NASA Technical Memorandum 85896

A Hardware/Software Simulation for the Video Tracking System of the Kuiper Airborne Observatory Telescope

Glenn A. Boozer, Darrell D. McKibbin, Michael R. Haas and Edwin F. Erickson

February 1984

NASA

National Aeronautics and
Space Administration

NASA Technical Memorandum 85896

A Hardware/Software Simulation for the Video Tracking System of the Kuiper Airborne Observatory Telescope

Glenn A. Boozer, Boozerco, San Jose, California
Darrell D. McKibbin, Ames Research Center, Moffett Field, California
Michael R. Haas, Mycol, Incorporated, Sunnyvale, California
Edwin F. Erickson, Ames Research Center, Moffett Field, California

NASA
National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

TABLE OF CONTENTS

1.0 SUMMARY

This simulator was created so that C-141 Kuiper Airborne Observatory (KAO; ref. 1) investigators could test their Airborne Data Acquisition and Management System (ADAMS; ref. 2) software on a system which is generally more accessible than the ADAMS on the plane. An investigator can currently test most of his data acquisition program using the data computer simulator (ref. 2) in the Cave. (The "Cave" refers to the ground-based computer facilities for the KAO and the associated support personnel.) The main Cave computer is interfaced to the data computer simulator in order to simulate the data-Exec computer communications (ref. 2). However, until now, there has been no way to test the data computer interface to the tracker. The simulator described here simulates both the KAO Exec and tracker computers with software which runs on the same Hewlett-Packard (HP) computer as the investigator's data acquisition program. A simulator control box is hardwired to the computer to provide monitoring of "tracker" functions, to provide an operator panel similar to the real tracker, and to simulate the 180° phase shifting of the chopper square-wave reference with beam switching. If run in the Cave, one can use their Exec simulator and this tracker simulator.

The investigator's data acquisition program needs no modification to use this simulator. The data program makes its normal calls to the standard tracker and Exec subroutines. These subroutines, in turn, call simulator versions of the tracker and Exec driver routines, which replace the standard versions at load time. These driver routines communicate with a background program which replaces the tracker and the Exec computers. This programming structure requires the multitasking environment of RTE IVb (ref. 3), HP's disk based operating system. An early version of the simulator software can also be run under BCS (ref. 4), HP's memory resident operating system. However, this version does not provide as good a simulation of the tracker, because the switches on the front panel of the simulator control box are not always live.

The simulator control box has buttons and indicator LEDs for right beam, left beam, nod enable, telescope steady, loss of track, Exec communications status, and tracker communications status. To test a data acquisition program with an infrared detector system and a chopped source on-line, run the chopper reference through the simulator control box. The reference will switch phase when beam switching occurs, thereby causing the demodulated detector signal to change polarity in the data recording system. Thus, practical testing of the entire experiment can be accomplished using the investigator's detector system and software, the ADAMS (data computer) simulator, and the present telescope simulator hardware and software.

2.0 GETTING STARTED

2.1 Files Required

The source files for use with the simulator are &NSIM, which contains the main simulator program, &NBLIB, which contains the drivers for the simulator control box, and &NSLIB, which contains replacement drivers for the standard tracker and Exec drivers (in the investigator's data acquisition program). Source listings for these routines are given in the Appendix. The routines &NSIM and &NSLIB are written in Fortran IV, but &NBLIB is written in HP Assembly (ref. 5). The Fortran files should work equally well under Fortran IV+.

2.2 Changes to the Data Acquisition Program

The telescope simulator software was designed so it can be used with an investigator's standard data acquisition program. The only changes required are to the procedure used to load it. However, as described below, one change to the investigator's data acquisition program is desirable.

When the investigator's data program is loaded and executed, the first call to a tracker or Exec subroutine will run the simulator program, NSIM. However, NSIM does not automatically stop when the data acquisition program is stopped. It must either be explicitly stopped before the data program is run again, or the data program must be slightly modified so that it will stop NSIM. The code required is:

```
C
C.....TELL THE SIMULATOR TO STOP
C
      CALL SMOVE(20HBR,NSIM        ,,,, ,1,20,IBUF,1)
      CALL MESSS(IBUF,7)
```

These lines of code should be put at the very end of the data program, just ahead of the 'STOP' or 'CALL EXEC(6)' statement. They can remain in the program permanently. If the simulator program is not running and this code is executed, it will be ignored by the operating system. When adding this code, remember that IBUF is an integer array which must be dimensioned 40 or more. Since this code is the last to be executed, it can use any user defined array that is sufficiently large.

2.3 Compiling and Loading the Programs

The simulator program, NSIM, can be compiled with:

```
RU,FTN4,&NSIM,,-
RU,ASMB,&NBLIB,,-
```

and loaded with:

```
RU,LOADR,#NSIM
```

where the file #NSIM contains the following commands:

```
OP,LB
OP,DC
RE,%NSIM
SE,%NBLIB
EN
```

Then type 'SP,NSIM' to save the program on disk. NSIM needs to be loaded only once, as it is the same for all investigators.

The data acquisition program and all of its subroutines can be compiled in the usual manner. To load the program, the KAO utility subroutines TBLT AND BLT must be replaced with the simulator versions of these same routines. To do this, simply comment the appropriate lines out of the loader procedure file with a double asterisk and relocate the module %NSLIB in their place. An example of a generalized loader procedure file using the simulator drivers is shown below. The lines '...' indicate the location of the user's normal loader commands.

```
ECHO
OP,...
OP,...
RE,...
SE,...
SE,...
...
...
**,LOAD TRACKER LIBRARY,**
...
...
**RE,R$BLT
**RE,R$TBLT
RE,%NSLIB
...
...
EN
```

To operate with the 'real' Exec when running on the Cave simulator, use the command 'RE,R$BLT' in the loader procedure file to relocate BLT and then use 'SE,%NSLIB' to search for the simulator version of the tracker routine TBLT.

2.4 Running the Programs

To run the data acquisition program using the telescope simulator, program NSIM must not be running, but must be RP'ed (ref. 6). To stop NSIM if it is running, execute the command 'SYBR,NSIM' from the file manager (ref. 6). If NSIM does not respond with "NSIM STOP", then it is 'stuck' and needs an 'OF,NSIM,8' command issued from the file manager two times. Usually the first 'OF,NSIM,8' command is sufficient, but the second one never hurts. NSIM can be RP'ed by typing 'RP,NSIM' from the file manager. If this command returns a FMGR 023 error, the program is already RP'ed. In this case, ignore the error and proceed with running the data acquisition program.

These commands can be put in a procedure file and 'transferred to' from the file manager. The following procedure file will work for running a program called NDATA:

```
:SYBR,NSIM
:OF,NSIM,8
:OF,NSIM,8
:RP,NSIM
:RU,NDATA
```

If the data acquisition program aborts with a SC05 error, NSIM was not RP'ed.

3.0 AN OVERVIEW OF THE SIMULATOR

This simulator was designed to simulate the rudimentary functions of the tracker and the Exec. It does not provide a complete and thorough simulation of either. However, it is very useful and has allowed us to find and correct many obscure errors and to fine-tune our software. The one major area not simulated is the raster-type tracker functions. These could be added at a later date if desired. The simulator functions currently supported are:

The tracker:

	Control the simulator control box LEDs
BNOD	Turn on nod mode
MOVE	Move the telescope (nod)
ERS	Turn off raster scan or nod mode
STEDY	Wait until telescope is steady
TRST	Update the tracker status area of Data CPU common
	Tracker communications up/down

The Exec:

	Read the Exec housekeeping buffer
	Exec communications up/down

The only Exec housekeeping words (ref. 2) that are updated are UT and RANG, Universal Time and Rotation ANGle, respectively. The RANG variable is incremented by 0.20 degrees each minute. The object label is defined as "NO OBJECT(SIMULATOR)".

A flow diagram for a data acquisition program's interfaces with program NSIM is given in figure 1. The only nonstandard routines in the data program are the driver routines TBLT and BLT, which handle tracker and Exec communications, respectively. The simulator versions of these routines call SEXEC (simulator EXEC), which communicates with the simulator program (NSIM) using bidirectional class READS and WRITES (ref. 7). The standard KAO routines do EXEC (ref. 7) reads and writes to the appropriate logical units instead.

4.0 OPERATING THE SIMULATOR

The simulator control box operates much like the tracker, although the lights are smaller and the buttons are in different places. When it is first turned on, the lights can be in one of three possible configurations:

1) All lights on,
2) All lights off, or
3 Some lights on/off.

The cases 1 and 2 are normal. Case 1 occurs when the computer has been booted and the simulator has not yet been run. Case 2 occurs when the simulator program turned off all of the lights before terminating. Case 3 means that the simulator program was aborted or is still running and needs to be stopped. To stop the program, check that no one else is using it and then type 'SYBR,NSIM'. If the message "NSIM STOP" is not displayed shortly, type 'OF,NSIM,8' twice.

The simulator program displays "NSIM ON" on the terminal when it is started. The lights on the simulator control box should then be as follows (from right to left):

Bit	Function	Status
0	Left Beam	Off
1	Right Beam	Off
2	Nod Mode	Off
3	Telescope Steady	On
4	Loss of Track	Off
5	Exec Communications	On
6	Tracker Communications	On

Pressing a switch associated with bits 3, 4, 5, or 6 should toggle the corresponding light. Bits 0 and 1 are mutually exclusive and, in addition, are controlled by bit 2. That is, if nod mode (bit 2) is turned off, then the left (bit 0) and right beam (bit 1) lights will both be off. If nod mode (bit 2) is enabled, either the left or right beam light will be on.

Note that bits 5 and 6, which simulate the Exec and tracker communications status, respectively, are additions not found on the real tracker. Bit 5 simulates loss of Exec communications and bit 6 simulates loss of tracker communications. The data terminal should display the appropriate error messages when this occurs. This can be a good test of the error handling portion of a data acquisition program. Interesting tests are to change beams or turn nod mode off while the data acquisition program is taking data. These types of errors do not generate a 'Loss of Track' error, so the data program may be fooled.

If the simulator control box is powered on and all of the lights are off, either everything is okay or the cable that connects the box to the computer is not connected properly. If NSIM runs and says "NSIM ON" and no lights come on, then the cable is probably disconnected.

If the lights come on in the correct configuration initially, but the switches are not live, then the cable is probably connected, but the NSIM program reacts as if the cable is disconnected. This can occur because NSIM was designed to run quite well without the simulator box. There are three conditions that can cause NSIM to act as if the the box is disconnected:

1) The simulator box is powered off,
2) The interconnecting cable is not connected at both ends, or
3) The toggle switch on the back of the control box is in the 'bit 13' position.

If one of these conditions exists, NSIM will set the tracker status to:

Exec communications	Up
Tracker communications	Up
Loss of track	Off
Telescope steady	On

To see NSIM set these defaults, toggle the 'bit 13' switch on the back of the simulator box.

5.0 PROGRAM NSIM, PROGRAMMER'S NOTES

The simulator program is divided into five sections:

1) Initialization
2) Exec CPU Common
3) Simulator Control Box
4) TBLT and BLT requests
5) Errors

A source listing is provided in the Appendix. NSIM has a priority of 19. To prevent it from dominating the CPU, it suspends itself for 20 msec within its main loop.

5.1 Initialization

The NSIM run command looks like this:

```
"RU,NSIM,LU,177,CLASS NUMBER"
```

The LU passed in the run string is the logical unit of the terminal running the data acquisition program. The '177' is used to make sure that NSIM was not run from the file manager because NSIM is designed to run from the data acquisition program through TBLT and BLT subroutine calls. The class number passed in the run string was assigned by the SEXEC subroutine from within the data acquisition program. When properly run, NSIM prints "NSIM ON" and enters its main loop. For a detailed description of the class I/O communications between NSIM and SEXEC, see section 7.3

5.2 EXEC CPU Common

The variable UT (Universal Time) is updated approximately every 20 msec. The variable RANG (Rotation ANGle) is incremented by 0.20 degrees each minute. The object label is defined as "NO OBJECT(SIMULATOR)".

5.3 Simulator Control Box

The simulator control box switches are polled approximately every 20 msec. If any of the three conditions which cause the box to be 'disconnected' are present (see section 4.0), NSIM will set the simulator to its default conditions. If the simulator box is 'connected', its switches can be used to change the Exec or tracker status.

5.4 TBLT and BLT Requests

The TBLT and the BLT requests are distinguished, as are the read and write requests. If there is a read request, the data is read. If there is a write request, a data header and the data are written.

5.5 Errors

If an error occurs in the initialization section, NSIM tries to deallocate the class number. If an unrecognizable data header buffer is read, NSIM reports "/NSIM: BAD DATA IN CLASS HEADER" on the data terminal and the contents of the header record are printed. The 'bad' class buffer is cleared from the class queue and a special error-reporting data header is sent to SEXEC. If there is not enough system available memory (SAM) (ref. 8) for NSIM to send the requested data to SEXEC, NSIM sends a message telling about the "SAM shortage" and then sends a special error-reporting data header to SEXEC. Since these errors occur seldom, if ever, subroutine SEXEC does not check for errors in the buffer, but simply returns normally.

6.0 THE SIMULATOR CONTROL BOX DRIVER

6.1 Overview

The control box driver is written in HP Assembly and uses the $LIBR and $LIBX system calls (ref. 7) to allow direct manipulation of the HP 12566B Microcircuit Interface card (ref. 9). This avoids the need for a special purpose driver that would have to be incorporated into the operating system, thus defeating the telescope simulator's main goal: transparency.

A source listing for the driver is given in the Appendix. The variable 'BOX' (located on line 131 of the driver) corresponds to the select code of the Microcircuit Interface when installed in the simulator CPU backplane. The select code of the slot containing the card can be found on the CPU chassis just to the right of the card. It is an octal number, as is the variable 'BOX' (Consult the Assembler Reference Manual (ref. 5) for information on using octal numbers).

6.2 Hardware Configuration

The HP 12566B Microcircuit Interface card with option 002, which provides for the transfer of positive-true signals to and from the I/O device (ref. 9), should be placed in an empty slot in the CPU backplane. If there is a priority interrupt fence, the card should be placed above it. Remember to turn off the power to the CPU before inserting or removing cards, or connecting cables. The system's behavior should be unaffected by the installation of this card.

7.0 THE SUBROUTINES SEXEC, TBLT, AND BLT

7.1 Overview

The source listings for SEXEC, TBLT, and BLT are in the Appendix and a diagram of program flow is given in figure 1. SEXEC is the simulator's EXEC subroutine. SEXEC does all the class I/O between the data acquisition program and NSIM. The arguments of subroutine SEXEC are different from those of an EXEC call. The simulator versions of TBLT and BLT are similar to the standard TBLT and BLT routines, except that the EXEC calls have been converted to SEXEC calls and there are fewer possible error returns.

7.2 Initialization

The first call to SEXEC requests a class number for use in communicating with NSIM and then schedules NSIM with an EXEC(10) call. This class number and the logical unit of the data terminal are passed in the run string.

7.3 Class I/O

A single class number is used to do bidirectional data transfers. NSIM polls the class number with a class GET that leaves the data in the class buffer. If the buffer is a header buffer and the header is intended for NSIM, then NSIM will take the header buffer from the class buffer and do as directed by the header. The header may indicate that a data buffer for NSIM is the next item in the class queue. In that case, NSIM will also take the data buffer from the class queue.

SEXEC puts a header buffer and a data buffer into the class queue. If SEXEC is expecting a response from NSIM, SEXEC polls the class number in much the same fashion as NSIM. When SEXEC gets the expected header buffer and data buffer, SEXEC returns the appropriate data to TBLT or BLT, whichever called SEXEC.

If SEXEC detects an error-reporting header buffer, it removes it from the queue and returns normally to its calling subroutine. This lack of error checking on the part of SEXEC is generally not a problem, as this type of class I/O has proven extremely reliable.

8.0 HARDWARE AND SOFTWARE REQUIREMENTS

The following hardware is required:

1) HP 2115, 2116 OR 2117 Computer (HP 1000 M, E, OR F), and
2) HP 12566B-002, Microcircuit Interface Card, Positive True (ref. 9),

and the following hardware is optional, but desirable:

3) Simulator control box,
4) Interconnecting cable.

The source code in the Appendix contains the interface card strap settings and a description of the cable wiring. Chapter 9 gives a more detailed description of the simulator control box and associated hardware.

The following software is required:

1) RTE-IVB Operating System,
2) &NSIM, Telescope Simulator,
3) &NBLIB, Simulator Control Box Driver,
4) &NSLIB, SEXEC and the Simulator versions of TBLT and BLT,
5) A Data Acquisition Program,
6) Regular Tracker Software.

9.0 SIMULATOR CONTROL BOX HARDWARE

The simulator control box provides monitoring of the HP 1000 computer output bits and provides push-button control of the input bits. Note that this unit may be used for any I/O monitoring purpose, as well as for telescope simulation.

A five-foot cable is provided to connect the simulator control box to the HP 1000 computer. At the HP computer a standard HP 48 pin connector (part of I/O card 12566-60025 option 002 kit) connects to the I/O card with HP cable 8120-1846 (also part of the I/O kit). The cable is terminated in a Cannon 20-41S MS 3126 F connector, which mates with a Cannon connector on the simulator control box. For laboratory work at Ames Research Center in Building 245, a similar cable is connected to four 10-point terminal blocks. These terminal blocks are wall mounted in the first floor breezeway of Building 245. The terminal blocks furnish test points for all input and output bits. At the terminal blocks a very long cable is connected to input bits 0 through 13. Bits 14 and 15 are not used because the cable does not contain sufficient wires. This cable enters Room 115 from the breezeway and is long enough to use in Room 111.

Within the simulator control box, the HP computer output bits 0 thru 13 sink current for light emitting diodes (LED). The LED's have current limiting resistors connected to the positive 5 volt supply. When an HP computer output bit is low (i.e., approximately 0 volts), the corresponding LED will be lit.

Bit 0 or 1 is selected as one input to an exclusive-or gate (ref. 10) by a rear panel switch. The other input to the exclusive-or gate is a TTL level, nominal square wave, chopper reference signal. This chopper signal is isolated by an optical coupler connected to a front panel mounted BNC connector. Thus the input chopper signal comes in through the BNC to the opto-isolator, which then transfers the signal to the exclusive-or gate. The selected bit, 0 or 1, then provides 0° or 180° phase shifting of the chopper signal. The output of the exclusive-or gate in normal or inverse phase is fed to a panel mounted BNC connector. For tracker simulation, bits 0 and 1 represent right and left beams and are, therefore, the inverse of each other. Thus, the phase of the chopper is switch selectable to match either beam position and inverts with beam change, simulating the phase shift of the detector signal relative to the chopper reference when the telescope chopper switches position.

Within the box, HP input bits 0 thru 13 are connected to panel mounted, grounded, normally open, push-button switches. Thus, pushing a button corresponding to a particular bit causes that input to go to 0 volts. No pull-up resistors are provided, since source currents from the HP computer inputs provide pull-up. Also, no debounce circuitry is provided and software must included debouncing.

Input bit 13 may be selected as a special, hookup test bit by using a second rear panel switch. This switch connects a transistor to input bit 13. When power is on, this transistor is on, and if both ends of the cable are connected, the HP computer will read input bit 13 as low (i.e., approximately 0 volts). If the simulator power is off, or one or both ends of the cable are disconnected, input 13 will read high (i.e., approximately 5 volts). Therefore, when the rear panel switch is closed, bit 13 indicates that the simulator is connected to the computer and is powered.

The simulator control box also contains a Datel model UPM-5/1000, 5.0 volt, 1.0 amp power supply, a power indicator light, a power switch, and a 0.5 amp fuse. Figures 2 through 5 and Table 1 show the front panel and circuit diagrams for the simulator box.

For tracker simulation the switches on the back of the box are in the ground and bit 0 positions. These are the upward positions of the switches.

10.0 APPENDIX, PROGRAM SOURCE LISTINGS

10.1 Source Listing for Program NSIM

```
FTN4
      PROGRAM NSIM(3,19), TRACKER AND EXEC SIMULATOR (831021.1525)
      IMPLICIT INTEGER(A-Z)
      REAL UT,RANG
      LOGICAL IFBRK
      EXTERNAL IFBRK
C
C
C  TRACKER COMMON
C
C  SEE BLOCK DATA FOR COMMENTS ABOUT WHAT TRACKER COMMON WORDS ARE USED
C  FOR WHAT.
C
      COMMON /TRACK/ TCOM(200)
C
C
C  EXEC COMPUTER COMMON
C
C  SEE BLOCK DATA FOR COMMENTS ABOUT WHAT EXEC COMMON WORDS ARE USED
C  FOR WHAT.
C
      COMMON /EXECC/ ECOM(500)
C
C
C  LET'S DEFINE SOME USEFUL VARIABLES.
C
      INTEGER TSUBF(15)
      INTEGER MYNAME(3)
      INTEGER IBUF(5)
      INTEGER ITIME(5)
C
C
C  SOME USEFUL EQUIVALENCES
C
      EQUIVALENCE(TCOM(54),IPAUS)
      EQUIVALENCE(TCOM(55),IRSON)
      EQUIVALENCE(TCOM(56),NODON)
      EQUIVALENCE(TCOM(57),LR)
      EQUIVALENCE(TCOM(58),LOSS1)
      EQUIVALENCE(TCOM(78),IRSX)
      EQUIVALENCE(TCOM(79),IMOVE)
      EQUIVALENCE(TCOM(80),IXNOD)
C
      EQUIVALENCE(ECOM(8),IYEAR)
      EQUIVALENCE(ECOM(151),UT)
      EQUIVALENCE(ECOM(165),RANG)
C
      EQUIVALENCE(IBUF(1),DEST)
      EQUIVALENCE(IBUF(2),ICODE)
      EQUIVALENCE(IBUF(3),CPU)
      EQUIVALENCE(IBUF(4),ISUBF)
      EQUIVALENCE(IBUF(5),SUBLEN)
C
C
C STORAGE FOR LAST SWITCH SETTINGS
      INTEGER LASTSW
C
      LOGICAL IBIT
C
C
C FUNCTIONS AND DATA
C
C LOGICAL FUNCTION TO TEST TO SEE IF A BIT IS ON
      IBIT(I,J)=(IAND(I,J).NE.0)
C
C DATA
      DATA EXUP/1/
      DATA TRKUP/1/
C
C
C  TRACKER SUBFUNCTION OFFSETS INTO TRACKER COMMON
C  0 MEANS I DO NOT KNOW WHAT THE OFFSET IS.
C
      DATA TSUBF/52,78,1,41,114,79,0,24,0,0,25,60,138,0,56/
C
C
C  LET'S SET UP SHOP.
C  FIRST, MAKE SURE THAT WE HAVE BEEN RUN FROM "NDATA" NOT THE KEYBOARD.
C
      CALL RMPAR(MYNAME)
      LU=MYNAME(1)
      I=MYNAME(2)
C
C
C  GET CURRENT SWITCH SETTING.  MAKE LAST SWITCH SETTING EQUAL TO CURRENT.
C
      CALL TSGET(LASTSW)
C
C
C  CLASS IS OUR CLASS NUMBER WITH THE 'NO WAIT' BIT SET.
C  CLASSC HAS THE "LEAVE THE BUFFER IN MEMORY BIT SET." JUST 'C'HECKING.
C
      CLASS=IOR(MYNAME(3),100000B)
      CLASSW=IAND(MYNAME(3),077777B)
      CLASSC=IOR(CLASS,40000B)
      CALL CHGLU(LU)
      CALL PNAME(MYNAME)
      WRITE(LU,11) MYNAME
11    FORMAT("/",2A2,A1,": ON")
C
C
C  WERE WE RUN BY NDATA?
```

```
C
      IF(I.EQ.177) GOTO 19
      WRITE(LU,10) MYNAME
10    FORMAT("/",2A2,A1,
     &": THIS PROGRAM MUST NOT BE RUN FROM THE FILE MANAGER")
      STOP
C
C
C  IS OUR NAME "NSIM"?.  IF NOT, KILL OURSELF.
C
19    IF(MYNAME(1).EQ.2HNS .AND. MYNAME(2).EQ.2HIM .AND.
     & MYNAME(3).EQ.2H  ) GOTO 20
      WRITE(LU,21) MYNAME
21    FORMAT("/",2A2,A1,": TYPE 'RP NSIM', THEN TRY AGAIN!")
      GOTO 30
C
C
C  START MAIN LOOP
C
C
C  CHECK TO SEE IF THE BREAK FLAG IS SET.
C  IF IT IS, TRY TO DEALLOCATE THE CLASS NUMBER.
C  TURN OFF ALL TRACKER LIGHTS
C
20    IF(.NOT. IFBRK(I)) GOTO 40
C
C
C  READ THE CLASS BUFFERS UNTIL ALL OF THEM HAVE BEEN READ.
C
30    CALL EXEC(21,CLASS,I,1)
      CALL ABREG(A,B)
      IF(A.GE.0) GOTO 30
C
C
C  DEALLOCATE THE CLASS NUMBER
C
      CALL EXEC(21,IAND(CLASS,017777B),I,1)
      CALL TSPUT(-1)
      STOP
C
C
C  UPDATE EXEC HOUSEKEEPING
C
40    CALL EXEC(11,ITIME,IYEAR)
      UT=FLOAT(ITIME(4))+FLOAT(ITIME(3))/60.0
     &+FLOAT(ITIME(2))/3600.0
     &+FLOAT(ITIME(1))/360000.0
C
C INCREMENT THE ROTATION ANGLE .20 DEGREES FOR EACH 1 MINUTE
C
      IF(NSEC.EQ.ITIME(3)) GOTO 50
      NSEC=ITIME(3)
      RANG=RANG+.20
      IF(RANG.GE.360.0) RANG=0.0
C
C
C  CHECK THE STATUS OF THE CONTROL BOX SWITCHES.
C
C
C READ SWITCHES ON BOX
C
50    CALL TSGET(NOWSW)
C
C
C IS THERE A CHANGE FROM THE LAST TIME WE LOOKED AT THEM?
C
      IF(NOWSW.EQ.LASTSW) GOTO 100
C
C
C THE SWITCHES ARE SOME HOW DIFFERENT NOW.
C IS THE CABLE CONNECTED TO THE BOX??
C
      IF(.NOT.IBIT(NOWSW,020000B)) GOTO 100
C
C THE CABLE IS DISCONNECTED.  SET SOFTWARE TO DEFAULT STATE.
C
C TRACKING ON STAR 1
C TELESCOPE STEADY.
C EXEC COMMUNICATIONS UP
C TRACKER COMMUNICATIONS UP
      IPAUS=1
      LOSS1=0
      EXUP=1
      TRKUP=1
      GOTO 110
C
C FIND WHICH SWITCH(ES) CHANGED
100   CHNGSW=IXOR(NOWSW,LASTSW)
C
C SET TRACKER TO CURRENT STATE AS PER THE SWITCHES
      IF(IBIT(CHNGSW,1B) .AND. IBIT(NOWSW,1B)) LR=0
      IF(IBIT(CHNGSW,2B) .AND. IBIT(NOWSW,2B)) LR=1
      IF(IBIT(CHNGSW,4B) .AND. IBIT(NOWSW,4B)) NODON=MOD(NODON+1,2)
      IF(IBIT(CHNGSW,10B) .AND. IBIT(NOWSW,10B)) IPAUS=MOD(IPAUS+1,2)
      IF(IBIT(CHNGSW,20B) .AND. IBIT(NOWSW,20B)) LOSS1=MOD(LOSS1+1,2)
      IF(IBIT(CHNGSW,40B) .AND. IBIT(NOWSW,40B)) EXUP=MOD(EXUP+1,2)
      IF(IBIT(CHNGSW,100B) .AND. IBIT(NOWSW,100B)) TRKUP=MOD(TRKUP+1,2)
C
C SAVE CURRENT SWITCH PATTERN FOR LATER
110   LASTSW=NOWSW
C
C SET TRACKER TO CURRENT SOFTWARE STATE
      LED=1B
      IF(LR.EQ.1) LED=2B
      LED=LED+4B
      IF(NODON.EQ.0) LED=0B
      IF(IPAUS.EQ.1) LED=LED+10B
      IF(LOSS1.EQ.1) LED=LED+20B
      IF(EXUP.EQ.1) LED=LED+40B
      IF(TRKUP.EQ.1) LED=LED+100B
C
C THE LEDS ARE OFF FOR 1 BITS AND ON FOR 0 BITS.
C WE SHALL INVERT THE BITS TO ACCOUNT FOR THIS.
      LED=NOT(LED)
C
C SET THE LED'S TO THEIR CURRENT VALUES.
      CALL TSPUT(LED)
C
C
C=============================================================
C=============================================================
C
C
C  NOW FOR THE HARD PART.
C  LET'S READ OUR CLASS NUMBER TO SEE IF THERE IS ANYTHING THERE FOR US.
C
      CALL EXEC(21,CLASSC,IBUF,5)
      CALL ABREG(A,B)
C
C
C  DEST= INTENDED DESTINATION OF THE CLASS BUFFER.
C     0 = UNKNOWN TRANSFER.  TRACKER IGNORES THESE.  NDATA TREATES THEM
C         AS A REQUEST TO RETRANSMIT THE DATA.
C     1 = FROM NDATA TO NSIM
C     2 = FROM NSIM  TO NDATA
C  ICODE= READ OR WRITE TO EXEC OR TRACKER
C     1 = READ
C     2 = WRITE
C  CPU= DATA TRANSFER IS INTENDED FOR TRACKER OR EXEC
C     1 = TRACKER
C     2 = EXEC
C  ISUBF= SUBFUNCTION OF THE CALL
C  SUBLEN= SUBFUNCTION BUFFER LENGTH
C
C
C  CHECK TO SEE IF THERE IS A BUFFER FOR US.
C
      IF(A.LT.0 .OR. DEST.NE.1) GOTO 10000
C
C
C  THE BUFFER IS OURS, LET'S READ IT AND REMOVE IT FROM THE BUFFER.
C
      CALL EXEC(21,CLASS,IBUF,5)
C
C
C  THERE IS A BUFFER OUT THERE FOR US.  IS IT A READ OR A WRITE?
C
      IF(ICODE.EQ.1) GOTO 1000
      IF(ICODE.EQ.2) GOTO 2000
      GOTO 20000
C
C
C  IT IS A READ FROM US.  IS IT AN EXEC OR TRACKER READ?
C
1000  IF(CPU.EQ.1) GOTO 3000
      IF(CPU.EQ.2) GOTO 4000
      GOTO 20000
C
C
C  IT IS A WRITE TO US.  IS IT AN EXEC OR TRACKER WRITE?
C
2000  IF(CPU.EQ.1) GOTO 5000
C     IF(CPU.EQ.2) GOTO 20000
      GOTO 20000
C
C
C  THIS IS A READ FROM THE TRACKER.  SEND THE HEADER FIRST.
C
3000  CONTINUE
      IF(TSUBF(ISUBF).EQ.0) GOTO 20000
      DEST=2
C     ICODE=ICODE
C     CPU=CPU
C     ISUBF=ISUBF
C     SUBLEN=SUBLEN
      CALL EXEC(20,0,IBUF,5,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.-2) GOTO 20100
C
C
C  SEND THE DATA
C
      IF(TRKUP.EQ.0) GOTO 3100
      CALL EXEC(20,0,TCOM(TSUBF(ISUBF)),SUBLEN,TRKUP,I,CLASS)
      CALL ABREG(A,B)
      GOTO 3200
3100  CALL EXEC(20,0,0,1,TRKUP,0,CLASS)
      CALL ABREG(A,B)
3200  IF(A.EQ.-2) GOTO 20100
      GOTO 10000
C
C
C  THIS IS A READ FROM THE EXEC.  SEND THE HEADER FIRST.
C
4000  IF(ISUBF.NE.8) GOTO 20000
      DEST=2
C     ICODE=ICODE
C     CPU=CPU
C     ISUBF=ISUBF
C     SUBLEN=SUBLEN
      CALL EXEC(20,0,IBUF,5,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.-2) GOTO 20100
C
C
C  SEND THE DATA
C
      IF(EXUP.EQ.0) GOTO 4100
      CALL EXEC(20,0,ECOM,SUBLEN,EXUP,I,CLASS)
      CALL ABREG(A,B)
      GOTO 4200
4100  CALL EXEC(20,0,0,1,EXUP,0,CLASS)
      CALL ABREG(A,B)
4200  IF(A.EQ.-2) GOTO 20100
      GOTO 10000
C
C
C  THIS IS A WRITE TO THE TRACKER.  READ THE DATA BUFFER.
C
5000  IF(TSUBF(ISUBF).EQ.0) GOTO 20000
      IF(TRKUP.EQ.0) GOTO 5100
      CALL EXEC(21,CLASSW,TCOM(TSUBF(ISUBF)),SUBLEN)
      GOTO 5200
5100  CALL EXEC(21,CLASSW,I,1)
      GOTO 20
C
C
C  LET'S PRETEND THE TRACKER DOES WHAT EVER NDATA WANTS IT TO.
C
5200  IRSON=IRSX
      NODON=IXNOD
      IF(IMOVE.NE.0) LR=MOD(LR+1,2)
      IMOVE=0
      GOTO 20
C
C
C  PAUSE THE PROGRAM FOR 20 MILLISECONDS
C  THEN GO BACK TO THE START OF THE LOOP
C
10000 CALL EXEC(12,0,1,0,-2)
      GOTO 20
C
C
C  WE HAD AN ERROR.  BAD HEADER ERRORS WILL COME IN 2'S,MOST OF THE TIME.
C
20000 WRITE(LU,20010) MYNAME,IBUF
20010 FORMAT("/",2A2,A1,": BAD DATA IN CLASS HEADER:",6I5)
C
C
C CLEAR OUT THIS 'BAD' BUFFER WITH OUR NAME ON IT.
C
      CALL EXEC(21,CLASS,I,1)
C
C
C  SEND A 'HUH' BUFFER TO NDATA.  TELL IT WE GOT JUNK.
```

```
C
20020 I=0
      CALL EXEC(20,0,I,1,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.-2) GOTO 20100
      GOTO 20
C
C
C  WE ARE HAVING A SAM SHORTAGE.  TELL THE WORLD.
C
20100 WRITE(LU,20110) MYNAME
20110 FORMAT("/",2A2,A1,": THERE IS A SAM SHORTAGE OVER HERE!")
      GOTO 20020
      END
      BLOCK DATA
      IMPLICIT INTEGER(A-Z)

      REAL UT,RANG
C
C
C  TRACKER COMMON
C
      COMMON /TRACK/ TCOM(200)
C
C
C  EXEC COMPUTER COMMON
C
      COMMON /EXECC/ ECOM(500)
C
C
C
C     THE DEFINITION OF TRACKER COMMON CAN BE FOUND IN THE FILE CALLED
C     "T*TRSC"
C
C     THE EXEC COMMON IS DEFINED IN THE "ADAMS HANDBOOK", APPENDIX A.
C
C
C
C  SOME USEFUL EQUIVALENCES
C
      EQUIVALENCE(TCOM(54),IPAUS)
      EQUIVALENCE(TCOM(55),IRSON)
      EQUIVALENCE(TCOM(56),NODON)
      EQUIVALENCE(TCOM(57),LR)
      EQUIVALENCE(TCOM(58),LOSS1)
      EQUIVALENCE(TCOM(78),IRSX)
      EQUIVALENCE(TCOM(79),IMOVE)
      EQUIVALENCE(TCOM(80),IXNOD)
C
      EQUIVALENCE(ECOM(8),IYEAR)
      EQUIVALENCE(ECOM(151),UT)
      EQUIVALENCE(ECOM(165),RANG)
C
```

```
      DATA ECOM(171) /2HNO/
      DATA ECOM(172) /2H O/
      DATA ECOM(173) /2HBJ/
      DATA ECOM(174) /2HEC/
      DATA ECOM(175) /2HT(/
      DATA ECOM(176) /2HSI/
      DATA ECOM(177) /2HMU/
      DATA ECOM(178) /2HLA/
      DATA ECOM(179) /2HTO/
      DATA ECOM(180) /2HR)/

      DATA IPAUS/1/

      DATA RANG/-.20/

      END
```

10.2 Source Listing for the Simulator Control Box Driver

```
ASMB,R
      NAM TSBOX,7 G. BOOZER  TRACKER SIMULATOR BOX DRIVERS (830912.0908)
*
*   &NBLIB    SIMULATOR BOX DRIVER LIBRARY
*
      ENT TSPUT,TSGET
      EXT .ENTR,$LIBR,$LIBX
*
*
*  TSPUT -- TRACKER SIMULATOR PUT.
*
*           OUTPUT A WORD TO THE MICROCIRCUIT INTERFACE CARD.
*
*           CALLING SEQUENCE:
*
*                   CALL TSPUT(I)
*                                  WHERE I IS A WORD WITH BITS
*                                  CORRESPONDING TO OUTPUT BITS.
*
*                                  BIT  |  FUNCTION
*                                 ----- | ----------
*                                   0   |  L LED
*                                   1   |  R LED
*                                   2   |  NOD MODE LED
*                                   3   |  STEADY LED
*                                   4   |  LOT LED
*                                   5   |  EXEC COMMUNICATIONS LED
*                                   6   |  TRACKER COMMUNICATIONS LED
*                                   7   |  N/U   NOT USED NOW BUT IS WIRED
*                                   8   |  N/U
*                                   9   |  N/U
*                                  10   |  N/U
*                                  11   |  N/U
*                                  12   |  N/U
*                                  13   |  N/U
*                                  14   |  N/C   NOT CONNECTED IN CABLE
*                                  15   |  N/C
*
*
*
*
*
      SKP
*
*  TSGET -- TRACKER SIMULATOR GET.
*
*           INPUT A WORD FROM THE MICROCIRCUIT INTERFACE CARD.
*
*           CALLING SEQUENCE:
*
*                   CALL TSGET(J)
*                                  WHERE J IS A WORD WITH BITS
*                                  CORRESPONDING TO INPUT BITS.
*
*                                  BIT  |  FUNCTION
*                                 ----- | ----------
*                                   0   |  LEFT BEAM SWITCH
*                                   1   |  RIGHT BEAM SWITCH
*                                   2   |  NOD MODE SWITCHCH
*                                   3   |  TELESCOPE STEADY SWITCH
*                                   4   |  LOT SWITCH
*                                   5   |  EXEC COMMUNICATIONS SWITCH
*                                   6   |  TRACKER COMMUNICATION SWITCH
*                                   7   |  N/U   NOT USED NOW BUT IS WIRED
*                                   8   |  N/U
*                                   9   |  N/U
*                                  10   |  N/U
*                                  11   |  N/U
*                                  12   |  N/U
*                                  13   |  GROUND
*                                  14   |  N/C   NOT CONNECTED IN CABLE
*                                  15   |  N/C
*
*                 NOTE:   BIT 13 IS AT GROUND POTENTIAL <<ONLY>> WHEN
*                         THE CABLE IS PLUGGED INTO THE CONTROL BOX.
*                         WHEN EITHER END OF THE CABLE IS DISCONNECTED,
*                         BIT 15 IS FLOATING.
*
      SKP
*
*========================================================================
*
*     MICROCIRCUIT INTERFACE STRAP SETTINGS
*
*========================================================================
*
*   STRAP  |  CONNECTION
*  -------  | ------------
*     W1    |  D/C        DON'T CARE  (PREFER FACTORY DEFAULT)
*     W2    |  D/C
*     W3    |  D/C
*     W4    |  B
*     W5    |  N/C        NO CONNECTION
*     W6    |  N/C
*     W7    |  N/C
*     W8    |  N/C
*     W9    |  A
*
*
*
*
      SKP
*
I     NOP
TSPUT NOP
      JSB .ENTR
      DEF I
      JSB $LIBR     TURN OFF INTERRUPT SYSTEM
      NOP             AND MEMORY PROTECT
      LDA I,I       GET WORD TO BE OUTPUT
      OTA BOX       OUTPUT TO MICROCIRCUIT CARD IN SELECT CODE BOX
      JSB $LIBX     TURN ON INTERRUPT SYSTEM AND MEMORY PROTECT
      DEF TSPUT     RETURN TO CALLING ROUTINE.
*
*
*
      SKP
*
J     NOP
TSGET NOP
      JSB .ENTR
      DEF J
      JSB $LIBR     TURN OFF INTERRUPT SYSTEM
      NOP             AND MEMORY PROTECT
      LIA BOX       INPUT FROM MICROCIRCUIT CARD IN SELECT CODE BOX
      STA J,I       PUT RESULT INTO CALLING PROGRAM BUFFER
      JSB $LIBX     TURN ON INTERRUPT SYSTEM AND MEMORY PROTECT
      DEF TSGET     RETURN TO CALLING ROUTINE.
*
*
BOX   EQU 12B       SELECT CODE FOR MICROCIRCUIT CARD
*
*
      END
```

10.3 Source Listings for the Subroutines TBLT, BLT, and SEXEC

The original lines of code are commented out with the 'CB' character string.

```
FTN4X
```

```
      SUBROUTINE TBLT(IRW,ISUBF,IBUF,ILEN,IERR)
     *,GAB &NSLIB Block transfer to Tracker(831021.1217)
      INTEGER SEXEC
CB    DIMENSION IA(2)
CB    EQUIVALENCE (AB,IA),(IB,IA(2))
      DATA LUTRK/20/
      IERR = 7
      IF (IRW.NE.1 .AND. IRW.NE.2) RETURN
CB    CALL EXEC(IRW+100000B,LUTRK+ISUBF*64,IBUF,ILEN)
CB    GO TO 900
CB11  CALL EXEC (13,LUTRK,IST1,IST2,IST3)
CB    IEQT5 = IAND (IST1,377B)
CB    IF (IEQT5 .EQ. 0 ) IERR = 1
CB    IF (IEQT5 .EQ. 8 ) IERR = 5
CB    IF (IEQT5 .EQ.16 ) IERR = 2
CB    IF (IEQT5 .EQ.32 ) IERR = 6
CB    IF (IEQT5 .EQ.64 ) IERR = 4
CB    IF (IEQT5 .EQ.128) IERR = 3
CB    RETURN
CBC  Exec error
CB900 CALL ABREG(IA,IB)
CB    WRITE(1,903) AB,IRW,LUTRK,ISUBF,ILEN
CB903 FORMAT('EXEC error in TBLT: 'A4
CB   *'EXEC('I3'+100000B,'I2'+'I2'*64,IBUF,'I4')')
C
C
C
      IERROR=SEXEC(IRW,LUTRK+ISUBF*64,IBUF,ILEN)
C
C ASSUME NO ERRORS
      IERR=1
C
C IF ERROR, ASSUME TRACKER COMMUNICATIONS FAILURE
      IF(IERROR.EQ.0) IERR=4
      RETURN
      END
C
      SUBROUTINE BLT(IRW,ISUBF,IBUF,ILEN,IERR)
     *,GAB &NSLIB Block transfer to EXEC   (831021.1217)
      INTEGER SEXEC
C
      DATA LUEXEC/10/
C
      IERR = 7
      IF (IRW.NE.1 .AND. IRW.NE.2) RETURN
CB    CALL EXEC(IRW+100000B,LUEXEC+ISUBF*64,IBUF,ILEN)
CB    GO TO 900
CB11  CALL EXEC (13,LUEXEC,IST1,IST2,IST3)
CB    IEQT5 = IAND (IST1,377B)
CB    IERR = 1
CB    IF ( IAND(IEQT5, 10B) .EQ. 10B  ) IERR = 5
CB    IF ( IAND(IEQT5, 20B) .EQ. 20B  ) IERR = 2
CB    IF ( IAND(IEQT5, 40B) .EQ. 40B  ) IERR = 6
CB    IF ( IAND(IEQT5,100B) .EQ. 100B ) IERR = 4
CB    IF ( IAND(IEQT5,200B) .EQ. 200B ) IERR = 3
CB    RETURN
CBC  Exec error
CB900 CALL ABREG(IA,IB)
CB    WRITE(1,903) IA,IB,IRW,LUEXEC,ISUBF,ILEN
CB903 FORMAT('BLT --> '2A2' Error on '
CB   +'EXEC('I3'+100000B,'I2'+'I2'*64,IBUF,'I4')')
C
C
C
      IERROR=SEXEC(IRW,LUEXEC+ISUBF*64,IBUF,ILEN)
C
C ASSUME NO ERRORS
      IERR=1
C
C IF ERROR, ASSUME EXEC COMMUNICATIONS FAILURE
      IF(IERROR.EQ.0) IERR=4
      RETURN
C
C
      INTEGER FUNCTION SEXEC(ICOD,ICNWD,IBUFF,ILEN)
     *, G. BOOZER  TRACKER & EXEC SIMUL. COMM. (831021.1217)
      IMPLICIT INTEGER(A-Z)
C
C
C SEXEC            COMMUNICATE WITH THE EXEC/TRACKER SIMULATOR PROGRAM
C                  CALLED "NSIM" USING CLASS I/O.
C
C
C
C THIS FUNCTION SIMULATES THE EXEC CALLS MADE BY THE DATA
C COMPUTER TO THE EXEC COMPUTER AND THE TRACKER COMPUTER.
C ALL CALLS ARE ROUTED TO "NSIM" VIA CLASS I/O
C
C
C
      COMMON ICOM(750)
      EQUIVALENCE(ICOM(10),LU)
C
C
      INTEGER IBUF(5),IBUFF(1)
C
C
      EQUIVALENCE(IBUF(1),DEST)
      EQUIVALENCE(IBUF(2),ICODE)
      EQUIVALENCE(IBUF(3),CPU)
      EQUIVALENCE(IBUF(4),ISUBF)
      EQUIVALENCE(IBUF(5),SUBLEN)
C
C LU 10 IS THE EXEC COMPUTER
C
C LU 20 IS THE TRACKER COMPUTER
```

```
C
C
      LOGICAL FIRST
      DATA FIRST/.TRUE./
C
C IS THIS THE FIRST TIME THROUGH?
1     IF(.NOT. FIRST) GOTO 100
      FIRST=.FALSE.
C
C IF FIRST TIME THROUGH, GET A CLASS NUMBER AND RUN "NSIM"
      CLASS=100000B
      CALL EXEC(17,0,I,1,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.NE.-1) GOTO 20
      WRITE(LU,10)
10    FORMAT("/SEXEC: OUT OF CLASS NUMBERS")
      STOP
20    CLASS=IOR(CLASS,120000B)
      CLASSC=IOR(CLASS,160000B)
      CLASSW=IAND(CLASS,077777B)
      CALL EXEC(21,CLASS,I,1)
C
C RUN "NSIM"
      CALL EXEC(10,6HNSIM  ,LU,177,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.0) GOTO 100
C
C ERROR IN SCHEDULING "NSIM", DEALLOCATE THE CLASS   AND QUIT.
      WRITE(LU,30)  A
30    FORMAT("/SEXEC: ERROR  ",I4," ON SCHEDULING 'NSIM'")
      CALL EXEC(21,IAND(CLASS,017777B),I,1)
      STOP
C
C ASSUME SEXEC CALL IS SUCCESSFUL.
100   SEXEC=1
C
C
C
C LET'S START THE SHOW
C GET THE LU
      EXLU=IAND(ICNWD,77B)
C
C GET THE SUBFUNCTION NUMBER
      ISUBF=IAND(ICNWD,177700B)/100B
C
C IS IT A CALL TO THE EXEC?
      CPU=0
      IF(EXLU.EQ.10) CPU=2
C
C IS IT A CALL TO THE TRACKER?
      IF(EXLU.EQ.20) CPU=1
C
C SET DESTINATION CODE
```

```
      DEST=1
C
C SET ICODE
      ICODE=ICOD
C
C SET SUBLEN
      SUBLEN=ILEN
C
C SEND THE HEADER
      CALL EXEC(20,0,IBUF,5,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.-2) GOTO 20000
C
C IF WE ARE SENDING DATA TO "NSIM" WE HAD BETTER GET ON WITH IT.
C WE CAN RETURN AFTER WE ARE DONE HERE.
      IF(ICODE.EQ.1) GOTO 115
      CALL EXEC(20,0,IBUFF,ILEN,I,I,CLASS)
      CALL ABREG(A,B)
      IF(A.EQ.-2) GOTO 20000
      RETURN
C
C LOOP UNTIL WE GET OUR HEADER
110   CALL WAIT(20)
115   CALL EXEC(21,CLASSC,IBUF,5)
      CALL ABREG(A,B)
      IF(A.LT.0 .OR. (DEST.NE.0 .AND. DEST.NE.2)) GOTO 110
      CALL EXEC(21,CLASS,IBUF,5)
C
C WE GOT OUR HEADER.  LET'S READ THE DATA.
      CALL EXEC(21,CLASSW,IBUFF,ILEN,SEXEC)
      RETURN
C
C NO ROOM IN SAM, TELL THE WORLD ABOUT THE SHORTAGE.
20000 WRITE(LU,20010)
20010 FORMAT("/SEXEC: NOT ENOUGH SAM TO FILL MY NEEDS")
      GOTO 1
      END
```

11.0 REFERENCES

1. C-141 Gerald P. Kuiper Airborne Observatory Investigator's Handbook, 1983. NASA/Ames Research Center, Moffett Field, California. (Contact L. C. Haughney, MS 211-12).

2. C-141 Gerald P. Kuiper Airborne Observatory Airborne Data Acquisition and Management System (ADAMS) User's Handbook, 1983. NASA/Ames Research Center, Moffett Field, California. (Contact L. C. Haughney, MS 211-12).

3. Getting Acquainted With RTE IVB, 1980. Hewlett-Packard Manual, Part No. 92068-90001, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

4. A Pocket Guide to the 2100 Computer, Chapter 3, 1978. Basic Control System Reference Manual, Hewlett-Packard Manual, Part No. 5951-4423, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

5. RTE IVB Assembler Reference Manual, 1980. Hewlett-Packard Manual, Part No. 92067-90003, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

6. RTE IVB Terminal User's Reference Manual, 1983. Hewlett-Packard Manual, Part No. 92068-90002, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

7. RTE IVB Programmer's Reference Manual, 1983. Hewlett-Packard Manual, Part No. 92068-90004, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

8. RTE IVB System Manager's Manual, 1983. Hewlett-Packard Manual, Part No. 92068-90006, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

9. Operating and Service Manual for the 12566B, 12566B-001, 12566B-002, and 12566B-003 Microcircuit Interface Kits, 1976. Hewlett-Packard Manual, Part No. 12566-90015, Hewlett Packard Company, Data Systems Division, 11000 Wolfe Road, Cupertino, CA 95014.

10. John L. Hughes, Digital Computer Lab Workbook, 1969. Digital Equipment Corporation, 146 Main Street, Maynard, MA 01754.

TABLE 1

TRACKER SIMULATOR WIRING LIST

TO HP						FROM HP					
TB1						TB3					
	48-PIN		20-41-SPIN				48-PIN		20-41-SPIN		
BIT #	HP CONN		MS 3126			BIT #	HP CONN		MS 3126		
0	1	BRN	1	WH	A	0	A	BRN/WH	1	WH	S
1	2	RD	2	BLK	B	1	B	RD/WH	2	RD	T
2	3	OR	3	BRN	C	2	C	OR/WH	3	BRN	U
3	4	YEL	4	BLK	D	3	D	YEL/WH	4	RD	V
4	5	GRN	5	RD	E	4	E	GRN/WH	5	OR	W
5	6	BL	6	BLK	F	5	F	BL/WH	6	RD	X
6	7	PUR	7	OR	G	6	H	PUR/WH	7	YEL	Y
7	8	SLV	8	BLK	H	7	J	SLV/WH	8	RD	Z
8	9	BLK	9	YEL	J	8	K	BLK/WH	9	GRN	a
	24	WH'S (COM)	10					WH'S (COM)	10		
TB2						TB4					
	48-PIN		20-41-SPIN				48-PIN		20-41-SPIN		
BIT #	HP CONN		MS 3126			BIT #	HP CONN		MS 3126		
9	10	BLK/SLV	1	BLK	K	9	L	RD/OR	1	RD	b
10	11	BLK/BRN	2	GRN	L	10	M	RD/YEL	2	BL	c
11	12	BLK/RD	3	BLK	M	11	N	BRN/RD	3	RD	d
12	13	BLK/OR	4	BL	N	12	P	BRN/OR	4	WH	e
13	14	BLK/YEL	5	BLK	P	13	R	BRN/YEL	5	GRN	f
	15	BLK/GRN	6	NC			S	BRN/GRN	6	NC	
	16	BLK/BL	7	NC			T	BRN/BL	7	NC	
	23	BLK/PUR	8	NC			22	BRN/PUR	8	NC	
	AA	RD/GRN	9	NC			Z	BRN/SLV	9	NC	
	24	WH'S (COM)	10	GRN	R		BB	WH'S (COM)	10	BL	g

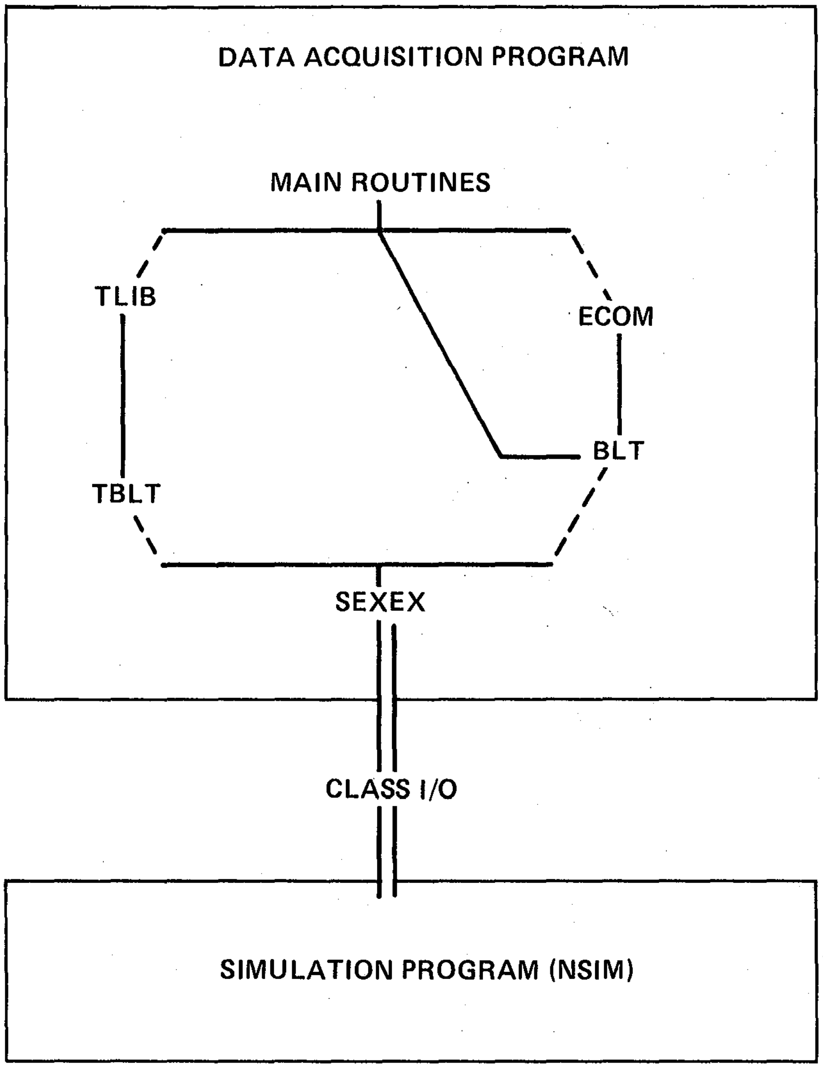


Figure 1.- Data acquisition program.

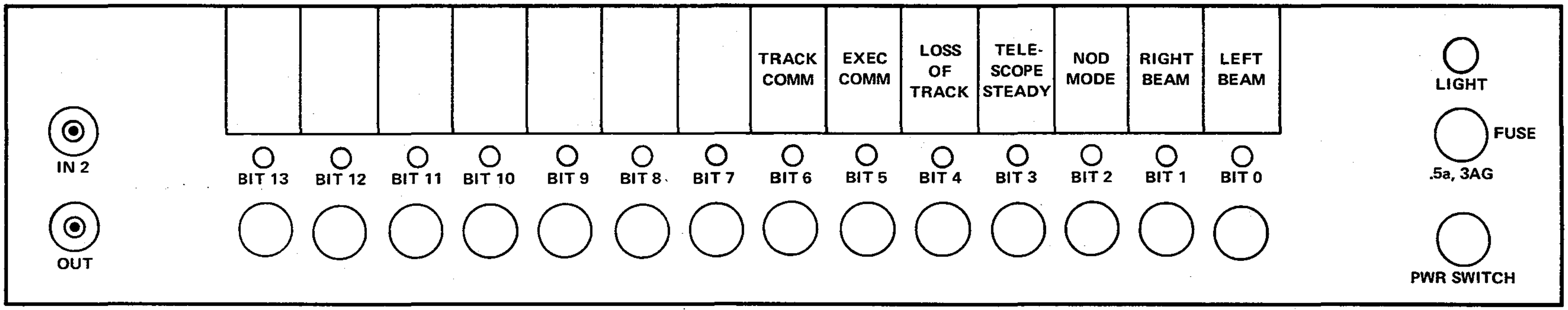
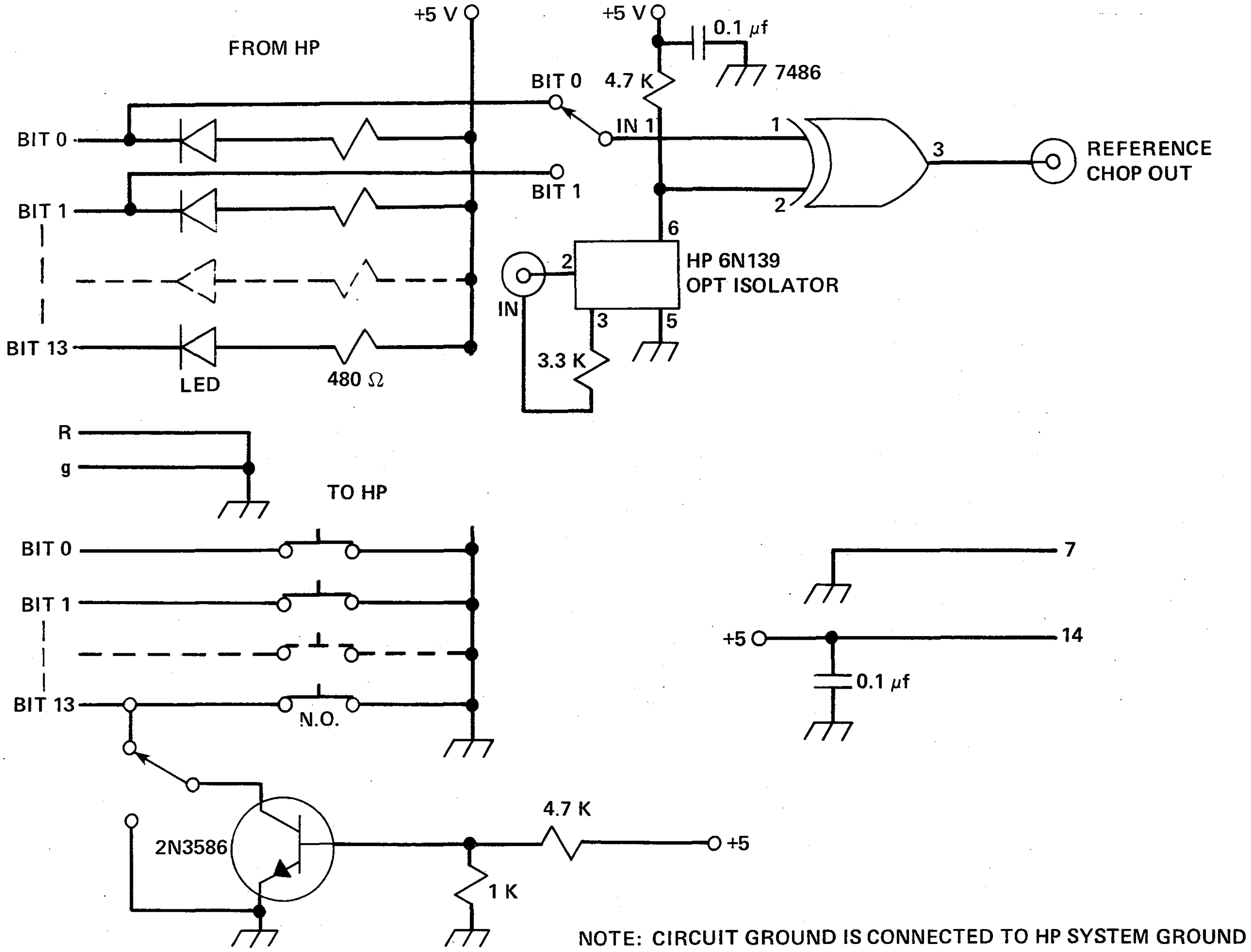


Figure 2.- KAO tracker simulator front panel.



NOTE: CIRCUIT GROUND IS CONNECTED TO HP SYSTEM GROUND

Figure 3.- KAO tracker simulator circuit diagram.

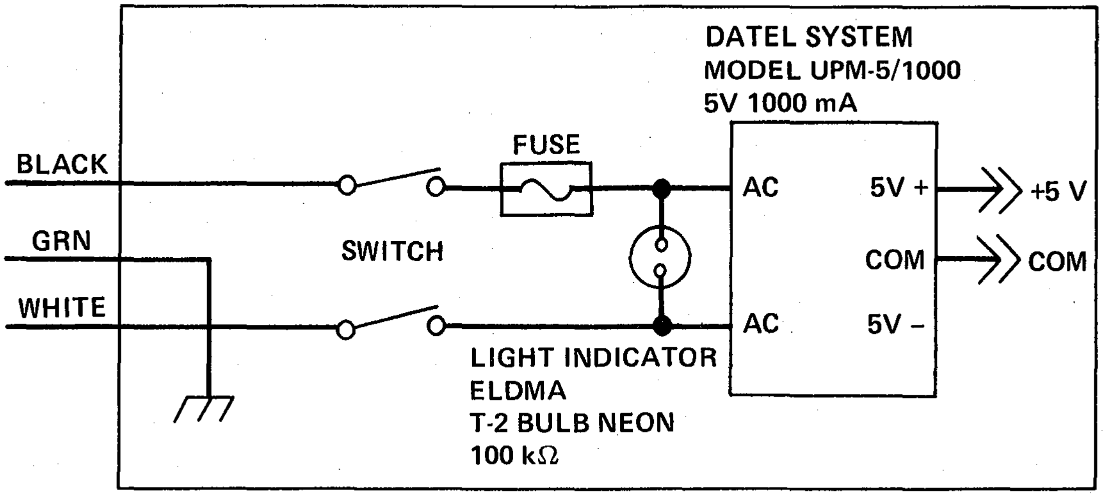


Figure 4.- KAO tracker simulator power supply.

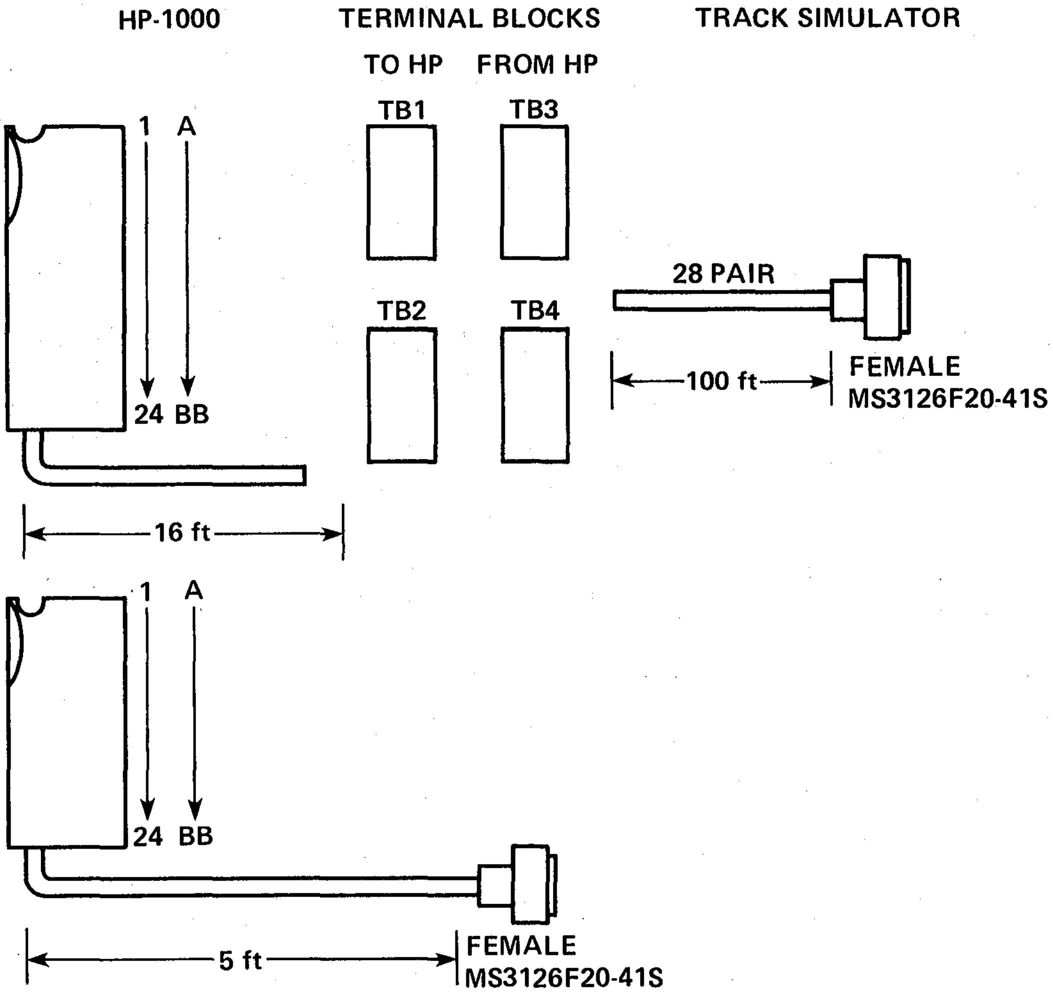


Figure 5.- KAO tracker simulator cables.

1. Report No. NASA TM 85896	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle A Hardware/Software Simulation for the Video Tracking System of the Kuiper Airborne Observatory Telescope	5. Report Date February 1984
	6. Performing Organization Code ATP
7. Author(s) Glenn A. Boozer*; Darrell D. McKibbin** Michael R. Haas*** and Edwin F. Erickson**	8. Performing Organization Report No. A-9662
9. Performing Organization Name and Address *Boozerco, San Jose, CA **Ames Research Center, Moffett Field, CA ***Mycol, Incorporated, Sunnyvale, CA	10. Work Unit No. T-4644
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington DC, 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code 352-02-03

15. Supplementary Notes

Point of contact: Darrell D. McKibbin, Ames Research Center, MS 245-6,
Moffett Field, CA (415) 965-5231 or FTS 448-5231

16. Abstract

This simulator was created so that C-141 Kuiper Airborne Observatory (KAO, ref. 1) investigators could test their Airborne Data Acquisition and Management System (ADAMS; ref. 2) software on a system which is generally more accessible than the ADAMS on the plane. An investigator can currently test most of his data acquisition program using the data computer simulator (ref. 2) in the Cave. (The "Cave" refers to the ground-based computer facilities for the KAO and the associated support personnel.) The main Cave computer is interfaced to the data computer simulator in order to simulate the data-Exec computer communications (ref. 2). However until now, there has been no way to test the data computer interface to the tracker. The simulator described here simulates both the KAO Exec and tracker computers with software which runs on the same Hewlett-Packard (HP) computer as the investigator's data acquisition program. A simulator control box is hardwired to the computer to provide monitoring of "tracker" functions, to provide an operator panel similar to the real tracker, and to simulate the 180° phase shifting of the chopper square-wave reference with beam switching. If run in the Cave, one can use their Exec simulator and this tracker simulator.

17. Key Words (Suggested by Author(s)) Kuiper airborne observatory tracking simulator Hardware/software Procedures	18. Distribution Statement Unlimited Subject Category: 89

19. Security Classif. (of this report) Uncl.	20. Security Classif. (of this page) Uncl.	21. No. of Pages 36	22. Price* A02

*For sale by the National Technical Information Service, Springfield, Virginia 22161