DATAS Hardware
Diagnostic Tests

John Zvanya

October 1990

DOT/FAA/CT-TN90/44

Technical Report Documentation Page

1. Report No. **DOT/FAA/CT-TN90/44**	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle DATAS HARDWARE DIAGNOSTIC TESTS **TECHNICAL REFERENCE**		**5. Report Date** **Oct-90** **6. Performing Organization Code** **ACD-320**
7. Author(s) **John Zvanya**		**8. Performing Organization Report No.** **DOT/FAA/CT-TN90/44**
9. Performing Organization Name and Address **U. S. Department of Transportation** **Federal Aviation Administration** **Technical Center** Atlantic City International Airport, N.J. 08405		**10. Work Unit No. (TRAIS)** **11. Contract or Grant No.** **T2001F** **13. Type of Report and Period Covered**
12. Sponsoring Agency Name and Address **U.S. Department of Transportation** **Federal Aviation Administration** **Technical Center** **Atlantic City International Airport, N.J. 08405**		**Technical Note** **14. Sponsoring Agency Code**
15. Supplementary Notes		

16. Abstract

This document is reference material for personnel using the Data Link and Analysis System (DATAS) for Hardware Diagnostic Testing.

This is the second of a series of documents to be published on DATAS.

17. Key Words **DATAS** **Diagnostic Tests/DATAS**		**18. Distribution Statement** **This document is available to the public through the National Technical Information Service, Springfield, VA 22161**	
19. Security Classif.(of this report) **Unclassified**	**20. Security Classif.(of this page)** **Unclassified**	**21. No. of Pages** **46**	**22. Price**

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

EXECUTIVE SUMMARY

This document provides reference material for personnel using the Data Link Test and Analysis System (DATAS) for hardware diagnostic testing.

Included in this document is a brief overall description of the DATAS, and a thorough description of how to operate the hardware diagnostic tests.

This is one of a series of technical notes published on the DATAS.

INTRODUCTION

BACKGROUND.

Since the beginning of the federal operation of the air traffic control (ATC) system in 1936, the amount of air traffic has increased dramatically. In 1981 the Federal Aviation Administration (FAA) assessed their current status and it was predicted that by the year 2000 air traffic would double. Faced with this situation, the FAA conceived the National Airspace System (NAS) Plan. The NAS Plan included replacing the entire existing air traffic system with new automated and computer enhanced systems. These systems included an Advanced Automation System (AAS) that will provide computer assistance to air traffic controllers, increased reliability via a digital Data Link between ground sensors and aircraft which will be provided by a Secondary Surveillance Radar (SSR) known as the Mode Select Beacon System (Mode S), which will replace the current Air Traffic Control Radar Beacon System (ATCRBS). Delivery of the Mode S sensors will be in the l990's.

The FAA Technical Center has a major role in the implementation of the NAS Plan. Many of the systems are being designed and/or tested at the Center. From the design and integration of such complex systems spawns the growth of small test systems such as the Data Link Test and Analysis System (DATAS).

The DATAS was originally conceived at the Technical Center by members of the Data Link project whose responsibilities include verification of Data Link systems reliability, interface protocols and system capacities. The DATAS was designed and fabricated at the FAA Technical Center to provide such test and analysis capabilities.

DATAS is capable of testing all components of the Data Link system. These components include: ATCRBS and Mode S transponders, avionics Data Link processors (ADLP3, and all Data Link system interfaces. It will also provide the capability of Mode S sensor simulation and 1030 and 1090 megahertz (MHz) radio frequency (RF) environment analysis for all beacon transmissions including Traffic Alert and Collision Avoidance Systems (TCAS). The DATAS has the capability of RF signal analysis within the frequency range of 950 to 1200 MHz.

DESCRIPTION OF EQUIPMENT

HARDWARE.

Figure 1 shows a very general block diagram of a DATAS System. Physically, the system consists of three racks: the RF unit, the

DATAS hardware section (digital components), and the computer rack (each of which is approximately 19 inches x 12 inches x 19 inches), a dish antenna (which is approximately 50 inches in diameter), and, at minimum, one video display terminal.

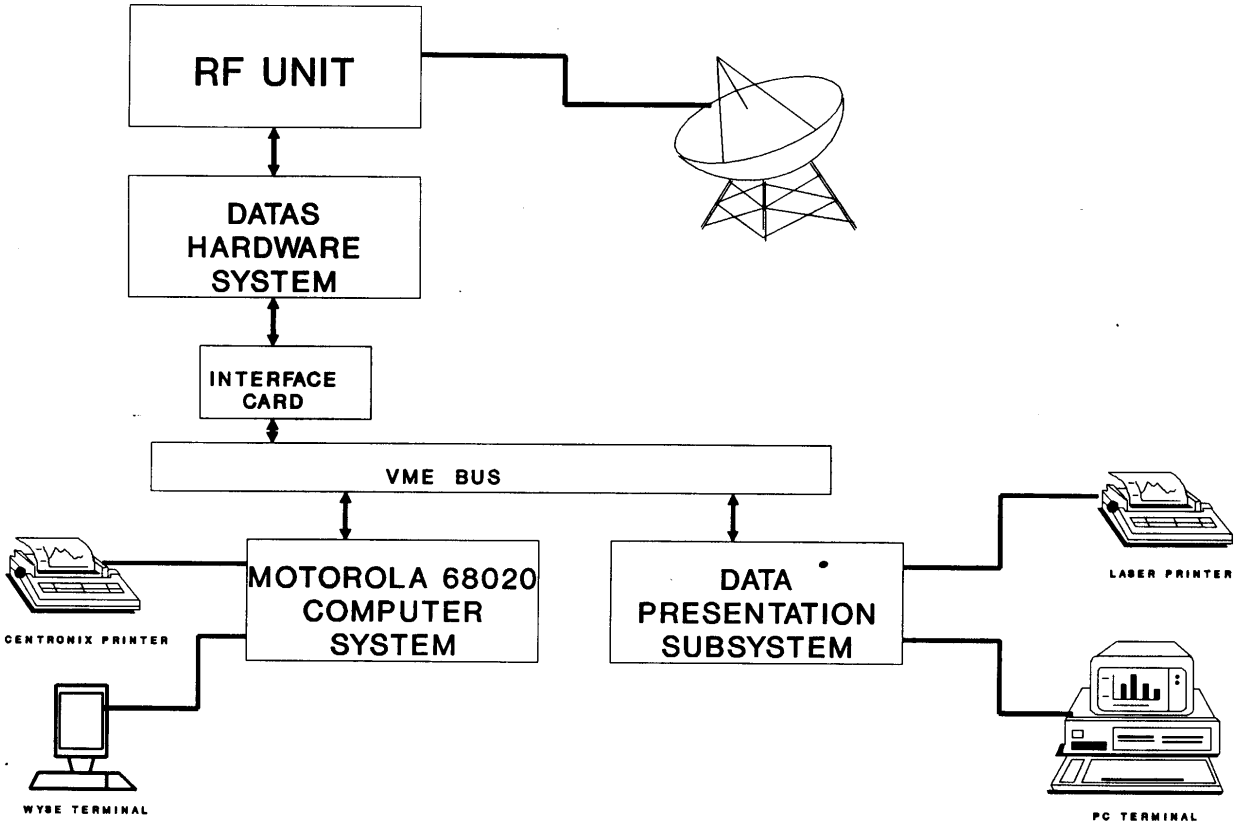


FIGURE 1. DATAS

The Motorola 68020 computer uses a Motorola MVME 135A central processing unit (CPU). This processor card consists of a 68020 32-bit microprocessor operating at 20 MHz. There is 4 megabytes of on-board random access memory (RAM). The card also contains a 68881 floating point coprocessor. The memory management unit has been disabled because there is too much overhead involved for some of the memory input/output (I/O) functions that the system will be required to perform with the Mode S testing functions.

The computer contains an MVME50 card for timing functions. The card provides up to 1 microsecond resolution.

The computer system uses a WYSE 75 (or compatible) terminal and any centronix interface printer. The lab system for software development uses a Fujitsu M304X series printer.

The DATAS may also contain an optional data presentation subsystem for data storage, data reduction, and report generation. The presentation subsystem is actually an independent personal computer (PC) system that plugs into the Versa Module Extended (VME) Bus rack and interfaces to the VMEbus. An Xycom XVMEÄ682 VMEbus PC Advanced Technology (AT) processor module is used for this purpose. This subsystem requires its own graphics display terminal. Optional peripherals include a floppy disk drive and a Hewlett Packard (HP) Laser Jet II printer or equivalent.

SOFTWARE.

The Motorola computer systems were used for the software development of the DATAS. The operating system of the DATAS is Motorola VERSADOS, version 4.6. VERSADOS operates on the various CPU boards and systems offered by Motorola and is a real-time operating system that offers the facilities to operate in a real-time domain.

The DATAS programs were written in C language, except for a few of the lower level interrupt handling functions which were written in assembly language. The compiler used was the Alcyon C68 C compiler.

SYSTEM USER'S GUIDE

SYSTEM START-UP PROCEDURE.

When the user powers on the system, the system performs the boot procedure and activates VERSAdos which, upon initialization, will prompt the user for several inputs. If the user does not get a prompt on the screen, one or two troubleshooting techniques can be tried. The first troubleshooting technique is that the

WYSEÄ75 terminal is in the BLK mode; which means the terminal is not on line. The second one might be that the power of the 68020 computer system is off. If all is performing properly, the user sees 135 bug> on the terminal. The user enters **BO 0 8 <CR>** (<CR>=carriage return key). If the user does not see enter default system volume: user = after a few seconds, then the user performs a sequence of options. Make sure that all of the boards, hardware system and 68020 system, are seated properly in their slots. Make sure the power is off when performing this task. Then power on the system. If the system still acts in the same manner as before, turn off the power. One of the hardware boards has an external interrupt that has to be addressed. Pull out one board at a time on the hardware system, except the first board, and continue with the boot-up procedure.

Replace the board that failed with a new board and continue. When the user sees on the terminal enter **default system volume: user** =, the user responses with the volume and user number which contains the Hardware Diagnostic Routines. The correct response is **sys:302**. If a carriage return is entered, the default response is selected (SYS:0). Do not worry, the correct user number can be obtained by entering use **302 <CR>** after the date and time prompt are entered. The system will then prompt you for the current date with **enter date (mm/dd/yy).** (Note: mm = month, dd = day, yy = year.)

The system must be dated since the current date is used as part of the file names created by the diagnostic routines. The system will then prompt for the current time with enter time (hr:min)=. The time of day is a critical entry for the diagnostic routines. If the user wants to run the diagnostic routines in the morning and then in the afternoon and powers off the system between the two runs, then the default time will be the same as the morning and afternoon runs.

OVERVIEW OF DIAGNOSTIC ROUTINES.

After the user has established the correct start-up procedure and the system is running, the user can now run the diagnostic routines. DATAS hardware has to be able to pass two diagnostic tests in order for the system to be used. The routines are called memotest, which is the memory test diagnostics, and hwtest which is the hardware functional test diagnostics. After the diagnostics are activated, a menu is displayed on the terminal. By using the up and down arrows, the user will be able to scroll to any selection. The user can scroll to the selection that tests the complete system or each board separately. Pressing <CR> starts the selected diagnostics. When the diagnostic routine is testing a particular board or group of boards, the terminal displays test in progress after the selection, and then

displays a pass or fail condition when the test is completed. The diagnostic routine called memotest first checks for memory errors throughout the system. The diagnostic routine hwtest checks the operational aspect of the system. It sends a known ATCRBS reply through the RF unit and loop back the same signal to the system for analysis of pulse width, spacing, amplitude, frequency, code, and delay. When the system tests are completed, a file is generated with the results. If further examination of the test results are needed, the user can list the test results on the terminal by entering list mtst.dmmddyy for the memory data, and data.dmmddyy for the system data.

DETAILED DESCRIPTION OF THE MEMORY TEST DIAGNOSTIC ROUTINES. The main function is called memotest with memosub, arrypatt, and arryrdwt as subfunctions. The file memotest.cf shows how the functions are linked. Memolibr.h and hwlibr.h are library files that support all of the functions in the memory tests diagnostic package.

Each board has a set of memory addresses with read and write accesses permitted. In order for the memory to be validated, a memory check will be performed on a per board basis. The board must be installed in the DATAS unit under test for proper operation. The array card_exist[l2] contains a value of 1 if the card is to be tested and O if not. For all boards not in the system, the corresponding element in the card_exist array should be 0.

[0] - nothing
[1] - Transmitter #l
[2] - Transmitter #2
[3] - Transmitter #3
[4] - DPSK #1
[5] - DPSK #2
[6] - DPSK #3
[7] - Mode S
[8] - Decoder #l
[9] - Raw Data Card
[10] - nothing
[11] - nothing

The resultant data from the test performed is stored in a file called 302.mtst.dmmddyy.AA. The file is appended for every run for a particular day.

The function called arrypatt.cc generates a fixed data pattern that is stored in global memory and is used for all of the memory tests. The array contains 256 data points that are 32 bits wide. There are three parts to this function. The first part will shift two l's to the left with a fast and slow bit, example:

```
 11 = 3          1001 = 9
101 = 5          1010 = A
110 = 6          etc.
```

The data contained in the array will be as follows:

```
[O] =  Ox00030003  hex
[1] =  Ox00050005  hex
     etc.
```

The upper 16 bits contains the same value as the lower 16 bits. This generates 120 data elements. The second part shifts two O's and the rest all ones to the left with a fast and slow bit, example:

```
1111 1111 1111 1100 = FFFC HEX
1111 1111 1111 1010 = FFFA HEX
1111 1111 1111 1001 = FFF9 HEX
1111 1111 1111 0110 = FFF6 HEX
                  etc.
```

The data contained in the array is as follows:

```
[120] =  OxFFFCFFFC HEX
[121] =  OxFFFAFFFA HEX
          etc.
```

The last part will exercise alternating 5's and A's for the last 16 addresses.

The function lists on the terminal all of the tests performed. If the first selection of the tests is selected (page 2 of the Diagnostics), then the RF unit memory will be checked for validity. The remaining tests on page 1 checks the hardware system. The memory addresses for each board are stored in a file hwlibr.h (table 1). Each board is analyzed on an individual basis or the complete system.

Page two of the menu contains diagnostics that checks out the RF unit and the control addresses. The file arryrdwt.cc contains the control function for execution of the diagnostics. This function will read and write a number of addresses and check to see if the data is correct. The following variables are utilized:

```
DATA_ARRAY     =    Starting address of the fixed data
                    that are written to memory.
HW_ADDR   =         Starting memory address.
ADDR_TOTAL     =    The amount of address locations
                    exercised.
MASK_OUT_BITS  =    The bits that are ignored when the
                    data are read and checked.
```

The function checks out 16 addresses at a time. The variable chk_off_bound computes the number of addresses being used and determines if the number is a multiple of 16. If not, it passes

TABLE 1. DATAS MEMORY AND CONTROL ADDRESSES

1.	INTERR_CONTR	OXEF800000	INTERRUPT CONTROL WORD
2.	TRANS_CONTRL	OXEF800004	TRANSMITTER CONTROL WORD
3.	EXT_OUT_CONT	OXEF800008	EXTERNAL OUTPUT JACKS
4.	DECONl	OXEF800010	DECODER TAP CONTROL
5.	DECON2	OXEF800014	DECODER TAP CONTROL & TOLERANCE
6.	DECON3	OXEF800018	DECODER TAP 1-4 SET-UP
7.	DECON4	OXEF80001C	DECODER TAP 5-8 SET-UP
8.	DECON5	OXEF800020	DECODER STORAGE CONTROL
9.	RF_CONTROL	0XEF800100	BIT 0 = SELECT DIAG/VSWR MODE BIT 1 = SELECT REFLECTED POWER BIT 2 = DISABLE INTERFERENCE INPUT BIT 3 = SPLIT BENCH MODE ENABLE BIT 4 = SELECT EXTERNAL INPUT FOR RECEIVER BITS 5 Ä 23 = SPARES BITS 24 Ä 31 = KEY (TO PROGRAM THE RF_CONTROL SWITCH, 3 FUNCTIONS HAVE TO BE PERFORMED: EF800100 = "00" RESET EF800100 = "07" KEY # 1 EF800100 = "06" KEY # 2
10.	MODE_S_ID	OXEF81C000	EXPECTED MODE S ID.
11.	RAW_CONTROL	OXEF80000C	STATUS CONTROL OF THE RAW DATA
12.	RAW_DATA	OXEF810000	RAW DATA STORAGE
13.	DECODED_DATA	OXEF824000	DECODED DATA
14.	SNAP_DATA	OXEF828000	SNAP SHOT DATA
15.	FREQ_l_CHANN	0XEF800200	CHANNEL 1 FREQU.OUTPUT
16.	FREQ_2_CHANN	0XEF800300	CHANNEL 2 FREQU.OUTPUT
17.	FREQ_3_CHANN	0XEF800400	CHANNEL 3 FREQU.OUTPUT
18.	LOC_OSC_FREQ	OXEF800500	LOCAL OSC. FREQUENCY
19.	RF_1_CHANNEL	0XEF800600	CHANNEL 1 RF ATTENUATOR
20.	RF_2_CHANNEL	0XEF800700	CHANNEL 2 RF ATTENUATOR
21.	RF_3_CHANNEL	0XEF800800	CHANNEL 3 RF ATTENUATOR
22.	TRAN_1_CONTRL	0XEF802000	CHANNEL 1 MEMORY ADDRESS
23.	TRAN_2_CONTRL	OXEF804000	CHANNEL 2 MEMORY ADDRESS

24.	TRAN_3_CONTRL	0XEF806000	CHANNEL	3	MEMORY ADDRESS
25.	DPSK_1_CONTRL	0XEF830000	CHANNEL	1	DPSK ADDRESS
26.	DPSK_2_CONTRL	0XEF831000	CHANNEL	2	DPSK ADDRESS
27.	DPSK_3_CONTRL	0XEF832000	CHANNEL	3	DPSK ADDRESS

the remainder. This value is used to analyze the remaining memory locations.

Each 16 hardware addresses will be exercised using the fixed data array. The first 16 addresses are loaded with the contents of the first 16 elements of the fixed data array. The contents of memory are checked. The next 15 iterations, the memory is reloaded with the contents of the fixed data, offset by the current iterations number (figure 2). After completion of this cycle the same memory locations are tested using the next 16 fixed data locations (16-31). This will continue until all of the 256 data points in the fixed data array have been used. The function will increment the hardware memory address to 16 - 31 and start the process again from the fixed data array address 0 - 15. By rotating the data and verifying 16 addresses, the data and address paths for the memory locations will be checked for validity.

Hardware Addresses		**Fixed Data Addresses**
First Pass	(0-15)	(0-15)
Second Pass	(0-15)	(1-15,0)
Third Pass	(0Ä15)	(2-15,0,1)
Fifteenth Pass	(0-15)	(15,0-14)
Sixteenth Pass	(0-15)	(16-31)
.....	(0-15)	(240-255)

FIGURE 2. MEMORY TEST DATA AND ADDRESSES

RUNNING OF THE MEMORY DIAGNOSTIC TESTS. The user enters memotest <CR>. This loads the memory diagnostic routine. The terminal displays page 1 of the list of tests (figure 3).

	RUN	STATUS / CONTROL	MEMORY TEST
	RUN	TRANSMIT CONTROL 1	MEMORY TEST
	RUN	TRANSMIT CONTROL 2	MEMORY TEST
	RUN	TRANSMIT CONTROL 3	MEMORY TEST
	RUN	DPSK 1 CONTROL	MEMORY TEST
	RUN	DPSK 2 CONTROL	MEMORY TEST
	RUN	DPSK 3 CONTROL	MEMORY TEST

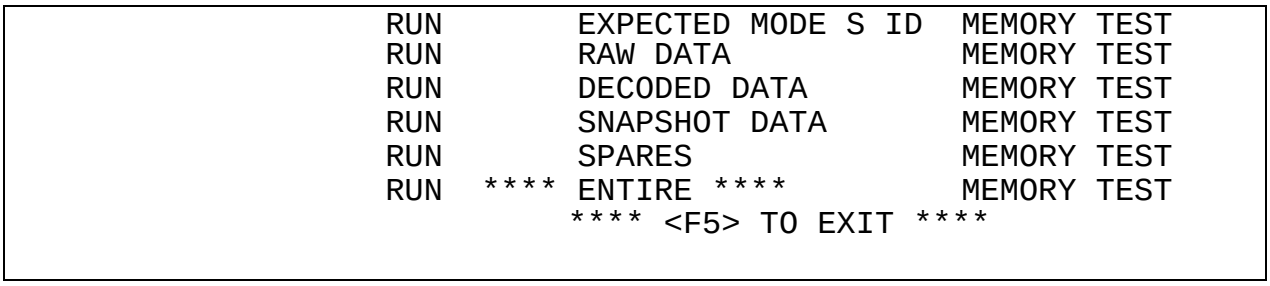


FIGURE 3. MEMORY DIAGNOSTIC MENU PAGE 1

By using the up and down arrow keys the user scrolls to any selection on the screen. The selection executed is in bold print. By pressing <CR> this executes the selection. If page 2 is desired, Run Status/Control Memory Test (selection is in bold print). Pressing <CR>, the terminal displays a new list of tests (figure 4).

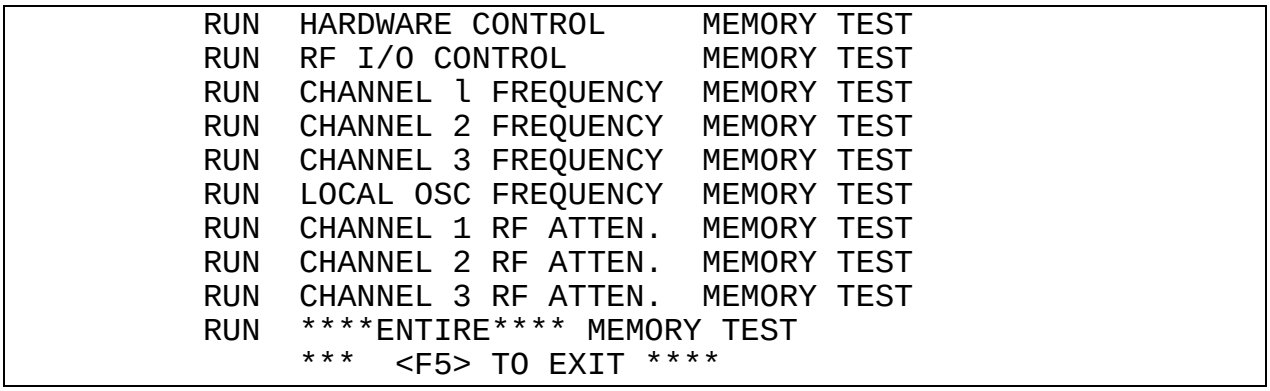


FIGURE 4. MEMORY DIAGNOSTIC MENU PAGE 2

When the user wants to either return to page l or exit the routine, <F5> key will perform this function. The user can only exit from the memory diagnostic routine from page l. When a particular selection is selected, the screen area to the right of the selection displays Test in Progress (figure 5).

This indicates the particular operation is in progress. When the routine has completed its memory check and no errors have been detected, the pass condition is displayed to the right of the selection (figure 5). If errors are encountered, then the failed condition is displayed with the amount of errors (figure 5). The user will use the entire selection on page l and 2 when running a comprehensive memory tests. When the tests are completed, pressing <F5> key will exit the routine.

TROUBLESHOOTING TECHNIQUES. The operation of the memory diagnostic results in two possible outcomes. Either the memory's

address/data is incorrect or the diagnostic program aborts due to a memory access error. Let us examine the least extreme condition first. By examining the data stored during a particular test, the user is able to determine the memory location error. There are a lot of combinations that exist when errors are encountered. For discussion purposes, an example error is provided.

List mtst.dmmddyy.aa <CR>, displays on the terminal the memory test results. Enter Ctrl S to stop the scrolling; pressing any other key starts the scrolling again. If no errors exist, the data file will be as shown in figure 6.

```
  RUN STATUS / CONTROL
  RUN TRANSMIT CONTROL 1 MEMORY TEST    TEST IN PROGRESS
  RUN TRANSMIT CONTROL 2
  RUN TRANSMIT CONTROL 3
  RUN DPSK 1 CONTROL
  RUN DPSK 2 CONTROL
  RUN DPSK 3 CONTROL
  RUN EXPECTED MODE S ID
  RUN RAW DATA
  RUN DECODED DATA
  RUN SNAPSHOT DATA
  RUN SPARES
  RUN **** ENTIRE ****
  **** <F5> TO EXIT ****
```

a. After Selection

```
  RUN STATUS / CONTROL
  RUN TRANSMIT CONTROL 1  MEMORY TEST        PASSED
  RUN TRANSMIT CONTROL 2
  RUN TRANSMIT CONTROL 3
  RUN DPSK 1 CONTROL
  RUN DPSK 2 CONTROL
  RUN DPSK 3 CONTROL
  RUN EXPECTED MODE S ID
  RUN RAW DATA
  RUN DECODED DATA
  RUN SNAPSHOT DATA
  RUN SPARES
  RUN **** ENTIRE ****
  **** <F5> TO EXIT ****
```

b. After Successful Completion

```
RUN STATUS / CONTROL
RUN TRANSMIT CONTROL 1 MEMORY TEST FAILED XX # OF LOCATIONS
```

```
RUN TRANSMIT CONTROL 2
RUN TRANSMIT CONTROL 3
RUN DPSK  1  CONTROL
RUN DPSK  2  CONTROL
RUN DPSK  3  CONTROL
EXPECTED MODE S ID
RUN RAW DATA
RUN DECODED DATA
RUN SNAPSHOT DATA
RUN SPARES
RUN **** ENTIRE ****
****<F5> TO EXIT ****
```

c. After Failures

FIGURE 5. MEMORY DIAGNOSTIC MENU DISPLAY

```
              TRAN_1_CONTRL       NO ERRORS WERE DETECTED
              TRAN_2_CONTRL       NO ERRORS WERE DETECTED
              TRAN_3_CONTRL       NO ERRORS WERE DETECTED
              DPSK_l_CONTRL       NO ERRORS WERE DETECTED
              DPSK_2_CONTRL       NO ERRORS WERE DETECTED
              DPSK_3_CONTRL       NO ERRORS WERE DETECTED
              MODE _S_ID          NO ERRORS WERE DETECTED
              RAW_DATA            NO ERRORS WERE DETECTED
              DECODED_DATA        NO ERRORS WERE DETECTED
              SNAP_DATA           NO ERRORS WERE DETECTED
              INTERR_CONTRL       NO ERRORS WERE DETECTED
              TRANS_CONTRL        NO ERRORS WERE DETECTED
              EXT_OUT_CONT        NO ERRORS WERE DETECTED
              RAW_CONTROL         NO ERRORS WERE DETECTED
                    DECONl        NO ERRORS WERE DETECTED
                    DECON2        NO ERRORS WERE DETECTED
                    DECON3        NO ERRORS WERE DETECTED
                    DECON4        NO ERRORS WERE DETECTED
                    DECON5        NO ERRORS WERE DETECTED
              RF_CONTRL           NO ERRORS WERE DETECTED
              FREQ_1_CHANN        NO ERRORS WERE DETECTED
              FREQ_2_CHANN        NO ERRORS WERE DETECTED
              FREQ_3_CHANN        NO ERRORS WERE DETECTED
              LOC_OSC_FREQ        NO ERRORS WERE DETECTED
              RF_l_CHANNEL        NO ERRORS WERE DETECTED
              RF_2_CHANNEL        NO ERRORS WERE DETECTED
              RF 3_CHANNEL        NO ERRORS WERE DETECTED
```

a. With No Errors

TIME	ERROR #	ADDRESS	FIXED DATA	HARDWARE DATA

```
08:00:00  1         EF802000  0X00030003     0X00030002
08:00:00  2         EF802000  0X00050005     0X00030004
08:00:00  3         EF802000  0X00090009     OX00090008
08:00:00  4         EF802000  OX00011011     OX00110010
```

b. With Errors

FIGURE 6. MEMORY TEST DATA FILE

Now let us examine a hardware board failure uncovered from the memory diagnostics. The user examines the Transmitter Number 1 card, which has a starting address of Hexadecimal EF802000. When errors are encountered, the data file displayed is as shown in figure 6. The pattern generated is Hexadecimal, 03, 05, 06, 09, OA, 0C, 11, etc. As the user can see, by analyzing the data, the least significant bit, bit 0, is tied low (always a zero). Because the least significant bit (LSB) was always 0, it passed on the 06 and OA data points. The user has now found a potential wiring problem with Transmitter Board Number 1.

The user uses a routine called CHK to examine and debug the memory address location in error. CHK can exercise one memory location or a number of locations, with any data combination the user desires (figure 7).

```
ENTER  ADDRESS <->dec , <=> inc
ENTER  DATA/  (*WRITE/READ READ*)
READS  THE DATA FROM LAST ADDRESS ENTERED (ANY KEY TO EXIT)
WRITES THE DATA FROM LAST ADDRESS ENTERED (ANY KEY TO EXIT)
READS  THE DATA FROM LAST ADDRESS ENTERED(10,000 TIMES)
WRITES THE DATA FROM LAST ADDRESS ENTERED(10,000 TIMES)
READS/WRITES DATA FROM 0 THROUGH 600,000 (ONE ADDRESS)
READS/WRITES FROM 0 - 600,000 FOR A NUMBER OF LOCATIONS
SHIFT 1 BIT TO THE LEFT (all zeros & 1 bit set) (100 TIMES)
SHIFT 1 BIT TO THE LEFT (all ones  & 1 zero)    (100 TIMES)
ENTER NEW BITS TO MASK OUT IN HEXADECIMAL
****<F5>  TO  EXIT ****
```

FIGURE 7. CHK ROUTINE'S MENU

The CHK routine menu is similar to the memory test menu. The up and down arrow keys are used to scroll to a particular question and the <CR> key is used to execute the function. The <F5> key exits the routine. To enter the address in error, use the arrow keys to select Enter Address and press <CR>. The terminal displays enter address in the lower left hand corner. (Make sure the CAP lock key is on). Enter the address OEF802000 where the error has occurred (Hexadecimal characters) and press <CR>. After eight characters are entered, there is an implied <CR>. If a mistake is made, a backspace or Delete key corrects one character at a time. Enter DATA and press <CR>. Enter the DATA is displayed. Enter the desired data (FFFFFFFF) and the routine writes the data to the address EF802000. The program reads back the address Data three times and displays the data on the terminal.

In the example the data read back would be FFFFFFFE, verifying the problem to be the least significant bit. Then the user selects the program that will write the address an infinite

number of times with the same data and verifies using an oscilloscope, the least significant bit is the problem. After the faulty component is replaced, the card is retested using the same diagnostic procedure. In the case where the memory tests aborts the system, a more serious error condition exist on the card that may involve handshaking signals.

HARDWARE DIAGNOSTICS.

The hardware diagnostic routine consists of many functions. The diagnostic hardware load file is called hwtest.lo. The command file which links the object files is called hwtest.cf.

During the actual testing of the hardware system, many waveforms are loaded into Transmitter #1 or Transmitter #2 for the loop back testing to occur. The file called cardparm.h contains four predetermined waveforms. These waveforms are generated through Transmitters #1 and #2 and looped back to a particular area to fully examine the hardware logic. Figure 8 gives the path of the loop back testing.

The normal path is through Transmitters #1 and #2 which are set up to transmit a pulse train through the RF unit. An external unit detects this signal and transmits a reply. The reply is received through the RF unit. The reply is detected and sent to the analog board, the raw data board, Decoder #1, and finally to the Decoder #2 board. In the diagnostic loop back mode, each board is examined independently or as a partial group of boards.

The hardware diagnostic package is used to check-out the DATAS hardware for all types of interrogations, interrupt control, and different pulse repetition frequencies using the system clock or the hardware clock. When programming the system to transmit a pulse train, the programmer must adhere to the DATAS transmitter setup structure (lead edge, trail edge, pam, reply window, etc.).

DATAS TRANSMITTER SETUP PARAMETERS.

The transmitter is configured by selection of six parameters that are controlled as follows:

1. Function Field

   a. DELAY 0
   b. LEAD_EDGE 1
   c. TRAIL_EDGE 2
   d. JUMP 3
   e. SPARES 4-7

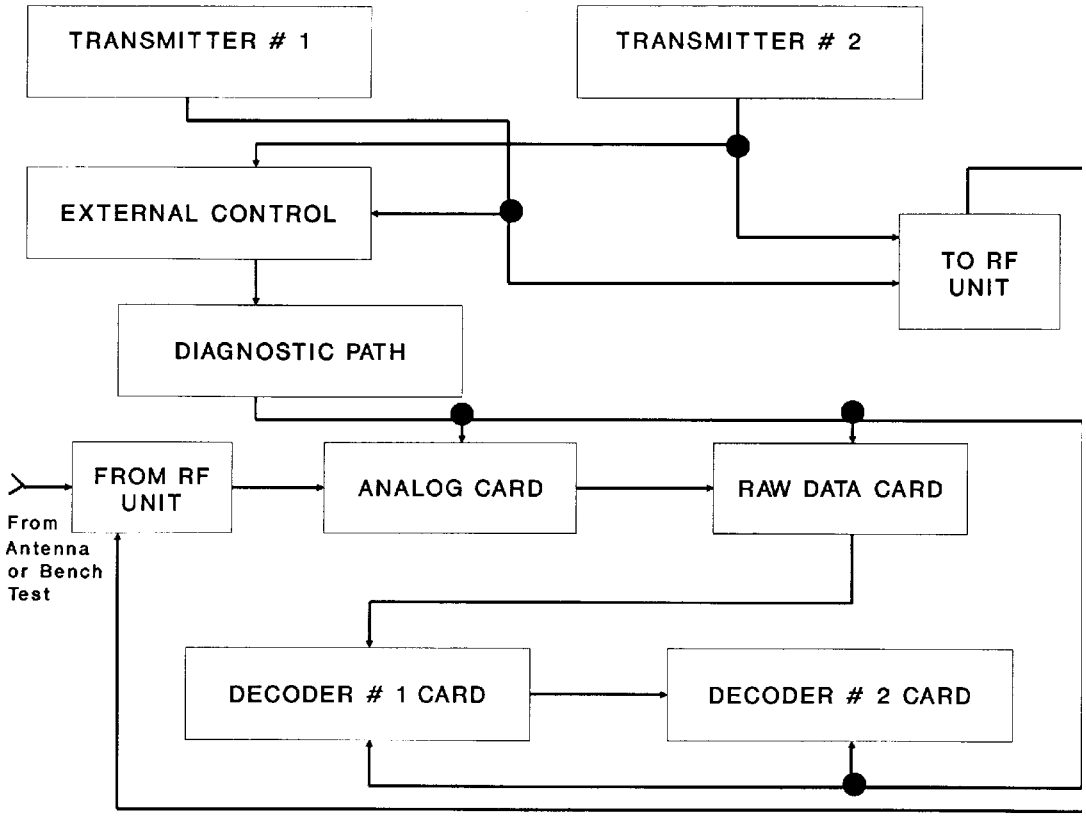


FIGURE 8. DATAS HARDWARE BLOCK DIAGRAM

2. Time Field

   a. 16 bits (2 16th power = 65536)
   b. Maximum delay (~25.00 ns time 65536 = 1.6384 ms)

3. Attenuator Select

   a. Eight unique attenuator selections

4. Channel Select

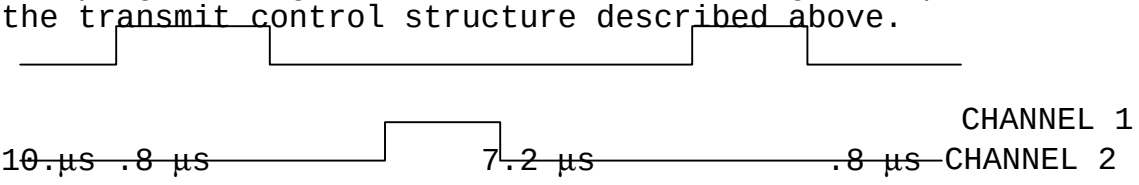
5. Action Field

   a. NOTHING 0
   b. PAM
   c. REPLY_WIND 2
   d. MODE_S_BLK 3
   e. END 4
   f. SAMPLE_WIND 5
   g. SPARES 6-7

6. Transmit Field

   a. NO_ACTION 0
   b. REFERENCE 1
   c. INTERRUPT 2
   d. INC_MODE_S_ADD 3
   e. INTER_STATUS 4
   f. RESTART_MODE_S 5

The programmer generates a Mode A interrogation pulse train using the transmit control structure described above.



12.0 µs .8 µs
12.1

Function	Time	Attenuator	Channel	Action	Transmit
LE	10.0	1	1	PAM	No Action
TE	0.8	1	1	PAM	No Action
LE	7.2	1	1	PAM	No Action
LE	0.8	1	1	PAM	No Action
DELAY	0.1	1		END	No Action
LE	12.0	1	2	PAM	No Action
TE	0.8	1	2	PAM	No Action
DELAY	0.1	1	2	END	No Action

After an interrogation is formed, the functions below take control to convert the structure into the DATAS bit stream.

GEN_ACTION - This function performs the task of setting up the control parameters; PAM, REPLY WINDOW, END, etc.

GEN_PULSE - This function performs the task of setting up the pulse parameters; DELAY, LEAD EDGE, TRAIL EDGE.

CHK_CHANNEL - This function performs the task of counting the number of pulses that are generated on each channel. A channel value of 0 is used to signify the end of an interrogation structure.

GEN_TIME - This function performs the task of setting up the spacing between waveform functions.

GEN_ATTN - This function performs the task of setting up the attenuator controls.

GEN_TRANS - This function performs the task of setting up the TRANSMIT FIELD, INTERRUPTS, REFERENCE, etc.

INITIAL CONDITIONS. Before any memory loading is achieved, all transmitting channels must be turned off. The procedure is as follows:

```
INTERR_CONTR   OXEF800000 = 0x0L; INTERRUPT CONTROL WORD
                            Turn off all interrupts
TRANS_CONTRL   OXEF800004 = 0x000000; TRANSMITTER CONTROL
TRANS_CONTRL   OXEF800004 = 0x202020; STOP ALL 3 CHANNELS
```

For the initial load, two control words are written to the transmitter control address. The reason being is the value of the latches are not determined on power up. The hardware needs a high to low or low to high transition to give a lead edge. If the bit is in a high state and the user writes a 1 to the latches, the lead edqe never occurs, causing the channels to be in a free state.

START PROCEDURE. To start any channel the procedure will be as follows:

```
TRANS_CONTRL        OXEF800004 = 0x000000; TRANSMITTER CONTROL
TRANS_CONTRL        OXEF800004 = 0x010101; START ALL 3 CHANNELS
     bits 0 - 7     channel 1
                                bits  8 - 15     channel 2
                                bits 16 - 23     channel 3
```

STOP PROCEDURE. To stop any channel the procedure will be as follows:

```
TRANS_CONTRL   OXEF800004 = 0x202020; STOP ALL 3 CHANNELS
                              bits 0 - 7     channel 1
                              bits  8 - 15     channel 2
                              bits 16 - 23     channel 3
```

INTERRUPT PROCEDURE. To have interrupts enabled, two initial conditions are required. All eight software interrupt vectors must be set and the interrupt control word programmed correctly. The software interrupt handler program is called 300.exec and handles eight interrupt vectors from the DATAS hardware. The address at this time is OX2FFB00 and is byte addressable. Setting address 2FFB00 - 2FFB08 to a value of OX01 corresponds to no interrupts occurred. Setting the interrupt control word to a 1 enables all interrupts.

```
     OX2FFB00 - OX2FFB08 = OX01;
                OXEF800000 = OXO1L; INTERRUPT CONTROLWORD (enable
                                                          interrupts)
```

CHECK FOR INTERRUPTS. To check for interrupts that occurred, follow this procedure:

```
     CHANNEL  #  1 INTERRUPT FLAG ADDRESS = 2FFB00;
     CHANNEL  #  2 INTERRUPT FLAG ADDRESS = 2FFB01;
     SET INTERRUPT  # 1 FLAG TO A ONE 2FFB00 = OX01;
```

Check for interrupt. If the flag is equal to zero, an interrupt vector from DATAS occurred.

```
     WHILE  (INTERRUPT  #  1 FLAG ! = 0)
     RESET INTERRUPT  #  1 FLAG TO A ONE  2FFB00 = OX01;
     AND WAIT FOR ANOTHER INTERRUPT.

     EXAMPLE PROCEDURE IN "C":

FOR

     CHAR     *INTERRUPT_VECTOR_l =  OX01;
     CHAR     *INTERRUPT_VECTOR_2 =  OX01;

     LONG      *TRANS_CONTRL = OXOOL;  /*have hardware in correct
     start state */
```

```
     LONG      *TRANS_CONTRL = OX2001L;  /*start channel #l and
     make sure channel 2 is off
               (INTERRUPT_VECTOR_1 != 0)     /*CHECK TO SEE I F
               INTERRUPT OCCURRED
                                 */
```

DIAGNOSTIC LOOP-BACK PATH SETUPS.

ANALOG BOARD CONTROL. The user selects the analog card from the menu. The raw data card is set up to give the time from lead edge to lead edge of every pulse; the diagnostic input path is through the analog card (figure 10). All values preceded with an 0X are Hexadecimal values; all values with OXEF8XXXXX are hardware addresses in the DATAS system.

```
      OXEF80000C  =  0X44000;

               BITS 14 - 16   TIMING REF. FOR PULSE POSITION
                          0 = LE OF REPLY WINDOW
                          1 = LE OF LAST PULSE
                          2 = REF. TIMING CHANNEL
                          4 - 7 = SPARES

               BIT 17    RAW DATA BOARD (DIAGNOSTIC MODE)
               BIT 18    ANALOG BOARD (DIAGNOSTIC MODE)
               BIT 19    DISABLE DYNAMIC THRESHOLD
```

To program the analog card to select the diagnostic mode, bit 18 is set and the external output control word is programmed to determine the type of input the card selects.

```
OXEF80000C  =  0X44000;  SET UP DIAGNOSTIC MODE
```

EXTERNAL OUTPUT JACKS. There are four external output jacks programmed from 16 different possibilities. External output #1 is the signal used for the loop-back diagnostics.

```
EXT_OUT_CONT OXEF800008       EXTERNAL OUTPUT JACKS
             BITS 0 - 3       EXTERNAL OUT #
             BITS 4 - 7       EXTERNAL OUT # 2
             BITS 8 - 11      EXTERNAL OUT # 3
             BITS 12 - 15     EXTERNAL OUT # 4
OXEF800008  =  0X00004;   SET UP FOR PAM  # 1
*ONLY USE BITS 0 - 3 IN THE DIAGNOSTIC MODE.*
```

The 16 different selections are defined as follows:

```
REPLY WINDOW                    = 0
TIMING REFERENCE CHANNEL # 1    = 1
TIMING REFERENCE CHANNEL # 2    = 2
TIMING REFERENCE CHANNEL # 3    = 3
PAM CHANNEL # 1                 = 4
PAM CHANNEL # 2                 = 5
PAM CHANNEL # 3                 = 6
DECODE # 1                      = 7
DECODE # 2                      = 8
DECODE # 3                      = 9
DECODE # 4                      = 10
DECODE # 5                      =

DECODE  #  6                              = 12
DECODE  #  7                              = 13
DECODE  #  8                              = 14
SPARES                                    = 15
```

RAW DATA CONTROL. The user selects the raw data card from the menu.

```
OXEF80000C  =  0X04000;
     BITS 14 - 16   TIMING REF. FOR PULSE POSITION
          0  = LE OF REPLY WINDOW
          1  = LE OF LAST PULSE
          2  = REF. TIMING CHANNEL
          4  - 7  = SPARES
     BIT  17        RAW DATA BOARD (DIAGNOSTIC MODE)
     BIT  18        ANALOG   BOARD (DIAGNOSTIC MODE)
     BIT  19        DISABLE DYNAMIC THRESHOLD
```

To program the raw data card to select the diagnostic mode, bit 17 is set and the external output control word is programmed to determine the type of input the card selects.

```
OXEF80000C   =   0X24000;   SET UP DIAG. MODE WITH REF. TIME
OXEF800008   =  0X00004;  SET UP FOR PAM # 1
ONLY USE BITS 0  -  3 IN THE DIAGNOSTIC MODE.
```

DECODER TAP CONTROLS. The decoder tap control gives total flexibility to the user for determining the type of replies received. There are a total of five decoder taps and each are software programmable.

```
DECON 1   OXEF800010     DECODER TAP CONTROL
DECON 2   OXEF800014     DECODER TAP CONTROL
                         & TOLERANCE
DECON3    OXEF800018     DECODER TAP 1 - 4 SETÄUP
DECON4    OXEF80001C     DECODER TAP 5 - 8 SETÄUP
DECON5    OXEF800020     DECODER STORAGE CONTROL
```

DECONl BITS 0 - 7   TAP DECODER # 1
DECONl BITS 8 - 15   TAP DECODER # 2
DECONl BITS 16 - 23   TAP DECODER # 3
DECONl BITS 24 - 31   TAP DECODER # 4
DECON2 BITS 8 - 15   TAP DECODER # 5

The time increment for the decoder taps is 100 ns, and each decoder tap is added to the previous time field to yield the total decoder tap time. For example:

TAP DECODER # 1 = 2.0µs (IF ENABLED DECLARES BRACKET 2.0µs)
TAP DECODER # 2 = 8.0µs (IF ENABLED DECLARES BRACKET 8.0µs)
TAP DECODER # 3 = 10.0µs (IF ENABLED DECLARES BRACKET 10.0µs)
TAP DECODER # 4 = 12.0µs (IF ENABLED DECLARES BRACKET 12.0µs)
TAP DECODER # 5 = 20.3µs (IF ENABLED DECLARES BRACXET 20.3µs)

FORMULA = 102 HEX - (TIME / 100 (ns)) Ä previous times
2.0µs / 100ns = 20 clocks

102 HEX - (2.0µs / 100ns) - 0 CLOCKS = ED HEX 20 CLOCKS
102 HEX - (8.0µs / 100ns) - 20 CLOCKS = C5 HEX 60 CLOCKS
102 HEX - (10.0µs / 100ns) - 80 CLOCKS = ED HEX 20 CLOCKS
102 HEX - (12.0µs / 100ns) - 100 CLOCKS = ED HEX 20 CLOCKS
102 HEX - (20.3µs / 100ns) - 120 CLOCKS = AE HEX 83 CLOCKS
TOTAL TIME 203 CLOCKS
20.3µs FOR TAP DECODER # 5
OXEF800010 = OXEDEDC5ED
OXEF800014 = OXOOOOAE00

SET-UP DECODE TOLERANCE. The decode tolerance time is a value in 100 ns steps, which is added and subtracted to the decoder taps. This value enables the user to tighten or open the tolerance window on the decoder taps. For example:

TOLERANCE TIME (100 ns) = 101 HEX - RESOLUTION (100 ns)

200 ns TIME TOLERANCE: 101 HEX - (200/100 ns) = FE HEX
OXEF800014 = 0XOOOOAFFF

SET-UP DECODE CONTROL. Once the decoder tap control values are determined, the user selects the decoder tap to enable. The user

selects one of five values from the decoder taps and can enable any combination of decoders (1 to 8). For example:

```
DECON3
BITS 0 - 7 DECODER # 1 CONTROL
          bit 0 - TAP # 1 ENABLED FROM THE TAP DECODER
          bit 1 - TAP # 2 ENABLED FROM THE TAP DECODER
          bit 2 - TAP # 3 ENABLED FROM THE TAP DECODER
          bit 3 - TAP # 4 ENABLED FROM THE TAP DECODER
          bit 4 - TAP # 5 ENABLED FROM THE TAP DECODER
          bits 5 - 7 SPARES
BITS      8 - 15    DECODER # 2 CONTROL
BITS      16 - 23   DECODER # 3 CONTROL
BITS      24 - 31   DECODER # 4 CONTROL
DECON4
BITS      0 - 7     DECODER # 5 CONTROL
BITS      8 - 15    DECODER # 6 CONTROL
BITS      16 - 23   DECODER # 7 CONTROL
BITS      24 - 31 DECODER # 8 CONTROL
```

SELECT DECODER #1 FOR DIAGNOSTIC INPUT.

```
DECON4     BIT 31        ENABLE DIAGNOSTIC VIDEO TO DECODER # 1
```

DECODER STORAGE CONTROL. Certain parameters are stored in memory for analysis when a bracket is declared. Selecting the type of data to be stored in memory are listed below.

```
DECON5     OXEF800020         DECODER STORAGE CONTROL

BITS  0 -  2   TIME REFERENCE FOR RANGE COUNTER
               0 - LE OF REPLY WINDOW
               1 - LAST DECODE TIME
               2 - TIMING REFERENCE ( CHANNEL # 1 )
               3 - 7 SPARES

BIT 3          INHIBIT ATCRBS DECODE DURING MODE S
BIT 4          INHIBIT MODE S. DECODE DURING ATCRBS
BIT 5          SPARE
BIT 6          SPARE
BIT 7          SPARE
BIT 8          STORE DECODE # 1 TIME OF ARRIVAL (TOA)
BIT 9          STORE DECODE # 3 TIME OF ARRIVAL (TOA)
BIT 11         STORE DECODE # 4 TIME OF ARRIVAL (TOA)
BIT 12         STORE DECODE # 5 TIME OF ARRIVAL (TOA)
BIT 13         STORE DECODE # 6 TIME OF ARRIVAL (TOA)
BIT 14         STORE DECODE # 7 TIME OF ARRIVAL (TOA)
BIT 15         STORE DECODE # 8 TIME OF ARRIVAL (TOA)
BIT 16         STORE CODE DATA FOR DECODE # 1
BIT 17         STORE CODE DATA FOR DECODE # 2
BIT 18         STORE CODE DATA FOR DECODE # 3
```

```
BIT 19          STORE CODE DATA FOR DECODE # 4
BIT 20          STORE CODE DATA FOR DECODE # 5
BIT 21          STORE CODE DATA FOR DECODE # 6
BIT 22          STORE CODE DATA FOR DECODE # 7
BIT 23          STORE CODE DATA FOR DECODE # 8
BIT 24          ENABLE DIAGNOSTIC VIDEO TO DECODER # 2
BITS 25 - 27    DIAGNOSTIC DECODE INPUT SELECT
                0 - DECODER # 1
                1 - DECODER # 2
                2 - DECODER # 3
                3 - DECODER # 4
                4 - DECODER # 5
                5 - DECODER # 6
                6 - DECODER # 7
                7 - DECODER # 8
BITS 28 - 31    SPARES
```

RF SET-UP CONTROL PARAMETERS. The RF unit has three frequency sources for transmitting and a local oscillator for the receiver. The hardware addresses are given below. To set the frequency

source to a given value, a decimal value is loaded equal to the frequency transmitted.

FREQ_l_CHANN	OXEF800200	CHANNEL 1 FREQU. OUTPUT
FREQ_2_CHANN	OXEF800300	CHANNEL 2 FREQU. OUTPUT
FREQ_3_CHANN	OXEF800400	CHANNEL 3 FREQU. OUTPUT
LOC_OSC_FREQ	OXEF800500	LOCAL OSC. FREQUENCY

Example:

```
OXEF800200 = 1090000  FREQUENCY #1 = 1090 MHz
OXEF800300 = 1030000  FREQUENCY #2 = 1030 MHz
```

Each transmitting channel has eight selectable attenuator values. Depending on what values are preset in the attenuators, the result of different power level outputs are achieved. See Calibration File.

RF_l_CHANNEL	OXEF800600	CHANNEL 1 RF ATTENUATOR
RF_2_CHANNEL	OXEF800700	CHANNEL 2 RF ATTENUATOR
RF_3_CHANNEL	OXEF800800	CHANNEL 3 RF ATTENUATOR

Examples:

```
OXEF800600 = OXOL       ALL OF THE VALUES FOR
OXEF800604 = OXOL       TRANSMITTER #l ARE SET
OXEF800608 = OXOL       FOR MAXIMUM POWER OUTPUT.
OXEF80060C = OXOL
```

OXEF800610 = OXOL
OXEF800614 = OXOL
OXEF800618 = OXOL
OXEF8006lC = OXOL

DETAILED HARDWARE DIAGNOSTIC ROUTINES.

The hardware diagnostic package consists of many functions. This document provides a detailed explanation of each function that deals with a particular hardware board. The main routine is called hwtest. This routine does most of the parameter initialization.

In order to run the hardware diagnostic package, the user enters hwtest <CR>. The routine reads in a calibration file and initializes a data output file. It then initializes the system for the interrupt handler, RF Frequencies, Transmitter #1 and #2, and provides a menu on the terminal (figure 9).

Figure 8 shows the diagnostic loop-back paths available. Each hardware board is analyzed independently or in groups if selected from the diagnostic path. Out of the four hardware boards; Analog, Raw Data, Decoder #1, and Decoder #2, only two boards have physical memory location allotted. They are the Raw Data and the Decoder #2 boards. When the user checks out the Analog or the Decoder #l boards, the data analyzed is routed to the Raw Data and Decoder #2 boards, respectively.

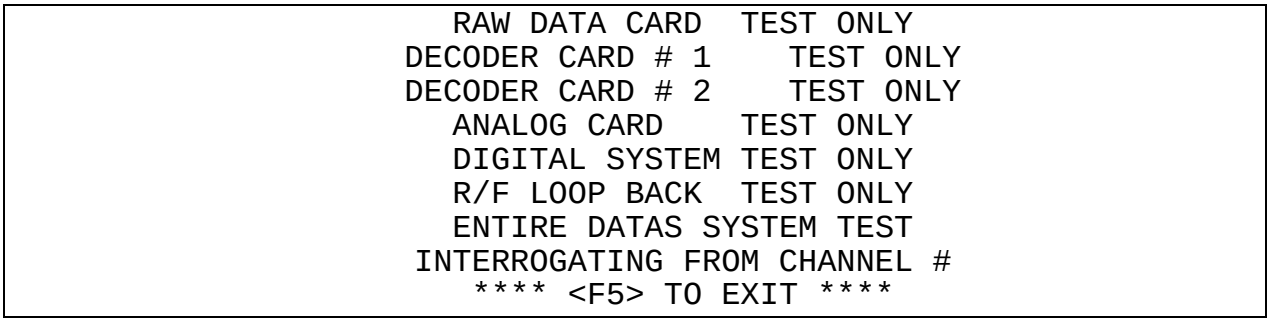


FIGURE 9. MENU FOR THE HWTEST DIAGNOSTIC ROUTINES

The function hwtest.cc controls the display, monitor, keyboard entry, and calls the proper function upon user selection. If the user selects the Raw Data Card Test only, this routine loads Channel #1 with an ATCRBS reply. See table 2 for the waveform Pattern loaded to produce the test ATCRBS reply.

The function called as a result of Raw Data Card Test selection is CARDlMENU contained in the source file CARDlFUN.CC. This routine sets up the Raw Data Card to select the diagnostic mode,

turns off all interrupts, and stops all transmitting channels. After all of the parameter initialization is completed, the test is started. The function RAW_CARD is contained in the source file RAWCARD.CC. This function starts the transmitter selected and runs for 25 interrupts. Correct pulse width, delay, and spacing are checked on each interrupt. The first pulse in the ATCRBS code pulse train is increased 25 ns and checked again for correct spacing. This continues until the first pulse width has reached (25 ns x 234 = 5850 ns).

When Decoder Card #l is selected, the next function that will be executed is CARD2_MENU, (CARD2FUN.CC). This routine sets up Decoder #l to select the diagnostic mode and initialize all of the decoder taps to a predetermined time value. The data collection commences when the routine CARD_l_DECODE, (DEClCARD.CC) is activated.

CARD_l_DECODE moves out the first pulse loaded into Transmitter #l and checks if a bracket is declared in the decoder card. If a bracket is declared, proper pulse spacing and delay are checked. This continues for 24 ~s intervals of 100 ns.

TABLE 2. ATCRBS REPLY

LE,	200,	0,	1,	REPLY_WIND,	NO_ACTION
LE,	200,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION
TE,	450,	1,	1,	PAM,	NO_ACTION
LE,	1000,	1,	1,	PAM,	NO_ACTION

```
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       1000,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       1000,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       1000,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       1000,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       1000,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
LE,       3900,     1,           1, PAM,           NO_ACTION
TE,       450,      1,           1, PAM,           NO_ACTION
TE,       2000,     0,           1, REPLY_WIND,    NO_ACTION
DELAY,    30000,    0,           1, END,      INTERRUPT

PULSE     TIME      ATTENUATOR CHANNEL ACTION     TRANSMIT
                    FIELD      SELECT  TYPE       TYPE
(LE,TE)   (NANO     P1 = 1
(DELAY)   SECONDS   P3 = 3    1,2,3     PAM,REPLY INTERRUPT
                    FIELD 0 - 7         WINDOW

0,        0,        0,           0,      0,            0,
```

When Decoder Card #2 test is selected, the next function executed is CARD3_MENU, (CARD3FUN.CC). This function sets up Decoder #2 to select the diagnostic mode. Data collection commences when the CARD_2_DECODE, (DEC2CARD.CC) is activated. This routine receives 16 pulses that are transmitted from Transmitter #1. The function checks for proper pulse spacing and delay. This continues until all eight selectable decoders are satisfied.

When the ANALOG CARD is selected, the function executed is CARD_4_MENU, (CARD4FUN.CC). This function sets up the ANALOG CARD to select the diagnostic mode. Data collection commences when the RAW CARD, (RAWCARD.CC) is activated. (See RAW_CARD explanation).

When the digital system test is selected, the functions executed are CARD6A_MENU, (CARD6AFU.CC) and CARD6B_MENU, (CARD6BFU.CC). The test function CARD6A_MENU sets up the system in two modes. The first sets up the RAW DATA card in the diagnostic mode and

checks for proper bracket spacing (see CARD_l_DECODE) and reruns the same test as mentioned above. The test function CARD6B_MENU sets up the decoder cards to check for bracket code data. The function BRK_CARD, (BRKCARD.CC) is activated. This function turns off one bit at a time in the code train and checks for proper code data.

When the RF Loop Back Test is selected, the function executed is CARD5_MENU (CARD5FUN.CC). This places the RF unit in the loop-back mode and executes the function RF_CARD, (RFCARD.CC). The function is the same as the RAWCARD test sequence with two differences. The two differences are frequency and amplitude parameters. The function checks for proper frequency and amplitude, and varies the amplitude at six different values.

The last selection in the menu is Entire DATAS System Test. This selection executes all of the functions mentioned above, one at a time.

HARDWARE DIAGNOSTIC OPERATION/TROUBLESHOOTING TECHNIQUES.

Prior to running the hardware diagnostics, the DATAS memory should be verified by running the memory diagnostics. In order to run the hardware diagnostics, the user enters **Hwtest** <CR> on the keyboard. This activates the Hardware Diagnostic Package. A menu appears on the display (figure 9). The user can select any seven of the test options by using the up and down cursor keys. When the selection is in bold print, the selection can be executed by pressing the <CR> key. The user may checkout each board separately, groups of boards, or the complete system.

The user runs through all of the selections one at a time and will be shown some troubleshooting techniques to help solve board related problems. The user selects RAW DATA CARD TEST ONLY and presses <CR>. TEST IN PROGRESS will appear on the display adjacent to the question (figure 10).

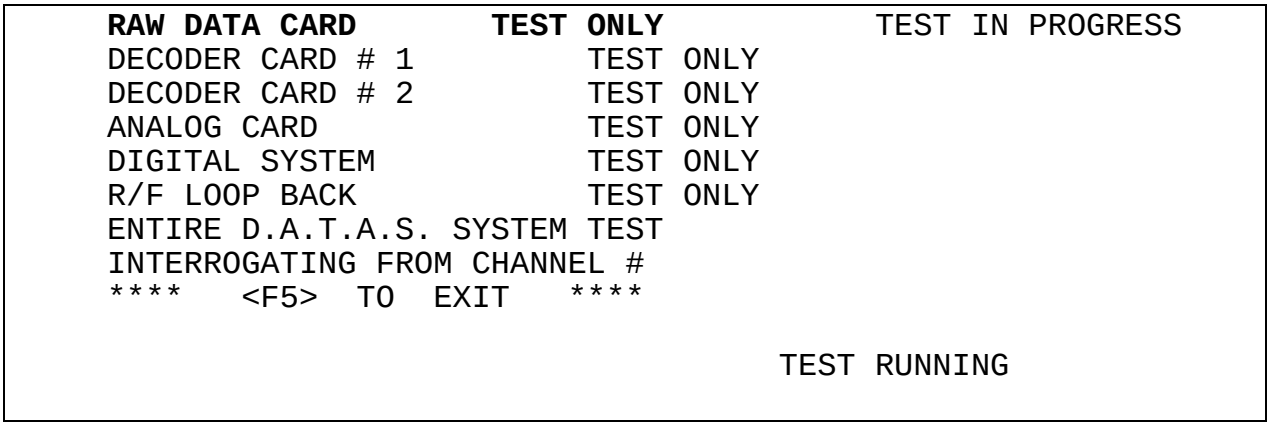


FIGURE 10. MENU FOR THE HARDWARE DIAGNOSTIC ROUTINE (RUNNING)

TEST RUNNING will be displayed in the lower right hand corner. If the TEST IN PROGRESS appears and TEST RUNNING does not, then there is a problem with the interrupt handler routine. Press the Break key to exit from the hardware diagnostic menu. No data are stored for analysis unless the interrupt handler is functioning properly. There may be one of two problems causing this failure. The first one is the least extreme of the two problems. The interrupt handler program was never executed. This program is in SYS:300.... By typing USE 300 <CR>, gets the user into User Number 300. To execute the interrupt handler routine, the user enters EXEC <CR>. The interrupt routine is loaded and operational. Exit the routine and return to User Number 302. USE 302 <CR> performs this task. Type HWTEST again and go through the same procedure as stated above. If TEST RUNNING does not appear on the screen, Transmitters #1 or #2 has a hard failure. Replace one board at a time and run HWTEST again until TEST RUNNING appears on the screen. After successful completion of the test, TEST IN PROGRESS is erased and a PASSED TEST is displayed in its place. If the card is failing, TEST RUNNING is replaced with ERRORS ARE BEING STORED ON DISK. After completion of the test, TEST IN PROGRESS is replaced with FAILED TEST. The first six options of the menu work in the same manner of displaying successful completion or failures to the screen. The last option, ENTIRE DATAS SYSTEM TEST, executes the first six options, one at a time.

If the diagnostics all pass, the file called Data.Dmmddyy.aa contains the data stored for that particular day (figure 11).

FAILURE CONDITIONS FOR EACH CARD.

TRANSMITTER #l/RAW DATA CARDS. If the user has completed the HWTEST diagnostic package and failures occurred for every board, the user can review the data for every board. The first board the user examines is the RAW DATA CARD. In figure 8 (DATAS Hardware Block Diagram), Transmitter #l transmits a pulse train to the external control logic, which then routes the signal to the RAW DATA CARD.

TIME = CURRENT TIME OF TEST BEING EXECUTED

TIME PASSED RAW_DATA CARD IN THE PULSE TO PULSE MODE
TIME PASSED RAW_DATA CARD REFERENCED TO THE REPLY WINDOW
TIME PASSED DECODER #l CARD REFERENCED TO THE DECODER #l CARD
TIME PASSED DECODER #2 TEST USING DECODE STORAGE #l
TIME PASSED DECODER #2 TEST USING DECODE STORAGE #2
TIME PASSED DECODER #2 TEST USING DECODE STORAGE #3
TIME PASSED DECODER #2 TEST USING DECODE STORAGE #4

```
TIME   PASSED DECODER #2 TEST USING DECODE STORAGE #5
TIME  PASSED DECODER #2 TEST USING DECODE STORAGE #6
TIME   PASSED DECODER #2 TEST USING DECODE STORAGE #7
TIME   PASSED DECODER #2 TEST USING DECODE STORAGE #8
TIME   PASSED ANALOG CARD IN THE PULSE TO PULSE MODE
TIME   PASSED ANALOG CARD REFERENCED TO THE REPLY WINDOW
TIME   PASSED DECODER #l CARD REFERENCED TO THE RAW CARD
TIME   PASSED DECODER #l CARD REFERENCED TO THE ANALOG CARD
TIME   PASSED DECODER #l CARD CHECKING BRACKET CODE
TIME   PASSED CHANN #1 R_F LOOP AT -20 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -20 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #1 R_F LOOP AT -25 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -25 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #1 R_F LOOP AT -30 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -30 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #1 R_F LQOP AT -35 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -35 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #1 R_F LOOP AT -40 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -40 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #1 R_F LOOP AT -45 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #1 R_F LOOP AT -45 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -20 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -20 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -25 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -25 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -30 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -30 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -35 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -35 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -40 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -40 (REF.TO REPLY WINDOW)
TIME   PASSED CHANN #2 R_F LOOP AT -45 (PULSE TO PULSE MODE)
TIME   PASSED CHANN #2 R_F LOOP AT -45 (REF.TO REPLY WINDOW)
```

FIGURE 11. DATA FILE GENERATED FOR HWTEST

The first of two problems that the user might encounter with the RAW DATA CARD is in the number of pulses detected. The data format of the stored data file is as follows:

```
ADDRESS DATA WAS INVALID (OX00??)
*** DETECTED ?  OUT OF ?  PULSES"
```

If this occurs, one of three boards may have a failure. Transmitter #l might not be transmitting the correct number of pulses. Transmitter #2 might not be initialized to route the

output jacks to their appropriate destination. The last possibility is the RAW DATA CARD is faulty. Change each board one at a time and rerun the hardware diagnostics and replace the faulty board.

Another problem that might occur is incorrect pulse width, spacing, and/or delay. As depicted in figure 12, the pulse width read was 400 ns for the first and second pulses. The actual pulse width should be 450 ns +/- one clock (25 ns). The problem board is the RAW DATA CARD. Replace the board and rerun the test to obtain successful results.

The data format that stored in the data file is shown in figure 12.

	WIDTH		SPACING		#		
TIME	(READ)	(ACTUAL)	(READ)	(ACTUAL)	PULSES	MODE	PULSE #
08:00	400	450	1450	1450	16	1	0
08:00	400	450	1450	1450	16		
08:01	FAILED	** RAW_DATA CARD IN THE PULSE TO PULSE MODE					

FIGURE 12. FAILURES IN THE RAW DATA CARD

ANALOG CARD. After detailed analysis of the RAW DATA CARD, the user examines the ANALOG CARD, since both the ANALOG and RAW DATA CARDS are interrelated. If the RAW DATA CARD or Transmitter #l was faulty and replaced, ignore all of the data stored for the ANALOG CARD. As shown in figure 8, the diagnostic signal is routed to the ANALOG CARD, then to the RAW DATA CARD for data analysis. Remember, the ANALOG CARD has no memory allocation on the board, therefore, it uses the RAW DATA CARD memory to validate proper pulse width, spacing, and delay. Rerun the ANALOG CARD TEST. This card can give you problems if the ANALOG CARD is not aligned properly. (See alignment procedure for the ANALOG CARD.) If the ANALOG CARD is aligned correctly and still fails (see figure 13), replace the analog board and rerun the test.

	WIDTH		SPACING		#		
TIME	(READ)	(ACTUAL)	(READ)	(ACTUAL)	PULSES	MODE	PULSE #
8:00	400	450	1450	1450	16	1	0
8:00	400	450	1450	1450	16	1	1
8:01	FAILED	** ANALOG CARD IN THE PULSE TO PULSE MODE					

FIGURE 13. FAILURE IN THE ANALOG CARD

DECODER #2 BOARD. Figure 8 shows that Transmitter #l transmits a known signal through the diagnostic path to the Decoder #2 board. By scrolling to the Decoder #2 card test and pressing the <CR> key, the diagnostic testing for this board starts. This routine gets an input pulse string from Transmitter #1 (figure 14) and checks for correct pulse spacing and delay.

After the <CR> key is pressed, TEST IN PROGRESS is displayed to the right of the selection and TEST RUNNING is displayed in the lower right hand corner. The function runs for 10 interrupts and checks for correct pulse spacing and delay, then it moves out the second to sixteenth pulses, 100 ns and runs again. This continues for 3,000 iterations, or 300 µs. After that test is completed, the function runs again for the other seven possible decode types. This runs for 10 interrupts and only moves out the second to sixteenth pulse, 200 iterations or 20 µs. The TEST RUNNING flashes seven times, showing that the routine is cycling through the test sequence. If the passed criteria is met, the terminal displays PASSED TEST to the right of the selection and the data stored in the file is shown in figure 15.

If errors are being encountered, the message ERRORS ARE BEING STORED ON DISK is displayed on the terminal in the lower right hand corner. There are three possibilities that can exist if the card is faulty. The first possibility is the number of decodes detected is not equal to the number of pulses transmitted by Transmitter #l (figure 16).

The second possibility is the number of time of arrivals (READ#DEC) detected is not equal to the number of pulses (figure 16).

The last possibility is the time of arrival times for each pulse are not equal to the pulse string times transmitted. See figure 6.

DECODER #1 BOARD. Figure 8 shows that Transmitter #l transmits a known signal through the diagnostic path to the Decoder #l Board. By pressing the <CR> key the Decoder #l card diagnostic is selected and starts execution. The diagnostic checks to see if a bracket is declared. A bracket is two pulses spaced 20.3 µs. The routine loads a pattern of two pulses spaced 0.9 µs apart (figure 17).

```
LE,       200,      0,        1, REPLY_WIND, NO_ACTION
LE,       200,      1,        1, PAM,        NO_ACTION
TE,       200,      1,        1, PAM,        NO_ACTION
LE,       1250,     1,        1, PAM,        NO_ACTION
```

```
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250     1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       1250,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       4150,    1,        1, PAM,        NO_ACTION
TE,       200,     1,        1, PAM,        NO_ACTION
LE,       2000,    0,        1, REPLY_WIND, NO_ACTION
DELAY,    30000,   0,        1, END, INTERRUPT
0,        0,       0,        0, 0,          0,

PULSE     TIME     ATTENUATOR CHANNEL ACTION TRANSMIT
                   FIELD      SELECT  TYPE   TYPE
(LE,TE) (NANO      Pl = 1
(DELAY) SECONDS    P3 = 3     1,2,3 PAM,REPLY INTERRUPT
                   FIELD 0 - 7      WINDOW    REFERENCE
```

FIGURE 14. PREDETERMINED PULSE STRING TO ANALYZE DECODER #2 CARD

```
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #l
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #2
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #3
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #4
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #5
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #6
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #7
    (TIME)    PASSED DECODER #2 TEST USING DECODE STORAGE #8
```

FIGURE 15. FILE STORAGE FOR DECODER #2

```
  (TIME)  COUNTED # DEC=15 READ # DEC=16 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=15 READ # DEC=16 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=15 READ # DEC=16 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=15 READ # DEC=16 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=15 READ # DEC=16 # PULSES=16 FLAG-AD=0
```

a. Number of Decodes Invalid

```
  (TIME)  COUNTED # DEC=16 READ # DEC=15 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=16 READ # DEC=15 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=16 READ # DEC=15 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=16 READ # DEC=15 # PULSES=16 FLAG-AD=0
  (TIME)  COUNTED # DEC=16 READ # DEC=15 # PULSES=16 FLAG-AD=0
```

b. Number of Time Arrivals Invalid

```
TIME PULSE #DECODES #ADDR OVERFLOW      ADDR       SW        HW
                                       (HEX)    (TIME)    (TIME)

8:00 1       16     16        0         EF824028  1400      1650
8:00 2       16     16        0         EF82402C  2900      3100
8:00 3       16     16        0         EF824030  4300      4550
```

c. Time of Arrival Time Invalid

FIGURE 16. ERROR STRING FOR DECODER #2

```
    LE,       200,      0,    1, REPLY,WIND      NO_ACTION
    LE,       200,      1,    1, PAM,            NO_ACTION
    TE,       200,      1,    1, PAM,            NO_ACTION
    LE,       700,      1,    1, PAM,            NO_ACTION
    TE,       200,      1,    1, PAM,            NO_ACTION
    TE,       1000,     0,    1, REPLY_WIND,     NO_ACTION
    DELAY,    30000,    0,    1, END,            INTERRUPT
    0,        0         0     0, 0,         0,
```

FIGURE 17. PREDETERMINED PULSE STRING TO ANALYZE

TEST IN PROGRESS is displayed next to the selection TEST RUNNING is displayed in the lower right hand corner of the screen. This routine runs 100 iterations verifying pulse spacing of a generated pulse train starting at a spacing of .9 μs. It then increases the spacing 100ns and runs the test again. This continues for 240 iterations. When completed, two conditions can occur. If the passed criteria is met,

TIME - PASSED DECODER #l CARD REFERENCED TO THE DECODER #l CARD

is stored in the data file and PASSED TEST is displayed to the right of the selection. If the failed condition is displayed, FAILED TEST, the data file stores the errors encountered and the message

TIME ** FAILED ** DECODER #l CARD REFERENCED TO THE DECODER #l CARD.

The user lists the errors on the terminal to analyze the problem. Two error conditions can occur. One condition is the Decoder #1 card declared no brackets (figure 18).

```
TIME  #DECODES=?  BRK WIND=?  LOOP# ?OUT OF ?  FLAG.AD-?

08:00 #DECODES=0  BRK WIND=.9 LOOP# 1 OUT OF 100 FLAG.AD
```

FIGURE 18. ERROR STRING FOR DECODER #1
(BRACKETS NOT DECODED)

The user sees the number of decodes is equal to zero. The user has determined from previous tests, Transmitter #1, Raw Data, Decoder #2 and the Analog cards passed for validity. Transmitter #1 and Decoder #2 were already verified to be in working condition, therefore, Decoder #1 is the problem. Replace the board and rerun the test. The other error condition is the bracket declared is not equal to the pulse string generated (figure 19).

TIME	PULSE#	#DECODES	#ADDR	OVERFLOW	ADDR (HEX)	SW (TIME)	HW (TIME)
08:00	0	1	1	0	EF824028	1400	1600
08:00	0	1	1	0	EF824028	1400	1600

FIGURE 19. ERROR STRING FOR DECODER #l
(BRACKET TIME IN ERROR)

The user sees the bracket time spacing is in error, therefore, replace Decoder #1 and rerun the test.

ENTIRE DIGITAL SYSTEM. The entire digital system operates by transmitting a known signal through the diagnostic path. The waveforms are fed to the RAW DATA CARD and the ANALOG board. The user selects the ENTIRE DIGITAL SYSTEM TEST and presses <CR> key. As mentioned before, the TEST IN PROGRESS and TEST RUNNING is displayed on the screen as an indicator to the user.

If the passed criteria is met, PASSED TEST is displayed and the data is stored in the file (figure 20).

```
(TIME) PASSED DECODER #l REFERENCED TO THE RAW CARD
(TIME) PASSED DECODER #l REFERENCE TO THE ANALOG CARD
(TIME) PASSED DECODER #1 CHECKING BRACKET CODE
```

FIGURE 20. DATA STORAGE FOR ENTIRE DIGITAL SYSTEM TEST

If a failure condition is indicated, the only possibility is that the bracket code is invalid. All of the boards of DATAS are validated for successful results, except for the bracket code. The list of the data stored are shown in figure 21.

TIME	(CODE PULSE)	(CODE CHECK)	DATA	OVERFLOW	#DECODES	#BRACKETS
08:00	FFFC	FFFF	FFFC	0	1	1

FIGURE 21. ERROR STRING DECODER #2 (BRACKET CODE)

The code transmitted (code check) should be the same as the code read (code pulse). The error is in Decoder #2 card. Replace that board and rerun the test again.

TRANSMITTER #2. The same diagnostics that are usually run using Transmitter #1 can run with Transmitter #2. Using the up or down arrow keys, the user selects TRANSMIT FROM CHANNEL #1. Press <CR> key to change to Channel #2. Select using the up or down arrows to the ENTIRE DIGITAL SYSTEM TEST and press <CR> key.

As mentioned before, TEST IN PROGRESS and TEST RUNNING is displayed on the terminal. If a failed condition is displayed on the terminal, replace the Transmitter #2 board and rerun the

test. After the completion of the ENTIRE DIGITAL SYSTEM TEST for Transmitter #l is completed and successful, the only possible cause for ENTIRE DIGITAL SYSTEM TEST to fail would be a failed Transmitter #2 board.

RF LOOP BACK TEST. The RF loop back test verifies the system through the RF front end. Figure 8 shows that Transmitter #1 transmits a known signal to the RF unit. The RF unit returns the signal to the analog board via the normal path. To operate the user scrolls to the R. F. LOOP BACK TEST, and presses <CR> key. If the diagnostic is running properly, TEST IN PROGRESS and TEST RUNNING is displayed on the terminal. The test runs at six different amplitude values. If the test is successful, the terminal displays PASSED TEST and the data file contains the data as shown in figure 22.

```
(TIME) PASSED CHANN #1 R_F LOOP AT -20 (PULSE TO PULSE MODE)
(TIME) PASSED CHANN #1 R_F LOOP AT -20 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -25 (PULSE TO PULSE MODE)
(TIME) PASSED CHANN #1 R_F LOOP AT -25 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -30 (PULSE TO PULSE MODE)
(TIME) PASSED CHANN #1 R_F LOOP AT -30 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -35 (PULSE TO PULSE MODE)
(TIME) PASSED CHANN #1 R_F LOOP AT -35 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -40 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -40 (PULSE TO PULSE MODE)
(TIME) PASSED CHANN #1 R_F LOOP AT -45 (REF.TO REPLY WINDOW)
(TIME) PASSED CHANN #1 R_F LOOP AT -45 (PULSE TO PULSE MODE)
```

FIGURE 22. DATA FILE STORAGE FOR R F LOOP-BACK TEST

If the test fails, there are many possible errors that could cause the test to fail. An example of a few failures is shown in figure 23.

```
 TIME  WIDTH  (ACTUAL)  SPACING  ACTUAL  # PULSES  MODE  PULSE #
       (READ)           (READ)

 8:00   400     450      1450    1450       16       0      0
 8:00   400     450      1450    1450       16       0      1
 ADDRESS DATA WAS INVALID 4  ***DETECT 4 OUT OF 16 PULSES
 8:01   -18.5 = -20 (AMPLITUDE) 1090.001=1090.00(FREQ.) PULSE #1
 8:00   -25.1 = -25 (AMPLITUDE) 1087.001=1090.00(FREQ.) PULSE #5
```

FIGURE 23. ERROR STRING FOR R_F LOOP-BACK TEST

The two most probable causes are the pulse width is in error and/or the address data was invalid (detected x out of 16 pulses). Also, the user may encounter an out of tolerance pulse amplitude or frequency. If this error is detected, the whole RF

chassis should be replaced. Replace the chassis and run the test again.

ENTIRE DATAS SYSTEM TEST. This test runs all of the tests previously mentioned with one exception. The test runs through all of the tests using Transmitter #1 as the transmitting source and then runs through the entire test again using Transmitter #2 as the transmitting source.