information the ETSG generates a memory map of the seeker image plane. This image plane is then convolved with the seeker scan pattern. For scan patterns other than reticles, scan signals must be supplied from a source external to the ETSG.

The digital signal which results from the convolution of the seeker image plane with the seeker scan pattern is converted to an analog signal, ripple filtered, and output as the simulated detector signal.

This signal passes through the seeker (bread-board) preamplifiers and is processed to generate the gyro procession command.

The AD-4 analog computer uses the procession command to produce guidance commands. The CDC-6600 calculates new air frame coordinates from the guidance commands.

The CDC-6600 communicates the updated target image plane coordinates to the ETSG via the Direct Cell Interface.

For a more detailed description of ETSG hardware subsystem refer to "Electronic Target Signal (ETSG): Hardware Development" [10] written by Robert Burt, Research Associate, The University of Alabama in Huntsville (to appear).

Other documents containing information relative to the ETSG are listed in the reference portion of this document.

CHAPTER TWO

THE INITIALIZATION PROCESSES AND CODES

2.0 Introduction

During initialization of the ETSG, parameters which define a given seeker and particular targets are entered. The ETSG generates a reference image which is stored in Random Access Memory (RAM) for each source.

The target lookup RAM is a 64 x 64 block of 8 bit memory for each target. The values stored in RAM are normalized so as to provide the highest resolution map that the target will require during a given scenerio. This reference is scaled in size, intensity and orientation during the run to simulate the target signature for various combinations of the dynamic parameters.

The flowchart in Figure 1 shows the main programs which perform the initialization process. Each of these programs is discussed individually in the following portions of this chapter. The input parameters used by the initialization processor are defined in Section 2.1.

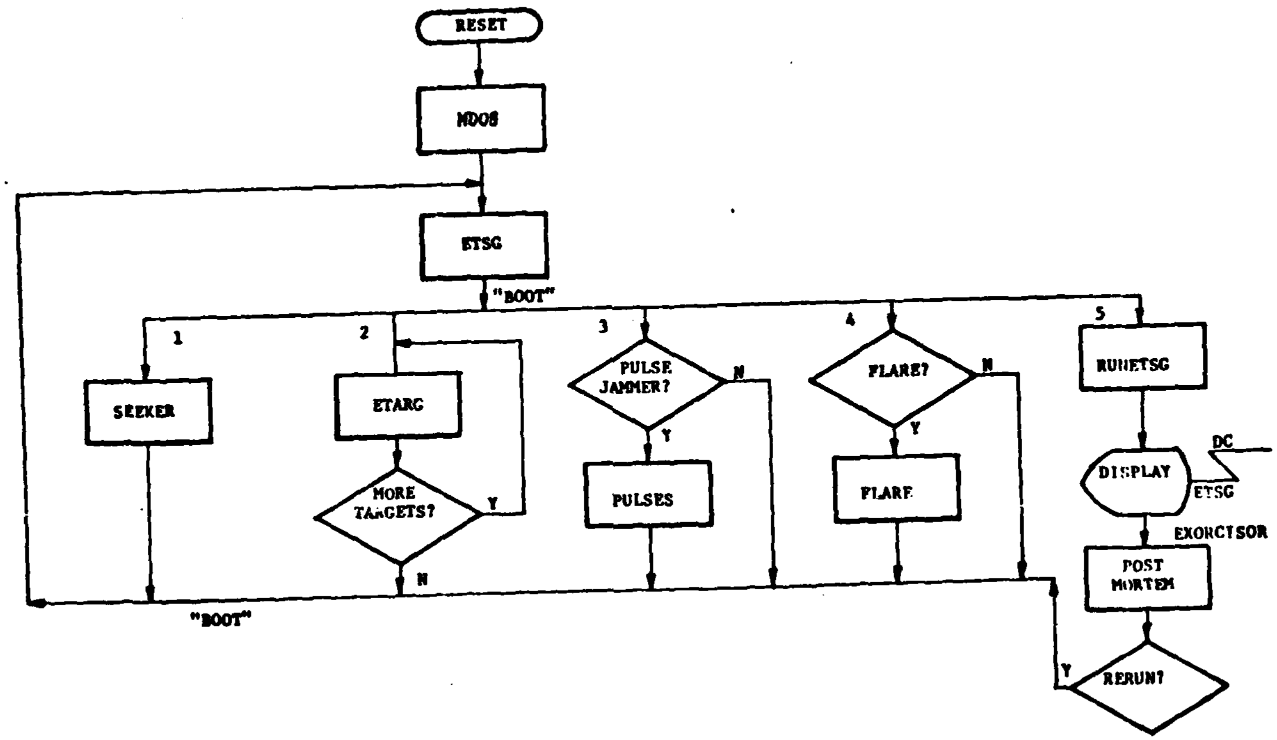


Figure 2.1: Initialization Flowchart

2.1 Simulation Initialization Parameters

The input parameters for the initialization phase are described in Table I. The variables which are internal to the initialization software are listed in Appendices III through VII.

Table I

SIMULATION INITIALIZATION PARAMETERS

INITIAL INPUTS	
SEEKER PARAMETERS	
Type	Rosette, Conical or Center Spun
FOV	Scaled to IFOV
Blur	0.5 mrad. Minimum
NEFD	Any Value
SNR for Track	1 to 10^{10}
Reticle Scan Rate	100 rps $\pm$ 20 rps
System Responsivity	Any Value
SOURCE PARAMETERS	
Shape	Elliptical, Rectangular, or Triangular
Size	Any Size (Linear Dimensions)
Aspect Ratio	1:1 to 32:1
Intensity Gradients	Programmable
Spectral Bands	Any Two
Intensity Polarity	Plus or Minus
Programmable Intensity	Complex
Maximum Range	Meters
Minimum Range	Meters
PULSE JAMMER	
Rep Rate	20 kHz Maximum
Sweep Time	Scan Rate $\pm$ 20%
Duty Cycle	Maximum 50%
Period	1.6 Sec Maximum
FLARES	
Intensity vs. Time	20 Seconds Maximum

2.2.1 M6800 Diskette Operating System (MDOS)

The M6800 Diskette Operating System (MDOS) is an interactive operating system that obtains commands from the system console. These commands are used to move data on the diskette, to process data, or to activate user-written processes from diskette.

In MDOS, a diskette file is a set of related information that is recorded more or less contigously on the diskette. The information can be actual machine instructions that comprise a command or a user program. The information can also be textual data, object program data, or any of the forms described in the following discussion of file name conventions.

The standard format for specifying file names, suffixes, and logical unit numbers is:

< file name > . < suffix > : < logical unit number >

where the period (.) and colon (:) serve to delimit the start of the suffix and logical unit number fields, respectively.

Logical unit numbers identify the drive that contains the file. Since each diskette carries with it its own directory, different files with identical names and suffixes can reside on different diskettes. The following is a list of suffixes and the file type specified by each.

Suffix	Implied Meaning
AL	Assembly listing file
CF	Chain Procedural file
CM	Command file
ED	EDOS - converted file
LO	Loadable memory - image file
LX	EXbug loadable file
RO	Relocatable object file
SA	ASCII source file
SY	Internally - used system file

To initialize MDOS power must first be applied to the EXORciser and to the diskette drive unit. No diskette should be in the drive while power is being turned on or off on either the drive or the EXORciser. Once the power is on, the following steps must be followed:

1. EXbug must be initialized and configured for the proper speed of the system console. If power has been turned on for the first time, EXbug initialization is automatically performed by the power-up interrupt service routine in EXbug. If power is already on and MDOS is to be re-initialized, then either the ABORT or RESTART pushbuttons on the EXORcisers front panel must be depressed to initialize EXbug. The prompt "EXBUG V. R." will be displayed by EXbug indicating it is waiting for operator input. "V" indicates the version and "R" the revision number of the EXbug monitor in the system.

2. An MDOS diskette (one shipped from Motorola or one that has been properly prepared by the user must be placed in drive zero. The door on the drive unit must then be closed in order for the diskette to begin rotating.

3. The EXbug I command "MAID" must be entered. An asterisk (*) prompt will be displayed once MAID has been activated.

4. The MAID command "E800;G" must be entered. This command will give control to the diskette controller at the specified address. The controller will initialize the drive electronics and then proceed to read the Bootblock into memory. Once the Bootblock has been loaded, control is transferred to it. The Bootblock will then attempt to load into memory the remainder of the resident operating system.

During ETSG initialization the ETSG Supervisory Program is executed from MDOS by typing ETSG and a carriage return at the system console.

2.2.2 ETSG Supervisory Program

The ETSG Supervisory Program is the main driver for all the ETSG software. It initializes all hardware and controls the flow of all ETSG software execution. The ETSG main driver calls the subroutine, CKINIT, to perform a hardware check and if necessary, hardware initialization.

CKINIT checks the value stored in the Peripheral Interface Adapter (PIA) at the extended memory address $CBF8 ($ indicates a hexadecimal number, i.e. base 16). If the value is zero, then it is assumed that a power up restart has been performed, a power failure has occurred, or a hardware abort has occurred. The PIAs initialized by CKINIT and the default values for these PIAs are shown in Table II.

After CKINIT the ETSG driver checks the system error flag. Based on this information and the operator's response, a decision is made in reference to these four options:

1. Initialize new system.
2. Perform error restart.
3. Restart with previous targets.
4. Continue initialization process with present system.

Then the initialization sequence continues either in the "auto sequence" or "manual select" mode, depending on the operator's preference. Each phase of initialization is handled by a different program. "Boot" transfers control from each program to the other. The flow chart in Figure 2 gives a detailed description of the ETSG Supervisory Program.

As one can see from the flow chart the next program in the initialization sequence is SEEK.

TABLE II PIAS TO INITIALIZE

		PIA Address	PIA Initialization Values		Default Values
TPIAS	PIA	$CBAO,	$FF,	$04,	$00
	PIA	$CBA2,	$FF,	$04,	$00
	PIA	$CBA4,	$FF,	$04,	$00
	PIA	$CBA6,	$FF,	$04,	$00
	PIA	$CBAD,	$00,	$04,	$00
	PIA	$CBAE,	$00,	$04,	$00
	PIA	$CBBO,	$FF,	$04,	$00
	PIA	$CBB2,	$FF,	$04,	$00
	PIA	$CBB4,	$FF,	$04,	$00
	PIA	$CBB6,	$FF,	$04,	$00
	PIA	$CBB8,	$FF,	$04,	$00
	PIA	$CBBA,	$FF,	$04,	$00
	PIA	$CBBC,	$FF,	$06,	$00
	PIA	$CBBE,	$FF,	$04,	$00
	PIA	$CBCO,	$FF,	$06,	$00
	PIA	$CBC2,	$FF,	$04,	$00
	PIA	$CBC4,	$FF,	$04,	$00
	PIA	$CBC6,	$FF,	$04,	$00
	PIA	$CBC8,	$FF,	$04,	$00
	PIA	$CBCA,	$FF,	$04,	$00
	PIA	$CBCC,	$FF,	$06,	$00
	PIA	$CBC#,	$FF,	$04,	$00
	PIA	$CBDO,	$FF,	$06,	$01
	PIA	$CBD2,	$FF,	$06,	$8E
	PIA	$CBD4,	$FF,	$04,	$01
	PIA	$CBD6,	$FF,	$04,	$3E
	PIA	$CBDB,	$00,	$00,	$00
	PIA	$CBDA,	$00,	$00,	$00
	PIA	$CBDC,	$00,	$00,	$00
	PIA	$CBDE,	$00,	$00,	$00
	PIA	$CBF8,	$FF,	$04,	$00
	PIA	$CBFA,	$FF,	$04,	$FF

TABLE II (CONT'D)

	PIA Address	PIA Initialization Values		Default Values
PIA	$CBFC,	$FF,	$04,	$00
PIA	$CBFE,	$FF,	$04,	$00
PIA	$CEEC,	$FF,	$04,	$00
PIA	$CEEE,	$FF,	$04,	$00
PIA	$CFF0,	$0F,	$04,	$00
PIA	$CEF2	$0F,	$04,	$00
PIA	$CEF4,	$FF,	$04,	$00
PIA	$CEFB,	$FF,	$04,	$00
PIA	$0000,	$00,	$00,	$00

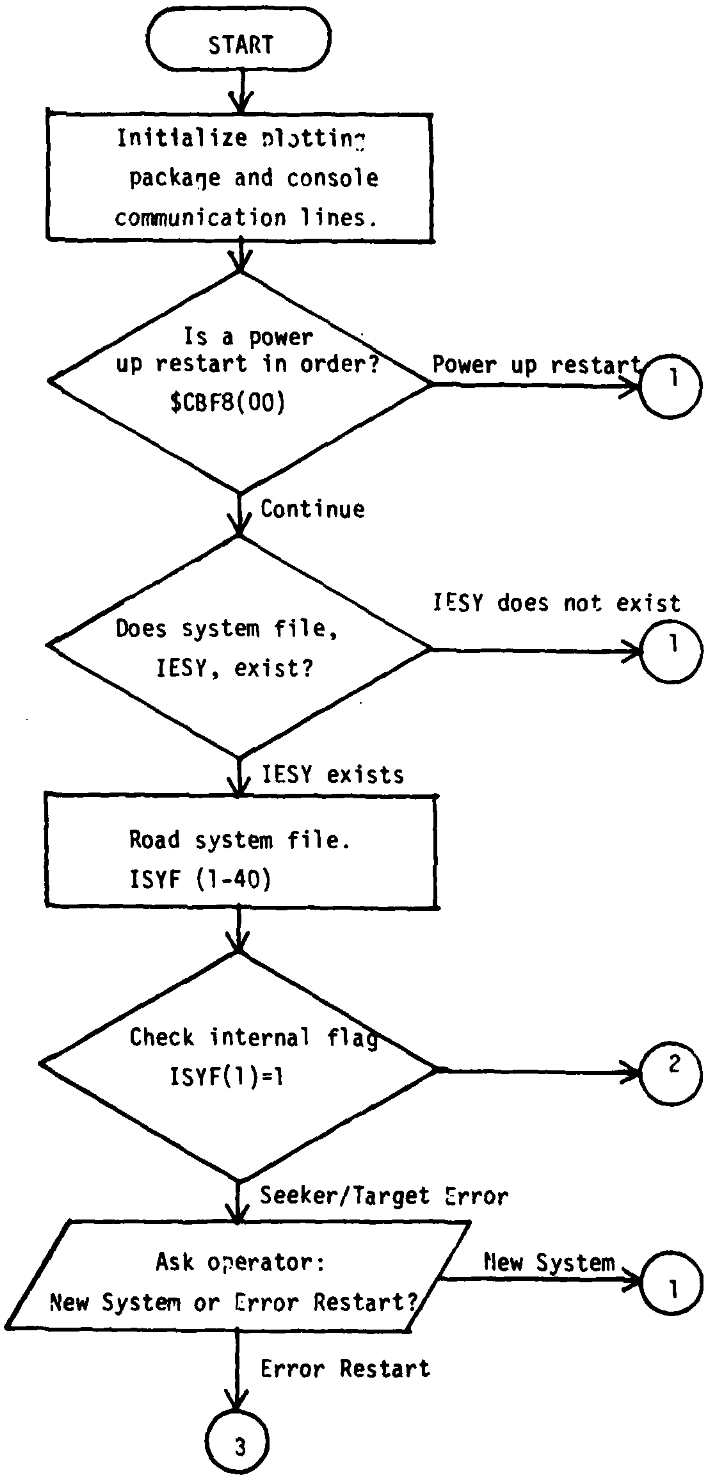


Figure 2 : ETSG Supervisory Program (Main Driver).

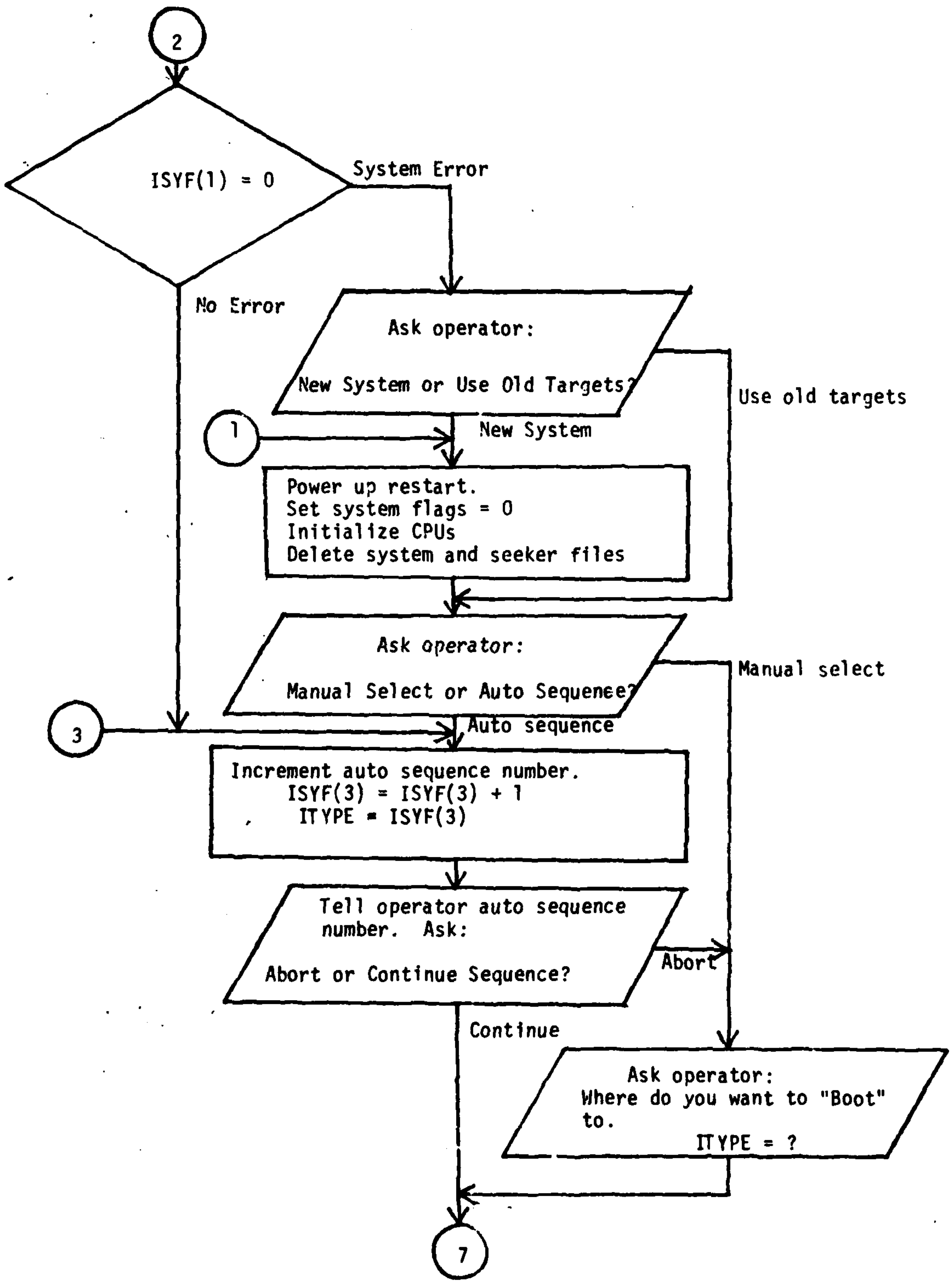


Figure 2 (Continued)

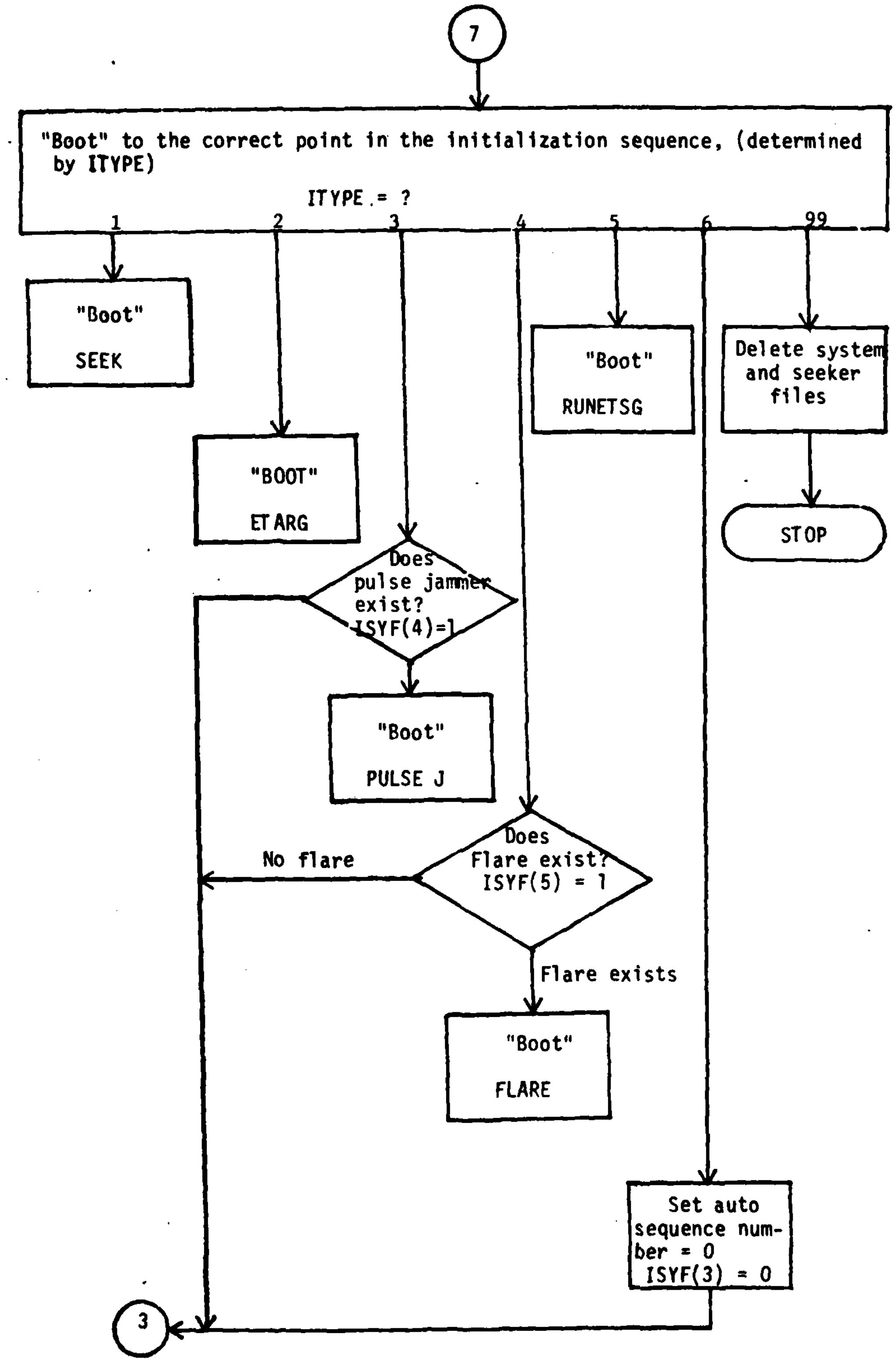


Figure 2 (Continued)

2.2.3 SEEK

SEEK is an interactive FORTRAN program which reads the input data to define a particular seeker. Calculations are performed to determine for the particular seeker if the minimum system signal, SMNSY, times seeker responsivity, ARES, exceeds the minimum DAC output voltage, VO.

When this condition is satisfied, the quantities;

IPBGL - Background Level

IPNSL - Programmable Noise Source Level

IAC10 - Analog Scale Factor Adjust

IB1 - Exponent Scale Factor Adjust

MSIGN - Exponent Scale Factor Command

are calculated and stored in PIAs by the subroutine STAOC. The information stored by STAOC is utilized as DAC controls for the analog boards.

The seeker data is then stored in a diskette file and control is returned to the main driver, ETSG, via "Boot."

The flow chart in Figure 3 (pages 22-29) gives a more detailed account of the processes performed by SEEK.

The "Notes on SEEK Calculations" at the end of this section is a step by step listing with explanatory notes of the calculations performed by SEEK.

The next program in the initialization sequence is ETARG.

Notes on SEEK Calculations:

*Indicates an input variable.

*FOVD = FOV1	Field of view side to side (degrees).
*BLRM = FOV2	Blur diameter (M radian)
BLRR = BLRM*0.001	Blur diameter (rads.).
DPR = 57.2957795	(degree/rad.)
DPM = 0.057295775	(degree/m rad.)
BLRD = BLRM*DPM	Blur diameter (degrees)
NPWN = 2	

NFOV=IRND (NPWN*FOVD/BLRD) Number of discrete points in blur diameter.

Note: IRND is a function which rounds floating point numbers to integer values. It always rounds so as to increase absolute magnitude.

FPPD=NFOV/FOVD Points per degree in field of view.

*TDPC = FOV3 Number of degrees per count for target coordinate.

*IRCSW Rosette or Conscan switch.

If IRCSW = 1 load rosette seeker.

If IRCSW = 2 load conscan seeker.

TCDPC = 4.0/128.0 Minimum number of degrees per count for target coordinate.

Note TDPC $\geq$ TCDPC

if not, default to TDPC = TCDPC.

*FOVTEM = TDPC*256* Temporary variable

Note: FOVTEM $\geq$ FOVD

TCPPC = FPPD*TCDPC points in field of view per count of target coordinate

SBLRM = NPWN/FPPD/DPM Scaled blur diameter.

ICPC8 = FOVTEM/FOV1*256 Number of target coordinate counts across field of view.

Rosette minimum/maximum limits.

NPWN =	1	2	3
MINX	0	0	1
MINY	0	0	1
MAXX	63	62	62
MAXY	63	62	62

*MXSCR Maximum scan rate (Hertz)

MSCRD = 115 Minimum scan rate (Hertz)

*RNEFD(ICH) = RESP (ICH,2) Noise equivalent flux density (watts/cm^2)

Note: ICH is the channel number 1 or 2.

*SNRT (ICH) = RESP (ICH,5) Minimum signal to noise ratio to track.

*BKRD (ICH) = RESP (ICH,3) Background intensity at aperture (watts/cm^2)

*ATTN (ICH) = RESP (ICH,4) Atmospheric attenuation coefficient (1/Km)

ANOIZ (ICH) = RNEFD (ICH) * ARES (ICH) Programmable noise level

SIGMN (ICH) = RNEFD (ICH) * SNRT (ICH) Minimum signal at aperture

*If NFSC (ICH) = 1 scale to NEFD.

If NFSC (ICH) $\neq$ 1 then input:

*SIGMN (ICH) = RESP (ICH,7) Minimum system signal at aperture (watts/cm^2)

SMNSY = SIGMN (ICH) Minimum system signal at aperture

SMXSY = DYNRNG*SMNSY Maximum system signal at aperture (watts/cm^2)

VMIN = SMNSY * ARES (ICH) Minimum detector voltage.

C2 = VMIN/VO Seeker volts to DAC volts scale factor.

Note: VO = 6.1 E - 4 Minimum DAC voltage

CO = 64 Seeker irradiance to FNS.

CM2PM2 = 1.0 E - 4 Cm^2/m^2

```
C5(ICH) = CO*CM2PM2/SMNSY/NPWN/NPWN
     Note:  C2 = VMIN/VO > 1

            RLOG2 = ALOG (2.0)
            RL218 = 0.8480
```

Note: $\frac{\log(X)}{\log(2)} = \log_2(X)$

```
            MSIGN = 1
     Note:  In STAOC MSIGN is tested to determine if add to exponent occurs.
     If VMIN/VO = 1 then;
         IB1 = 0
     IAC10 = 1023
         AT2 = 1.0

     If VMIN/VO > 1 then;
     T1 = ALOG(C2)/RLOG2
```

Note: $T1 = \log_2(VMIN/VO)$

```
         IT1 = T1
     Note: Change real to integer
         FT1 = T1 - IT1
```

If $FT1 \leq 0.8480$ then;

```
         IB1 = IT1 + M51GN
     IAC10 = 1023
         AT2 = POWER (2.0, FT1)
```

Note: POWER (a,x) = a^x

```
     If FT1 > 0.8480 then
       IT1 = IT1 + MSIGN
       IB1 = IT1
       FT2 = FT1 - 1
```

$AT2 = 2^{FT2}$

```
       DZ = AT2*1024
       IAC10 = IRND (DZ)
```

If $IAC10 \gtrsim 1024$ then;

```
       IAC10 = 1023
        IB1 = IT1 + MSIGN
        IAC10 = 1023
```

$AT2 = 2^{FT1}$

```
       VBGMX = 10.0
       VBGAB = BKRD (ICH) VBGMX*16383
        IPBGL = IRND (VBGAB)
```

```
If IPBGL > 16383 then IPBGL = 16383
VNZAB = VMIN/VNZMX*255
 IPNSL = IRND (VNZAB)

If IPNSL > 255 then IPNSL = 255
ISKRCK = IFLAGS (22)
ISKRCK = 10 * (BLRM + FOVD)/(ARES(1)*SIGMN(2) + ARES(2) * SIGMN(1))
```

Analog Scale Factor Adjust IAC10

$T1 = \log_2(SNRT*RNEFD*ARES/6.1E-4)$

IT1 = T1

Note: Real to integer

FT1 = T1 - IT1

Note: Truncate whole number.

FT2 = FT1 - 1

$IAC10 = 2^{FT2} * 1024$

If IQC10 ≥ 1024

then IAC10 = 1023

Stored at:

CBCA	
CBC8	for J channel
CBBA	for K channel
CBB8	

If Ftl ≤ RL 218 = 0.8480

then IAC10 = 1023

Background Level IPBGL

IPBGL = IRND (BKRD/10.0*16383)

If IPBGL > 16383 then

IPBGL = 16383

AND High byte with $3F

EOR High byte with $3F

EOR Low byte with $FF

Store at:

CBCE	for K channel
CBCC	
CBBE	for J channel
CBBC	

Note: Subroutine CBV in STAOS reorders the bits to compensate for a hardware design problem.

Exponent Scale Factor Adjust IB1

$$IB1 = 16 * \left(\frac{\ln \left(RNEFD*SNRT*ARES/6.1E-4 \right)}{\ln 2} + 1 \right)$$

AND with $FO

Store at:

CBB0	for J channel
CBB4	
CBC4	for K channel

Exponent Scale Factor Command MSIGN

MSIGN = 1

CBB2/20	for J channel
CBB6/20	
CBC6/20	for K channel

Programmable Noise Source Level IPNSL

IPNSL = IRND(SNRT*RNEFD*ARES/5*255)

If IPNSL > 255

then IPNSL = 255

CBC2	for K channel
CBC0	for J channel

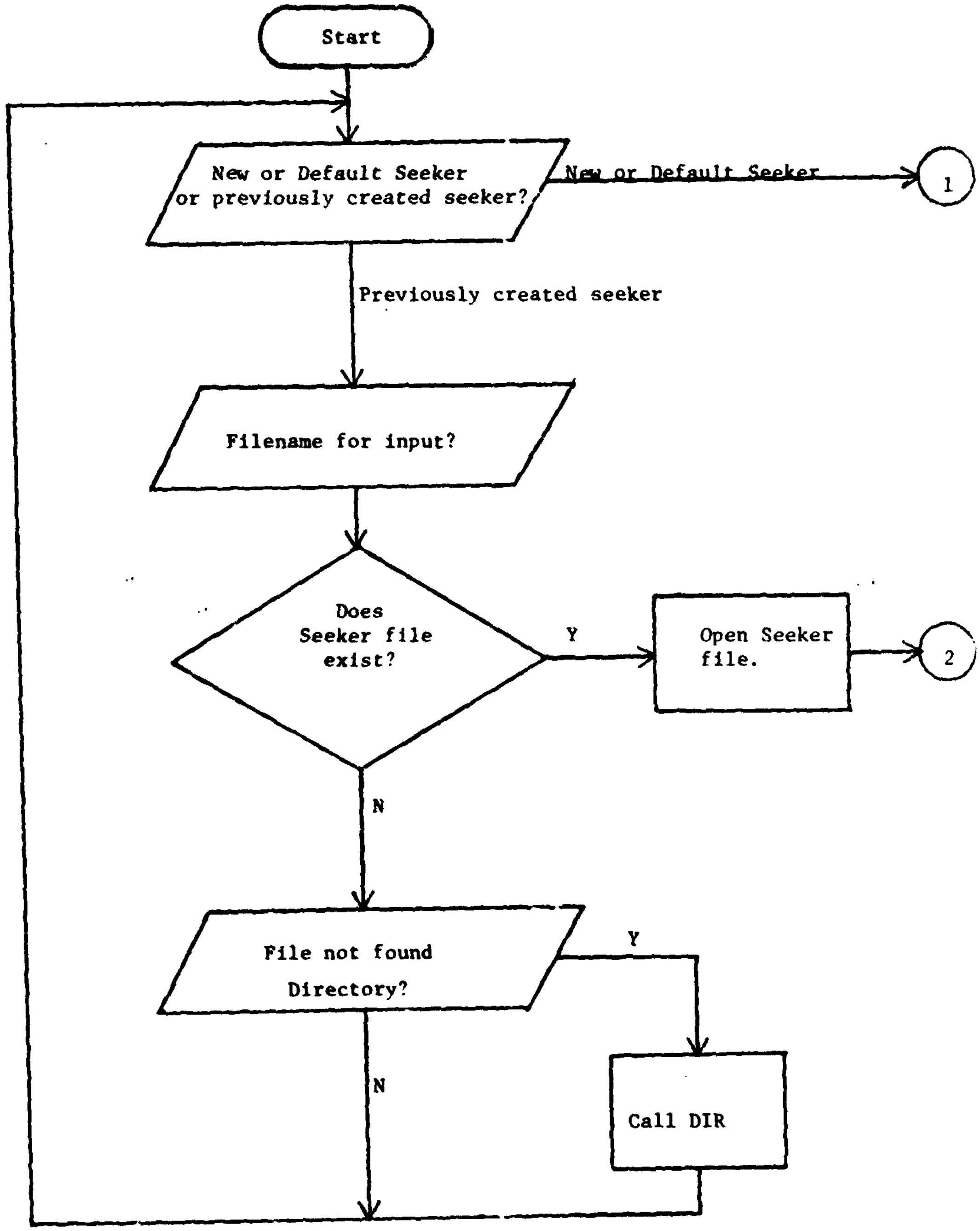


Figure 3 : SEEK

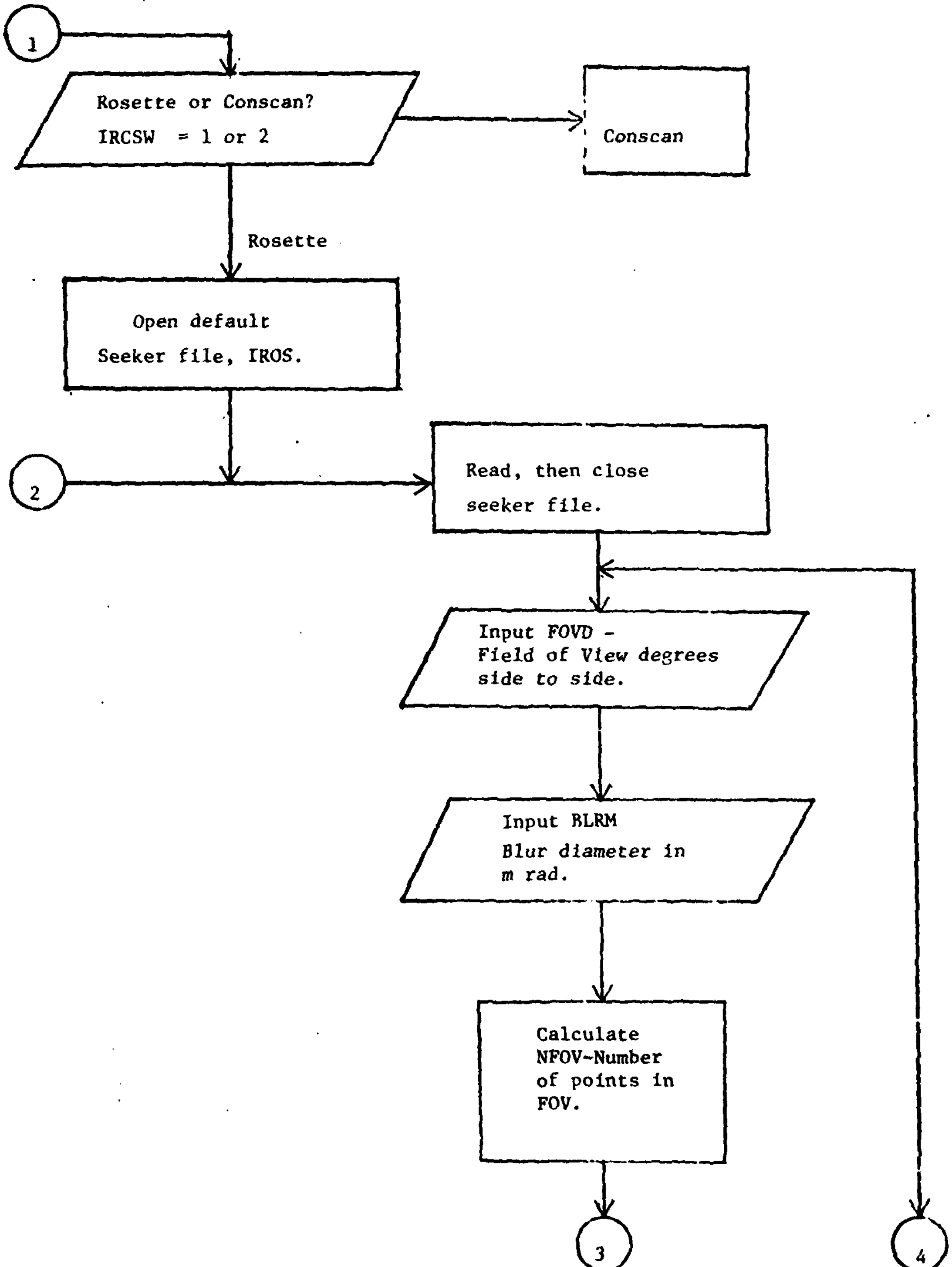


Figure 3 (Continued)

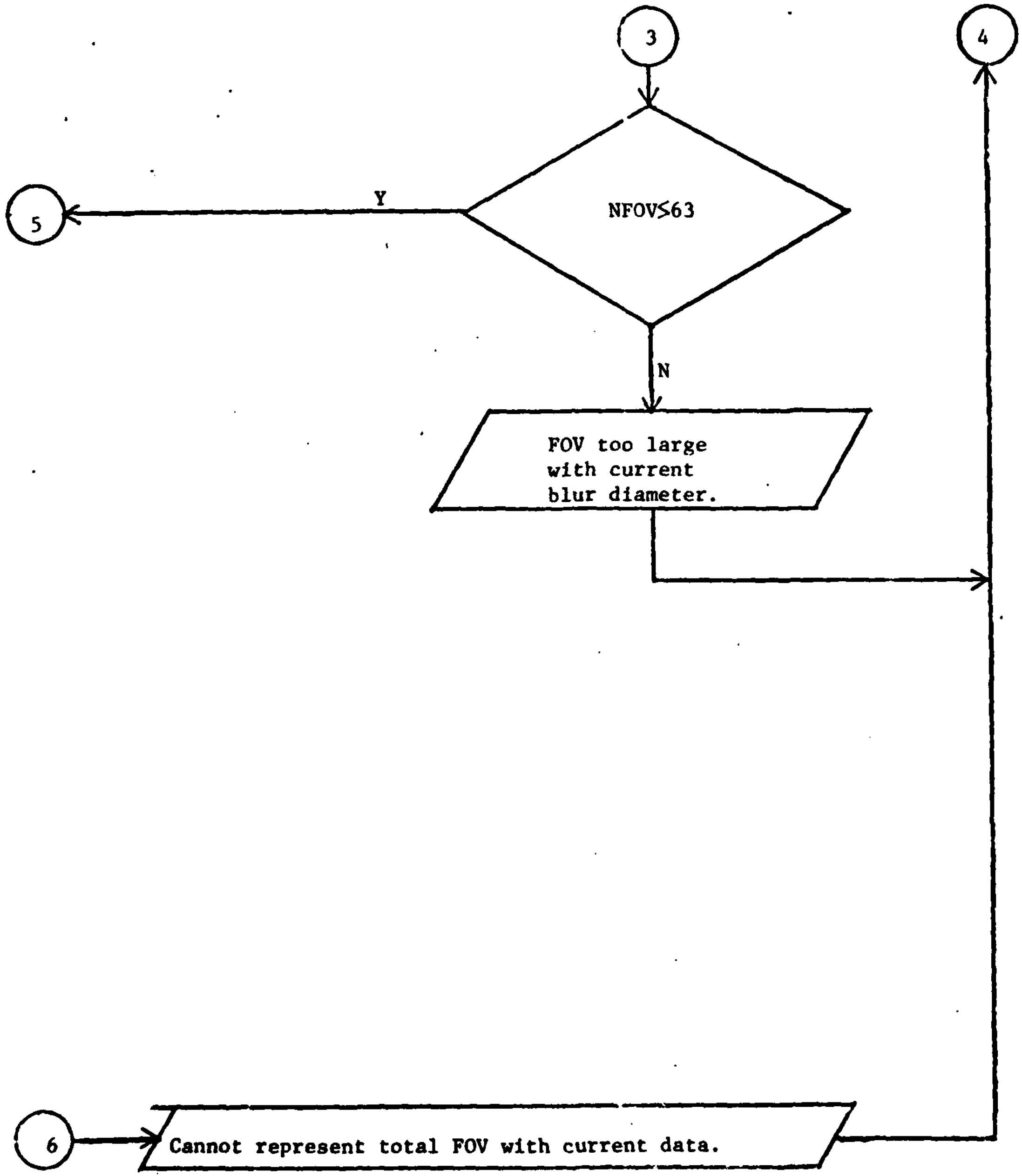


Figure 3 (Continued)

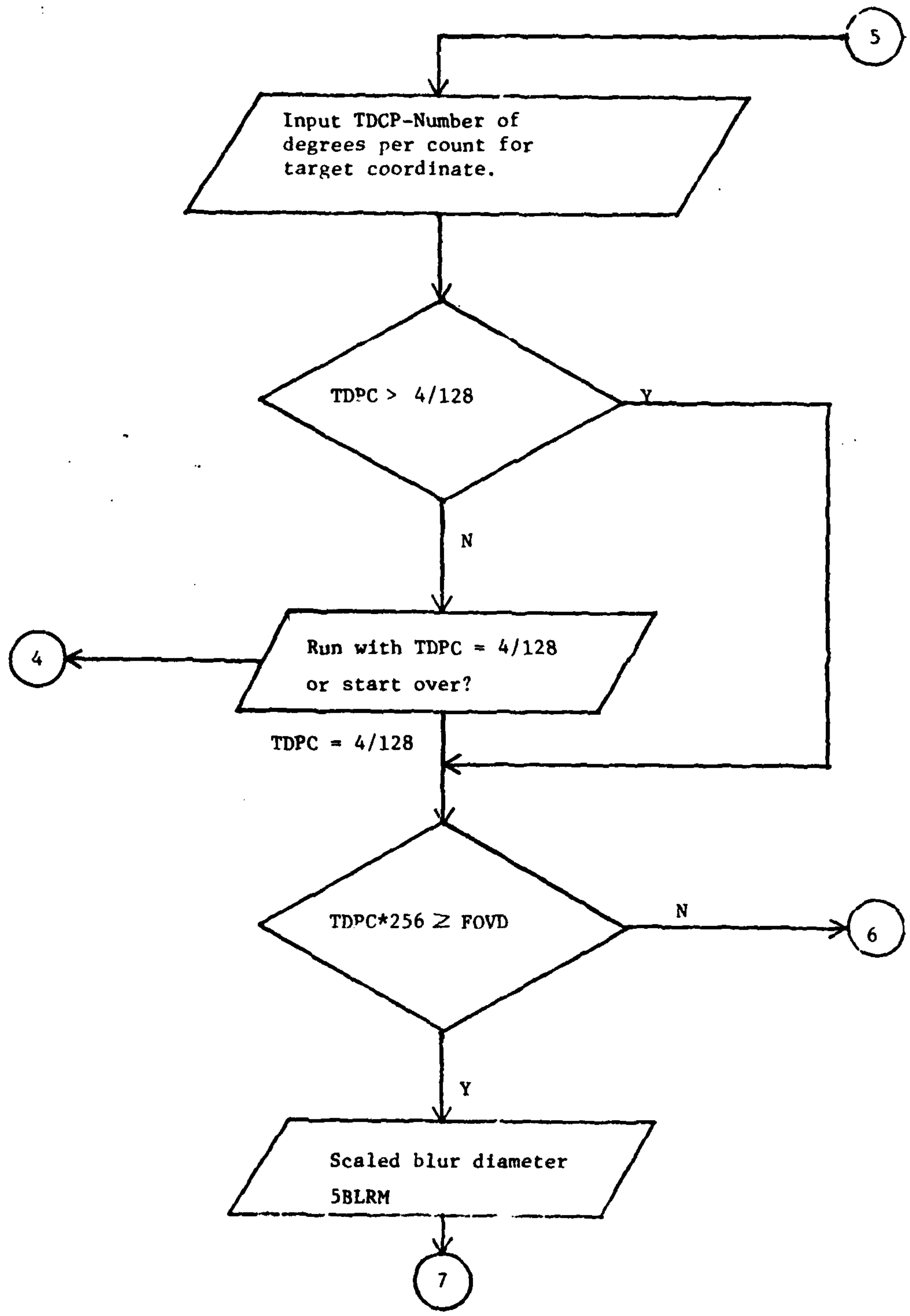


Figure 3 (Continued)

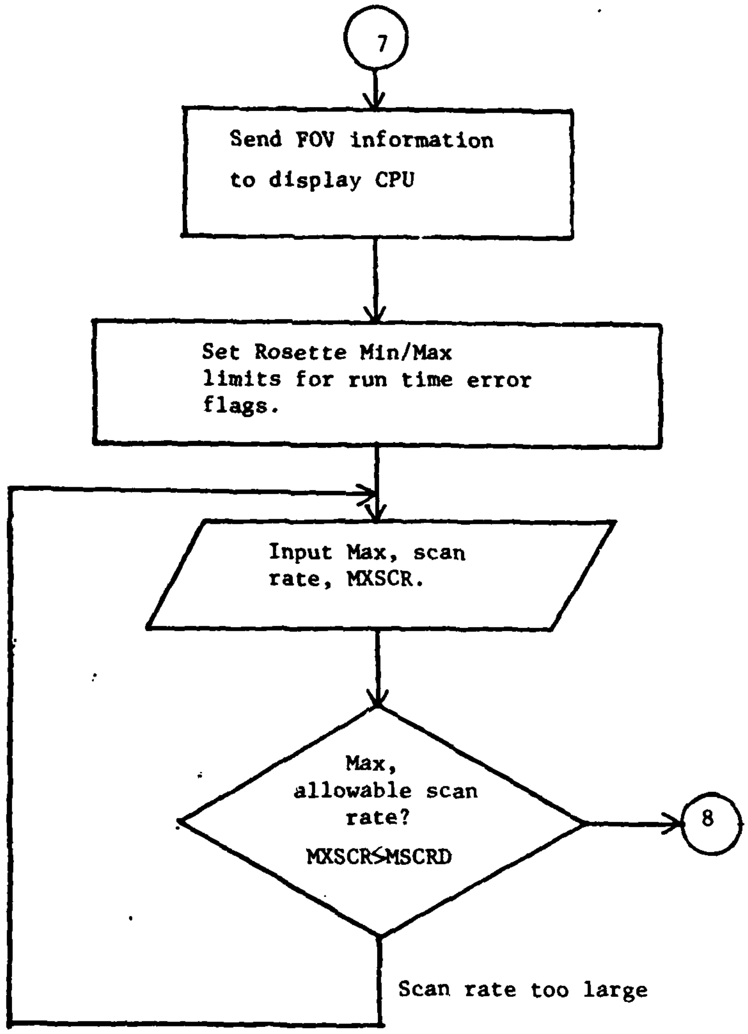


Figure 3 (Continued)

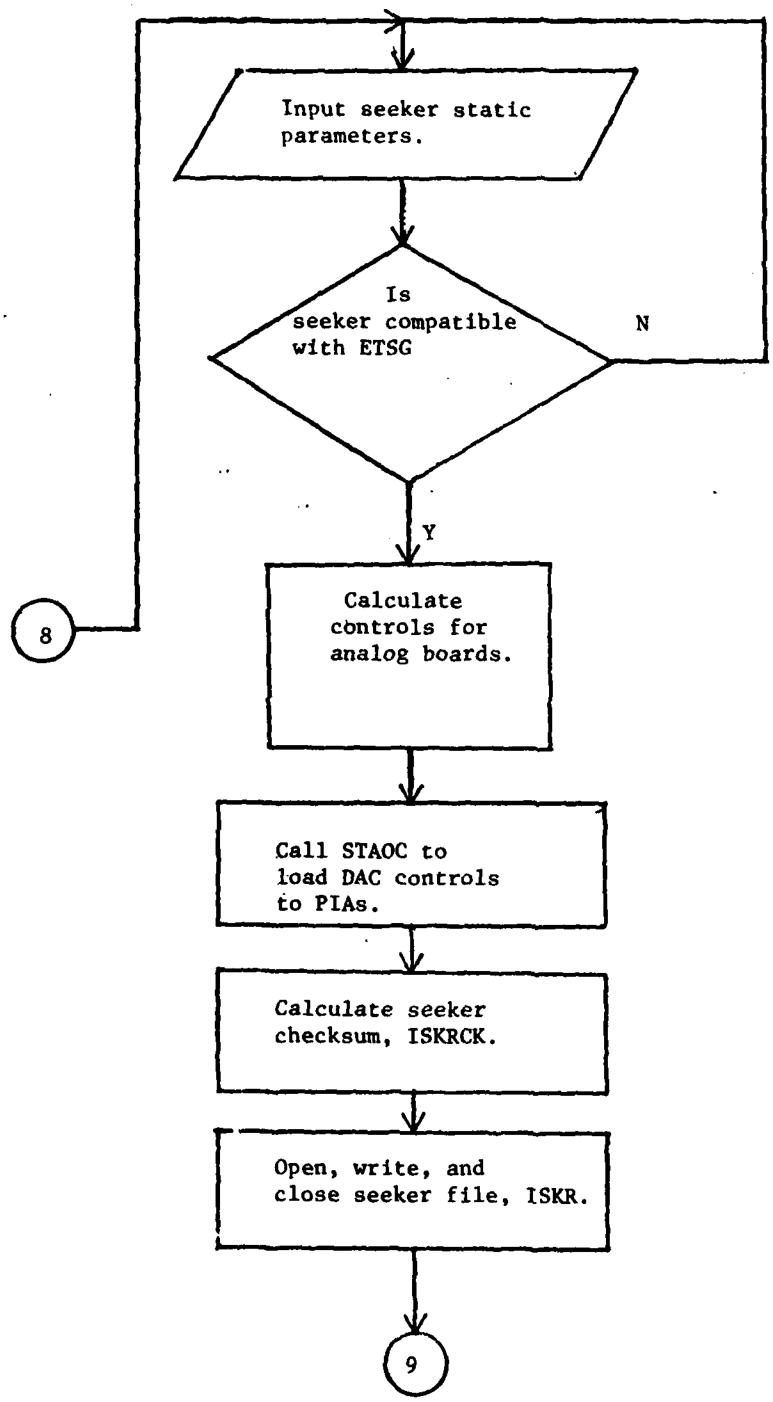


Figure 3 (Continued)

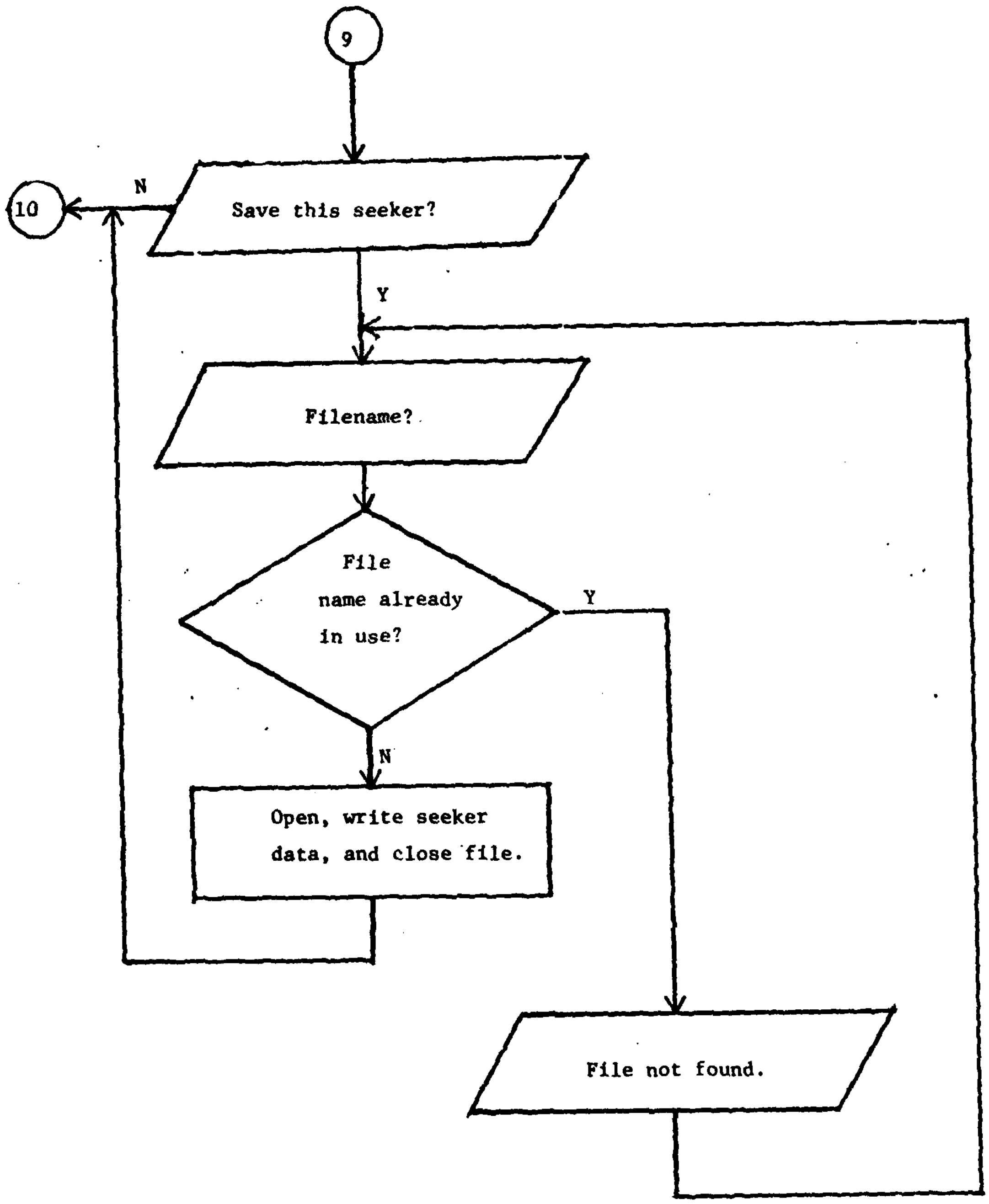


Figure 3 (Continued)

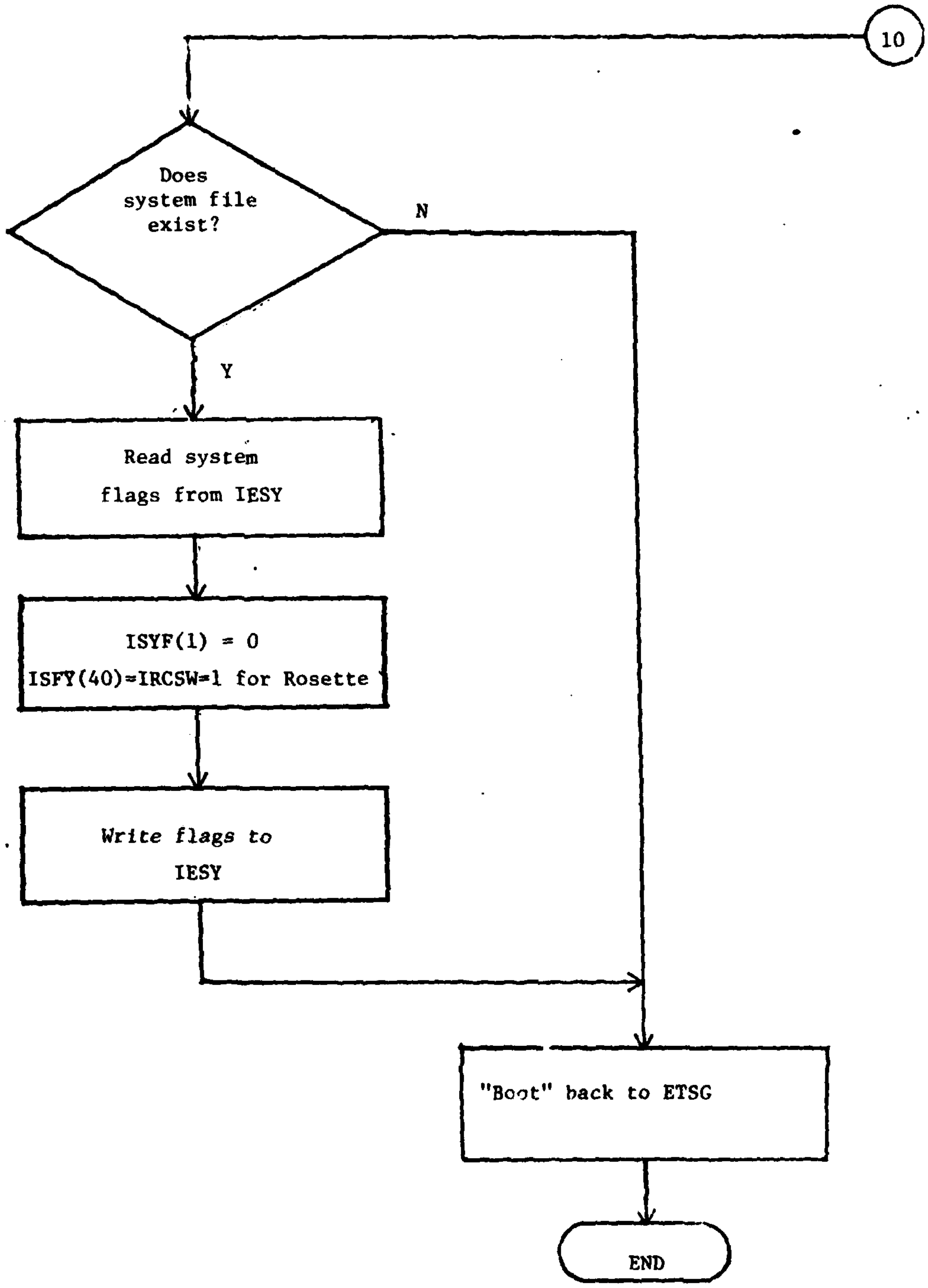


Figure 3 (Continued)

2.2.4 ETARG

The interactive FORTRAN routine ETARG, controls the generation of all simple and complex targets. ETARG specifics which targets are flares or pulsed jammers and also assigns target channels and polarities.

The general information flow in ETARG is depicted in the flowchart in Figure 4 (see pages 32-38). A detailed account of the values calculated in ETARG is presented in the "Notes on ETARG Calculations" at the end of this section.

Notes on ETARG Calculations:

```
     * Indicates an input variable
     * TSZX = TRG(1)                    Target size X (meters).
     * TAR = TRG (3)                    Target aspect ratio.
              If TAR = 0 then TSZY = TSZX/TAR.
              If TAR = 0 then
     * TSZY = TRG2                      Target size Y (meters).
             and
       TAR = TSZX/TSZY
                 TAR > 1
     * ISC = IFLAGS (3)                  Channel number
     * IPOLTY = 1
              If ISC = 2 then
     * IPOLTY = + 1 or - 1     for UV targets.
     * RJT = TRG (4)                    Target radiance (watts/steradian)
     TATTN = ATTN (ISC)/1000  Atmospheric attenuation coefficient from SEEK.
RJTP = RJT - BKRD(ISC)*10000*EXP(TATTN)
     RJTP = RJTP*IPOLTY                  Contrast Radiance (watts/steradian)
         RJTP ≥ 0
RT = RJT/SIGMN (ISC)/10000
        RT = SQRT (RT)                  Clear air track range (meters)
     * RMAX = TRG(7)                    Maximum target range (meters)
                DYNRNG = 3.57E9
TATTN = -ATTN (ISC)/1000*RMAX
SMNT = RJTP*EXP(TATTN)             Minimum target signal
SMNT = SMNT/RMAX/RMAX/10000
```

```
SMXSY = DYNRNG*SIGMN(ISC)  Maximum system signal.
AGS = SMNT/SI[illegible]MN(ISC)
                  AGS > 1
                  T1 = TSZX
                  IMAX = 64
PPM = (IMAX-1)/T1             Points/meter in TLR
*If the target is not a plane, ITSW ≥ 5,
then:
     T1 = TSZY*TSZY/4 +TSZX*TSZX
and  PPM = (IMAX - 1)/SQRT(T1)
otherwise:
RC0 = TSZX/BLRM*1000
RC1 = TSZX/BLRM*1000*NPWN/(NPWN + 1)
RC2 = TSZY/BLRM*1000*NPWN/(NPWN + 1)
RMNR = BLRM/NPWN/PPM*1000     Range of 1 to 1 resolution. (meters)
*RMIN = TRG(8)                Minimum target range (meters).
TATTN = -ATTN(ISC)/1000*(RMAX-RMIN)
SMXV = SMNT*EXP(TATTN)
SMXV = SMXV*RMAX/RMIN*RMAX/RMIN
ZA = SMXSY/SMXV
If ZA ≥ 1                 System will overflow
   ZA ≥ 128               Probable overflow
   ZA ≥ 1608.5            Possible overflow
   ZA ≥ 1608.5            No overflow
RKMX = 599.0/PPM          Maximum value for key points (meters).
*RKXM = TRG(5)            X key point (meters).
*RKYM = TRG(6)            Y key point (meters).
*ITCLR = IFLAGS(10)       Target color
*ISRVT = IFLAGS(11)       True target flag
*IPJ = IFLAGS(5)          Pulse jammer flag
*IFL = IFLAGS(6)          Flare flag
      If IPJ = 1 then ISYF(4) = 1
*IPRI        from    STTP  Target priority
     ID6 = IPRI + 6
     ISYF (ID6) = 0
```

If IFL = 1 then

ISYF(5) = 1

and ISYF(6) = ISYF(6) + 1

and ISYF (ID6) = 1

Clear intensity accumulators:

ZSUM(4) = 0

ZCNT(4) = 0

PMX(4) = 0

PMN(4) = 5.0E10

Set point target view

CST =9.0

Note: INVERT is a function which converts floating point numbers to the ETSG internal Floating Point Number System (FNS).

ITMP = INVERT(CST)

This number goes to the point target lookup RAM.

RLOG2 = ALOG(2.0)

RFAVG = ZSUM(4)/ZCNT(4)

PAVG = RFAVG/9.0*PMN(4)

PMX(4), PMN(4), and PAVG are output at the console during initialization.

Equations for initialization interface variables listed below may be found in Appendix II.

RRAN

ISFR

ISFP

AL2E

FOVS

ACSF

PTSS

TGT1

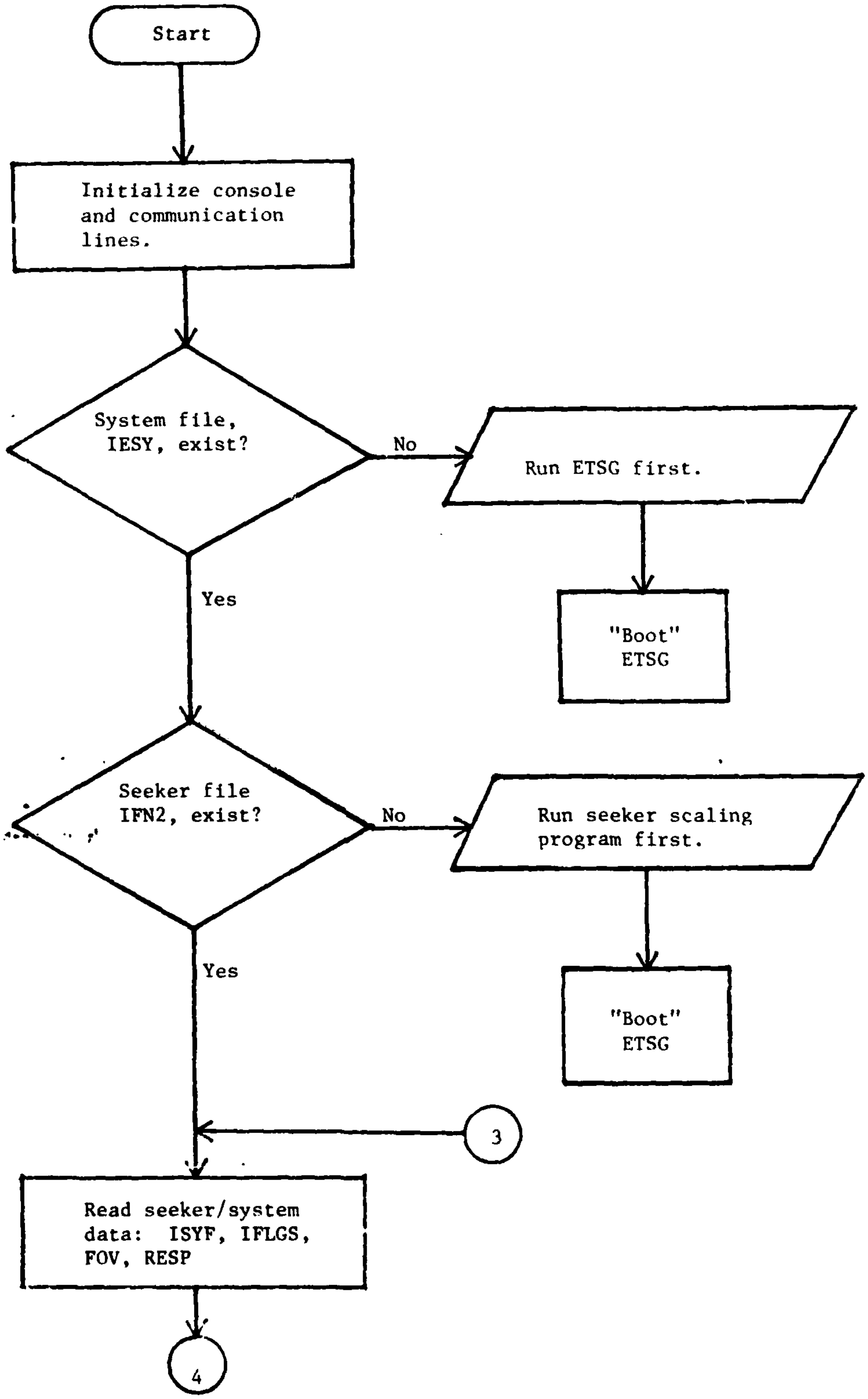


Figure 4: ETARG

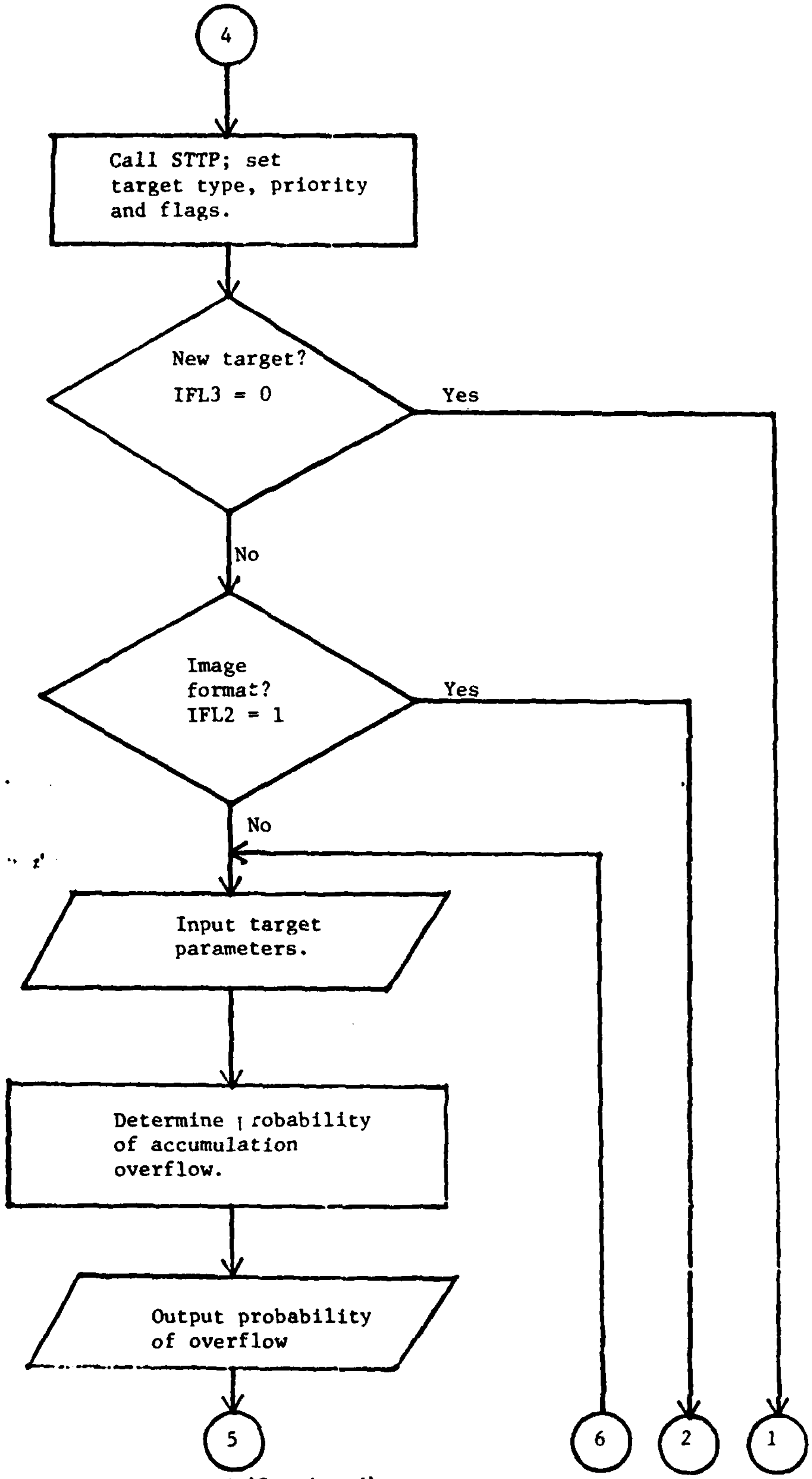


Figure 4 (Continued)

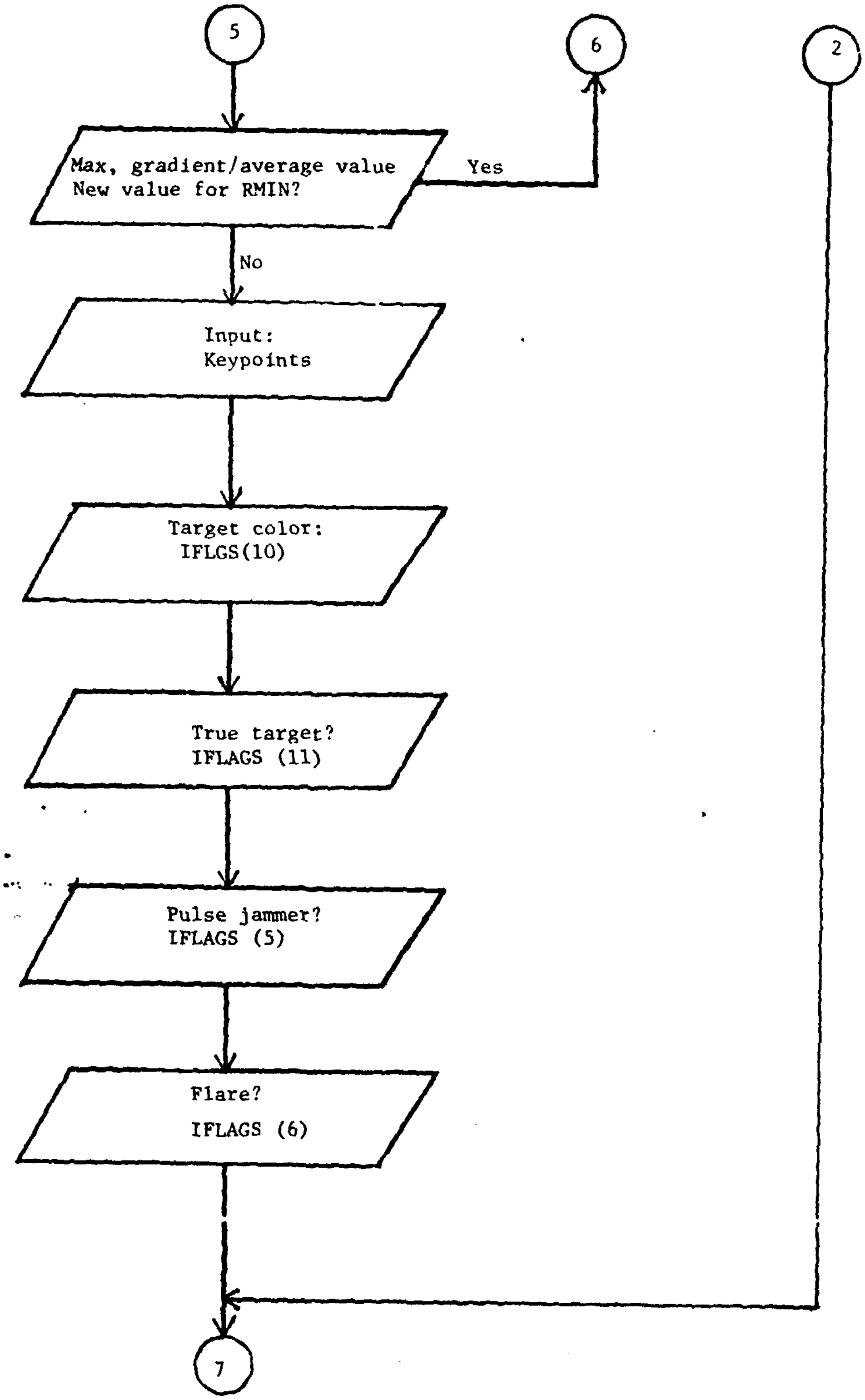


Figure 4 (Continued)

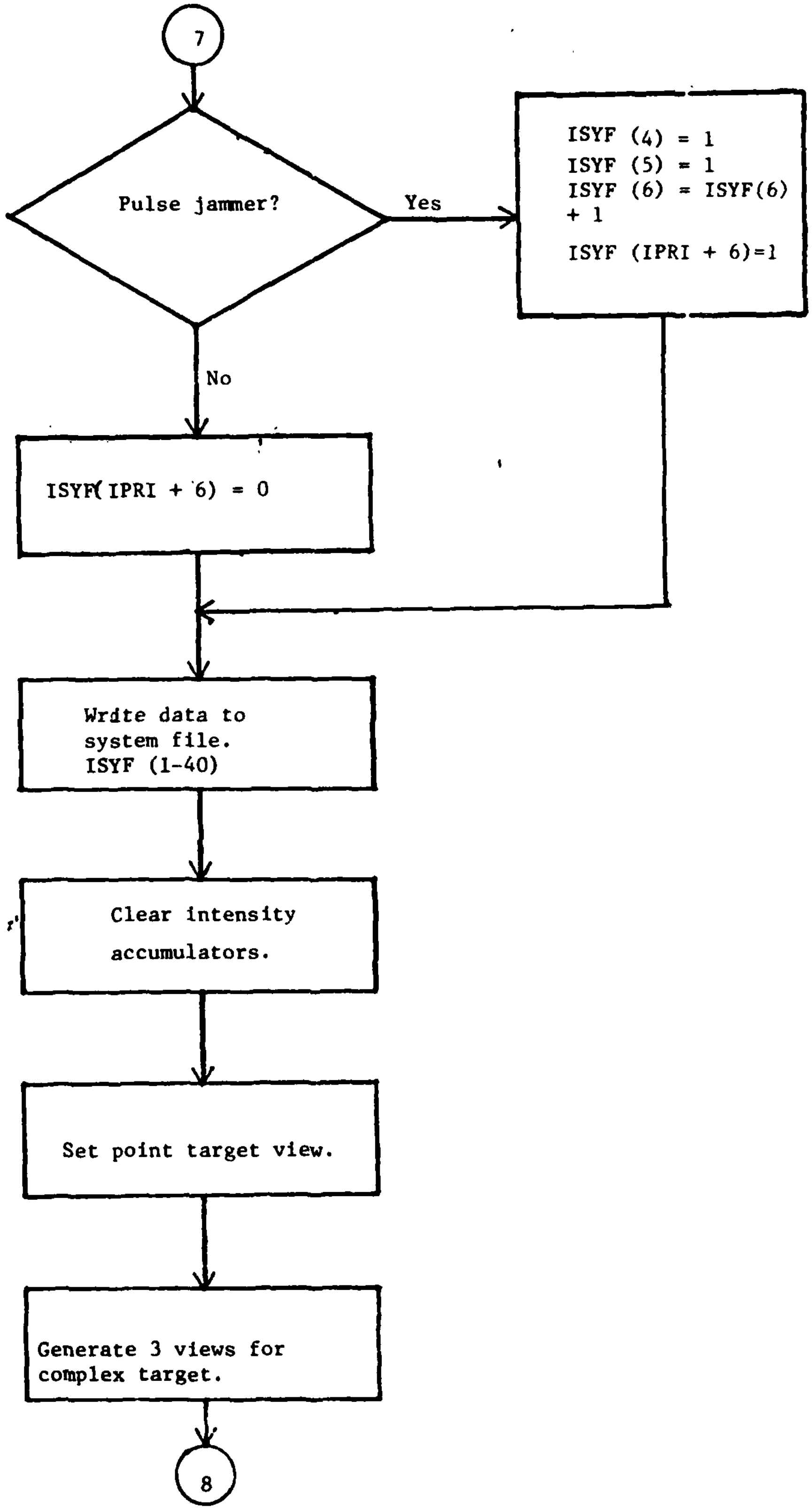


Figure 4 (Continued)

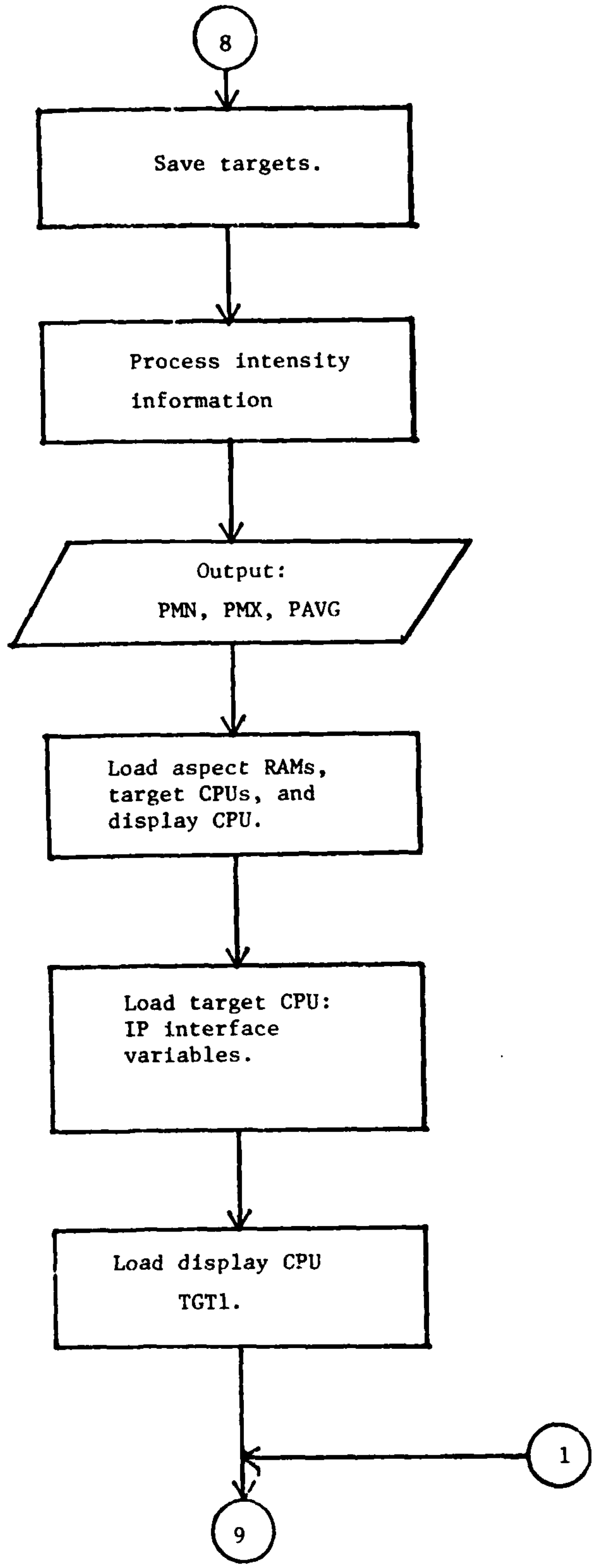


Figure 4 (Continued)

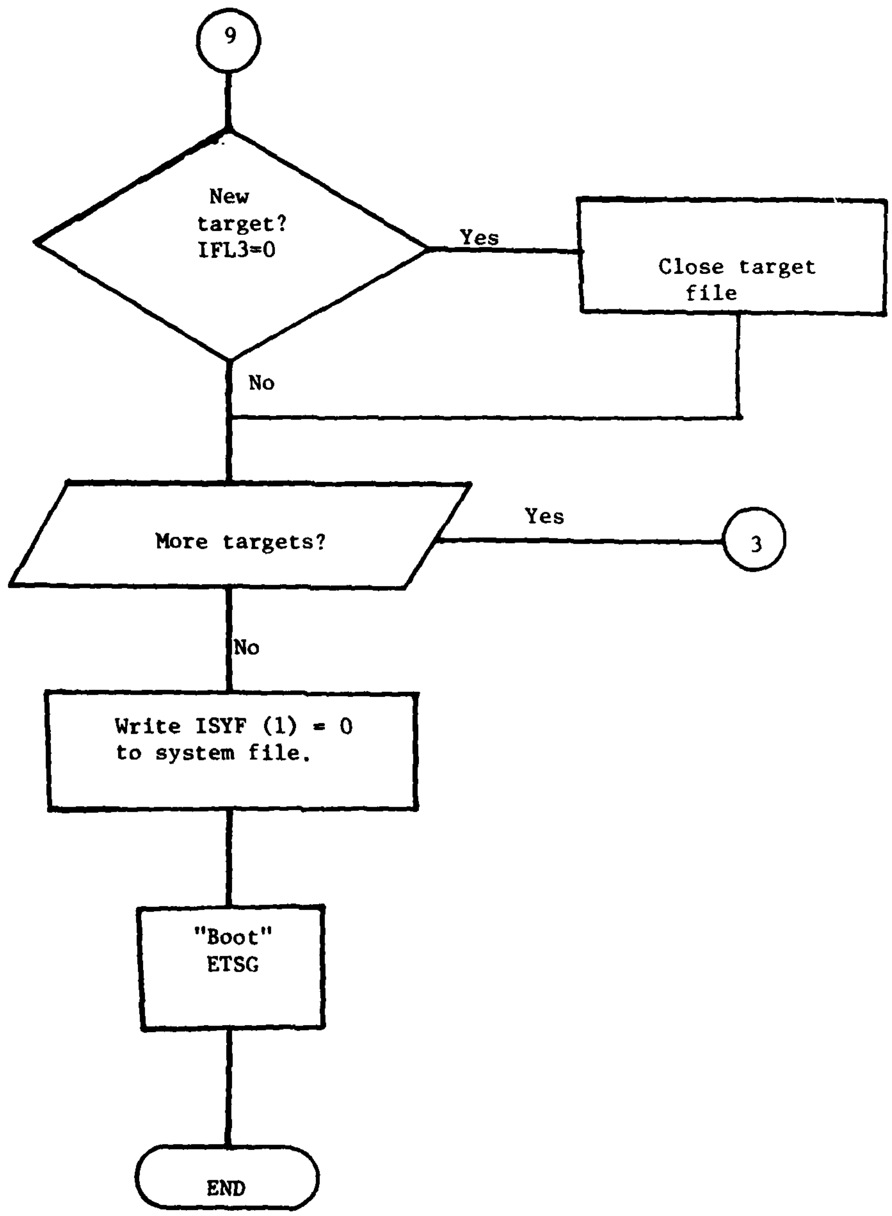


Figure 4 (Continued)

2.2.5 PULSEJ

PULSEJ is a FORTRAN program which generates the time history for a given pulsed jammer. The pulsed jammer is defined by the following input parameters:

NPULSE - number of pulses

PDUTY - duty cycle (%)

SFREQ - start spin frequency

EFREQ - end spin frequency

STIME - sweep time

The pulsed jammer time history is generated by an iterative process which increments time by a factor which is the reciprocal of the current spin frequency. For each cycle the number of strobe "on" and strobe "off" points is calculated and stored in memory. The strobe time history is recorded in 32K bits of memory. Since the strobe "on/off" flag, JSIG, is a FORTRAN integer, 64K bytes of storage is required.

Notes on PULSEJ Calculations

* denotes input variable.

* NPULSE	Number of pulses
* PDUTY	Duty cycle (per cent)
* SFREQ	Start Spin Frequency(Hertz)
* EFREQ	End frequency (Hertz)
* STIME	Sweep time (Seconds)

SFREQ > 64

EFREQ > 64

If: (2*SFREQ*NPULSE-SFREQ)*(2*PDUTY/100) > 1000 then frequency/NPULSE is too high or PDUTY is too low.

DT = 1.0/32767 (cycle/bits)

DT is a scale factor which is used to divide sweep time into 32K bits.

DUTY = PDUTY/100.0

Converts % to fractions of a cycle.

SRATE = (EFREQ-SFREQ)/STIME

Average change in frequency per unit time.

NPARTS = 2*NPULSE-1

Number of parts in strobe history.

TIME = 0 Initialize time to zero.

CFREQ = SFREQ + SRATE*TIME Current frequency at any given point in time.

CT = 1.0/CFREQ Current spin cycle time

NPTS = CT*DUTY/DT Strobe time per given cycle multiplied by total number of points.

MPTS = (NPTS/NPARTS) + 0.5 The number of points that the strobe is "on" per spin cycle.

When JSIG(IP) = 0 strobe is "off"

When JSIG(IP) = 1 strobe is "on"

NPTS = CT*(1-DUTY)/DT The number of points that the strobe is "off" per spin cycle.

TIME + CT/2 - STIME SO Test to see if sweep time has been used up.

Refer to Fig. 5 for a functional diagram of the PULSEJ.

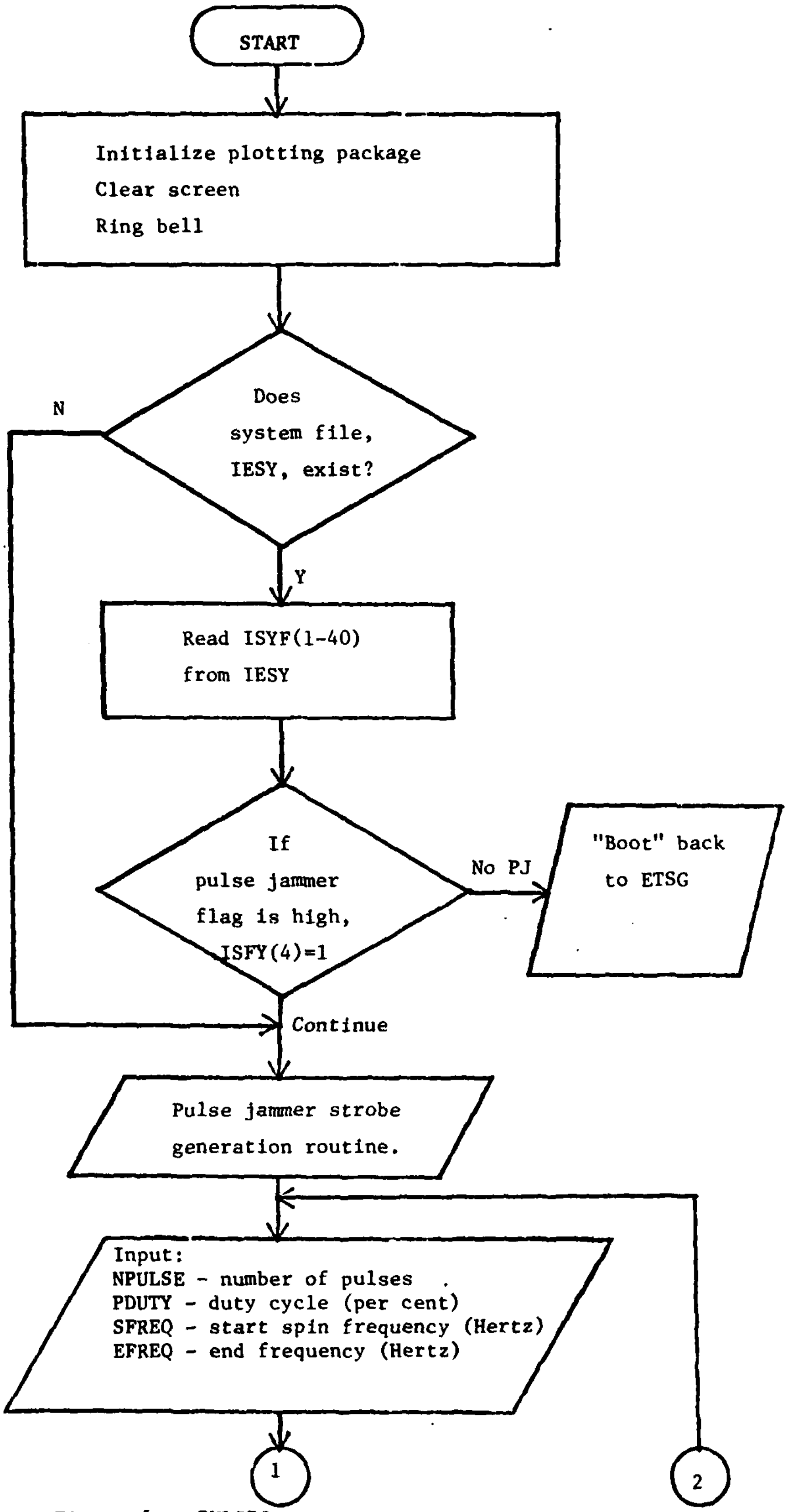


Figure 5 : PULSEJ

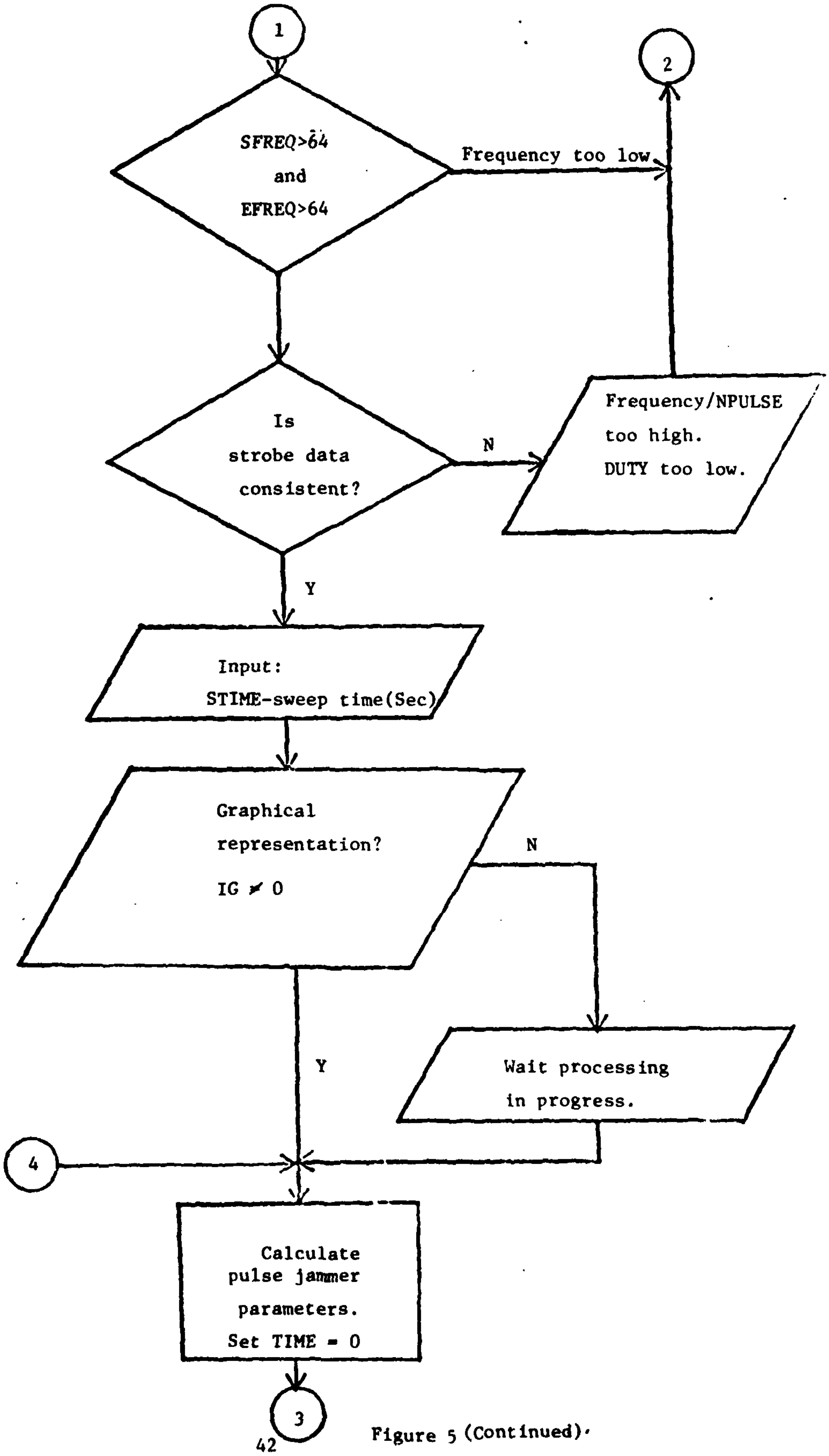


Figure 5 (Continued).

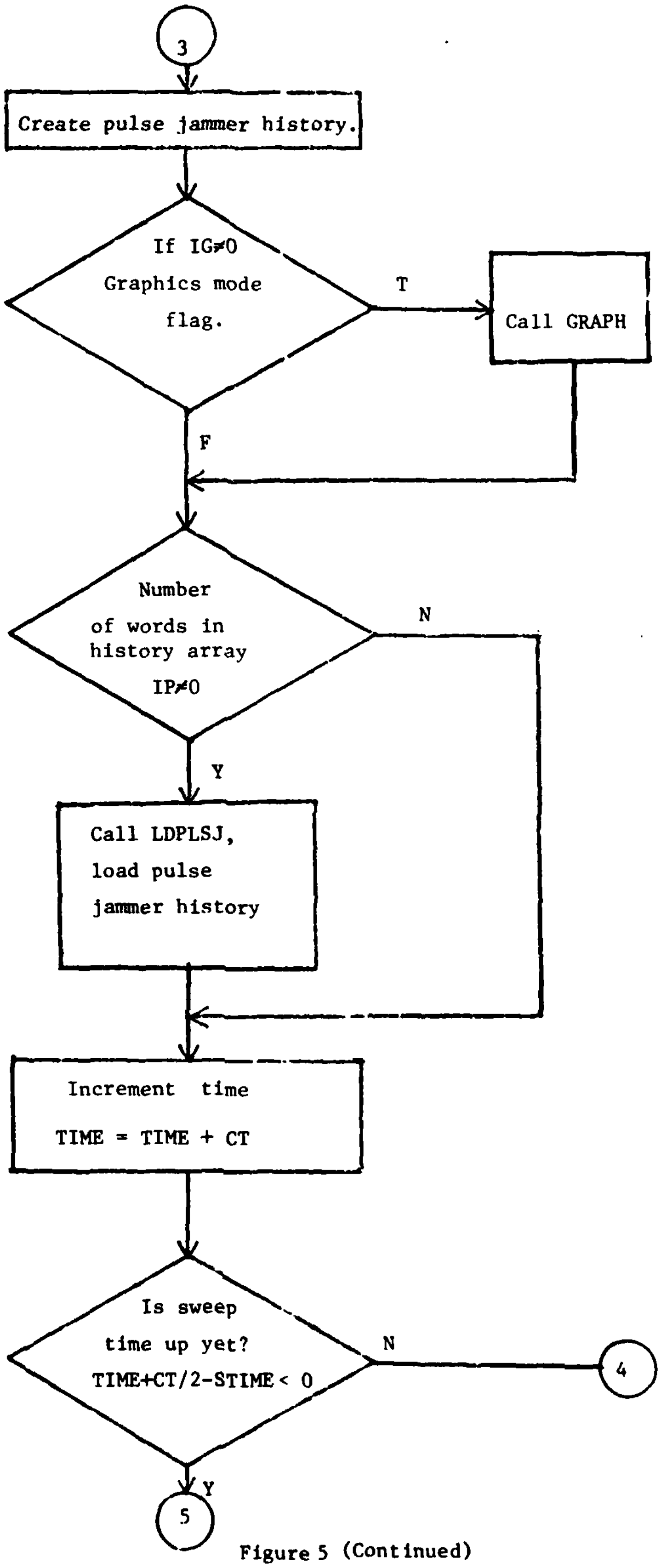


Figure 5 (Continued)

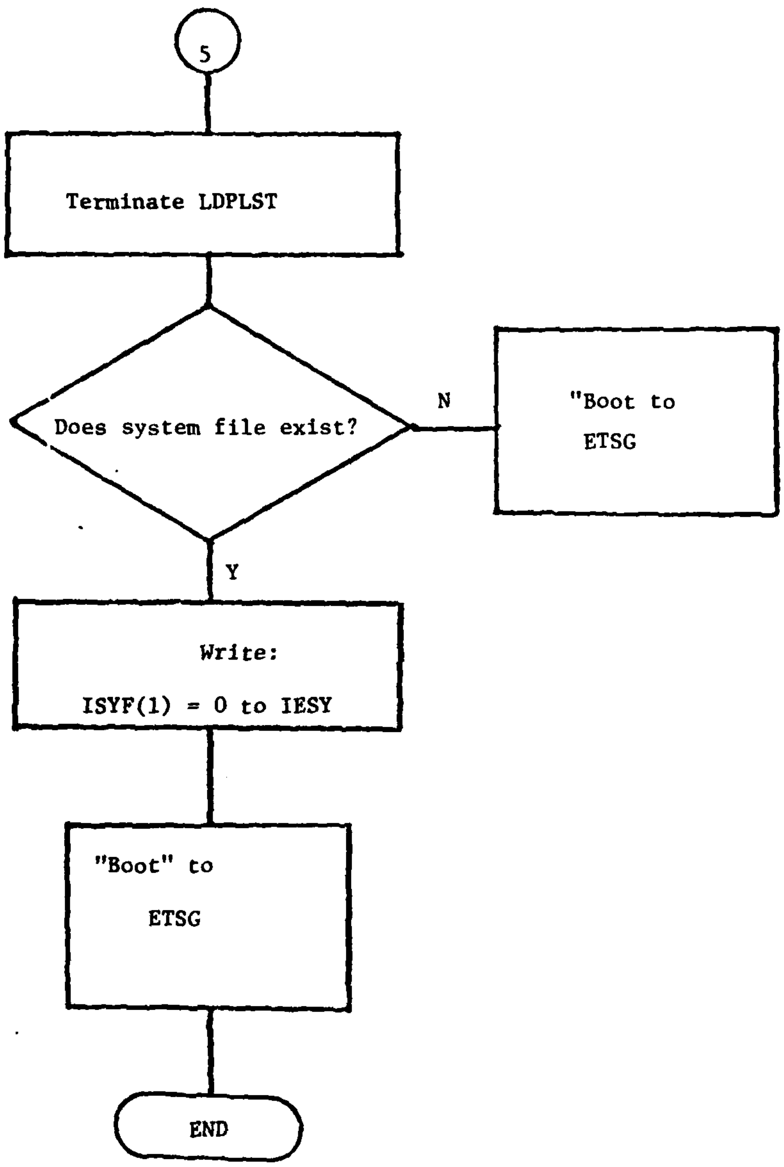


Figure 5 (Continued)

2.2.6 FLARE

FLARE initializes the flare time history. The time history is entered as up to twenty pairs of intensity and time data. One time history is used for all sources designated as flares, but each individual flare may be activated independently. The specified pairs of time and intensity data are processed by the target CPUs to update the flare absolute intensity during each frame. Refer to Fig. 6 for a functional diagram of the FLARE.

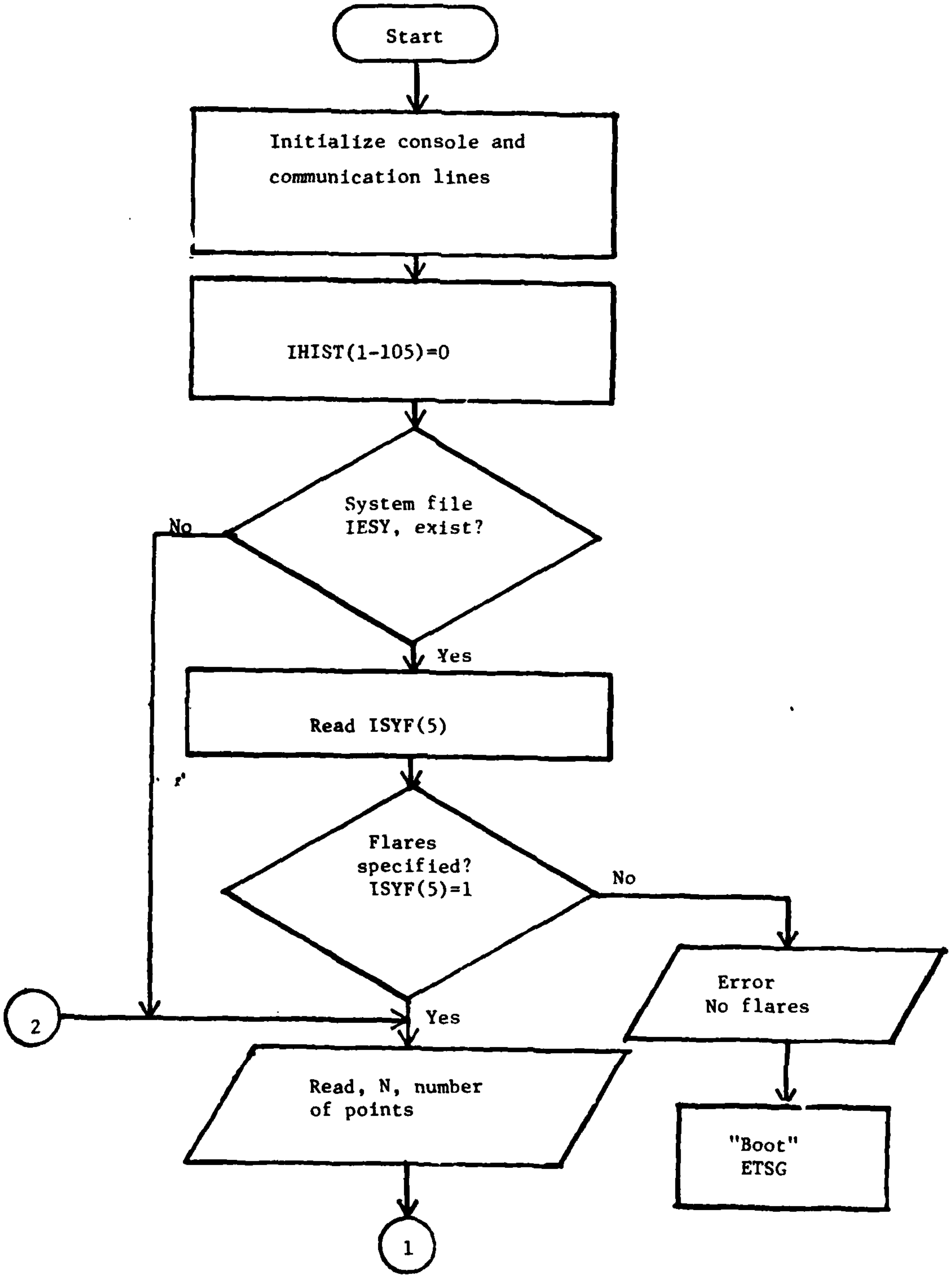


Figure 6 : FLARE

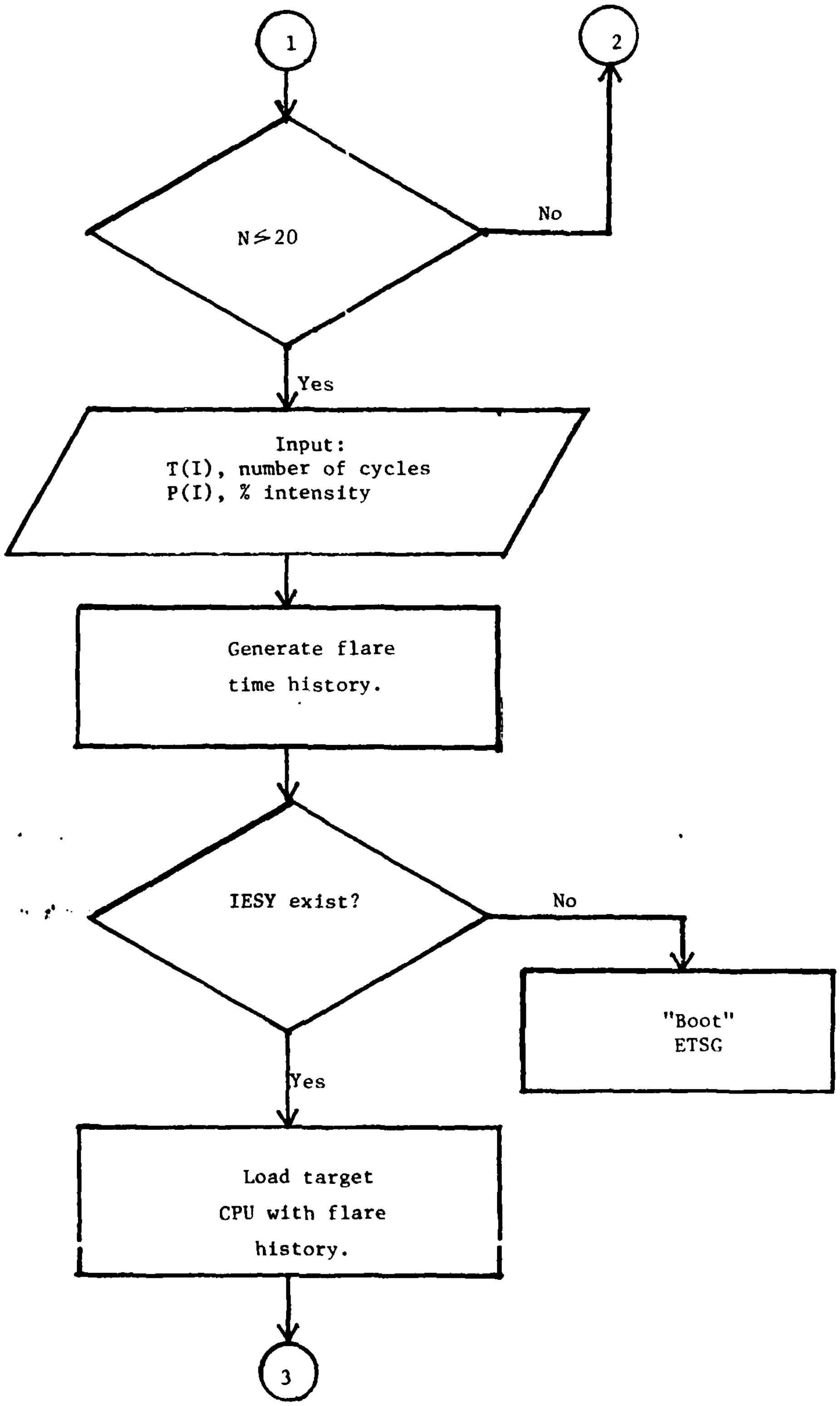


Figure 6 (Continued)

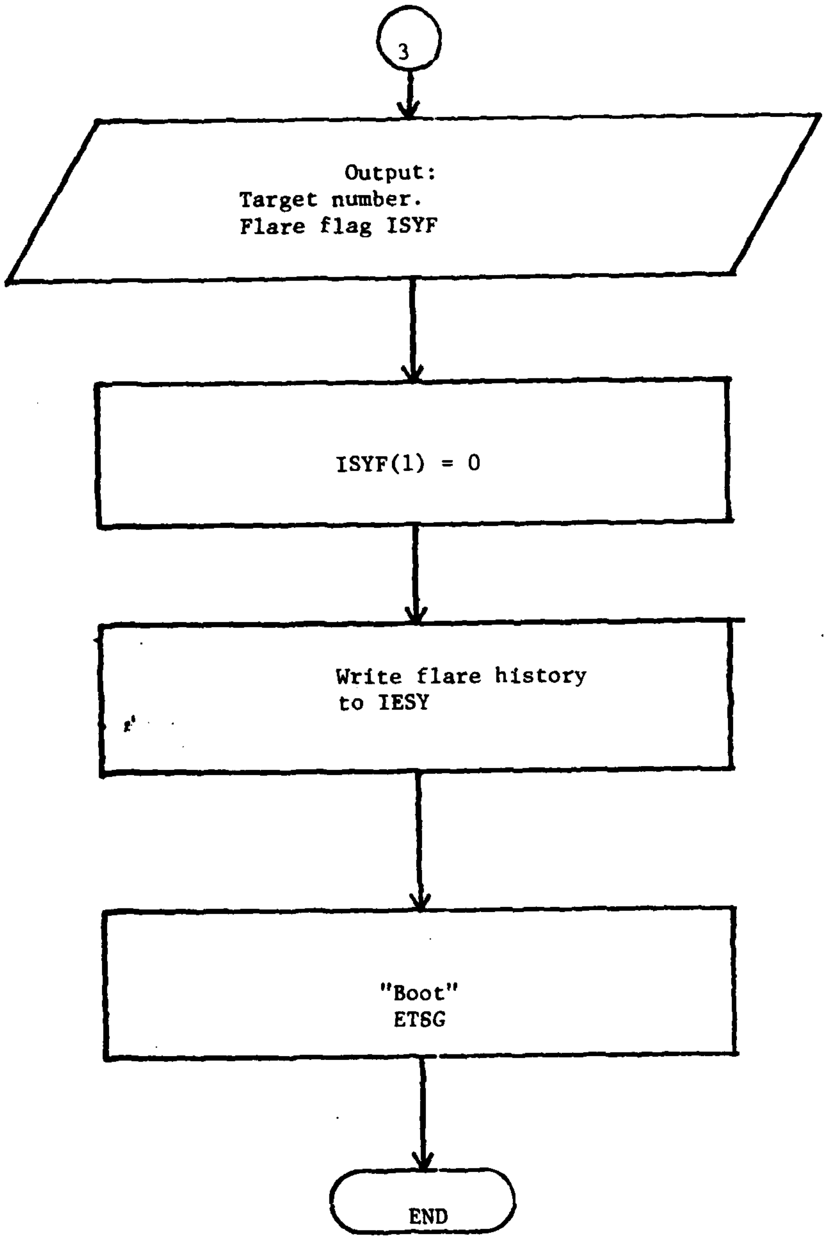


Figure 6 (Continued)

2.2.7 RUNETSG

RUNETSG transfers control from the initialization processor to the ETSG Hardware. It is the last program in the initialization sequence. Refer to Fig. 7 for a functional diagram.

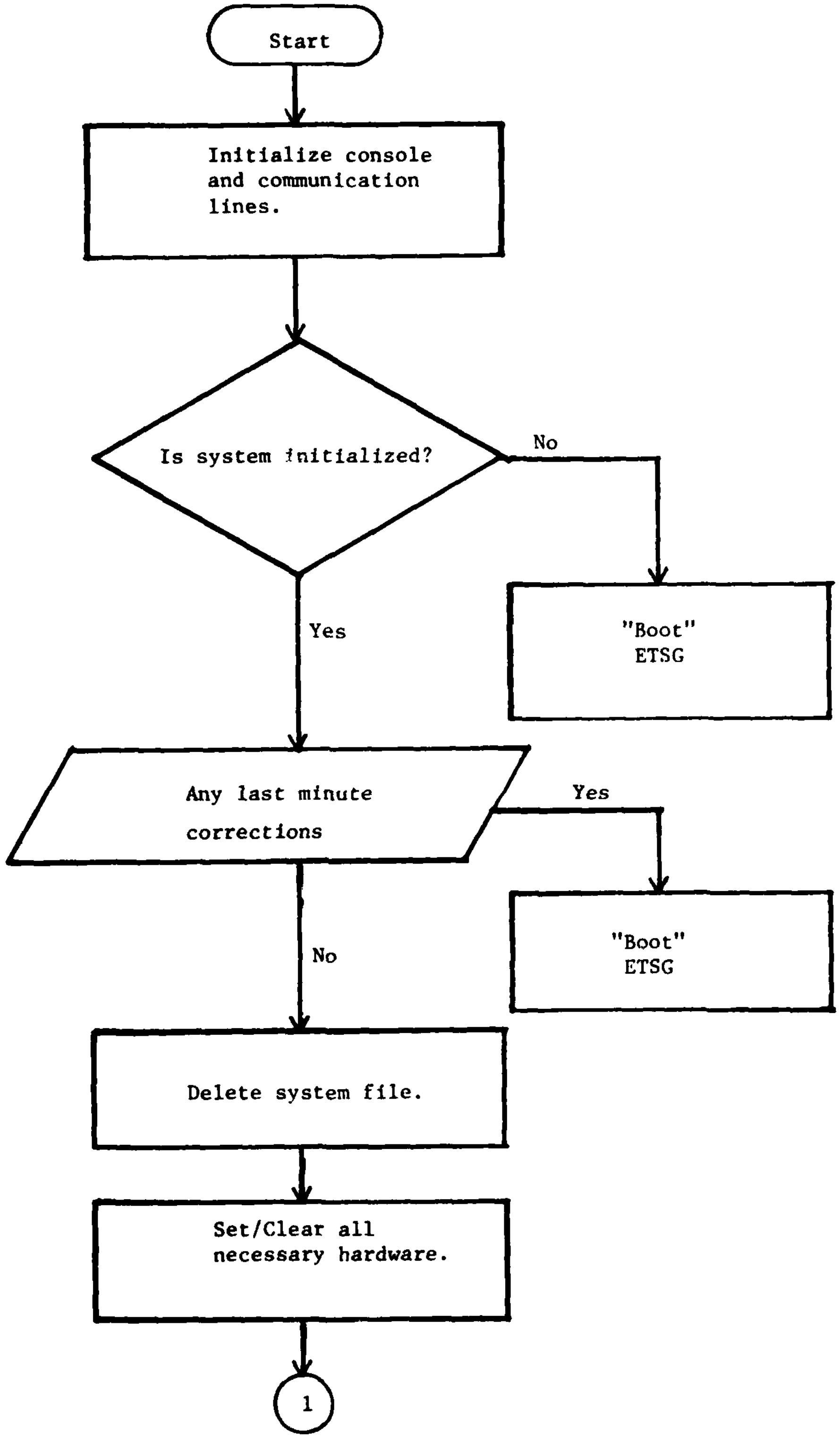


Figure 7 : RUNETSG

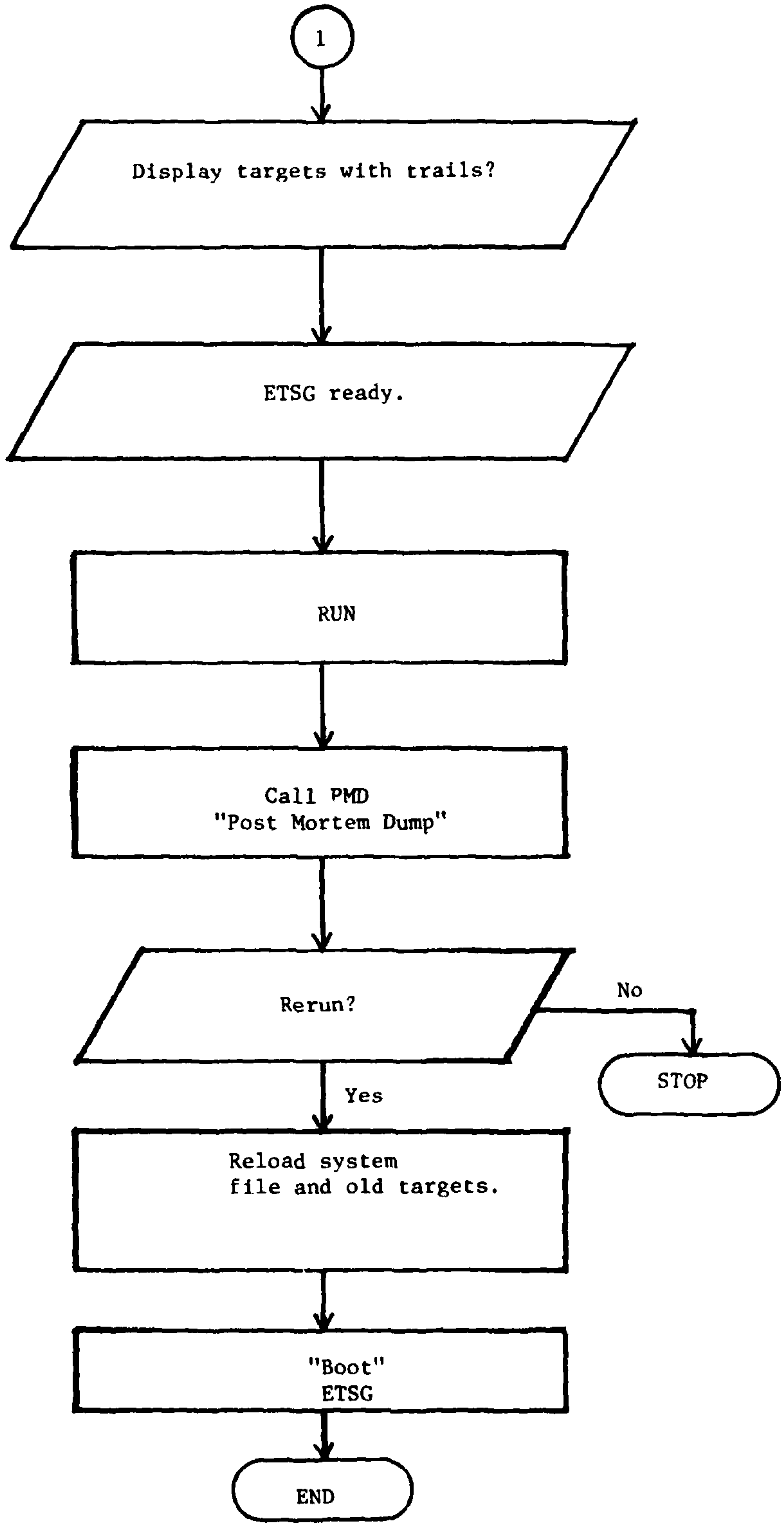


Figure 7 (Continued)

CHAPTER THREE

REAL-TIME COMPUTER MODEL (TARGET CPU SOFTWARE)

3.0 Introduction

Target CPU Driver, TCD is the main driver for the ETSG target CPU. Its primary purpose is to generate the coordinate values and step sizes for the target loaders. For the derivations and memory locations of target coordinate values and initialization interface variables calculated by the target CPU code refer to Appendices One and Two respectively.

TCD has three modes of operation, RUN, DEBUG with PRESET, and DEBUG.

RUN Mode

While operating in RUN mode, the target CPU is fulfilling its primary purpose. It is generally in this mode while the ETSG is running. While in this mode, it will use the dynamic variables supplied by the DCB, the static variables supplied by the IP, and some internal variables kept by *TCD* and generate the target loader/intensity factor output values.

The steps involved are:

1. Wait till data is supplied by the DCB.
2. Generate intensity scale factor.
3. Generate target loader values.
4. If target still valid (inside FOV, no intensity factor overflow, and target valid from DCB), load calculated values into latches. If target invalid, set target to point target outside FOV.
5. Go to step 1.

DEBUG with PRESET

If the debug target flag is set to 1, the target CPU will enter this mode of operation. While in this mode, the CPU will set the static variables usually set by the IP, preset all necessary local variables, and set up a block of memory for use of the debug target. The target CPU then enters the debug target mode.

DEBUG Target Mode

While in debug target mode, the actions of the target are directed by a block of memory. This block consists of 9 different variables:

1, 2) A rotation angle increment and period. These two numbers allow the target to rotate CW or CCW at any desired speed.

3, 4) A range decrement and period. These two numbers allow the target to *zoom* in and out (i.e., appear to grow and shrink) at a selected speed.

5) A range overshoot limit. As the range decrement value is usually positive, and the algorithm involved does not check for negative ranges, it will appear that the target has flown through the viewer. This value places a negative limit on the range.

6, 7) An aspect increment and period. These two allow the targets aspect value to change with any desired rate and direction.

8) A delay factor. This value is a delay to be placed at the end of any cycle. (Normally 0)

9) The control byte. This is the value that the DCB would usually place in the control byte at address 7. It contains the go flag, the flare flag, and an invalid target flag.

The steps involved are:

1) Delay for delay factor time.
2) If time period exhausted for rotation, change rotation angle by indicated amount.
3) If time period exhausted for range, change range by indicated amount.
4) If range is negative enough, reset range to positive value.
5) Ditto for aspect angle. If aspect angle went through edge value, change rotation by 180 degrees.
6) Set control byte to specified value.
7) Pretend to be RUN mode target.

3.1 Target CPU Code

The following is a listing of the subroutines which constitute the target CPU software. A brief description of each subroutine's function is presented. Those variables operated on by each subroutine are designated as "Entry" and those calculated are labeled "Exit." At the end of this section is a flow chart which shows the interaction between these subroutines.

TCD - Target CPU Drive - TCD provides the main line processing and start for the ETSG Target CPUs.

PRS - Preset - PRS clears all necessary internal variables to allow for correct initialization processor interaction. It will also remove the target from the field of view.

Exit (VALD) = 0, Not Valid
(DBUG) = 0, Not Debug
(CONT) = 0, Just in Case
(ERRF) = 0, No Errors Encountered Yet
(CYCL) = 0, No Cycles Finished Yet
Target outside of field of view

INT - Initialize, Preset, and Wait for Go - INT sets all variables to correct assumed values and waits for a go signal from the DCB.

Entry (VALD) = Valid Target From IP
(DBUG) = Debug Target From IP
(FS) = Flare Status
FS = 0, Flare Turned Off
= 1, Flare Turned On
= FF, Flare Turned Off by Program

Exit Go Signal Cleared. Flare Pointer Set If Flare Turned On

Calls IFH

RCK - Check Point Target - RCK determines whether or not the target is a point target. It also calls on L2R to calculate the range.

Entry (RRAN , RRAN + 1) = Resolution range (2 Bytes)
(RRAN , RRAN + 1) = Current Range (2 Bytes)

Exit (PT) = 00, If not a point target
01, If point target
(LR) = Log Base 2 (Range)

Uses A, B

Calls L2R

IFH - Initialize Flare History - IFH Sets the pointers for the flare history array.

Exit - (RC) = Repeat count for first value

(ST) = Step for first point

(IX) = Pointer for first point

(LS) = Current log output value

(TS) = (LS)

Uses A, X

L2R - Log Base 2 of the Range - L2R computes the log (Base 2) of the target range. The algorithm is as follows.

1. Find the largest bit set--this is the power of two for the number
2. Extract the next 6 bits, these are used as a fractional log. This entails a 64 byte lookup table

Entry (RANG, RANG + 1) = Range

Exit (LR + 0) = Interger (Log 2 (Range))

(LR + 1) = Fractional (Log 2 (Range))

Uses A, B, X, T0, T1

ISF - Calculate Intensity/Range Scale Factor - ISF calculates the basic intensity/range scaling factor. The ISF is determined by the following equations:

1. For resolved targets: ISF = EXP (-ALPHA*RANGE)
2. For half-resolved targets: (Unimplemented) ISF = EXP(-ALPHA*RANGE)/RANGE
3. For unresolved targets: ISF = EXP(-ALPHA*RANGE)/RANGE**2

In addition to the range scale factor,flares have a time loss factor. All calculations are based on the log (base 2) of the range. Conversion from logs to the ETSG Floating Point Number System (FNS) is trivial because the log is the FNS number to the first three bits, which is all that is necessary.

Entry - (ALZE) = Log2(E)*ALPHA

(ISFR) = Implied bias if resolved

(ISFP) = Implied bias if point target

Exit (ISFO) = ISF, FNS

Uses A, B, X, TO- Tl, T2, T3

Calls TSF

TSF - Time Scaling Factor - TSF calculates the time scaling factor. This is the value by which the TSF is to be decremented due to time (for flares)

Entry (RC) = Repeat Count for current LS value (ABPO)

(ST) = Current Sine Term (for corrections) (APBO)

(LS) = Log Scale Value (ABP8)

(TS) = Log Scale Value (Corrected) (ABP10)

(IX) = Current Flare Index (into table)

Exit - All above values updated

(A, B) = TSF Value. (ABP8)

Uses A, B, X, T4, T5

CAV - Calculate Aspect Values - CAV sets the X key point depending upon the current value of the aspect angle and determines the correct aspect ratio RAM to use. It will perform a table look up in TKPT to find the correct value of the keypoint.

Entry - (ASPC) = Aspect Angle

(PLUM) = Complex Target Flag

(PT) = Point Target Flag

Exit - (XK) = X key point

(YK) = Y key point

Uses TO, Tl, T2, T3

CXY - Calculate X and Y Coordinates - CXY calculates the X and Y coordinates of the keypoint for the TLR. This coordinate is just the Azimuth and elevation scaled upward by a predetermined scale factor

Entry (AZIM) = Target Azimuth

(ELEV) = Target Elevation

(ACSF) = Elevation/Azimuth Scale Factor

Exit (XC, XC + 1) = X Coordinate ABP5

(YC, YC + 1) = Y Coordinate ABP5

Uses A, X

CSS - Calculate Step Size - CSS calculates the X and Y step sizes used by the target loader to index into the target lookup RAM (TLR). These values are independent of the aspect angle.

Entry (RANGE, RANGE + 1) = Range of target (2 Bytes)
(FOVS) = Field of View Scaling Factor

Exit (XM, XM + 1) = TLR Step Size X with respect to X
(XN, XN + 1) = TLR Step Size X with respect to Y
(YM, YM + 1) = TLR Step Size Y with respect to X
(YN, YN + 1) = TLR Step Size Y with respect to Y

Uses TO

Calls TSC

CZC - Calculate Target Map Zero Coordinate - CZC calculates the value of the target map zero coordinate within the TLR coordinate system. The equations used for the coordinate transdformation are:

1. X' = -(XC*XM + YC*XN - XK)
2. Y' = -(XC*YM + YC*YN - YK)

Where XC = X coordinate of target (scaled azimuth)
YC - Y coordinate of target (scaled elevation)

Entry (XM) = DELTA XM
(XN) = DELTA XN
(YM) = DELTA YM
(YN) = DELTA YN
(XK) = Key Point X
(YK) = Key Point Y

Exit (XO) = Target Map Zero X Coordinate
(YO) - Target Map Zero Y Coordinate

Uses A, B, X, TO-T7

Calls TSC

TSC - Sine/Cosine Calculation Routine - TSC calculates sin/cos values for an angle. The angle is assumed to be an 8-bit positive number 0-255, which corresponds to an angle of 0-360 degrees.

Entry (A) = Angle

Exit (SN, SN + 1) = Sin(A) ABP14

(CS, CS + 1) = Cos(A) ABP14

Uses A, B, X, T3, T4, T5

TVT - Test Valid Target - TVT checks and insures that the target is in fact valid. If not, TVT sets the target out of the field of view and sets all step sizes to 0. This effectively removes the target from consideration.

Entry (VT) = Valid Target Flag

0 = Not Valid

1 = If Valid

LTL - Load Target Loader - LTL transfers to the target loader the following values:

1. The target map zero coordinate WRT to the TLR.
2. All four incremental values
3. Set aspect select values (13th latch)
4. Set complete bit (13th latch)

Uses A, B, X

ERR - Check Internal Errors - ERR performs a short self check to determine if any detectable errors have occurred. It check the following:

1. The Multiplier
2. RAMs T0-T7

If an error is detected, the CPU is hung

Exit (CYCL) = (CYCL) + 1

(ERRF) = (ERRF) + Applicable error flags.

Refer to Fig. 8 for a functional diagram.

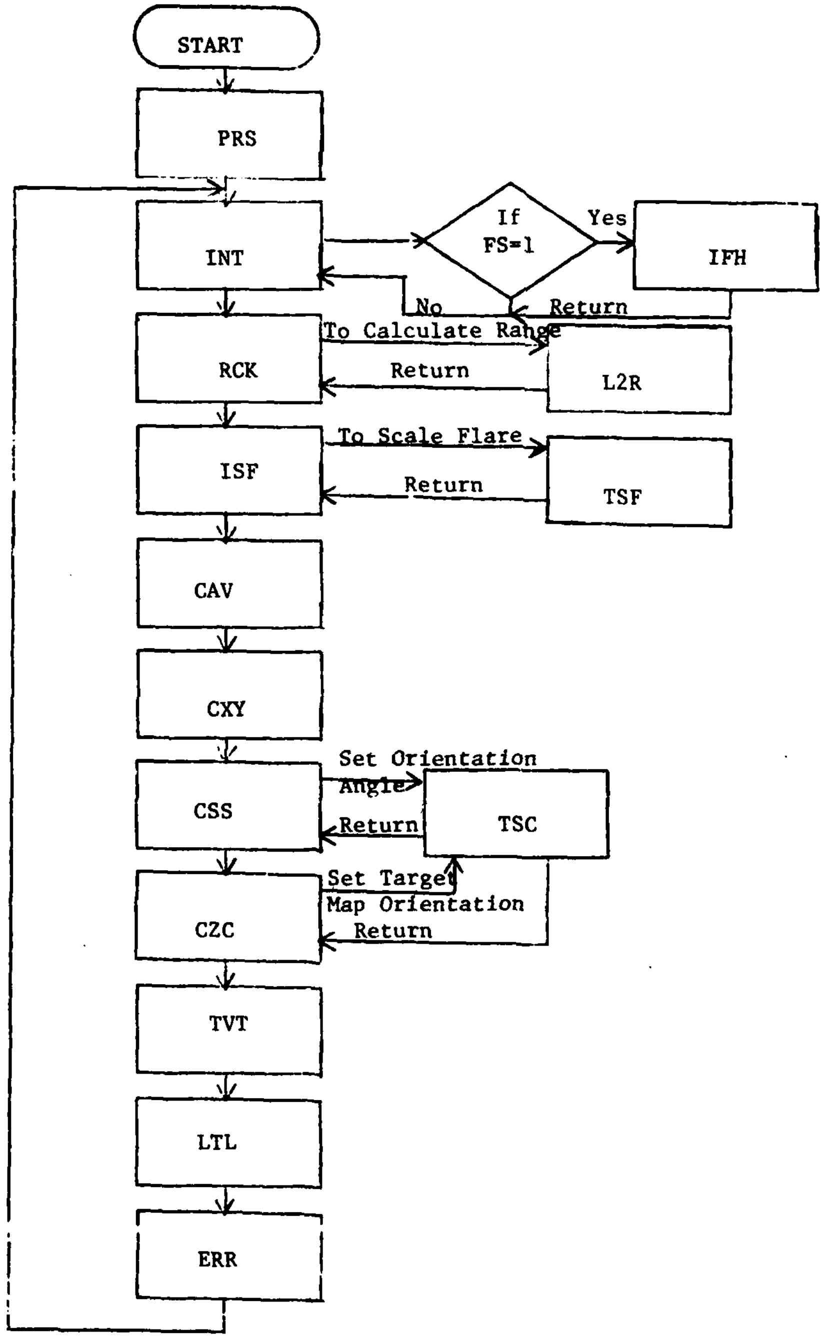


Figure 8 : Target CPU Code

CHAPTER FOUR

CONCLUSIONS AND RECOMMENDATIONS

4.0 Conclusions and Recommendations

The ETSG Target CPU firmware is complete and totally functional. The authors do not anticipate the need for firmware changes unless further development necessitates alterations in the coordinate transformation on target mapping algorithms. All changes made in Target CPU firmware as well as changes in Initialization software subsequent to May 5, 1981 are documented in the ETSG program listings and in the author's daily log.

The Initialization software has been revised from Motorola FORTRAN revision number 2.20 to FORTRAN 3.10. Some "debugging" is required for this most recent revision of the ETSG software.

It is our recommendation that the development of this software be continued and that the diagnostic software presentl" in development be completed.

REFERENCES

[1] Barnett, C. E. and Long, T. N., Integration of a Hybrid Simulation for a Small Air Defense Missile, Georgia Institute of Technology, Atlanta, Georgia, April 1980.

[2] Barnett, C. E., Long, T. N. and Wallace, C. T., Stinger-POST Hybrid Simulation Integration (U), Georgia Institute of Technology, Atlanta, Georgia, February 1981.

[3] Barnett, C. E., Simulation System Target Parameters and Logic Definition, Georgia Institute of Technology, Atlanta, Georgia, June 1980.

[4] Cantrell, Gerald and Kheir, N. A., A Digital Target Model for Use With a Stinger-POST Guidance Simulation, Final Technical Report The University of Alabama in Huntsville, Report No. 272, prepared for System Simulation and Development Directorate, US Army Missile Command, Redstone Arsenal, Alabama, Contract No. DAAK-40-79-D-0031, Delivery Order #0004, January 1981.

[5] Hudson, Richard D., Jr., Infrared System Engineering, Wiley, New York, 1969.

[6] Sinclair, M. J., Electronic Target Signal Generator (ETSG) Hardware Design and Fabrication, Georgia Institute of Technology, Atlanta, Georgia, April 1980.

[7] Sinclair, M. J., Electronic Target Signal Generator (ETSG) Integration and Test, Georgia Institute of Technology, Atlanta, Georgia, March 1981.

[8] Sinclair, M. J. and Riley, G. E., Electronic Target Signal Generator (ETSG), Design and Analysis, Georgia Institute of Technology, Atlanta Georgia, May 1980.

[9] Wolfe, W. L. and Zissis, G. J., The Infrared Handbook, Environmental Research Institute of Michigan, Michigan, 1978.

REFERENCES (Cont'd)

[10] Burt, R., and Kheir, N. A., Electronic Target Signal Generator (ETSG): Hardware Development, Final Technical Report, The University of Alabama in Huntsville, Under preparation for System Simulation and Development Directorate, US Army Missile Command, Redstone Arsenal, Alabama, Contract No. DAAH-01-81-D-A006, Delivery Order #0004.

Appendix I

Target Coordinate Variables

Variable Name	Description	Origin	Location
XO	Target Map Origin (TLR)	CZC	D051 D052
YO	Target Map Origin (TLR)	CZC	D053 D054
XK	X keypoint (TLR)	CAV	D055 D056
YK	Y keypoint (TLR)	CAV	D057 D058
XC	Current X Coordinate	CXY	D059 D05A
YC	Current Y Coordinate	CXY	D05B D05C
XM	TLR Step Size X wrt. X	CSS	D05D D05E
XN	TLR Step Size X wrt. Y	CSS	D05F D060
YM	TLR Step Size Y wrt. X	CSS	D061 D062
YN	TLR Step Size Y wrt. Y	CSS	D063 D064

Notes On Target Coordinate Calculations

These notes and equations may be used to verify that correct numbers are being calculated and stored in the target CPU RAM for target coordinate calculations. Substitution of the appropriate IP input variables and direct cell interface variables into the equations will generate correct values for each coordinate variable. Scale factors for the hardware multiply are absorbed into the equations. Care should be taken in applying these equations in order that number base conventions are observed. All numbers in the equations are decimal or base ten numbers. All results must be converted to hexadecimal.

CXY

The subroutine LSHL operates on FNUMB and ISHFT and yields the results IRSLT and ILEFT.

FNUMB = IRND(NPWN*FOV/(BLRM*180*10^{-3}/ π))/FOVD/32

All unknowns are IP input variables.

ISHFT = 15

IRSLT = ACSF	D094 D095
ILEFT = SRAC - 7	
CNTX = 8448*2^{7-SRAC}	D098 D099
CNTY = 8192*2^{7-SRAC}	D09A D09B
XC = (ACSF*AZIM)/2^{15} + CNTX	D059 D05A
YC = (ACSF*ELEV)/2^{15} + CNTY	D05B D05C

AZIM and ELEV are direct cell interface variables.

XK = 0 For a point target PT = 0 D055

D056

YK = 0 D057

D058

The values for XK are calculated by ASPGEN and stored in a table location \$0200 + $|ASPC|$/4.

PPM = 63/TSZX Points/meter for simple target.

PPM = $63/\sqrt{(TSZY/2)^2 + TSZX^2}$ Points/meter for a complex target.

TSZX and TSZY are IP input variables.

XK = PPM*RKXM*16*SIN(ACOS(($|ASPC|$/4 - 1)/16))

YK = KEYY = KYP = PPM*RKYM*16

RKXM and RKYM are initialization input variables via ETARG.

ASPC is a direct cell interface variable.

CSS

To compute COS(ORNT);

1. Convert ORNT to decimal
2. $ORNT_{10} * 1.41$
3. $COS\ (1.41*ORNT_{10})$

$SS = FOVS*RANG*2^{SRFV}$

$SS = RANG*2^{5}*64/NPWN/(TSZX/BLRM*1000)$

ORNT and RANG are direct cell interface variables. All other unknowns are IP input variables.

$XM = SS*COS(ORNT)*2^{-15}$	D05D D05E
$YM = -SS*SIN(ORNT)*2^{-15}$	D061 D062
$XN = SS*SIN(ORNT)*2^{-15}$	D05F D060
$YN = SS*COS(ORNT)*2^{-15}$	D063 D064

CZC

$$X0 = -((XC*XM + YC*XN)*2^{SRAC-15} - XK)$$ D051

D052

$$Y0 = -((XC*YM + YC*YN)*2^{SRAC-15} - YK)$$ D053

D054

Appendix II
Initialization Interface Variables

Initialization Interface Variables

DBUG	Debug target flag	(1,FF=DEBUG)	D080
VALD	Valid target flag	(1 = Valid)	D081
FLAR	Target flare flag	(1 = Flare)	D082
PLUM	Complex target flag	(1 = Complex)	D083
CYCL	Cycle count		D09E
ERRF	Error flag		D09F

RRAN Resolution Range D084 D085

RCO = TSZX/BLRM*1000

RRAN = RCO*NPWN/(NPWN + 1)

LRAN Linear Resolution Range D086 D087

(not used)

KEYY Y Key Point D088 D089

IMAX = 64

Simple target	Complex target
T1 = TSZX	T1 = TSZY*TSZY/4 + TSZX*TSZX
PPM = (IMAX-1)/T1	PPM = (IMAX - 1)/SQRT(T1)

KEYY = PPM*RKYM*16

ISFR Intensity Scale Factor Pias for Resolved Target D08A D08B

CO = 64

CM2PM2 = 1.0E - 4

SMSY = RNEFD*SNRT

```
     RCO = TSZX/BLRM*1000
     REAVG = PAVG/PMN(4)*9
     RJJP = IPOLTY*(RJT-BKRD*100000*EXP(ATTN/1000))
      C5 = CO*CM2PM2/SMNSY/NPWN/NPWN
ISFR = IRND(((ALOG(C5*RJJP/RCO/RCO/RFAVG)/ALOG/2))*2**8
```

ISFP	Intensity Scale Factor Bias for a Point Target	D08C D08D

```
          CST = 9.0
           CO = 64
       CM2PM2 = 1.0 E-4
        SMNSY = RNEFD*SNRT
           C5 = CO*CM2PM2/SMNSY/NPWN/NPWN
         RJJP = (RJT - BKRD*10000*EXP(ATTN/1000)*IPOLTY
         ISFP = IRND((ALOG(C5*RJJP/CST)/ALOG(2))*2**8
```

AL2E	ALPHA*LOG2(E)	D08E D08F

$1/\ln(2) = \log_2(e) = 1.442695041$

$2^{22} = 4194304$

AL2E = 1.442695041*ATTN/1000*4194304

FOVS	Field of View Scaling Factor	D090 D091
SRFV	20 Shift Applied to FOVS	D092 D093

RCO = TSZX/BLRM*1000

FRMP = 64.0/NPWN/RCO

LSHL/FTMP,20)

FTMP = FOVS*$2^{SRFV-20}$

ACSF	Angle to Coordinate Scale Factor	D094 D095

DPM = 0.0572957795

BLRD = BLRM*DPM

NFOV = IRND(NPWN*FOVD/BLRD)

FPPD = NFOV/FOVD

TCDPC = 4/128

TCPPC = FPPD*TCDPC

LSHL(TCPPC,15)

$TCPPC = ACSF*2^{20-JTMP}$

SRAC	22 Shift Applied to ACSF	D096 D097
	SRAC = JTMP + 7	
CNTX	Shifted X Center Coordinate	D098 D099

CNTX = 528*16/2**JTMP

CNTY	Shifted Y Center Coordinate	D09A D09B
PTSS	Point Target Step Size	D09C D09D

PTSS = 64/NPWN*2**4

Appendix III

System Flags

SYSTEM FLAGS

```
ITSW        = IFLGS(1) TARGET TYPE
ISVSW       = IFLGS(2) TARGET GEOMETRY FOR COMPLEX TARGET SIDE VIEW
ISC         = IFLGS(3) TARGET TYPE 1= LONG WAVE,K-CHL 2= SHORT WAVE,J-CHL
IPOLTY      = IFLGS(4) POLARITY
IPJ         = IFLGS(5) PLUSE JAMMER  FLAG
IFL         = IFLGS(6) FLARE FLAG
IPRI        = IFLGS(7) PRIORITY
IPLM        = IFLGS(8) PLUME FLAG
ITN         = IFLGS(9) VIEW NUMBER FOR COMPLEX TARGET
ITCLR       = IFLGS(10) TARGET COLOR
ISRUT       = IFLGS(11) TRUE TARGET FLAG
IGLISW(1)   = IFLGS(12)
IGLISW(2)   = IFLGS(13) TARGET INTENSITY GRADIENT FLAG
IGLISW(3)   = IFLGS(14)
ISKRSV      = IFLGS(15) SEEKER CHECK VALUE FROM STTP
IRCSW       = IFLGS(16) 1=ROSETTE 2=CONSCAN
NPWN        = IFLGS(17) NUMBER OF POINTS IN ONE DIMENSION OF BLUR DIAMETE
MXSCR       = IFLGS(18) MAXIMUM SCAN RATE FOR CONSCAN
NFSC(1)     = IFLGS(19) SCALE TO NEFD CHANNEL 1
NFSC(2)     = IFLGS(20) SCALE TO NEFD CHANNEL 2
ISROT       = IFLGS(21) CONSCAN SEEKER ROTATION
ISKRCK      = IFLGS(22) SEEKER CHECKSUM VALUE
```

SYSTEM FLAGS

```
ISYF( 1) = SYSTEM ERROR FLAG
ISYF( 2) = MANUAL SELECT FLAG
ISYF( 3) = AUTO SEQUENCE NUMBER
ISYF( 4) = STROBE EXISTS
ISYF( 5) = FLARES EXIST
ISYF( 6) = NUMBER OF FLARES
ISYF( 7) = TARGET  1 IS A FLARE
ISYF( 8) = TARGET  2 IS A FLARE
ISYF( 9) = TARGET  3 IS A FLARE
ISYF(10) = TARGET  4 IS A FLARE
ISYF(11) = TARGET  5 IS A FLARE
ISYF(12) = TARGET  6 IS A FLARE
ISYF(13) = TARGET  7 IS A FLARE
ISYF(14) = TARGET  8 IS A FLARE
ISYF(15) = TARGET  9 IS A FLARE
ISYF(16) = TARGET 10 IS A FLARE
ISYF(17) = TARGET 11 IS A FLARE
ISYF(18) = TARGET 12 IS A FLARE
ISYF(19) = TARGET 13 IS A FLARE
ISYF(20) = TARGET 14 IS A FLARE
ISYF(21) = TARGET 15 IS A FLARE
ISYF(22) = TARGET 16 IS A FLARE
ISYF(23) = TARGET 17 IS A FLARE
ISYF(24) = TARGET 18 IS A FLARE
ISYF(25) = TARGET 19 IS A FLARE
ISYF(26) = TARGET 20 IS A FLARE
ISYF(40) = SEEKER TYPE
```

Appendix IV

Target Parameters

TARGET PARAMETERS

```
TSZX = TRG(1)  TARGET SIZE X (METERS)
TSZY = TRG(2)  TARGET SIZE Y  (METERS)
TAR  = TRG(3)  TARGET ASPECT RATIO
RJT  = TRG(4)  TARGET RADIANCE (WATTS/STERADIANS)
RKXM = TRG(5)  X KEY POINT (METERS)
RKYM = TRG(6)  Y KEY POINT (METERS)
RMAX = TRG(7)  MAXIMUM RANGE (METERS)
RMIN = TRG(8)  MINIMUM RANGE (METERS)
PPM  = TRG(9)  POINTS PER METER IN T.L.R.
RC1  = TRG(10) RESOLUTION RANGE (METERS)
RMNR = TRG(11) RANGE OF 1:1 RESOLUTION (METERS)
RJTP = TRG(12) CONTRAST RADIANCE (WATTS/STERADIANS)
```

Appendix V

Intensity Data

INTENSITY DATA

** I = IPLMTH

```
CN(1,I) = PEAK         [ALL]
CN(2,I) = EDGE         [EX(T),E(R,E)]
CN(3,I) = EDGE         [Y(T ONLY)]
CN(4,I) = BREAK PT     [XB(T),BRK(E),YB(R)]
CN(5,I) = BRK VALUE    [EX(T),E(R,E)]
CN(6,I) = BREAK PT     [YB(T ONLY)]
CN(7,I) = BRK VALUE    [EY(T ONLY)]
```

ACCUMULATED INTENSITY VALUES

```
ZSUM = ZS(1,I)      [ I=4, TOTAL FOR ALL VIEWS ]
ZCNT = ZS(2,I)      [ I=4, TOTAL FOR ALL VIEWS ]
PMX  = ZS(3,I)      [ I=4, MAX FOR ALL VIEWS   ]
PMN  = ZS(4,I)      [ I=4, MIN FOR ALL VIEWS   ]
```

Appendix VI

Seeker Parameters

SEEKER PARAMETERS

```
RNEFD(I) = RESP(I,1) NOISE EQUIVALENT FLUX DENSITY (WATTS/CM↑2)
ARES(I)  = RESP(I,2) SYSTEM RESPONSIVITY (VOLTS/WATTS/CM↑2))
BKRD(I)  = RESP(I,3) BACKGROUND IRRADIANCE (WATTS/CM↑2)
ATTN(I)  = RESP(I,4) ATMOSPHERIC ATTENUATION COEFFICIENT (1/KM)
SNRT(I)  = RESP(I,5) SIGNAL TO NOISE RATIO TO TRACK
ANOIZ(I) = RESP(I,6) SYSTEM NOISE LEVEL
SIGMN(I) = RESP(I,7) MINIMUM SIGNAL AT APERTURE
C5       = RESP(I,8) SEEKER IRRADIENCE TO FNS SCALE FACTOR
```

Appendix VII

Field of View Data

FIELD OF VIEW DATA

```
FOV(1) = FOVD [R];RFOVD [C]
FOV(2) = BLRM      [R,C] BLUR DIAMETER (MILLIRADIANS)
FOV(3) = TDPC      [R,C] TARGET IEGREES PER COUNT
FOV(4) = --,ENTR  [C]
FOV(5) = --,EFOVD [C]
FOV(6) = TCPPC     [R,C] TARGET COORDINATE POINTS PER COUNT
FOV(7) =
FOV(8) =
```

Appendix VIII

Initialization Processor Subroutines

Compiled by Donn Hall

Initialization Processor Subroutines

ACOS - Arccosine Function

Input (x)

ADFLT - Real Array Default Function

Input (A, I, J, M)

ALP - Argument List Processor - ALP is an assembly routine which is designed to process the argument list of an abortran subroutine.

Input - (A) = Number of Arguments.

ANMD - Set Alpha-Numeric Mode

APKT - Intensity Target Display

Input - (IA, IR, MX, NLVLS)

Call - (INIT, PAGE, GREY)

ASIN - Arcsine Function

Input (x)

ASPGEN - Plume Aspect Generator

Inputs - (AR, KEYX) Input files - (IFLGS, SA:B, TRG.SA:0)

Call - (LDASF, LDTCPO)

AXES - Flare History Display

Inputs (X, Y, N)

Call - (INIT, PAGE, PLOT, CRSR)

BELL - Sound 150 MS BELL

BLNK - Set/Clear Blink Mode

Input (OP) OP = 1, Bunk mode on : OP = 0 Clear Blink Mode

CKINIT - Check initialization - CKINIT is a FORTRAN callable assembly routine which is designed to check if a total system initialization is in order. A system initialization may be necessary for any of the following reasons:

1. A power up restart has been done on the ETSG
2. A power failure which cause reset of the PIAS
3. A hardware abort (restart)

If any of these three reasons are present, CKINIT will initialize all PIAS and return a initialize required FLAS.

Input - (IVAL) = 0, if initialization was necessary

1, if initialization was not necessary

Call - (ALP, TPIAS)

CLRTMP - Clear Target Map - CLRTMP is a FORTRAN callable assembly routine designed to reset the target maps to a clear state. It will write zeros to both halfs of both channels of the target map.

CNVERT - Convert ETSG Floating Point to MOTORODA Floating Point

Input - (INUMB)

COLR - Set Background/Foreground Color

Input - (BACK, FORE)

CPS - Check Plot Status - This subroutine checks to see if the terminal is currently in plot mode or in a plot submode, but leaves "PLTF" set CPS always leaves the interface plot mode set (PLTF)

CRSR - Set Cursur Position

Input (COLM, LINE)

DIR - Directory of SEEKERS and TARGETS - DIR produces a listing of all TARGETS and SEEKERS previously recorded in memory.

DLY - Delay For Specified Time. - DLY will wait for a specified time. This delay is in increments of 10 incroseconds with a minimum of 40 microseconds delay.

Input - (B) B = Number of 10 microsecond delays

DPLX - Set Half/Full Duplex

Input - (MODE) MODE = 0, Half Duplex: MODE = 1, Full Duplex

DRC - Draw Boresight Circles - DRC Draws two circles on the monitor/display. The routine is entirely table driven. All values for the X/Y coordinate values for the circle points have been precalculated

Call - (TCRD/OUT)

DRX - Draw Boresight Crosshairs - DRX places A "+" in the center of the monitor display.

ETARG - ETSG Target Generator Program - ETARG, in cooperation with the user, sets all the static parameters for a given target.

Input Files - (IFLGS. SA:0, CN.SA:0, TRG.SA:0, FOV,SA:0, RESP.SA:0,
ZS.SA:0, ETSG.CM:0, ESYS.SA:0, DSKR.SA:0 SCROIJ7Z.QR:0)

Call - (FILTST, MLOAD, STTP, BELL, INVERT, LDPTIG, PAGE, GENTRG, DELF,
ASPGEN, STTGCH, STTSGN, STSTBB, LDTCPV, LSHL, LDDSPC)

ETSG - Driver For ETSG Initialization - ETSG initializes, in cooperation with the operator (user), all seeker and target static parameters by call ins other subroutines.

Input Files - (ESYS:SA:0, DSKR.SA:0, Various user defined variables,
Seek.CM:0, ETARG.CM:0, PULSE.CM:0, FLARE.CM:0, RUNETSG.
CM:0)

Call - (INIT, PAGE, BELL, CKINCT, FILTST, INITCP, MLOAD, DELF, SEEK,
ETARG, PULSE, FLARE, RUNETSG.

ETSGGO - Set Ready/Run Modes

Call - (READY, RUN)

FLAG - Set/Clear Flag (Enable/Disable Erase)

Input - (IFLAG) IFLAG = 0 Clear; IFLAG = 1 Set

FLARE - FLARE Generation Program - FLARE sets all parameters for flare type targets.

Input Files - (ESYS.SA:0, ETSG.CM:0)

Call - (KEYIN, INIT, PAGE, BELL, FILTST, MLOAD, AXES, LDTCPU)

GENTRG - General Target Generator - GENTRG is called by "ETARG" to produce the targe- image based upon the parameters set in "ETARG".

Input Files (IFLGS.SA:0, CN.SA:0, TRG.SA:0, ZS.SA:0

Input - (ITYPE, SIZEX, SIZEY, IFLZ)

Call - (MX, IRND, INVERT, OUTFLT, SAVTRG)

GRAPH -

Input - (JSIG,N)

Call - (PAGE, PLOT, CRSR)

GREY - Provide GREY Scale Character.

Input (IX, IY, IV)

IX = Character Column (See CRSR)

IY = Character Line (See CRSR)

IV = GREY Scale Valve (1 to 55)

GRSC - GREY Scale Value (Table)

IADET - Integer Array Default Function

Input - (I,A,I,J,M)

INIT - Initialize Plotting Package

INITCP - Initialize CPU

Call - (LDDSPC, LDTCPU)

INVERT - Convert Motorola Floating Point Numbers to ETSG Floating Point Numbers and return the result as an Integer.

Input - (RNUMB)

Call - (SAA)

IRND - Real to Integer Rounding Function

Input - (X)

LDASP - Load Target Aspect Ram - LDASP is a FORTRAN callable routine designed to transfer data from the initialization processor to a select target CPU aspect Ram

Input - (ITRGT, IVIEW, IARRY)

Call - (ALP, SEA, MDV, CEA)

LDDSPC - Load Display Processor - LDDSPC is a FORTRAN callable routine designed to transfer data from the initialization processor to the display processor of the ETSG system. It also presets other values for the display CPU.

Input - (ITARG, ICOLR, MINAR, MAXAR)

Input - (0, ISCLF, 0, 0)

Call - (ALP, SEA, MDV, CEA)

LDNTRR _ Load Null Track Radios Ram - LDNTRR is a FORTRAN callable routine designed to transfer data from the initialization processor to the null track (reticle rotation) rams of the ETSG system.

Input - (ICHNL, ICONT, IDATA)

Call - (ALP, SEA, MDV, CEA)

LDPLSJ - Load Pulse Jammer - LDPLSJ is a FORTRAN vallable routine designed to enable the initialization processor to load the bit pattern used to describe the pulse jammer for the ETSG system.

Input - (IARRAY, NWORDS)

Call - (ALP, SEA, CEA)

LDPTTG - Load Target Lookup Ram with a Point Target - LDPTTG is a FORTRAN callable routine designed for transfer data from the initialization processor to a selected target CPUs lookup ram point target

Input - (ITARG, IDATA)

Call - (ALP, SEA, MDV, CEA)

LDRET - Load Reticle Maps - LDRET is a FORTRAN callable routine designed to transfer data from the initialization processor to the reticle maps of the ETSG system.

Input - (ICHNL, IDATA)

Call - (ALP, SEA, MDV, CEA)

LSHL - Left Shift with Limit - LSHL will shift a given floating point number left up to a supplied number of bits while retaining integer value limits on the result.

Input - (FNUM, ISHFT) - Output - (IRSLT, ILEFT)

Call - (ALP)

LDTCPU - Load Target CPU - LDTCPU is a FORTRAN callable routine designed to transfer data from the initialization processor to a selected target processor of the ETSG system.

Input - (ITARG, IARRY, NWORD, IOFFS, ISIZE)

Call - (ALP, SEA, MDV, CEA)

<u>LDTLR</u> - (Load Target Lookup Ram) - LDTLR is a FORTRAN callable routine designed to transfer data from the initialization processor to a selected target CPUS lookup ram

Input - (ITRGT, IARRY, IVIEW, IROWN)

Call - (ALP, SEA, MDV, CEA)

<u>NRT</u> - Null Track Radius - Generates coordinates for null track radius hardware

Input - (NFOVR, ENTR, ISROT)

Call - (LDNTRR)

<u>OUT</u> - Output Character to Monitor - Out ships one character to the monitor with a delay of 53 MS. If this is insufficient time for the control character in question, a further delay must be implemented.

Input - (A) A = Character to send

Calls - (DLY)

<u>OUTC</u> - Ship Character to Intecolor (Terminal)

Input - (A) A = Character to ship)

<u>OUTP</u> - Output Character with Programmable Delay

<u>OUTPLT</u> - Output Subroutine for Display

Input - (IPTG, JJ, IPKP, NLVLS)

Call - (LDTLR, APKT, PKT)

<u>OUTS</u> - Output Character with Standard Delay

Input (A)

<u>PAGE</u> - Clear Screen

<u>PCT</u> - Reticle Point Counter - Counts the number of points in the reticle to insure that it does not exceed the field of view

Input - (IA,IR) IA = Total field of view

IR = Radius of reticle (if the scan is a square scan

IR = Half the width of scan)

TART,ETARG sub module)

<u>PICT</u> - Target Display

Input - (IA, IR, MX, NLVLS)

Call - (INIT, PAGE, COLR, PLOT, DRSR, TEXT, GREY, ANMD)

<u>PLOT</u> - ETSG Plotting Package (Driver Routine)

Input - (ARG1, ARG2, ARG3)

Call - (ANMD, BELL, BLNK, COLR, CRSR, DPLX, FLAG, GREY, INIT, PAGE, PLOT, ROLL, TEXT, ALP, BSCT)

<u>PLOT</u> - Move * Pen * To (X, Y) coordinates

Input - (X,Y,P)

X = X coordinate value (0 to 159)

Y = Y coordinate value (0 to 191)

P = Z Move * Pen * Down P = 3 Move * Pen * Up.

PMD - Parameter Mapping and Overflow check - PMD initializes CRT, loads post processing data, prints headings, displays data, checks rosette limits, checks intensity overflow flags

Input - (   )

Call - (INIT, PAGE, RDDSPC, RDTCPU, LDTCPU, BI)

PMS - Plot Mode Start - PMS is called to initiate interface plot mode

Call - (OUTC)

PMT - Plot Mode Terminate - PMT is called to terminate the interface plot mode

Call - (OUTC)

PRS - Process Preset - PRS initializes the ACIA for terminal I/O and programs the PIA

Call - (OUT)

PULSEJ - Pulse Jammer (Strobe) History Generator - PULSEJ generates all necessary parameters for pulse jammer (Strobe) targets

Input Files - (ESYS.SA:0, ETSG.CM:0)

Call - (KEYIN, INIT, BELL, PAGE, FILTST, MLOAD, GRAPH, LDPLSJ, CRSR)

RDDSPC - Read Display CPU - RDDSPC is a FORTRAN callable routine designed to transfer data from the display processor to the initialization processor after an ETSG run

Input - (IFLAG, IMNAR, IMINR, IMAXR, IMXAR)

Call - (ALP, SEA, MDV, CEA)

RDTCPU - Read Target CPU - RDTCPU is a FORTRAN callable routine designed to transfer data from a selected target CPU to the initialization processor after an ETSG run.

Input - (ITRGT, IARRY, NWORD, IDFFS, ISIZE)

Call - (ALP, SEA, MDV, CEA)

RDTMP - Read Target Map - RDTMP is a FORTRAN callable routine designed to read the target maps one line at a time.

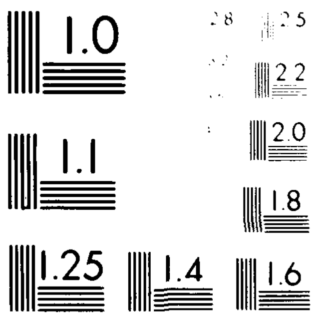
Input - (ICHNL, ITMAP, ILINE, IARRY)

Call - (ALP)

AD-A112 149 ALABAMA UNIV IN HUNTSVILLE SCHOOL OF SCIENCE AND ENG--ETC F/G 14/2
ELECTRONIC TARGET SIGNAL GENERATOR (ETSG) SOFTWARE DEVELOPMENT.(U)
OCT 81 P F PRITCHETT, N A KHEIR DAAH01-81-D-A006
UNCLASSIFIED UAH-296 DRSMI-RD-CR-82-4 NL

2 OF 2
AD A
112149

END
DATE
FILMED
04-82
DTIC



MICROCOPY RESOLUTION TEST CHART

READY - Enable ETSG to Run - READY is a routine which will set the ETSG in run mode and set the READY line (To the CDC 6600) high.

RETGEN - Multi-Size Reticle Generator - RETGEN creates a reticle of the size asked for by the user.

Input - (NPTS) NPTS = Number of points for width

Call - (PCT, CRSR, PAGE, FILTST)

ROLL - Set Terminal in Roll Mode

RPS - Restore Plot Status - RPS sets the terminal in the plot submode specified by "PLTF." Used in conjunction with "CPS" it allows a non-plot function to be issued from within a plot mode. If plot sub-mode is specified, the interface plot is left set.

Input - (PLTF) PLTF = Plot submode desired

RUN - Final terminal preparation - RUN is the last routine called by the ETSG initialization software. It prepares the terminal for the run and turns control over to the display processor. Control is returned to the calling routine when the terminal is once again handed over to the initialization software. Final terminal preparation consist of the following:

1. Clear the screen
2. Draw two concentric circles (FOV Representations)
3. Draw crosshairs between the circle

Call - (DRC, DRX)

RUNETSG - Initialize system to run - After completion of target and seeker loading "RUNETSG" initializes the system to run. When initialization is complete a command is sent to the monitor allowing the user to start the run

Input Files - (ESYS.SA:0, ETSG.CM:0)

Call - (KEYIN, INIT, PAGE, BELL, FILTST, BOOT, OPENF, DELF, LDTCPU, CLRTMP, STSEEK, READY, READA, RUN, CRSR, PMD, CLOSEF)

SAA - Set Argument Addresses - SAA is a routine that sets aside an address for the result of an arithmetic process and enables that result to be read back into the calling routine

Input - (RSLT)

SEEK - Set Seeker Parameters - SEEK, in cooperation with the user, sets all static parameters for the seeker

Input File - (IFLGS.SA:0, RESP.SA:0, FOV.SA:0, DSKR.SA:0, DROS.SA:0, ICON.SA:0, ESYS.SA:0, ETSG.CM:0)

Call - (KEYIN, INIT, PAGE, BELL, FILTST, DIR.LDDSPC, STAROS, NTR, RETGEN, STOAC, MLOAD)

SAVTRG - Save Target Parameters and/or Image

SRM - Set Run Mode - SRM is called to terminate the initialization process. It turns the display over to the display processor, and starts the run.

Call - (OUT, DLY, INIT)

STAROS - Set Rosette Scan X/Y Amplitudes - STAROS is a FORTRAN callable routine which will allow the initialization processor to set the amplitude (MIN/MAX, X and Y values) for the rosette scan

Input - (IXMIN, IXMAX, IYMIN, IYMAX)

Call - (ALP)

STOAC - Set Analog Output Controls - STOAC programs the PIAs, DACs etc., which controls the analog output of the ETSG. The values set by STOAC include:

1. The background level
2. The noise source level
3. The analog scale factor adjust

Input - (ICHNL, IPBGL, IPNSL, IASFA, IESFA, IESFC)

Call - (ALP)

STSEEK - Set Seeker Type - STSEEK is the ETSG interface with the PIA that controls the simulated seeker type the ETSG is currently rising

Input - (ITYPE)

Call - (ALP)

STSTRB - Set Strobe Flag for Target

Input - (ITRGT, ISTRB)

STTGCH - Set Target Channel

Input - (ITRGT, ICHNL)

STTGPM - Set Strobe Flag, Target Channel and TARG Polarity, (DRIVER, ROUTINE)

Call - (STTGCH, STTSGN, STSTRB)

STTP - Set Target Type and Priority - STTP sets the target type, priority, and generation flags.

Input Files - (IFLGS.SA:0, DTRI.SA:0, DELL.SA:0, DRECT.SA:0, DPLUM.SA:0)

Call - (FILTST, DIR)

STTSGN - Set Target Sign

Input - (ITRGT, ISIGN)

TEXT - Send Text to Terminal - This subroutine ships characters to the terminal bypassing the FORTRAN I/O package--This allows cursur addressing of text on the screen (Via CRSR)

Input - (INFO, NUMB)

TPIAS - Table of PIAS to Initialize

TRCD - Table of Coordinates for Boresight Circles

Appendix IX

ETSG.BAS

A Basic Program Which Emulates Some Internal ETSG Functions

Developed by

Paul F. Pritchett

and

Donn Hall

```
10DIMX(100),Y(100),A(10),A$(4),B(8),D(4),H$(4),W(4),W$(4),C(4,1)
20 DIM X$(4),Y$(4),F$(4),G$(4)
30PRINT"THE FOLLOWING IS A LIST OF SUBROUTINES EMULATING THE ";
40PRINT"ETSG SOFTWARE.  TYPE IN THE NUMBER CORRESPONDING TO THE ";
50PRINT"SUBROUTINE YOU WANT TO RUN."
60PRINT"1.  THE MULTIPLIER."
70PRINT"2.  HEXADECIMAL TO DECIMAL CONVERTER."
80PRINT"3.  DECIMAL TO HEXADECIMAL CONVERTER."
90PRINT"4.  BINARY TO DECIMAL CONVERTER."
100PRINT"5.  DECIMAL TO BINARY CONVERTER."
110PRINT"6.  DECIMAL TO EBS CONVERTER."
120PRINT"7.  EBS TO DECIMAL CONVERTER."
130PRINT"8.  SUBROUTINE GSHO."
140PRINT"9.  SUBROUTINE IRNO."
150PRINT"10.  SUBROUTINE CAV."
160PRINT"11.  SUBROUTINE CSS."
170PRINT"12.  SUBROUTINE CXY."
180PRINT"13.  SUBROUTINE CZC."
190INPUTX(100)
200ONX(100)GOSUB280,480,790,1090,1210,1330,1640,1810,1920
210 [illegible](100)=[illegible]270,2120,270
220PRINT"<12> IF YOU WOULD LIKE TO RUN THE SAME PROGRAM PLEASE ";
230PRINT"TYPE A (1).  IF YOU WOULD LIKE TO RUN ANOTHER PROGRAM ";
240 PRINT "PLEASE TYPE A (2). PRESS (CR) TO END. "
260 [illegible]X(100)2[illegible],190
270STOP
280REM                    "THE MULTIPLIER"
290I=0:J=0
300PRINT"[illegible]"
310GOSUB[illegible]
320[illegible]=
330PRINT"[illegible]"
340GOSUB[illegible]
350J=0
360IFG<2^15[illegible]
370I=1
380G=G-2^15
390IFH<2^15[illegible]
400J=1
410H=H-2^15
420[illegible]=[illegible](G*H/2^15)
430K=I+J
440IFK<>1[illegible]
450[illegible]=[illegible]+2^15
460GOSUB[illegible]
470RETURN
480REM                    "HEX TO DEC CONVERTER"
490[illegible]=[illegible]
500PRINT"[illegible] DIGITS [illegible]"
510[illegible]
520 IF [illegible]<3[illegible]
530[illegible]
540 [illegible]
550[illegible]
560[illegible]
570[illegible]
580[illegible]
590[illegible]
600 [illegible]
610[illegible]
620[illegible]
630[illegible]
```

```
401FHS(J,J)="7"THEND(J)=7
501FHS(J,J)="8"THEND(J)=8
601FHS(J,J)="9"THEND(J)=9
701FHS(J,J)="A"THEND(J)=10
801FHS(J,J)="B"THEND(J)=11
901FHS(J,J)="C"THEND(J)=12
700IFHS(J,J)="D"THEND(J)=13
710IFHS(J,J)="E"THEND(J)=14
720IFHS(J,J)="F"THEND(J)=15
730NEXTJ
740FORJ=1TO4
750D=D+D(J)*2^(4*(4-J))
760NEXTJ
770PRINTD
780RETURN
790REM                        "DEC TO HEX CONVERTER"
800PRINT"INPUT DECIMAL NUMBER,";:INPUTD
810I=4
820N=INT(D/16^(I-1))
830[illegible](5-I)=[illegible]
840[illegible]16^(I-1)
850I=I-1
860IF I=[illegible]
870[illegible]
880[illegible]$="1234"
890[illegible]
900IFA(I)=0THEN[illegible]="0"
910IFA(I)=1THEN[illegible]="1"
920IFA(I)=2THEN[illegible]="2"
930IFA(I)=3THEN[illegible]="3"
940IFA(I)=4THEN[illegible]="4"
950IFA(I)=5THEN[illegible]="5"
960IFA(I)=6THEN[illegible]="6"
970IFA(I)=7THEN[illegible]="7"
980IFA(I)=8THEN[illegible]="8"
990IFA(I)=9THEN[illegible]="9"
100[illegible]="A"
101[illegible]="B"
102[illegible]="C"
103[illegible]="D"
104[illegible]="E"
105[illegible]="F"
10[illegible]
10[illegible]
10[illegible]
10[illegible]                   "[illegible]"
11[illegible] NUMBER"
11[illegible]
11[illegible]
11[illegible]
11[illegible]
11[illegible]
11[illegible]
11[illegible]
11[illegible]
11[illegible]
12[illegible]
12[illegible]                   "[illegible]R"
12[illegible]                   [illegible]IS PROGRAM";
12[illegible]                   [illegible]"
12[illegible]
12[illegible]
12[illegible]
```

```
1270U=U-B(J)*2^J
1280J=J-1
1290IFJ<0THEN1310
1300GOTO1260
1310PRINTB(7);B(6);B(5);B(4);;B(3);B(2);B(1);B(0)
1320RETURN
1330REM                    "DECIMAL TO FNS CONVERTER"
1340A=0:N=0
1350PRINT"INPUT DECIMAL NUMBER"
1360INPUTN
1370IFN<=15THEN1480
1380I=[illegible]
1390[illegible]= /2
1400IF[illegible]<=15THEN1420
1410I=I+1:GOTO1390
1420J=7
1430A(J)=INT(I/2^(J-3))
1440I=I-A(J)*2^(J-3)
1450J=J-1
1460IFJ<=2THEN1520
1470GOTO1430
1480PRINT"EXP=0"
1490FORK=3TO7
1500A(K)=0
1510NEXTK
1520[illegible]=A(7)*1[illegible]A(6)*100000+A(5)*10000+A(4)*1000+A(3)*100
1530[illegible]=[illegible]
1540J=[illegible]
1550A(J)=INT([illegible]/2^J)
1560[illegible]=[illegible]-A(J)*2^J
1570J=J-1
1580IF[illegible]THEN16[illegible]
1590GOTO1550
1600[illegible]=[illegible]+A(2)*1[illegible]+A(1)*10+A(0)
1610[illegible]
1620[illegible]
1630[illegible]
1640[illegible]                    "FNS TO DECIMAL CONVERTER"
1650[illegible]=2
1660[illegible]"INPUT [illegible] DIGIT NUMBER"
1670[illegible]=3
1680[illegible]
1690[illegible]
1700[illegible]
1710[illegible]
1720[illegible]
1730[illegible]
1740[illegible]
1750[illegible]
1760[illegible]
1770[illegible]
1780[illegible]
1790[illegible]
1800[illegible]
1810[illegible]                    "[illegible]"
1820[illegible]
1830[illegible]
1840[illegible]
1850[illegible]
1860[illegible]
1870[illegible]
1880[illegible]
1890[illegible]",
```

```
1900PRINT"ILEFT=",A
1910RETURN
1920REM                              "SUBROUTINE IRND"
1930Z=0
1940PRINT"INPUT NUMBER TO BE ROUNDED(ETC.)"
1950INPUTZ
1960IFZ<0THEN1990
1970Z=INT(Z+.5)
1980GOTO2000
1990Z=INT(Z-.5)
2000PRINT"ROUNDED NUMBER =",Z
2010RETURN
2020REM                              "SUBROUTINE CSS"
2030REM CALCULATE SIEVESIZE
2040X=0:Y=0
2050PRINT"INPUT [illegible]S":INPUTE
2060PRINT"INPUT [illegible]":INPUTK
2070PRINT"INPUT SS%X":INPUTZ
2080PRINT"INPUT [illegible]":INPUTB
2090PRINT"IF RANGE IS IN HEX PRESS A (1).":INPUTG
2100[illegible]2120
2110PRINT"[illegible]":INPUTR:GOTO2130
2120PRINT"[illegible]":GOSUB480
2130[illegible]=0:G=0
2140PRINT"IF [illegible] IS IN HEX PRESS A (1).":INPUTC
2150[illegible]2170
2160PRINT"INPUT [illegible]":INPUTD:GOTO2190
2170PRINT"INPUT [illegible]":GOSUB480
2180[illegible]=0
2190S=[illegible]
2200[illegible]
2210[illegible]
2220[illegible]
2230[illegible]
2240[illegible]
2250[illegible]
2260[illegible]
2270[illegible]
2280[illegible]
2290[illegible]
2300[illegible]
2310[illegible]
2320[illegible]
2330[illegible]
2340[illegible]
2350[illegible]
2360[illegible]
2370[illegible]
2380[illegible]
2390[illegible]
2400[illegible]
2410[illegible]
2420[illegible]
2430[illegible]
2440[illegible]
2450[illegible]
2460[illegible]
2470[illegible]
2480[illegible]
2490[illegible]
2500[illegible]                       "[illegible]"
2510[illegible]
2520[illegible]
2530[illegible]
```

```
2540 PRINT "TAR?":INPUT R
2550 IF R=0 THEN 2620
2560 S(2)=S(1)/R:GOTO 2620
2570 PRINT "ISZY?":INPUT S(2)
2590 IF S(2)=0 THEN 2610
2600 R=S(1)/S(2):GOTO 2620
2610 PRINT "ISZY AND TAR CANNOT BOTH BE EQUAL TO ZERO!"
2615 GO TO 2540
2620 PRINT "VALID (CR) OR POINT TARGET(1)?"
2630 INPUT T(1)
2640 ON T(1) THEN 2750
2650 PRINT "SIMPLE (CR) OR COMPLEXITY?"
2660 INPUT T(2)
2670 ON T(2) THEN 2690
2680 F=63/S(1):GO TO 2700
2690 F=63/SQR((S(2)/2)^2+S(1)^2)
2700 PRINT "RRANGE?":INPUT R(1)
2710 PRINT "RAY[illegible]?":INPUT R(2)
27[illegible] PRINT "ASPECT?":INPUT A
27[illegible] [illegible](1)*F*[illegible](2*S^-1((ABS(A)/4-1)/16))
27[illegible] [illegible](2)*F:GOTO 2760
[illegible]
                                        "SUBROUTINE CYY"
[illegible]
                                        "SUBROUTINE ISKK"
[illegible]
```

```
4060 J=J*1
4070 X=T/M*1000
4080 Y=N*S
4090 C=64*1E-4/Y/3/3
5000 Z=C*J/X/X/([illegible])
5010 Z=LOG(Z)/LOG(2)
5020 Z=Z*2^6
5030 IF Z<0 THEN 5060
5040 Z=INT(Z+0.5)
5050 GO TO 5070
5060 Z=INT(Z-0.5)
5070 PRINT "IOFF=",Z
5080 RETURN
9999 END
```

Appendix X

ETSG Operating Instructions

Operating Instructions

1. Turn display console "ON."
2. Open doors to disk drive and remove any diskettes therein.
3. Turn disk drive "ON."
4. Insert diskette DPO in drive 0.
5. Insert diskette with appropriate seeker and target files in drive 1.
6. Close disk drive doors.
7. Depress EXORciser RESET button.
8. Type E800;G
9. Type ETSG at the console after the MDOS prompt = appears.
10. The ETSG initialization software is interactive and will now prompt the operator.

This instruction set assumes that the EXORciser is "on." If this is not true refer to the "power up" instructions in Appendix XI. For more explicit instructions refer to the "ETSG Operator Manual" which is generally kept near the ETSG.

Appendix XI

Frequently Used MDOS Commands

Compiled by

D. E. Bockstahler

and

G. R. Loefer

POWER UP:

I. Turn on CRT (switch on back, right rear)
II. Turn on EXORCISOR (key switch)
III. Turn on Disk Unit (red button on front)

BRINGING UP MDOS:

I. Slide System Disk into Drive 0 (left side)*
II. Slide User Disk into Drive 1 (right side)*
III. Close both doors on Disk Unit
IV. Type: 'MAID'** (no carriage return)
V. Type: 'E800;G'** (no carriage return)

'=' Equals Sign should come up when the system is ready.
If not, start over at Step IV.

*To load a disk: Hold disk carefully, (do not bend) with the label side up and the opening on one edge toward the disk drive. Slide the disk slowly and smoothly into the unit until it stops just past the door.

**NOTE: Command strings are enclosed in single ' ' quotes.
Lower case letters inside quotes are user selectable names.
Upper case letters inside quotes MUST be entered as shown.

POWER DOWN:

I. OPEN BOTH DISK DRIVE DOORS FIRST
II. Remove User Disk and return to box
III. Remove System Disk and return to box
IV. Turn OFF Disk Unit
V. Turn OFF EXORCISOR
VI. Turn OFF CRT

BAUD RATE:

I. Set desired BAUD Rate Switch on CRT and turn OFF the previously set rate.

II. Set matching BAUD Rate on the EXORCISOR. (switch is on the right rear)

FORTRAN QUICKIE:

I. Turn on CRT
II. Turn on EXORCISOR
III. Turn on Disk Unit
IV. Type: 'MAID'
V. Type: 'E800;G'
VI. Create Program File with Editor (store on Disk Unit 1)
VII. Type: 'CHAIN↔F4;FN%filename%'
VIII. To Execute Type: 'filename:1'

FREQUENTLY USED MDOS COMMANDS:

Note: ↔ means a space must be put here.

FORMAT:

PURPOSE: To prepare a new disk or wipe out an old one

I. Load Disk into Drive 1
II. Type: 'FORMAT'. RESPONSE: 'FORMAT DRIVE 1'
III. Type: 'Y' for YES. RESPONSE: 'LOCK OUT ADDITIONAL SECTORS '
IV. Type: 'N'

DOSGEN:

PURPOSE: To initialize a new disk

I. Load formatted disk into Drive 1 (if not already there)
II. Type: 'DOSGEN↔;TU' for a user disk or
Type: 'DOSGEN↔;T' for a system disk

DIR:

PURPOSE: List directory of files on a disk

I. Type: 'DIR' for directory of Drive 0 or
Type: 'DIR ↔:1' for directory of Drive 1

LIST:

PURPOSE: To list any ASCII file stored on a disk

I. Type: 'LIST↔filename' for a file on Drive 0 or
Type: 'LIST↔filename:1' for a file on Drive 1

filename: Name of file, including the suffix if not '.SA'

DEL:

PURPOSE: To delete a file from a disk

I. Type: 'DEL↔filename' for a file on Drive 0 or
Type: 'DEL filename:1' for a file on Drive 1

filename: Name of file, including suffix

COPY:

PURPOSE: To copy files (same disk or between disks)

I. Type: 'COPY↔filename1, filename2'

filename1: Name of source file, including suffix and drive number filename2: Name of new file, including suffix and drive number

NAME:

PURPOSE: To change a disk file name

I. Type: 'NAME↔filename1, filename2'

filename1: Name of old file, including suffix and drive number
filename2: Name of new file, including suffix and drive number

BACKUP:

PURPOSE: To make a complete copy of a disk and
To reorganize files thereon

I. Copy files to system disk in Drive 0
II. Place a formatted blank disk in Drive 1
III. Type; 'BACKUP↔;UR'. RESPONSE: 'BACKUP FROM DRIVE 0 TO 1'
IV. Type: 'Y' for Yes

EDIT:

PURPOSE: To edit ASCII files

I. Type: 'EDIT↔filename'

filename: Name of file, including suffix and Drive No.

II. Type: 'AAAAAAAAAAAAAA . . .$$' * (this loads the file)
(use repeat key)
III. See section on EDITOR for list of commands and a hints
and kinks list

*Note: $ means ESCape Key

Appendix XII

The 6800 Text Editor

Compiled by

D. E. Bockstahler

and

G. R. Loefer

TEXT EDITOR:

I. Command Summary: Table 2

II. EDITOR Messages: Table 3

III. Hints and Kinks

1. This is a CHARACTER editor and NOT A LINE editor like TED on the CYBER.
2. All characters, INCLUDING CARRIAGE RETURN, are legal characters to be edited.
3. A '$$' (hit ESCape key twice) marks the end of a command line.
4. Commands may be concatenated on one line (if you can keep track of them) without any extra delimiter characters.
5. MISTYPE? Use SHIFT-RUB (most consistant) or CNTRL-H (only in EDITOR) for BACKSPACE.
6. Use 'B' to position pointer at head of file.
7. Use 'Z' to position pointer at end of file.
8. Une n'T' to display n lines. Does not move pointer.
9. Use n'L' to skip n lines. 'L' positions the pointer JUST AFTER THE LAST CARRIAGE RETURN. 'L' counts carriage returns. n may be negative to backup lines.
10. To input a new program (or a new block of statements), use the 'I' command. Type one 'I', then enter the entire block of code as if using a typewriter and then type $$(ESC ESC). The entire block is entered all at once.
11. To input new lines between old ones, use n'L' to position the pointer AFTER THE LAST LINE TO PRECEED THE NEW LINES. It works like an 'INSERT BEFORE' command.
12. Use n'K' to delete n lines. Position pointer just after the last line to be kept.
13. Use 'C' to change a string within a line. Position pointer just ahead of line to be edited, (so that a 'T' will display the line). Use 'Ccurrentstring$newstring$-LLT' to change a string of characters and display the corrected line.

TABLE 2. EDITOR COMMAND SUMMARY

COMMAND	DESCRIPTION
* A	Append. Appends input text from the System Reader Device to the edit buffer.
* B	Beginning. Moves the edit buffer pointer to the beginning of the edit buffer.
* Cstring1$ string2	Change. Replaces the first occurrence of "string 1" with "string 2".
nD	Delete. Deletes n characters from the edit buffer.
E (tape)	End. Terminates an edit operation by writing the contents of the edit buffer to the output tape and copying the remainder of the input tape to the output tape. Returns control to the editor.
* E (disc)	End. Terminates an edit operation by writing the contents of the edit buffer to the output file and copying the remainder of the input file to the output file. Returns control to the disc operating system.
F (tape)	Tape Leader/Trailer. Writes 50 NULL characters into the system punch device.
F (disc)	The F command is ignored.
* Istring	Insert. Inserts characters or lines of text into the edit buffer.
* nK	Kill lines. Deletes n lines from the edit buffer.
* nL	Line. Moves the edit buffer point n lines.
nM	Move character pointer. Moves the edit buffer pointer n characters.
Nstring (tape)	Search File. Searches file for first occurrence of "string".
Nstring (disc)	Search File. Searches file for first occurrence of "string". If "string" is not found, returns control to the disc operating system.
nP	Punch. Punches n lines from the edit buffer to the System Punch Device.
* Sstring	Search. Searches the edit buffer for the first occurrence of "string"

*MOST OFTEN USED COMMANDS

$← ESC Key

TABLE 2. EDITOR COMMAND SUMMARY
(continued)

COMMAND	DESCRIPTION
* nT	Type. Types n lines from the edit buffer to the System Console Device.
X (tape)	EXbug. Returns control to EXbug.
X (disc)	The X command is an illegal command in the disc version of the editor.
* Z	End of edit buffer. Moves the edit buffer pointer to the end of the edit buffer.
Control H	Backspace. Causes the last character entered in the command mode to be typed on the System Console Device and deleted from the command.
Control X	Cancel. Causes all commands following the last prompt to be deleted and another prompt to be typed.

TABLE 3. EDITOR MESSAGES

MESSAGE	DESCRIPTION
M6800 RESIDENT EDITOR n.n	Printed upon initiation of editor. Revision is specified by n.n.
@	Prompt. Editor is waiting for a command.
????	Illegal command.
CAN'T FIND "string"	Editor cannot find the string specified by Search or Change command.
BELL	The editor rings the bell in the System Console Device when the user attempts to enter further commands into a full command buffer. The user must delete (backspace) two characters in order to terminate the command with two ESC characters.

14. Use '$string$' to search for a character string within the file. It starts searching from the current pointer position to the end of the file. The pointer will end up at the end of the string it found, (not at the beginning of the line). Use '-LL' to position at beginning of line.
15. Use 'BE' to end the editor program. *Do not use just an 'E', you might lose some of your file.*

Note: '$' means ESCape key.

FORTRAN:

NOTE: Be very careful to follow the manual when composing a FORTRAN program for the EXORCISOR. It falls short of ANSI Standard FORTRAN in a number of places (see Table 4).

I. Prepare FORTRAN programs using the EDITOR.

II. Programs must be complete within one file to be compiled and run. However, subroutines, etc. may be stored seperately and merged prior to compilation, or just before the Linking Loader command as shown below.

III. For a one file program in file 'prog.SA:1' DO:
'CHAIN↔F4;FN%prog%'
DO NOT store programs on Drive 0.
When finished, simply type: 'prog:1' to run the program.

IV. For MULTI-FILE programs, prog1.SA:1, sub1.SA:1, etc.
After making sure all old '.RO' files are deleted, DO:
'FORT↔prog1.SA:1'
'FORT↔sub1.SA:1'
'FORT↔etc.' (as many as there are)
'MERGE↔prog1.RO:1,sub1.RO:1,. . .,dest.RO:1'
dest: destination file name
'CHAIN↔RL;FN%dest%'
Then Type: 'dest:1' to run the program

TABLE 4. CONVERSION OF FORTRAN FROM CDC6600 TO EXORCISER

1. No program statement. For READ and WRITE to units other than CRT use OPENF and CLOSEF.
2. No blank lines in source file.
3. must be used for continuation in Column 1 (see special compile features of FORT 2.2).
4. INT and FLOAT functions do not exist. Simply assign to opposite type variable to switch types.
5. Variables and arrays are not initialized to zero.
6. Only one dimension statement per program block (use continuation).
7. No variable array dimensioning or accessing outside the dimension in subprograms.
8. No labeled common.
9. Can't use same variable in both data and common statements.
10. Some forms of data statement illegal.
11. No one line functions.
12. Parameters of functions, subroutines, and array indices must be constants or simple variables (no expressions).
13. Change Unit 5 (INPUT) to Unit 100 (from CRT keyboard).
14. Change Unit 6 (OUTPUT) to Unit 101 (to CRT screen).
15. NO FREE FORMAT WRITE.
16. FREE FORMAT INPUT and write a blank line use: 998 FORMAT( ).
17. No 'H' (HOLLERITH) format.
18. Use ' instead of " for format and data statement.
19. No spaces between format and open bracket:
    OK: FORMAT(NOT OK: FORMAT (
    applies to other statements with brackets also.
20. Keep computations simple, such as:
    Don't call a function twice on same line,
    Don't use lots of brackets ( ).,
    etc.

21. Keep special attention to IF statements that include computations, they don't always work.

22. Start all line numbers in column 1.

23. Code does not have to start in column 7.

24. 72 columns usable for FORTRAN.

25. Use X and Y in column 1 (special compile feature) to help de-bug programs with extra write statements.

Appendix XIII

Diskette Files

```
;S
DRIVE : 0   DISK I.D. : MDOS
BINEX   .CM
LIST    .CM
MDOSOV0 .SY
DIR     .CM
MERGE   .CM
RLOAD   .CM
MDOSOV4 .SY
MDOS    .SY
ABASIC  .CM
MDOSOV6 .SY
RASM    .CM
FREE    .CM
ROLLOUT .CM
EQU     .SA
DUMP    .CM
EXBIN   .CM
NAME    .CM
MDOSOV1 .SY
PATCH   .CM
ASM     .CM
BLOKEDIT.CM
ECHO    .CM
EDIT    .CM
LOAD    .CM
MDOSOV3 .SY
MDOSER  .SY
DEL     .CM
CHAIN   .CM
BACKUP  .CM
REPAIR  .CM
MDOSOV5 .SY
DOSGEN  .CM
EMCOPY  .CM
COPY    .CM
FORMAT  .CM
MDOSOV2 .SY
TOTAL DIRECTORY ENTRIES SHOWN : 036/$24

 :1
DRIVE : 1   DISK I.D. : ETSGDP0
ETARG   .CM
ETSG    .CM
PULSEJ  .CM
DCON    .SA
DELL    .SA
DRECT   .SA
DSKR    .SA
F       .SA
MJSTARG .SA
RUNETSG .CM
DROS    .SA
LI      .SA
DTRI    .SA
DPLUM   .SA
FLARE   .CM
SEEK    .CM
RUNETSG .SA
TOTAL DIRECTORY ENTRIES SHOWN : 017/$11
```

```
 :1
DRIVE : 1   DISK I.D. : JTR1
CKINIT   .RO
RDTCPU   .RO
ALP      .RO
PICT     .RO
LDTLR    .RO
GRAPH    .RO
STSEEK   .RO
PMD      .RO
LDDSPC   .RO
DIR      .RO
ETLB     .RO
ASPGEN   .RO
IRND     .RO
ARCTRIG  .RO
DFLT     .RO
RUNETSG  .CM
FTNLBX   .RO
LDRET    .RO
STTGPM   .RO
RUNETSG  .RO
LRUN     .CF
LDPTTG   .RO
GREY     .RO
LDPLSJ   .RO
STAROS   .RO
LSHL     .RO
RDTMP    .RO
STAROS   .SA
CLRTMP   .RO
VERT     .RO
ESYS     .SA
APICT    .RO
ETSGGO   .RO
LDTCPU   .RO
RDDSPC   .RO
AXES     .RO
STAOC    .RO
LDNTRR   .RO
LDASP    .RO
READWRIT.RO
BOOT     .RO
TOTAL DIRECTORY ENTRIES SHOWN : 041/$29
```

```
 :1
DRIVE : 1   DISK I.D. : SEEK
RETCL    .SA
LSK      .CF
NTR      .RO
DSKR     .SA
RETGEN   .SA
SEEK     .RO
NTR      .SA
SEEK     .CM
SEEK     .SA
RETGEN   .RO
TOTAL DIRECTORY ENTRIES SHOWN : 010/$0A
```

```
:1
DRIVE : 1   DISK I.D. : JTR1
CKINIT   .RO
RDTCPU   .RO
ALP      .RO
FOV      .SA
ZS       .SA
PICT     .RO
LDTLR    .RO
GRAPH    .RO
STSEEK   .RO
DIP      .RO
PMI      .RO
ETLB     .RO
LDDSPC   .RO
TRG      .SA
ASPGEN   .RO
IRND     .RO
ARCTRIG  .RO
DFLT     .RO
LDRET    .RO
STTGPM   .RO
RUNETSG  .RO
LRUN     .CF
CN       .SA
LDPTTG   .RO
GREY     .RO
LDPLSJ   .RO
STAROS   .RO
LSHL     .RO
RDTMP    .RO
IFLGS    .SA
CLRTMP   .RO
VERT     .RO
APICT    .RO
LDTCPU   .RO
RDDSPC   .RO
AXES     .RO
ETSGGO   .RO
STAOC    .RO
LDNTRR   .RO
LDASP    .RO
RESP     .SA
TOTAL DIRECTORY ENTRIES SHOWN : 041/$29


:1
DRIVE : 1   DISK I.D. : SCRATCH
PUNC     .LX
DSD      .LX
EMT      .CM
PUNC     .LO
PLOT     .SA
DSD      .LO
DSD      .SA
PLOT     .RO
OLDDSD   .LO
INTFAC   .SA
INTFACND.SA
TOTAL DIRECTORY ENTRIES SHOWN : 011/$0B
```

```
:1
DRIVE : 1   DISK I.D. : JTR2
CKINIT  .RO
RDTCPU  .SA
VERT    .SA
CLRTMP  .SA
AXES    .SA
LDTLR   .SA
GRAPH   .SA
RDDSPC  .SA
STAOC   .SA
LDNTRR  .SA
CKINIT  .SA
PACK    .SA
DIR     .SA
PMD     .SA
LDDSPC  .SA
STSEEK  .SA
LDPTTG  .SA
LDPLSJ  .SA
IRND    .SA
LDRET   .SA
LSHL    .SA
ARCTRIG .SA
DFLT    .SA
STAROS  .SA
STTGPM  .SA
LDTCPU  .SA
ETSGGO  .SA
LDASP   .SA
RUNETSG .SA
RDTMP   .SA
ALP     .SA
TOTAL DIRECTORY ENTRIES SHOWN : 031/$1F
:1
DRIVE : 1   DISK I.D. : FSTTST
S       .SA
T       .SA
PLUM    .SA
T02S01  .SA
T01S01  .SA
T03S01  .SA
TRET    .SA
PT      .SA
IELPS   .SA
RB      .SA
S01     .SA
ITRI    .SA
TOTAL DIRECTORY ENTRIES SHOWN : 012/$0C
:1
DRIVE : 1   DISK I.D. : ETARG
LTG     .CF
PICT    .SA
ASPGEN  .RO
ETS     .CM
ETARG   .RO
ETARG   .SA
ASPGEN  .SA
APICT   .SA
TOTAL DIRECTORY ENTRIES SHOWN : 008/$08
```

```
:1
DRIVE : 1   DISK I.D. : SEEK
RETCL    .SA
LSK      .CF
NTR      .RO
RETGEN   .SA
SEEK     .RO
NTR      .SA
SEEK     .CM
SEEK     .SA
RETGEN   .RO
TOTAL DIRECTORY ENTRIES SHOWN : 009/$09

:1
DRIVE : 1   DISK I.D. : GRL
LTE      .CF
FLARE    .SA
VARLIST  .SA
ETSG     .CM
PULSEJ   .CM
PULSEJ   .RO
ETSG     .RO
LPJ      .CF
ETSG     .SA
FLARE    .CM
PULSEJ   .SA
FLARE    .RO
LFLR     .CF
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D

:1
DRIVE : 1   DISK I.D. : NONAME
F3       .SA
F4       .SA
F5       .SA
ABASIC   .CM
DCBSIM   .SA
DCBSIM   .LO
FTNLBX   .RO
DCBSIM   .LX
DCBSIM   .CM
F1       .LX
F1       .LO
F1       .SA
F2       .SA
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D
:1
DRIVE : 1   DISK I.D. : ET2
FLARE    .SA
LTE      .CF
VARLIST  .SA
PULSEJ   .CM
ETSG     .CM
PULSEJ   .RO
ETSG     .RO
LPJ      .CF
ETSG     .SA
PULSEJ   .SA
FLARE    .CM
FLARE    .RO
LFLR     .CF
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D
```

```
:1
DRIVE : 1   DISK I.D. : ETSGDP0
ETSG     .CM
PULSEJ   .CM
DCON     .SA
DELL     .SA
ETARG    .CM
DRECT    .SA
F        .SA
DSKR     .SA
MJSTARG  .SA
RUNETSG  .CM
DROS     .SA
LI       .SA
DTRI     .SA
DPLUM    .SA
FLARE    .CM
SSSS     .SA
SEEK     .CM
RUNETSG  .SA
TOTAL DIRECTORY ENTRIES SHOWN : 018/$12
```

```
:1
DRIVE : 1   DISK I.D. : SDBASIC
SDASM    .CM
SDBCOM   .CM
TMTEST   .BA
TEST     .BA
TEST     .LX
RUNROS   .BA
SDEDIT   .CM
SDRUN    .CM
TOTAL DIRECTORY ENTRIES SHOWN : 008/$08
```

```
:1
DRIVE : 1   DISK I.D. : SYSTEM
KATE     .LO
P1       .SA
SEEKER   .SA
DUBLIN   .SA
P2       .SA
PROMPROG.CM
P3       .SA
TS1      .SA
TOTAL DIRECTORY ENTRIES SHOWN : 008/$08
```

```
:1
DRIVE : 1   DISK I.D. : TARGET
S        .SA
T        .SA
ABASIC   .CM
PLUM     .SA
TOTAL DIRECTORY ENTRIES SHOWN : 004/$04
```

```
 :1
DRIVE : 1   DISK I.D. : ET2
PICT     .SA
LTG      .CF
ETARG    .CM
ASPGEN   .RO
ETARG    .RO
ETARG    .SA
ASPGEN   .SA
APICT    .SA
TOTAL DIRECTORY ENTRIES SHOWN : 008/$08
```

```
 :1
DRIVE : 1   DISK I.D. : NONAME
F3       .SA
F4       .SA
F5       .SA
ABASIC   .CM
FTNLBX   .RO
F1       .LX
F1       .LO
F1       .SA
F2       .SA
TOTAL DIRECTORY ENTRIES SHOWN : 009/$09
```

DISTRIBUTION

	No. of Copies
IIT Research Institute ATTN: GACIAC 10 West 35th Street Chicago, IL 60616	1
DRSMI-LP, Mr. Voigt	1
-RD, Dr. Hallum	10
-RPR	15
-RPT, Record Copy	1
Reference Copy	1

ATE

LMED

-8