ELECTRONIC TARGET SIGNAL GENERATOR (ETSG) SOFTWARE DEVELOPMENT. (U)

OCT 81  P F PRITCHETT, N A KHEIR

DAAH01-81-D-0006
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.

Prepared for:
Systems Simulation and Development Directorate
US Army Missile Laboratory
Contract DAAH01-81-D-A006
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.
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

PREFACE ........................................ 1

CHAPTER ONE:  ETSG Concept and Function ................. 1
  1.0 Introduction ...................................... 1
  1.1 Overview of the ETSG ............................... 2

CHAPTER TWO:  The Initialization Processes and Codes .... 5
  2.0 Introduction ....................................... 5
  2.1 Simulation Initialization Parameters .................. 7
  2.2.1 MDOS ........................................... 9
  2.2.2 ETSG .......................................... 11
  2.2.3 SEEK ........................................... 17
  2.2.4 ETARG ......................................... 30
  2.2.5 PULSEJ .......................................... 39
  2.2.6 FLARE .......................................... 45
  2.2.7 RUNETSG ....................................... 49

CHAPTER THREE:  Real-Time Computer Model (Target CPU Software) ... 52
  3.0 Introduction ....................................... 52
  3.1 Target CPU Code ................................... 54

CHAPTER FOUR:  Conclusions and Recommendations .............. 60

REFERENCES: ......................................... 61

APPENDICES
  Appendix I  Target Coordinate Variables
  Appendix II  Initialization Interface Variables
  Appendix III  System Flags
  Appendix IV  Target Parameters
  Appendix V  Intensity Data
  Appendix VI  Seeker Data
  Appendix VII  Field of View Data
  Appendix VIII  Initialization Subroutine
TABLE OF CONTENTS (Cont'd)

APPENDICES (Cont'd)

Appendix IX  ETSC.BAS
Appendix X  ETSG Operating Instructions
Appendix XI  MDOS
Appendix XII  M6800 EDITOR
Appendix XIII  Diskette Files
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.
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 scenario. 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.
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

<table>
<thead>
<tr>
<th>INITIAL INPUTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEEKER PARAMETERS</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>FOV</td>
</tr>
<tr>
<td>Blur</td>
</tr>
<tr>
<td>NEFD</td>
</tr>
<tr>
<td>SNR for Track</td>
</tr>
<tr>
<td>Reticle Scan Rate</td>
</tr>
<tr>
<td>System Responsivity</td>
</tr>
<tr>
<td>SOURCE PARAMETERS</td>
</tr>
<tr>
<td>Shape</td>
</tr>
<tr>
<td>Size</td>
</tr>
<tr>
<td>Aspect Ratio</td>
</tr>
<tr>
<td>Intensity Gradients</td>
</tr>
<tr>
<td>Spectral Bands</td>
</tr>
<tr>
<td>Intensity Polarity</td>
</tr>
<tr>
<td>Programmable Intensity</td>
</tr>
<tr>
<td>Maximum Range</td>
</tr>
<tr>
<td>Minimum Range</td>
</tr>
<tr>
<td>PULSE JAMMER</td>
</tr>
<tr>
<td>Rep Rate</td>
</tr>
<tr>
<td>Sweep Time</td>
</tr>
<tr>
<td>Duty Cycle</td>
</tr>
<tr>
<td>Period</td>
</tr>
<tr>
<td>FLARES</td>
</tr>
<tr>
<td>Intensity vs. Time</td>
</tr>
</tbody>
</table>
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 contiguously 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.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Implied Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Assembly listing file</td>
</tr>
<tr>
<td>CF</td>
<td>Chain Procedural file</td>
</tr>
<tr>
<td>CM</td>
<td>Command file</td>
</tr>
<tr>
<td>ED</td>
<td>EDOS - converted file</td>
</tr>
<tr>
<td>LO</td>
<td>Loadable memory - image file</td>
</tr>
<tr>
<td>LX</td>
<td>EXbug loadable file</td>
</tr>
<tr>
<td>RO</td>
<td>Relocatable object file</td>
</tr>
<tr>
<td>SA</td>
<td>ASCII source file</td>
</tr>
<tr>
<td>SY</td>
<td>Internally - used system file</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>TPIAS</th>
<th>PIA Address</th>
<th>PIA Initialization Values</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$CBA0,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBA2,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBA4,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBA6,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBAD,</td>
<td>$00, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBAE,</td>
<td>$00, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB0,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB2,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB4,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB6,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB8,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBBA,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBBC,</td>
<td>$FF, $06, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBBE,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBC0,</td>
<td>$FF, $06, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBC2,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBC4,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBC6,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBC8,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBCA,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBCC,</td>
<td>$FF, $06, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBB4,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBDA,</td>
<td>$00, $00, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBDC,</td>
<td>$00, $00, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBDE,</td>
<td>$00, $00, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBDF,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>$CBFA,</td>
<td>$FF, $04, $00</td>
<td></td>
</tr>
<tr>
<td>PIA Address</td>
<td>PIA Initialization Values</td>
<td>Default Values</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>$CBFC,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CBFE,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CEEC,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CEEE,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CFF0,</td>
<td>$0F, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CEF2</td>
<td>$0F, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CEF4,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$CEFB,</td>
<td>$FF, $04, $00</td>
<td>$00</td>
<td></td>
</tr>
<tr>
<td>$0000,</td>
<td>$00, $00, $00</td>
<td>$00</td>
<td></td>
</tr>
</tbody>
</table>
START

Initializeplotting
package and console
communication lines.

Is a power
up restart in order?
$CBFB(00)$

Continue

Does system file,
IESY, exist?

IESY exists

Road system file.
ISYF(1-40)

Check internal flag
ISYF(1)=1

Seeker/Target Error

Ask operator:
New System or Error Restart?

New System

Error Restart

Figure 2: ETSG Supervisory Program (Main Driver).
ISYF(1) = 0

System Error

No Error

Ask operator:
New System or Use Old Targets?

New System

Power up restart.
Set system flags = 0
Initialize CPUs
Delete system and seeker files

Ask operator:
Manual Select or Auto Sequence?

Manual select

Auto sequence

Increment auto sequence number.
ISYF(3) = ISYF(3) + 1
ITYPE = ISYF(3)

Tell operator auto sequence number. Ask:
Abort or Continue Sequence?

Abort

Continue

Ask operator:
Where do you want to "Boot" to.
ITYPE = ?

Figure 2 (Continued)
"Boot" to the correct point in the initialization sequence, (determined by ITYPE)

ITYPE = ?

1. "Boot" SEEK

2. "BOOT" ET ARG

3. Does pulse jammer exist? ISYF(4) = 1

4. "Boot" PULSE J

5. Does Flare exist? ISYF(5) = 1

6. Flare exists

7. Set auto sequence number = 0 ISYF(3) = 0

STOP

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
- IBl - Exponent Scale Factor Adjust
- MSIGN - Exponent Scale Factor Command

are calculated and stored in PIA's 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 = BLRR*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.

FPFD=NFOV/FOVD Points per degree in field of view.
*TDPC = FOV3  Number of degrees per count for target coordinate.

*IRCSW Rosette or Consan 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 ≥ TCDPC.
if not, default to TDPC = TCDPC.

FOVTEM = TDPC*256  Temporary variable
Note: FOVTEM ≥ FOVD

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

SBLRM = NPWN/FPPD/DPM Scaled blur diameter.

ICPC8 = FOVTEM/FOVI*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/cm²)
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/cm²)

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

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

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

*If NFSC (ICH) ≠ 1 then input:

*SIGMN (ICH) = RESP (ICH,7) Minimum system signal at aperture (watts/cm²)

SMNSY = SIGMN (ICH) Minimum system signal at aperture

SMXSY = DYNRNG*SMNSY Maximum system signal at aperture (watts/cm²)

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
C0 = 64 Seeker irradiance to FNS.
Cm2PM2 = 1.0 E - 4 Cm²/m²
C5(ICH) = C0*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 ≤ 0.8480 then;
IB1 = IT1 + MSIGN
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 ≥ 1024 then;
IAC10 = 1023
IB1 = IT1 + MSIGN
IAC10 = 1023
AT2 = 2^IT1
VBGMX = 10.0
VBGAB = BKRD (ICH) VBGAB*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)*SIGHN(2) + ARES(2) * SIGHN(1))

Analog Scale Factor Adjust IAC10

\[ T1 = \log_2(CNBC*REFD*ARES/6.1E-4) \]

\[ IT1 = T1 \]  
Note: Real to integer

\[ FT1 = T1 - IT1 \]  
Note: Truncate whole number.

\[ FT2 = FT1 - 1 \]

\[ IAC10 = 2^{FT2} \times 1024 \]

If IQC10 \leq 1024
then IAC10 = 1023

Stored at:

CBCA

CBC8 for J channel

CBBA for K channel

CBB8

If FT1 \leq RL 218 = 0.8480
then IAC10 = 1023

Background Level IPBGL

\[ IPBGL = IRND (BKRD/0.0*16383) \]

If IPBGL \geq 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

CBCB

20
Note: Subroutine CBV in STAOS reorders the bits to compensate for a hardware design problem.

Exponent Scale Factor Adjust IB1

\[ IB1 = 16 \times \left( \frac{\ln \left( \frac{\text{RNEFD*SNRT*ARES}}{6.1E-4} \right) + 1}{\ln 2} \right) \]

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

\[ IPNSL = \text{IRND}(\text{SNRT*RNEFD*ARES/5*255}) \]

If IPNSL > 255
then IPNSL = 255

- CBC2 for K channel
- CBC0 for J channel
New or Default Seeker or previously created seeker?

Previously created seeker

Filename for input?

Does Seeker file exist?

Y

Open Seeker file.

N

File not found Directory?

Y

N

Call DIR

Start

Figure 3: SEEK
Rosette or Conscan?
IRCSW = 1 or 2

Rosette
Open default Seeker file, IROS.

Read, then close seeker file.

Input FOVD - Field of View degrees side to side.

Input RLDM Blur diameter in m rad.

Calculate NPOV - Number of points in FOV.

Figure 3 (Continued)
3

FOV too large with current blur diameter.

6

Cannot represent total FOV with current data.

Figure 3 (Continued)
Input TDPC—Number of degrees per count for target coordinate.

TDPC > 4/128

N

Run with TDPC = 4/128 or start over?

TDPC = 4/128

TDPC*256 ≥ FOVD

N

Y

Scaled blur diameter SBLRM

Figure 3 (Continued)
Figure 3 (Continued)
Input seeker static parameters.

Is seeker compatible with ETSG

Y

Calculate controls for analog boards.

Call STAOC to load DAC controls to PIAs.

Calculate seeker checksum, ISKRCK.

Open, write, and close seeker file, ISKR.

Figure 3 (Continued)

27
9

10 N

Save this seeker?

Y

Filename?

File name already in use?

Y

N

Open, write seeker data, and close file.

N

File not found.

Figure 3 (Continued)
Does system file exist?

Y

Read system flags from IESY

ISYF(1) = 0
ISFY(40)=1RCSW=1 for Rosette

Write flags to IESY

"Boot" back to ETSG

END

Figure 3 (Continued)

29
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)*1000*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*SIGN(ISC)  Maximum system signal.
AGS = SMNT/SV*DN(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*RMX/RMIN
ZA = SMXSY/SMXV

If ZA ≥ 1  System will overflow
ZA ≥ 128  Probable overflow
ZA ≥ 1608.5  Possible overflow
ZA ≥ 1608.5  No overflow
RRMM = 599.0/PPM  Maximum value for key points (meters).
*RRXM = TRG(5)  X key point (meters).
*RRYM = 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
      ETSC 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
Start

Initialize console and communication lines.

System file, IESY, exist?

Yes

Seeker file IFN2, exist?

Yes

Read seeker/system data: ISYF, IFLGS, FOV, RESP

Figure 4: ETARG

No

Run ETSG first.

"Boot" ETSG

No

Run seeker scaling program first.

"Boot" ETSG

3

4
Call STTP; set target type, priority and flags.

New target?
IFL3 = 0

Yes

No

Image format?
IFL2 = 1

Yes

No

Input target parameters.

Determine probability of accumulation overflow.

Output probability of overflow

Figure 4 (Continued)
Max, gradient/average value
New value for RMIN?

Yes

No

Input:
Keypoints

Target color:
IFLGS(10)

True target?
IFLGS (11)

Pulse jammer?
IFLGS (5)

Flare?
IFLGS (6)

Figure 4 (Continued)
7

Pulse jammer?

Yes

ISYF (1-40) = 1
ISYF (5) = 1
ISYF (6) = ISYF(6)
+ 1
ISYF (IPRI + 6) = 1

No

ISYF(IPRI + 6) = 0

Write data to system file.
ISYF (1-40)

Clear intensity accumulators.

Set point target view.

Generate 3 views for complex target.

8

Figure 4 (Continued)
Save targets.

Process intensity information

Output:
PMN, PMX, PAVG

Load aspect RAMs, target CPUs, and display CPU.

Load target CPU: IP interface variables.

Load display CPU TGT1.

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: \((2*\text{SFREQ}*\text{NPULSE}-\text{SFREQ}*(2*\text{PDUTY}/100)) > 1000\) 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.
S RATE = (E FREQ-S FREQ)/S TIME
Average change in frequency per unit time.
NP ARTS = 2*NPULSE-1
Number of parts in strobe history.
TIME = 0 Initialize time to zero.
CFREQ = SFREQ + S RATE*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/NP ARTS) + 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.
Initialize plotting package  
Clear screen  
Ring bell  

Does system file, IESY, exist?  

Y  
Read ISYF(1-40) from IESY  

If pulse jammer flag is high, ISFY(4)=1  
Continue  

"Boot" back to ETSG  
Pulse jammer strobe generation routine.  

Input:  
NPULSE - number of pulses  
P DUTY - duty cycle (per cent)  
SFREQ - start spin frequency (Hertz)  
EFREQ - end frequency (Hertz)  

Figure 5: PULSEJ
Is strobe data consistent?

Frequency too low

Frequency/NPULSE too high. DUTY too low.

Input:
STIME—sweep time (Sec)

Graphical representation?
IG ≠ 0

Wait processing in progress.

Calculate pulse jammer parameters.
Set TIME = 0

Figure 5 (Continued)
Create pulse jammer history.

If IG = 0
Graphics mode flag.
T
Call GRAPH

F

Number of words in history array IF = 0
N
Y

Call LDPLSJ, load pulse jammer history

Increment time
TIME = TIME + CT

Is sweep time up yet?
TIME + CT/2 - STIME < 0
N
4
Y
5

Figure 5 (Continued)
Figure 5 (Continued)

Terminate LDPLST

Does system file exist?

Y

Write:
ISYP(1) = 0 to IESY

"Boot" to ETSG

N

"Boot to ETSG"

END
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.
Initialize console and communication lines

IHIST(1-105)=0

System file IESY, exist?

Yes

Read ISYF(5)

Flares specified? ISYF(5)=1

No

Error No flares

Yes

Read, N, number of points

"Boot" ETSG

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.
Start

Initialize console and communication lines.

Is system initialized?

Yes

"Boot" ETSG

Any last minute corrections

No

"Boot" ETSG

Yes

Delete system file.

Set/ Clear all necessary hardware.

Figure 7: RUNETSG
1

Display targets with trails?

ETSG ready.

RUN

Call PMD "Post Mortem Dump"

Rerun?

Yes

Reload system file and old targets.

"Boot" ETSG

END

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 (VALID) = 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

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

Uses A, B, X, TO, 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

55
Exit (ISFO) = ISF, FNS
Uses A, B, X, TO- T1, 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 (ABP0)
(ST) = Current Sine Term (for corrections) (AP80)
(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, T1, 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 transformation 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-77
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 checks 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
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 presently in development be completed.
REFERENCES


REFERENCES (Cont'd)

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

AI.2
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.
The subroutine LSHL operates on FNUMB and ISHFT and yields the results IRSLT and ILEFT.

\[
FNUMB = \text{IRND}(NPWN*FOV/(BLRM*180*10^{-3}/\pi))/FOVD/32
\]

All unknowns are IP input variables.

ISHFT = 15

IRSLT = ACSF

ILEFT = \text{SRAC} - 7

CNTX = 8448*2^7-\text{SRAC}

CNTY = 8192*2^7-\text{SRAC}

XC = (ACSF*AZIM)/2^{15} + CNTX

YC = (ACSF*ELEV)/2^{15} + CNTY

AZIM and ELEV are direct cell interface variables.
CAV

\[ \begin{align*}
\text{XK} &= 0 & \text{For a point target PT} &= 0 & \text{D055} \\
\text{YK} &= 0 & \text{D056} \\
\text{D057} \\
\text{D058}
\end{align*} \]

The values for XK are calculated by ASPGEN and stored in a table location $0200 + \lceil \text{ASPC} \rceil /4$.

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

TSZX and TSZY are IP input variables.

\[ \begin{align*}
\text{XK} &= \text{PPM}\times\text{RKXM}\times16\times\sin(\arccos((\text{ASPC}/4 - 1)/16)) \\
\text{YK} &= \text{KEYY} = \text{KYP} = \text{PPM}\times\text{RKYM}\times16 \\
\text{RKXM} \text{ and RKYM are initialization input variables via ETARG.} \\
\text{ASPC is a direct cell interface variable.}
\end{align*} \]
To compute $\cos(\text{ORNT})$:
1. Convert ORNT to decimal
2. $\text{ORNT}_{10} \times 1.41$
3. $\cos (1.41\times \text{ORNT}_{10})$

$SS = \text{FOVS} \times \text{RANG} \times 2^{\text{SRFV}}$

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

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

$XM = SS \times \cos(\text{ORNT}) \times 2^{-15}$

$YM = -SS \times \sin(\text{ORNT}) \times 2^{-15}$

$XN = SS \times \sin(\text{ORNT}) \times 2^{-15}$

$YN = SS \times \cos(\text{ORNT}) \times 2^{-15}$
CZC

\[ \begin{align*}
X_0 &= -((X_C X_M + Y_C X_N) \cdot 2^{SRAC-15} - X_K) \\
Y_0 &= -((X_C Y_M + Y_C Y_N) \cdot 2^{SRAC-15} - Y_K)
\end{align*} \]
Appendix II

Initialization Interface Variables
Initialization Interface Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBUG</td>
<td>Debug target flag</td>
<td>(1, FF=DEBUG)</td>
</tr>
<tr>
<td>VALD</td>
<td>Valid target flag</td>
<td>(1 = Valid)</td>
</tr>
<tr>
<td>FLAR</td>
<td>Target flare flag</td>
<td>(1 = Flare)</td>
</tr>
<tr>
<td>PLUM</td>
<td>Complex target flag</td>
<td>(1 = Complex)</td>
</tr>
<tr>
<td>CYCL</td>
<td>Cycle count</td>
<td></td>
</tr>
<tr>
<td>ERRF</td>
<td>Error flag</td>
<td></td>
</tr>
</tbody>
</table>

RRAN
Resolution Range

\[ RCO = \frac{TSZX}{BLRM} \times 1000 \]
\[ RRAN = \frac{RCO \times NPWN}{NPWN + 1} \]

LRAN
Linear Resolution Range

(not used)

KEYY
Y Key Point

\[ IMAX = 64 \]

Simple target

\[ T1 = TSZX \]
\[ PPM = \frac{(IMAX-1)}{T1} \]
\[ KEYY = PPM \times RKYM \times 16 \]

Complex target

\[ T1 = TSZY \times TSZY \times 4 + TSZX \times TSZX \]
\[ PPM = \frac{(IMAX - 1)}{\sqrt{T1}} \]

ISFR
Intensity Scale Factor Pias for Resolved Target

\[ CO = 64 \]
\[ CM2PM2 = 1.0E - 4 \]
\[ SMSY = RNEFD \times SNRT \]
RCO = TSZX/BLRM*1000
REAVG = PAVG/PWN(4)*9
RJJP = IPOLTY*(RJT-BKRD*100000*EXP(ATTN/1000))
C5 = CO*CM2PM2/SMNSY/PWN/PWN
ISFR = IRND(((ALOG(C5*RJJP/RCO/RCO/RFAVG)/ALOG(2)))*2**8)

ISFP  Intensity Scale  D08C
Factor Bias for a Point Target  D08D
CST = 9.0
CO = 64
CM2PM2 = 1.0 E-4
SMNSY = RNEFD*SNRT
C5 = CO*CM2PM2/SMNSY/PWN/PWN
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) = log2(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/PWN/RCO
LSHL/FTMP,20)
FTMP = FOVS*2^SRFV-20

AII.3
ACSF Angle to Coordinate Scale Factor

\[ DPM = 0.0572957795 \]

\[ BLRD = BLRM \times DPM \]

\[ NFOV = IRND(NPWN \times FOVD / BLRD) \]

\[ FPPD = NFOV / FOVD \]

\[ TCDPC = 4 / 128 \]

\[ TCPPC = FPPD \times TCDPC \]

\[ LSHL(TCPPC, 15) \]

\[ TCPPC = ACSF \times 2^{20-JTMP} \]

SRAC 22 Shift Applied to ACSF

\[ SRAC = JTMP + 7 \]

CNTX Shifted X Center Coordinate

\[ CNTX = 528 \times 16 / 2^{JTMP} \]

CNTY Shifted Y Center Coordinate

PTSS Point Target Step Size

\[ PTSS = 64 / NPWN \times 2^{2 \times 4} \]
Appendix III
System Flags
### SYSTEM FLAGS

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITSW</td>
<td>iFLGS(1) TARGET TYPE</td>
</tr>
<tr>
<td>IWSW</td>
<td>iFLGS(2) TARGET GEOMETRY FOR COMPLEX TARGET SIDE VIEW</td>
</tr>
<tr>
<td>ISC</td>
<td>iFLGS(3) TARGET TYPE 1= LONG WAVE, K-CHL 2= SHORT WAVE, J-CHL</td>
</tr>
<tr>
<td>IPOLTY</td>
<td>iFLGS(4) POLARITY</td>
</tr>
<tr>
<td>IFJ</td>
<td>iFLGS(5) PLUSE JAMMER FLAG</td>
</tr>
<tr>
<td>IFL</td>
<td>iFLGS(6) FLARE FLAG</td>
</tr>
<tr>
<td>IPRI</td>
<td>iFLGS(7) PRIORITY</td>
</tr>
<tr>
<td>IFLM</td>
<td>iFLGS(8) PLUME FLAG</td>
</tr>
<tr>
<td>ITH</td>
<td>iFLGS(9) VIEW NUMBER FOR COMPLEX TARGET</td>
</tr>
<tr>
<td>ITCLR</td>
<td>iFLGS(10) TARGET COLOR</td>
</tr>
<tr>
<td>ISRUT</td>
<td>iFLGS(11) TRUE TARGET FLAG</td>
</tr>
<tr>
<td>IGLISW(1)</td>
<td>iFLGS(12)</td>
</tr>
<tr>
<td>IGLISW(2)</td>
<td>iFLGS(13) TARGET INTENSITY GRADIENT FLAG</td>
</tr>
<tr>
<td>IGLISW(3)</td>
<td>iFLGS(14)</td>
</tr>
<tr>
<td>IS:RSV</td>
<td>iFLGS(15) SEEKER CHECK VALUE FROM STTP</td>
</tr>
<tr>
<td>IRCW</td>
<td>iFLGS(16) 1=ROSETTE 2=CONSCAN</td>
</tr>
<tr>
<td>NFWSN</td>
<td>iFLGS(17) NUMBER OF POINTS IN ONE DIMENSION OF BLUR DIAMETE</td>
</tr>
<tr>
<td>NFXCR</td>
<td>iFLGS(18) MAXIMUM SCAN RATE FOR CONSCAN</td>
</tr>
<tr>
<td>NFSC(1)</td>
<td>iFLGS(19) SCALE TO NEFD CHANNEL 1</td>
</tr>
<tr>
<td>NFSC(2)</td>
<td>iFLGS(20) SCALE TO NEFD CHANNEL 2</td>
</tr>
<tr>
<td>ISROT</td>
<td>iFLGS(21) CONSCAN SEEKER ROTATION</td>
</tr>
<tr>
<td>IS:RECK</td>
<td>iFLGS(22) SEEKER CHECKSUM VALUE</td>
</tr>
</tbody>
</table>

### SYSTEM FLAGS

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISYF(1)</td>
<td>SYSTEM ERROR FLAG</td>
</tr>
<tr>
<td>ISYF(2)</td>
<td>MANUAL SELECT FLAG</td>
</tr>
<tr>
<td>ISYF(3)</td>
<td>AUTO SEQUENCE NUMBER</td>
</tr>
<tr>
<td>ISYF(4)</td>
<td>STROBE EXISTS</td>
</tr>
<tr>
<td>ISYF(5)</td>
<td>FLARES EXIST</td>
</tr>
<tr>
<td>ISYF(6)</td>
<td>NUMBER OF FLARES</td>
</tr>
<tr>
<td>ISYF(7)</td>
<td>TARGET 1 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(8)</td>
<td>TARGET 2 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(9)</td>
<td>TARGET 3 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(10)</td>
<td>TARGET 4 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(11)</td>
<td>TARGET 5 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(12)</td>
<td>TARGET 6 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(13)</td>
<td>TARGET 7 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(14)</td>
<td>TARGET 8 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(15)</td>
<td>TARGET 9 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(16)</td>
<td>TARGET 10 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(17)</td>
<td>TARGET 11 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(18)</td>
<td>TARGET 12 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(19)</td>
<td>TARGET 13 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(20)</td>
<td>TARGET 14 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(21)</td>
<td>TARGET 15 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(22)</td>
<td>TARGET 16 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(23)</td>
<td>TARGET 17 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(24)</td>
<td>TARGET 18 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(25)</td>
<td>TARGET 19 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(26)</td>
<td>TARGET 20 IS A FLARE</td>
</tr>
<tr>
<td>ISYF(40)</td>
<td>SEEKER TYPE</td>
</tr>
</tbody>
</table>
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)
PNIR = TRG(11) RANGE OF 1:1 RESOLUTION (METERS)
RJTP = TRG(12) CONTRAST RADIANCE (WATTS/STERADIANS)
Appendix V
Intensity Data

AV. 1
** INTENSITY DATA **

** I = I P L M T H

** 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), ERK(E), YB(R)] **

** CN(5, I) = ERK VALUE [EN(T), E(R, E)] **

** CN(6, I) = BREAK PT [YB(T ONLY)] **

** CN(7, I) = ERK 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] **

** PMAX = ZS(3, I) [I=4; MAX FOR ALL VIEWS] **

** PMIN = 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/CM²)
- **ARES(I)** = **RESP(I,2)** SYSTEM RESPONSIVITY (VOLTS/WATTS/CM²)
- **BKRD(I)** = **RESP(I,3)** BACKGROUND IRRADIANCE (WATTS/CM²)
- **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 IRRADIANCE 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 DEGREES 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

AVIII.1
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)

ASPCEN - Plume Aspect Generator
Inputs - (AR, KEYX) Input files - (IFLGS, SA:B, TRG.SA:O)
Call - (LDASF, LDTCP0)

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 halfls of both channels of the target map.

AVIII.2
CONVERT - Convert ETSG Floating Point to MOTOROLA 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 Cursor 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 microsecond. 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.
Call - (FILTST, MLOAD, STTP, BELL, INVERT, LDPTIG, PAGE, GENTRG, DELF, ASPGEN, STTGCH, STTSGN, STSTBB, LDTCFP, LSHL, LDSPC)

ETSG - Driver For ETSG Initialization - ETSG initialization, in cooperation with the operator (user), all seeker and target static parameters by call into other subroutines.
Call - (INIT, PAGE, BELL, CKINCT, FILTST, INITCP, MLOAD, DELF, SEEK, ETARG, PULSE, FLARE, RUNETSG.

AVI.3
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:O, ETSG.CM:O)
   Call - (KEYIN, INIT, PAGE, BELL, FILTST, MLOAD, AXES, LDTCPU)

GENTRG - General Target Generator - GENTRG is called by "ETARG" to produce the target image based upon the parameters set in "ETARG".
   Input Files (IFLGS.SA:O, CN.SA:O, 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, IARRAY)
   Call - (ALF, 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)
- 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 callable 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)

AVI.I.5
LDTLR - (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)

NRT - Null Track Radius - Generates coordinates for null track radius hardware

Input - (NFOVR, ENTR, ISROT)
Call - (LDNTRR)

OUT - 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)

OUTC - Ship Character to Intecolor (Terminal)

Input - (A) A = Character to ship

OUTP - Output Character with Programmable Delay

OUTPLT - Output Subroutine for Display

Input - (IPTG, JJ, IPKP, NLVLS)
Call - (LDTLR, APKT, PKT)

OUTS - Output Character with Standard Delay

Input (A)

PAGE - Clear Screen

PCT - 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)

PICT - Target Display

Input - (IA, IR, MX, NLVLS)
Call - (INIT, PAGE, COLR, PLOT, DRSR, TEXT, GREY, ANMD)

PLOT - 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)

PLOT - Move * Pen * To (X, Y) coordinates

AVIII.6
Input - (X,Y,P)
    X = X coordinate value (0 to 159)
    Y = Y coordinate value (0 to 191)
    P = 2 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 - (X,Y,P)
    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:O, ETSG.CN:O)
    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, MWORD, 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)
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, ESY.SA:0, ETSG.CM:0)

Call - (KEYIN, INIT, PAGE, BELL, FILTST, DIR.LDS, 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, T-SFA, 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)

STTCH - Set Target Channel

Input - (ITRGT, ICHNL)

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

Call - (STTCH, STTGM, 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 - (FILST, 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 cursor 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
100 IMX(100), (100), A(10), AS(4), B(U), D(4), HS(4), W(4), US(4), C(4, 1)
20 DIM AS(4), T(4), PS(4), US(4)
30 PRINT "THE FOLLOWING IS A LIST OF SUBROUTINES EMULATING THE ";
40 PRINT "EBCDIC SOFTWARE. TYPE IN THE NUMBER CORRESPONDING TO THE ";
50 PRINT "SUBROUTINE YOU WANT TO RUN."
60 PRINT "1. THE MULTIPLIER."
70 PRINT "2. EBCDIC TO DECIMAL CONVERTER."
80 PRINT "3. DECIMAL TO EBCDIC CONVERTER."
90 PRINT "4. BINARY TO DECIMAL CONVERTER."
100 PRINT "5. DECIMAL TO BINARY CONVERTER."
110 PRINT "6. BINARY TO EBCDIC CONVERTER."
120 PRINT "7. EBCDIC TO DECIMAL CONVERTER."
130 PRINT "8. SUBROUTINE LSHL."
140 PRINT "9. SUBROUTINE LRA."
150 PRINT "10. SUBROUTINE CAV."
160 PRINT "11. SUBROUTINE CSS."
170 PRINT "12. SUBROUTINE CIV."
180 PRINT "13. SUBROUTINE CIC."
190 PRINT "14. SUBROUTINE CIV."
200 PRINT "SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
210 PRINT "(1.) (2.) (3.) (4.) (5.) (6.) (7.) (8.) (9.) (10.) (11.) (12.) (13.) (14.)"
220 PRINT "27. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
230 PRINT "28. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
240 PRINT "29. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
250 PRINT "30. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
260 PRINT "31. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
270 PRINT "32. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
280 PRINT "33. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
290 PRINT "34. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
300 PRINT "35. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
310 PRINT "36. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
320 PRINT "37. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
330 PRINT "38. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
340 PRINT "39. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
350 PRINT "40. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
400 PRINT "41. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
410 PRINT "42. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
420 PRINT "43. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
430 PRINT "44. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
440 PRINT "45. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
450 PRINT "46. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
460 PRINT "47. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
470 PRINT "48. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
480 PRINT "49. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
500 PRINT "50. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
510 PRINT "51. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
520 PRINT "52. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
530 PRINT "53. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
540 PRINT "54. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
550 PRINT "55. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
560 PRINT "56. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
570 PRINT "57. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
580 PRINT "58. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
590 PRINT "59. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
600 PRINT "60. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
610 PRINT "61. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
620 PRINT "62. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
630 PRINT "63. SUB 74, 90, 1090, 1210, 1330, 1440, 1510, 1920"
DECIMAL TO PXS CONVERTER

151 = 0
152 = P0
153 = P1
154 = P2
155 = P3
156 = P4
157 = P5
158 = P6
159 = P7
160 = P8
161 = P9
162 = P10
163 = P11
164 = P12

PAS TO DECIMAL CONVERTER

165 = 0
166 = 1
167 = 2
168 = 3
169 = 4
170 = 5
171 = 6
172 = 7
173 = 8
174 = 9
175 = "_"
176 = "_"
177 = "_"
178 = "_"
179 = "_"
180 = "_"
181 = "_"
1909PLINT "LEFT".,A
1910PRINT
1911REM "SUBR71UE 1RNA"
1912Z=O
1913PRINT "1.PUT (1 PUT TO BE KNOWN AS "A")"
1914INPUT Z
1915PRINT"1990"
1916Z=Z+0
1917REM "1(2=.5)
1918INPUT T
1919REM "1(3=.5)
2000REM "SUB81U8E CSS"
2001INPUT A:SIZE
2002REM "SUB81U8E CSS"
2003INPUT Y=O
2004INPUT "PUT 210:'";E
2005INPUT "PUT 2.1":PUT
2006INPUT "PUT 2.1":PUT
2007INPUT "SUB81U8E CSS":PUT
2008INPUT "SUB81U8E CSS":PUT
2009INPUT "SUB81U8E CSS":PUT
2010PRINT "PRESS A (1)"":PUT
2011PRINT "PRES~ A (1)"":PUT
2100PRINT "PRESS A (1)"":PUT
2101PRINT "PRESS A (1)"":PUT
2102INPUT A:SIZE
2103INPUT B:SIZE
2104INPUT C:SIZE
2105INPUT D:SIZE
2106INPUT E:SIZE
2107INPUT F:SIZE
2108INPUT G:SIZE
2109INPUT H:SIZE
2110INPUT I:SIZE
2111INPUT J:SIZE
2112INPUT K:SIZE
2113INPUT L:SIZE
2114INPUT M:SIZE
2115INPUT N:SIZE
2116INPUT O:SIZE
2117INPUT P:SIZE
2118INPUT Q:SIZE
2119INPUT R:SIZE
2120INPUT S:SIZE
2121INPUT T:SIZE
2122REM "SUB81U8E CSS"
2123REM "SUB81U8E CSS"
2124REM "SUB81U8E CSS"
2125REM "SUB81U8E CSS"
AIX.5
2540 PRINT "YAK?"; \n2550 IF P=U THEN 2620
2560 S(2)=S(1)/S(2); \n2570 GOTO 2620
2580 IF P(S(1))=S(2) THEN 2610
2590 K=S(1)*S(2); \n2600 GOTO 2620
2610 PRINT "THAT AND THAT CANNOT BOTH BE EQUAL TO ZER0!" \n2615 G0 T0 2540
2620 PRINT "VALID (KR) IN POINT TARGA(1)?" \n2630 INPUT T(1)
2640 ON T(1) T0 2750
2650 PRINT "SPLICE (CH) IN C(1)FA(1)?" \n2660 INPUT T(2)
2670 ON T(2) T0 2640
2680 S=3/5*(S(1)/4=1/16)
2690 S=4*(S(1)/4=1/16)
2700 IF S=4 THEN 12
2710 S=12
2720 PRINT "C=MUL (TAK)=",A
2730 PRINT "T=",A
2740 PRINT "GTA: SISFL 13"
4001 J=1
4010 X=1/X#1000
4014 Y=Z
4014 C=6/11=2/11
5000 Z=X/Y/(1+Y/2)
5020 Z=Z-2+5
5030 X<0 B=.01
5040 Z=Z+2+5
5050 X=1/Z
5060 Z=5+5
5070 P: PEP-1 "LOFL!" A
5080 K=PSB
9991 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

AXI.1
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 MOOS:

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:

1. Set desired BAUD Rate Switch on CRT and turn OFF the previously set rate.
2. 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 MOS 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

1. 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

1. 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

1. 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)

1. Type: 'COPY++filename1, filename2'

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

AXI.5
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: 'AAAAAAAAAAAAA . . $$' * (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

AXI.6
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 '$$' (hit ESCape key twice) marks the end of a command line.
4. Commands may be concatenated on one line (if you can keep track of them) without any extra delimiter characters.
5. MISTYTYPE? Use SHIFT-RUB (most consistant) or CNTRL-H (only in EDITOR) for BACKSPACE.
6. Use 'B' to position pointer at head of file.
7. Use 'Z' to position pointer at end of file.
8. Use n'T' to display n lines. Does not move pointer.
9. Use n'L' to skip n lines. 'L' positions the pointer JUST AFTER THE LAST CARRIAGE RETURN. 'L' counts carriage returns. n may be negative to backup lines.
10. To input a new program (or a new block of statements), use the 'I' command. Type one 'I', then enter the entire block of code as if using a typewriter and then type $$ (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 PRECEDE 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 'Currentstring$newstring$-LLT' to change a string of characters and display the corrected line.

AXII.2
### TABLE 2. EDITOR COMMAND SUMMARY

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>* A</td>
<td>Append. Appends input text from the System Reader Device to the edit buffer.</td>
</tr>
<tr>
<td>* B</td>
<td>Beginning. Moves the edit buffer pointer to the beginning of the edit buffer.</td>
</tr>
<tr>
<td>* Cstring1$string2</td>
<td>Change. Replaces the first occurrence of &quot;string 1&quot; with &quot;string 2&quot;.</td>
</tr>
<tr>
<td>nD</td>
<td>Delete. Deletes n characters from the edit buffer.</td>
</tr>
<tr>
<td>E (tape)</td>
<td>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.</td>
</tr>
<tr>
<td>E (disc)</td>
<td>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.</td>
</tr>
<tr>
<td>F (tape)</td>
<td>Tape Leader/Trailer. Writes 50 NULL characters into the system punch device.</td>
</tr>
<tr>
<td>F (disc)</td>
<td>The F command is ignored.</td>
</tr>
<tr>
<td>* Istring</td>
<td>Insert. Inserts characters or lines of text into the edit buffer.</td>
</tr>
<tr>
<td>* nK</td>
<td>Kill lines. Deletes n lines from the edit buffer.</td>
</tr>
<tr>
<td>* nL</td>
<td>Line. Moves the edit buffer point n lines.</td>
</tr>
<tr>
<td>nM</td>
<td>Move character pointer. Moves the edit buffer pointer n characters.</td>
</tr>
<tr>
<td>Nstring (tape)</td>
<td>Search File. Searches file for first occurrence of &quot;string&quot;.</td>
</tr>
<tr>
<td>Nstring (disc)</td>
<td>Search File. Searches file for first occurrence of &quot;string&quot;. If &quot;string&quot; is not found, returns control to the disc operating system.</td>
</tr>
<tr>
<td>nP</td>
<td>Punch. Punches n lines from the edit buffer to the System Punch Device.</td>
</tr>
<tr>
<td>* Sstring</td>
<td>Search. Searches the edit buffer for the first occurrence of &quot;string&quot;.</td>
</tr>
</tbody>
</table>

*Most often used commands

$ ESC Key

AXII.3
### Table 2. Editor Command Summary (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Type. Types n lines from the edit buffer to the System Console Device.</td>
</tr>
<tr>
<td>X (tape)</td>
<td>EXbug. Returns control to EXbus.</td>
</tr>
<tr>
<td>X (disc)</td>
<td>The X command is an illegal command in the disc version of the editor.</td>
</tr>
<tr>
<td>*</td>
<td>End of edit buffer. Moves the edit buffer pointer to the end of the edit buffer.</td>
</tr>
<tr>
<td>Control H</td>
<td>Backspace. Causes the last character entered in the command mode to be typed on the System Console Device and deleted from the command.</td>
</tr>
<tr>
<td>Control X</td>
<td>Cancel. Causes all commands following the last prompt to be deleted and another prompt to be typed.</td>
</tr>
</tbody>
</table>

### Table 3. Editor Messages

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M6800 RESIDENT EDITOR n.n</td>
<td>Printed upon initiation of editor. Revision is specified by n.n.</td>
</tr>
<tr>
<td>@</td>
<td>Prompt. Editor is waiting for a command.</td>
</tr>
<tr>
<td>???</td>
<td>Illegal command.</td>
</tr>
<tr>
<td>CAN'T FIND &quot;string&quot;</td>
<td>Editor cannot find the string specified by Search or Change command.</td>
</tr>
<tr>
<td>BELL</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 separately 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.

AXI.6
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
DRIVE: 0  DISK I.D.: MDOS
BINDK  .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

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

AXIII.2
Drive: 1  Disk I.D.: JTR1

Drive: 1  Disk I.D.: SEEK

Total directory entries shown: 041/$29

Total directory entries shown: 010/$0A
DRIVE : 1 DISK I.D. : JTH:1
CK:INIT .RO
RD"CPU .RO
ALP .RO
FOV .SA
ZS .SA
PICT .RO
LD"LR .RO
GRAPPH .RO
SSTSEEK .RO
DIP .RO
PMI .RO
ETLB .RO
LDDSPC .RO
TRG .SA
ASPGEN .RO
IRMD .RO
ARCTRIG .RO
DFT .RO
LDRETS .RO
STTCGRN .RO
RUNETSG .RO
LRUN .CF
CH .SA
LDPTTG .RO
GREY .RO
LDPLS .RO
STARC .RO
LSH .RO
RDTMP .RO
IFLGS .SA
CLRTMP .RO
VERT .RO
APICT .RO
LDTCPU .RO
RDDSPC .RO
AXES .RO
ETSGGO .RO
STAG .RO
LDNTRR .RO
LDASP .RO
RESP .SA

TOTAL DIRECTORY ENTRIES SHOWN : 041/$29

DRIVE : 1 DISK I.D. : SCRATCH
PUNIC .LX
DSD .LX
EMT .CH
PUNIC .L0
PLOT .SA
DSD .L0
DSD .SA
PLOT .RO
OLDSD .LO
INTFAC .SA
INTFACND .SA
TOTAL DIRECTORY ENTRIES SHOWN : 011/$0B

AXII.4
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
ARCTTRIG .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

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

DRIVE : 1  DISK I.D. : ETARG
LTG .CF
PICT .SA
ASPGEN .RO
ET5 .CM
ETARG .RO
ETARG .SA
ASPGEN .SA
APICT .SA
TOTAL DIRECTORY ENTRIES SHOWN : 008/$08

AXIII.5
:1
DRIVE : 1  DISK I.D. : SEEK
RETI L .SA
LSK  .CF
NTR  .RO.
RE TGEN .SA
SEEK .RO
NTR  .SA
SEEK  .CM
SEEK  .SA
RE TGEN .RO
TOTAL DIRECTORY ENTRIES SHOWN : 009/$09

:1
DRIVE : 1  DISK I.D. : GRL
L T E .CF
FLARE .SA
VA RLIST .SA
ET SG .CM
PUL SEJ .CM
PUL SEJ  .RO
ET SG .RO
LP J .CF
ET SG  .SA
FLARE .CM
PUL SEJ .SA
FLARE  .RO
LFLR .CF
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D

:1
DRIVE : 1  DISK I.D. : NONAME
F 3  .SA
F 4  .SA
F 5  .SA
ABASIC .CH
DCBSIM .SA
DCBSIM .LO
FTNLBX  .RO
DCBSIM  .LX
DCBSIM .CM
F 1  .LX
F 1  .LD
F 1  .SA
F 2  .SA
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D

:1
DRIVE : 1  DISK I.D. : ET2
FLARE .SA
L T E .CF
VA RLIST .SA
PUL SEJ .CM
ET SG  .CM
PUL SEJ  .RO
ET SG .RO
LP J .CF
ET SG  .SA
PUL SEJ .SA
FLARE .CM
FLARE  .FD
LFLR .CF
TOTAL DIRECTORY ENTRIES SHOWN : 013/$0D

AXIII.6
DRIVE: 1  DISK I.D.: ETSGDPO
ETSG .CM
PULSEJ .CM
DCON .SA
DELL .SA
ETHALG .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

DRIVE: 1  DISK I.D.: SDBASIC
sdas .CM
sdbcom .CM
tmtest .BA
test .BA
test .LK
runros .BA
sedit .CM
sdrun .CM
TOTAL DIRECTORY ENTRIES SHOWN: 008/$08

DRIVE: 1  DISK I.D.: SYSTEM
kate .LU
P1 .SA
seeker .SA
dublin .SA
P2 .SA
promprog .CM
P3 .SA
tsi .SA
TOTAL DIRECTORY ENTRIES SHOWN: 008/$08

DRIVE: 1  DISK I.D.: TARGET
S .SH
T .SH
abasic .CM
plum .SA
TOTAL DIRECTORY ENTRIES SHOWN: 004/$04
AXIII.7
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

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

<table>
<thead>
<tr>
<th>Distributor</th>
<th>No. of Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT Research Institute</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: GACIAC</td>
<td></td>
</tr>
<tr>
<td>10 West 35th Street</td>
<td></td>
</tr>
<tr>
<td>Chicago, IL 60616</td>
<td></td>
</tr>
<tr>
<td>DRSMI-LP, Mr. Voigt</td>
<td>1</td>
</tr>
<tr>
<td>-RD, Dr. Hallum</td>
<td>10</td>
</tr>
<tr>
<td>-RPR</td>
<td>15</td>
</tr>
<tr>
<td>-RPT, Record Copy</td>
<td>1</td>
</tr>
<tr>
<td>Reference Copy</td>
<td>1</td>
</tr>
</tbody>
</table>