Floating Point Systems, Inc.

AP-120B

ARRAY TRANSFORM PROCESSOR

AP-120B

DIAGNOSTIC SOFTWARE

MANUAL

FLOATING POINT
SYSTEMS, INC.

AP-120B

DIAGNOSTIC SOFTWARE

MANUAL

FPS-7284-01

Published in the United States of America
Rev. January 1976

P.O. BOX 23489 PORTLAND, OR 97223 11000 S.W. 11TH STREET, BEAVERTON, OR 97005 (503) 641-3151 TLX: 360470 FLOATPOINT PTL

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT).

APARTH

FORM # 7278

AP-120B

APTEST

DIAGNOSTIC SOFTWARE MANUAL

Revision 1   11-23-75

Adds 'S' Command
Converts Codes to Hex
Adds tests 7,8,9 and 10

Revision 1.1   1-21-76
Corrects Typographical errors.
Updates Test 7,8,9,10 error messages.

SECTION 1
BRIEF DESCRIPTION;APTEST

1.1 BRIEF DESCRIPTION

This program exercises the Panel and DMA interface functions of the AP-120B. It tests all of the available memory inside the AP-120B with simple patterns and then with pseudo-random number patterns.

(INTENTIONALLY BLANK)

SECTION 2
RDS 500 OPERATING INSTRUCTIONS

2.1 RAYTHEON RTOS

Start the program with;

:QU, APPATH
:EX

The program will indicate its readiness to accept user commands by typing an "*". Typical user response at this point is to type "RWE (CR) ". The program will then begin to cycle through the test and will type a 'P' for every complete passes through the test. If an error is encountered, the program will type out the error and a code number. Refer to Section IV for a complete description of the error outputs and their interpretation. Given the user input of "RWE (CR) ", the program will type an "*" following the error and then wait for input from the keyboard. At this point the user can cause the test to loop on this error if he wishes to isolate the problem below the Board level using an oscilloscope. The following section describes the user input commands.

Note:

This program requires that the FPS supplied package of Teletype Control Routines (INPT) be extended onto the Disk along with APTEST in order for the above Queing sequence to work.

(INTENTIONALLY BLANK)

SECTION 3
USER COMMANDS

3.1 USER COMMANDS

The test program responds to a set of single letter commands, some of which are to be followed by one or two Hexadecimal integers. A string of commands may be typed on a line terminated by a carriage return. With the exception of the 'E' (Execute) command, the commands can be typed in any order on the line. The 'E' command should be typed last since commands following the 'E' will not be seen by the command string interpreter. Typing a Control C "↑C" will cause the current line to be ignored. Sense Switch Ø is used to interrupt test program execution and bring control back to the command input portion of the test program. The "Rubout" function is more specialized than in RTOS Teletype input as it can be used to delete only certain commands. Table 3-1 summarizes the commands recognized by this test program.

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
Annnn	Input the starting adress of another program in core to which control is to be transferred by the 'B' command.	Must be followed by a Hexadecimal integer specifying the desired absolute address.
B	Transfer program control to the address specified in the last preceding 'A' command.	
C	Type out the user input flags that have been set.	
D	Set the typeout Disable flag. Used for Scoping a hardware fault. Also disables the error detection.	
E	Execute the test program Used to transfer control from the command input section of the program to the test section.	This is the last command on the line that will be seen by the Command String Interpreter.
H	Set the unconditional Halt flag. Following the next 'E' command the program will execute one test case and return to command Input mode.	
I	Set the IO Reset flag. Test program Resets the AP-120B between each test case.	
L	Set the LOOP flag. Program loops on the last executed test case.	
Mnnnn	Define the AP-120B Memory Size.	Must be followed by a hexadecimal integer (2000=8K)
R	Reset all flags. Clears D,H,I,L,S and W.	
S	Set short form format for error typeout. Affects FIFFT only.	

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
W	Set Wait on error flag. If set, the test will return to command input mode after encountering and typing out an error.	
Xnnnn,nnnn	Reset the Random Number generator paramenters. Used to recreate a test case from the parameters typed out in an error message.	Must be followed by two hexadecimal integers on the same line.
Rubout	Used to selectively clear one of the flags (D,H,I,L,S,N,S, or W). slash "/".	Type the flag to be cleared followed by a rebout i.e., "LRubout" clears the Loop flag.
↑C	Used to delete and input command string if the "C" is typed before the Carriage Return.	

SPECIAL RDS 500 DEBUG COMMANDS

User Command Functions.

;	Type Absolute address of Calling Routine. Used to locate the Calling program in the absence of a load map.
.nnnn	Set Base address. The Debugger has a Base address register that allows the user to reference locations in his relocatable program using the output of the SYMII Assembler. Following the ";" command the user should compute the absolute beginning address of his program and input it to the debugger using the "." Command.

Example:

```
*; (CR)
6CAA
*.6CA5 (CR)
*
```

Explanation:

Call to Debugger was at 6CAA. This is the absolute address of the next instruction following the call. Assuming this instruction was at relative location 5, the user subtracts 5 from 6CAA and enters 6CA5 as the Base address. (Asterisks are typed by program)

Onnnn	Open the location nnnn+Base. The debugger will type out the contents (in hexadecimal) of the opened location.

To modify this location, the user simply types the desired hexadecimal value followed by one of the three pointer movement commands.

To leave it unchanged and open contiguous locations or return to command input, the user simply types one of the three pointer movements commands (see below).

Pointer Movement Commands:

(LF) (line feed) closes the currently open location and advances the pointer to the next location in memory. The debugger will type out the address and then the contents of the next location.

(↑CR) (up arrow, carriage return) closes the currently open location and moves the pointer to the preceding memory location. The location and contents are typed out in hexadecimal.

(CR) (carriage return) closes the currently open location and returns to command input mode. The program will type out an "*".

Example:

*025 (CR)	OPEN location 25
1234 123 (LF)	change to 123, go to 26
ØØ26 45AB (LF)	no change to ABF, go to 26
ØØ27 1564 ABF ↑(CR)	change to ABF, go to 26
ØØ26 45AB FF (CR)	change to FF, Quit

Address Calculation Commands:

=nnnn — Adds nnnn to Base and types out result. Converts relative addresses to absolute.

-nnnn — Subtracts Base from nnnn. Converts absolute addresses to relative.

Program Control Commands

Tnnnn — Trap. Sets a Breakpoint trap at relative location nnnn. The trap consists of two instructions.

```
SMB TRTN
JMP TRTN
```

These two instructions are inserted in the user program at the time that the "G" (GO) command is issued. When the trap is encountered, the routine TRTN will save the ACR and IXR, restore the two user instructions and return to command input mode. Once encountered the TRAP is removed and will not be set again until the user issues another 'T' command.

Gnnnn — GO. Starts the user program running at relative location nnnn after inserting a TRAP (if previously called for by user) and restoring the ACR and IXR.

P Proceed. Starts the user program running from the last TRAP location. ACR and IXR are restored. The overflow and compare flops are lost.

Example:

1)	*T45 (CR)	Set TRAP at 45
2)	*G25 (CR)	Start at 25
3)	TRAP 65AD ACR 8ØØØ IXR ØØ37	Prog. Encounters TRAP
4)	*P (CR)	Proceed from 45

At line 3 where the trap was encountered, the program types out the absolute Hexadecimal location of the trap and the contents of the ACR and the IXR.

NOTES:

1) Hexadecimal integers may consist of from 1 to n digits terminated by a non-numeric character. This character may be a comma, a carriage return or the next command letter (other than A,B,C,D or E). If more than 4 digits are typed, only the last 4 will be taken as the desired hexadecimal integer.

EXAMPLES:

1) For Normal operation type "RWE(CR)". This starts the test running so that it will return to command input mode when an error is encountered.

B) Looping on an error:
   1. After program has typed out an error and returning to command mode, Type "LE (CR) " to loop on the failed case to see if the error is solid. Type "LDE (CR) " to go into a scope loop with typeout disabled.
   2. After the error is corrected lift SSØ to interrupt the test. Type "D Rubout E (CR) " to check to make sure that it has been corrected.
   3. To Return to the full test, lift SSØ to interrupt and type "RWE (CR) " to proceed.

C) For an overnight Run type "RE (CR)". The program will type out all errors and proceed. This will leave a record of any failures that may have occured. The program will stop the test automatically if more that 64 errors occur. This is done so as to prevent excessive wear on the Teletype in the case of a catastrophic failure.

SECTION 4
APTEST ERROR MESSAGES

4.1 ERROR MESSAGES

4.1.1 TEST 1 checks the WC, HMA, CTRL, APMA, ASPR, and FN registers.

Error Message Format:

E nnnn A mmmm cccc

Where nnnn is the expected value,
mmmm is the actual value,
and cccc is a code number between Ø and 5.

TEST 1 Code Number description:

Code Number	Register under Test	Most likely failing board or boards
0	WC	Interface
1	HMA	Interface
2	CTRL	Interface
3	APMA	Interface
4	APSR	214
5	FN	214

4.1.2 TEST 2 and TEST 2A check the ability of the AP-120B Panel to access all of the internal registers that are available to it.

Error output format is the same as TEST 1 except that the code number will be in the range 6 to 19 (except for PS word 3 = 8010).

TEST 2 and TEST 2A Code Number description:

Code Number	Register under Test	most likely failing board or boards
6	PSA	212,214,210
7	SPA	210,201,214
8	MA	201,214
9	TMA	212,214
A	DPA	211
B	SPAD	201
C	APSTATUS	210,201 (if low 3 bits)
D	DA	Interface
E	PS word Ø	216,210,201
F	PS word 1	216,201
10	PS word 2	216,201
8010	PS word 3	216,201
11	DPX EXP	200L,211
12	DPX HMAN	200R
13	DPX LMAN	200L,200R
14	DPY EXP	200L,211
15	DPY HMAN	200R
16	DPY LMAN	200L,200R
17	MD EXP	215,202,213,210
18	MD HMAN	215,202,213
19	MD LMAN	215,202,213

4.1.3 TEST 3 performs a basic address test on all the memories in the AP-120B. The address of each memory location is written into that location until all locations have been written, then the entire memory is read back and verified.

TEST 3 error message format:

BASIC ADDR E nnnn A mmmm cccc

Where nnnn is the expected value and is also the address of the failing location,
mmmm is the actual result
and cccc is the code number bewteen 1A and 27 that identifies the memory under test.

See TEST 4 for the Code Number descriptions.

4.1.4 TEST 4 performs a random patterns test of each memory by filling it with pseudo-random numbers, and reading them back to check.

TEST 4 error message format:

RND PATTERNS E nnnn A mmmm CODE=cccc ADDR=aaaa
X, Y=xxxx yyyy

Where nnnn is the expected value,
mmmm is the actual value (note that only the bits in this word that lie within the width of the memory portion under test are checked)
cccc is a code number between 1A, and 27,
aaaa is the address of the failing location
and xxxx yyyy are the restart parameters for the random number generator.

TEST 3 and TEST 4 Code Number descriptions:

Code Number	Memory Under Test	most likely failing board or boards
1A	SPAD	201,210
1B	DPX EXP	200L,211
1C	DPX HMAN	200R
1D	DPX LMAN	200L,200R
1E	DPY EXP	200L,211
1F	DPY HMAN	200R
20	DPY LMAN	200L,200R
21	PSØ	216,212
22	PS1	216
23	PS2	216
24	PS3	216
25	MD EXP	215,201,210
26	MD HMAN	215
27	MD LMAN	215

4.1.5 TEST 5 checks the DMA to DMA interface by transferring Host memory contents into AP-120B Main Data memory in 16-bit integer mode and then checking the contents of MD against Host memory by using the AP-120B Panel.

TEST 5 error message #1:

MEMTST LOAD ERROR
WC wwww HMA hhhh APMA aaaa CTRL cccc

The correct values for wwww, hhhh, aaaa, and cccc, are ØØØØ, 2Ø8Ø, 2ØØØ, and 8Ø42, for an AP-120B with 8K of MD.

Where wwww, hhhh, aaaa,
and cccc are the respective contents of the AP-120B WC, HMA, APMA, and CTRL registers after the completion of the DMA transfer.

This error indicates that the transfer did not go to completion correctly. The interface board is the most likely failing Board in this case.

TEST 5 error message #2:

MTEST CHECK LOC=1111 E eeee A aaaa

Where 1111 is the MD location having incorrect contents,
eeee is the expected value (host memory contents)
and aaaa is the actual value.

This error generally indicates a failure in the Interface, or Format boards, however, the problem could be in Main Data memory itself, Board 215.

4.1.6 TEST 6 Reverses TEST 5 and stores a portion of MD back into Host memory and then checks for a correct transfer.

TEST 6 error message #1:

DMA STORE NOT DONE
WC wwww HMA hhhh APMA aaaa CTRL cccc

See description of TEST 5 error message #1.

TEST 6 error message #2:

MSTORE LOC=1111 E eeee A aaaa

Where 1111 is Host memory location having incorrect contents.
eeee is the expected (MD) value
and aaaa is the actual value.

This error generally indicates an Interface or Format board failure, however, the failure could be in MD, Board 215.

4.1.7 TEST 7 Tests RDS-500 single precision to AP-120B Floating Point format conversion.

Error Message:

TEST7/8 E eeee eeee eeee
A aaaa aaaa aaaa ANSP pppp

Where eeee is the expected AP-120B format result
aaaa is the actual result.
pppp is the pointer to the table entry in the listing. Consult "RAYTBL" in the listing to find the input RDS-500 format FPN.

4.1.8 TEST 8 Tests AP-120B to RDS-500 format conversion.

Error Message:

TEST7/8 E eeee eeee
A aaaa aaaa ANSP pppp

Where eeee is the expected
and aaaa is the actual RDS-500 format result.
pppp is a pointer to the table entry in the listing. Consult "APTBL" in the listing to find the AP-120B format number that is equivalent to the expected result.

4.1.9 Tests IBM short form to AP-120B floating point conversion.

Error Message:

```
TEST9/10 E eeee eeee eeee
A aaaa aaaa aaaa ANSP pppp
```

Consult "IBMTBL" in listing

4.1.10 TEST10 Tests AP-120B to IBM format conversion.

Error Message:

```
TEST 9/10 E eeee eeee
A aaaa aaaa ANSP pppp
```

Consult APTBL2 in listing

Notes on Tests 7, 8, 9 and 10:

1) The only difference in the error messages between Tests 7 and 8 and between Tests 9 and 10 is in the number of hexadecimal numbers typed out. For RDS-500 or IBM to AP-120B conversion (Tests 7 and 9), 3 words each of expected and actual results are typed out. For AP-120B to RDS-500 or IBM conversion (Tests 8 and 10) 2 words each are typed out.

2) Example of Test interpretation.

   Example message:

```
TEST9/10 E C11Ø ØØØØ
A 411Ø ØØØØ ANSP Ø6Ø7
```

Interpretation:

Since only 2 words are typed for the expected result, the error occured in Test 10. Go to "IBMTBL" in the listing of APTEST. Find the entry at relative location 607.

```
"    D    X'C11Ø',Ø      -1.0 ".
```

Find the corresponding entry in APTBL2

```
"    D    X'2Ø1',X'CØØ',Ø -1.0 ".
```

The entry in APTBL2 is the representation of -1.0 in AP-120B format that corresponds to the IBM 360 representation in IBMTBL. Where possible, the comments in both tables indicate the decimal value of the floating point number in question.

3) Errors in tests 7,8,9 and 10 can generally be traced to the interface or format Boards (228, 227, 226).

(INTENTIONALLY BLANK)

FORM #7279

AP-120B

APPATH

DIAGNOSTIC SOFTWARE MANUAL

Revision 1 11-22-75
Adds 'S' Command
Converts Codes to Hex.

Revision 1.1 1-21-76
Corrections of typographical errors.

SECTION 1
BRIEF PROGRAM DESCRIPTION;APPATH

1.1 BRIEF DESCRIPTION

This program tests the bulk of the internal data paths within the AP-120B. In contrast to the paths and registers tested by the program APTEST, the data paths and registers tested by the APPATH require the execution of AP-120B micro-instructions. Thus this test also effectively checks most of the AP-120B micro-instruction set.

(INTENTIONALLY BLANK)

SECTION 2
RDS 500 OPERATING INSTRUCTIONS

2.1 RAYTHEON RTOS

Start the program with:

:QU, APPATH
:EX

The program will indicate its readiness to accept user commands by typing an "*". Typical user response at this point is to type "RWE (CR) ". The program will then begin to cycle through the test and will type a 'P' for every complete pass through the test. If an error is encountered, the program will type out the error and a code number. Refer to Section IV for a complete description of the error outputs and their interpretation. Given the user input of "RWE (CR) ", the program will type and "*" following the error and then wait for input from the keyboard. At this point the user can cause the test to loop on this error if he wishes to isolate the problem below the Board level using an oscilloscope. The following section describes the user input commands.

Note:

This program requires that two other routines, APTBL1 and INPT, be extended onto the disk in order for the above Queing sequence to work.

(INTENTIONALLY BLANK)

SECTION 3
USER COMMANDS

3.1 USER COMMANDS

The test program responds to a set of single letter commands, some of which are to be followed by one or two Hexadecimal integers. A string of commands may be typed on a line terminated by a carriage return. With the exception of the 'E' (Execute) command, the commands can be typed in any order on the line. The 'E' command should be typed last since commands following the 'E' will not be seen by the command string interpreter. Typing a Control C "↑C" will cause the current line to be ignored. Sense Switch Ø is used to interrupt test program execution and bring control back to the command input portion of the test program. The "Rubout" function is more specialized than in RTOS Teletype input as it can be used to delete only certain commands. Table 3-1 summarizes the commands recognized by this test program.

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
Annnn	Input the starting address of another program in core to which control is to be transferred by the 'B' command.	Must be followed by a Hexadecimal integer specifying the desired absolute address.
B	Transfer program control to the address specified in the last preceding 'A' command.	
C	Type out the user input flags that have been set.	
D	Set the typeout Disable flag. Used for Scoping a hardware fault. Also disables the error detection.	
E	Execute the test program Used to transfer control from the command input section of the program to the test section.	This is the last command on the line that will be seen by the Command String Interpreter.
H	Set the unconditional Halt flag. Following the next 'E' command the program will execute one test case and return to command Input mode.	
I	Set the IO Reset flag. Test program Resets the AP-120B between each test case.	
L	Set the LOOP flag. Program loops on the last executed test case.	
Mnnnn	Define the AP-120B Memory Size.	Must be followed by a hexadecimal integer (2000=8K)
R	Reset all flags. Clears D,H,I,L,S and W.	
S	Set short form format for error typeout. Affects FIFFT only.	

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
W	Set Wait on error flag. If set, the test will return to command input mode after encountering and typing out an error.	
Xnnnn,nnnn	Reset the Random Number generator paramenters. Used to recreate a test case from the parameters typed out in an error message.	Must be followed by two hexidecimal integers on the same line.
Rubout	Used to selectively clear one of the flags (D,H,I,L,S,N,S, or W). slash "/".	Type the flag to be cleared followed by a rubout i.e., "LRubout" clears the Loop flag.
↑C	Used to delete and input command string if the " C" is typed before the Carriage Return.	

SPECIAL RDS 500 DEBUG COMMANDS

User Command Functions.

; Type absolute address of Calling Routing. Used to locate the Calling program in the absence of a load map.

.nnnn Set Base address. The Debugger has a Base address register that allows the user to reference locations in his relocatable program using the output of the SYMII Assembler. Following the ";" command the user should compute the absolute beginning address of his program and input it to the debugger using the "." Command.

Example:

```
*; (CR)
6CAA
*.6CA5 (CR)
*
```

Explanation:

Call to Debugger was at 6CAA. This is the absolute address of the next instruction following the call. Assuming this instruction was at relative location 5, the user subtracts 5 from 6CAA and enters, 6CA5 as the Base address. (Asterisks are typed by program)

Onnnn Open the location nnnn + Base. The debugger will type out the contents (in hexadecimal) of the opened location.

To modify this location, the user simply types the desired hexadecimal value followed by one of the three pointer movement commands.

To leave it unchanged and open contiguous locations or return to command input, the user simply types one of the three pointer movement commands (see below).

Pointer Movement Commands:

(LF) (line feed) closes the currently open location and advances the pointer to the next location in memory. The debugger will type out the address and then the contents of the next location.

↑(CR) (up arrow, carriage return) closes the currently open location and moves the pointer to the preceding memory location. The location and contents are typed out in hexidecimal.

(CR) (carriage return) closes the currently open location and returns to command input mode. The program will type out and "*".

Example:

*025 (CR)	OPEN location 25
1234 123 (LF)	change to 123, go to 26
ØØ26 45AB (LF)	no change, go to 27
ØØ27 1564 ABF ↑(CR)	change ABF, go to 26
ØØ26 45AB FF (CR)	change to FF, Quit

Address Calculation Commands:

=nnnn Adds nnnn to Base and types out result. Converts relative addresses to absolute.

-nnnn Subtracts Base for nnnn. Converts absolute addresses to relative.

Program Control Commands

Tnnnn Trap. Sets a Breakpoint trap at relative location nnnn. The trap consists of two instructions.

```
SMB TRTN
JMP TRTN
```

These two instructions are inserted in the user program at the time that the "G" (GO) Command is issued. When the trap is encountered, the routine TRTN will save the ACR and IXR, restore the two user instructions and return to command input mode. Once encountered the TRAP is removed and will not be set again until the user issues another 'T' command.

Gnnnn GO. Starts the user program running the relative location nnnn after inserting a TRAP (if previously called for by user) and restoring the ACR and IXR.

P Proceed. Starts the user program running from the last TRAP location. ACR and IXR are restored. The overflow and compare flops are lost.

Example:

1)	*T45 (CR)	Set TRAP at 45
2)	*G25 (CR)	Start at 25
3)	TRAP 65AD ACR 8ØØØ IXR ØØ37	Prog. Encounters TRAP
4)	*P (CR)	Proceed from 45

At line 3 where the trap was encountered, the program types out the absolute Hexadecimal location of the trap and the contents of the ACR and the IXR.

NOTES:

1) Hexadecimal integers may consist of from 1 to n digits terminated by a non-numeric character. This character may be a comma, a carriage return or the next command letter (other than A,B,C,D or E). If more than 4 digits are typed, only the last 4 will be taken as the desired hexadecimal interger.

EXAMPLES:

1) For Normal Operation type "RWE (CR)". This starts the test running so that it will return to command input mode when an error is encountered.

B) Looping on an error:
   1. After program has typed out an error and returning to command mode, Type "LE (CR)" to loop on the failed case to see if the error is solid. Type "LDE (CR)" to go into a scope loop with typeout disabled.
   2. After the error is corrected lift SSØ to interrupt the test. Type "D Rubout E (CR)" to check to make sure that it has been corrected.
   3. To Return to the full test, lift SSØ to interrupt and type "RWE (CR)" to proceed.

C) For an overnight Run type "RE (CR)". The program will type out all errors and proceed. This will leave a record of any failures that may have occurred. The program will stop the test automatically if more than 64 errors occur. This is done so as to prevent excessive wear on the Teletype in the case of a catastrophic failure.

SECTION IV
DETAILED DESCRIPTION OF TESTS PERFORMED AND THE ERROR MESSAGES

4.1 DETAILED DESCRIPTION OF TESTS

Each path is tested with the following data patterns; zero, all-bits on, and a single true bit which is passed through all bit positions in the path under test starting from the least significant bit and moving left to the most significant bit.

Error output format:

```
CODE cccc EXP eeee eeee eeee
ACT aaaa aaaa aaaa PLOC pppp
```

Where cccc is a code number that indicates the type of test being performed.

eeee etc is a 2 to 38-bit value indicating the expected result.

aaaa etc is a 2 to 38-bit value indicating the actual result.

and pppp is pointer to the test table entry corresponding to the current test case. This pointer can be used with the program listing if analysis beyond the level of that given in the following table is desired. The pointer is expressed relative to the base address of the table (APTBL1).

Description of error codes and board level Diagnostic indications.

HEX Code*	Unique Path or Paths under test	Most likely failing Board or Boards
40	PS Left half to DPBS,	210, 214
41	PS Right half (FP Value) to DPBS,	210, 211
42	DPBS to PS left half	201, 210
43	DPBS to PS Right half	201, 210
44	DPBS (approximation portion of mantissa) to SPAD	201, 210
45	DPBS (Exponent portion) to SPAD	201, 210
46	DPBS to I/O Bus to WC	Interface, 214
47	DPBS to I/O Bus to HMA	Interface
48	DPBS to I/O Bus to CTRL	Interface
49	DPBS to I/O Bus to APMA	Interface
4A	DPBS to I/O Bus to FMT	222
4B, 4C	Not implemented	
4D	MD to FMT	222, Interface
4E	DPX to A1, FA to M2 FM to DPY	200L, 200R 203, 204, 205 206, 207, 208
4F	DPX to A2 FA to DPY	200L, 200R 203, 205, 204
50	DPY to A1 FA to DPX	200L, 200R
51	DPY to A2 FA to MD	200L, 200R, 213
52	FA to A2	203, 205

* PROGRAM PRINTS IN OCTAL ∴ MUST CONVERT.
EX 101 = 41

Code	Unique Path or Paths under test	Most likely failing Board or Boards
53	DPY to M2 FM to DPX	200L, 200R 206, 207, 208
54	DPY to M1 FM to DPY	200L, 200R 206, 208
55	DPX to M2 FM to MD	200L, 200R, 213
56	DPX to M1 FM to A1	200L, 200R 203, 205
57	MD to M2 FM to M1	202 206, 208
58	MD to A2	202
59	TM to DPBS	217, 209
5A	TM to A1	209
5B	TM to M1	209
5C	VAL to DPBS (mantissa) SPFN to MA	211 201
5D	SPFN to DPBS (mantissa)	201
5E	VAL to DPBS (exponent) SPFN to DPBS (exponent)	211 201
5F	SPFN to A2 EXP	212, 200L
60	SPFN to A2 mantissa	212, 200L, 200R
61	Not implemented	
62	VAL to EXIT EXIT to PNLBUS PNLBUS to SPAD	212, 201
63	VAL + PSA to EXIT (Relative JSR)	212
64	SPFN(DPBS) to TMA TMA to EXIT	212
65	VAL to PSA (JMPA)	212

Code	Unique Path or Paths under test	Most likely failing Board or Boards
66	VAL + PSA to PSA (JMP)	212
67	EXIT to PSA (Return Jump)	212
68	TMA to PSA (JMPT)	212

NOTES:

Each path typically uses more paths than the unique ones indicated. The choice of indicated paths is based on the premise that previously tested paths have been verified and are thus no longer under test as the test proceeds.

The mnemonics used in the above Table are described in detail in the AP-120B programming manual.

RM #7280

AP-120B

FIFFT

DIAGNOSTIC SOFTWARE MANUAL

Revision 1   11-22-75
Adds 'S' Command
and short form message
Adds description of AP-120B
Hardware debug routines

SECTION 1
BRIEF DESCRIPTION;FIFFT

1.1 BRIEF DESCRIPTION

This program is intended primarily for use as a verification test of the AP-120B. It does not provide board level diagnostic indicators as do the diagnostic programs, APTEST and APPATH. Its use as a diagnostic requires the skill of a technician who is well versed in the theory of operation of the AP-120B.

The test is based on the fact that a forward Fourier Transform of a data set followed by an inverse Fourier Transform of the result of the forward transform should result in the original data set within a predictable error limit. Thus the name of the test "FIFFT," is an acronym for "Forward/ Inverse Fast Fourier Test".

The program is divided into two sections. The first section uses simple data sets that consist of only one non-zero point with all the remaining points set to zero. The second section utilizes a pseudo-random number generator to produce data sets. After the forward/inverse FFT process the pseudo-random number generator is restarted to recreate the input data set for comparison against the result of the process.

(INTENTIONALLY BLANK)

SECTION 2
RDS 500 OPERATING INSTRUCTIONS

2.1 RAYTHEON RTOS

Start the program with;

```
:QU, FIFFT
:EX
```

The program will indicate its readiness to accept user commands by typing an "*". Typical user response at this point is to type "RWE (CR) ". The program will then begin to cycle through the test and will type a 'F' for every 64 complete passes through the test. If an error is encountered, the program will type out the error and a code number. Refer to Section IV for a complete description of the error outputs and their interpretation. Given the user input of "RWE (CR) ", the program will type and "*" following the error and then wait for input from the keyboard. At this point the user can cause the test to loop on this error if he wishes to isolate the problem below the Board level using an oscilloscope. The following section describes the user input commands.

Note:

Since FIFFT utilizes the FORTRAN compatible output of APLINK, it is actually called from a small FORTRAN mainline.

```
CALL FIFFTF
END
```

and is entered at FIFFTF which is a FORTRAN compatible entry point.

2.2 LOADING INSTRUCTIONS

The following routines must be extended into the Disk:

FIFFTF	FORTRAN compatible Driver
INPT	FPS supplied package of Teletype Control Routines
FIFT	AP-120B micro-code
APEXEC	AP FORTRAN Executive.

To get the complete package linked together, run the above FORTRAN mainline with an ":FG" command. When the system responds with "REDY", type ":GO". In response to the "NAME?" query type "FIFFT". An absolute copy of the program is now available on the disk and can be run as described in Section 2.1.

(INTENTIONALLY BLANK)

SECTION 3
USER COMMANDS

3.1 USER COMMANDS

The test program responds to a set of single letter commands, some of which are to be followed by one or two Hexadecimal integers. A string of commands may be typed on a line terminated by a carriage return. With the exception of the 'E' (Execute) command, the commands can be typed in any order on the line. The 'E' command should be typed last since commands following the 'E' will not be seen by the command string interpreter. Typing a Control C "↑C" will cause the current line to be ignored. Sense Switch Ø is used to interrupt test program execution and bring control back to the command input portion of the test program. The "Rubout" function is more specialized than in RTOS Teletype input as it can be used to delete only certain commands. Table 3-1 summarizes the commands recognized by this test program.

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
Annnn	Input the starting address of another program in core to which control is to be transferred by the 'B' command.	Must be followed by a Hexadecimal integer specifying the desired absolute address.
B	Transfer program control to the address specified in the last preceding 'A' command.	
C	Type out the user input flags that have been set.	
D	Set the typeout Disable flag. Used for Scoping a hardware fault. Also disables the error detection.	
E	Execute the test program. Used to transfer control from the command input section of the program to the test section.	This is the last command on the line that will be seen by the Command String Interpreter.
H	Set the unconditional Halt flag. Following the next 'E' command the program will execute one test case and return to command Input mode.	
I	Set the IO Reset flag. Test program Resets the AP-120B between each test case.	
L	Set the LOOP flag. Program loops on the last executed test case.	
Mnnnn	Define the AP-120B Memory Size.	Must be followed by a hexadecimal integer (2000=8K)
R	Reset all flags. Clears D,H,I,L,S and W.	
S	Set short form format for error typeout. Affects FIFFT only.	

USER COMMAND	FUNCTION	RESTRICTIONS
W	Set Wait on error flag. If set, the test will return to command input mode after encountering and typing out an error.	
Xnnnn,nnnn	Reset the Random Number generator parameters. Used to recreate a test case from the parameters typed out in an error message.	Must be followed by two hexadecimal integers on the same line.
Rubout	Used to selectively clear one of the flags (D,H,I,L,S, or W). Rubout echoes as a slash "/".	Type the flag to be cleared followed by a rubout i.e., "LRubout" clears the Loop flag.
↑C	Used to delete an input command string if the "↑C" is typed before the Carriage Return.	

User Command Functions.

; Type Absolute address of Calling Routing. Used to locate the Calling program in the absence of a load map.

.nnnn Set Base address. The Debugger has a Base address register that allows the user to reference locations in his relocatable program using the output of the SYMII Assembler. Following the ";" command the user should compute the absolute beginning address of his program and input it to the debugger using the "." Command.

Example:

```
*; (CR)
6CAA
*.6CA5 (CR)
*
```

Explanation:

Call to Debugger was at 6CAA. This is the absolute address of the next instruction following the call. Assuming this instruction was at relative location 5, the user subtracts 5 from 6CAA and enters, 6CA5 as the Base address. (Astericks are typed by program)

Onnnn Open the location nnnn + Base. The debugger will type out the contents (in hexadecimal) of the opened location.

To modify this location, the user simply types the desired hexadecimal value followed by one of the three pointer movement commands.

To leave it unchanged and open contiguous locations or return to command input, the user simply types one of the three pointer movement commands (see below).

Pointer Movement Commands:

(LF) (line feed) closes the currently open location and advances the pointer to the next location in memory. The debugger will type out the address and then the contents of the next location.

↑(CR) (up arrow, carriage return) closes the currently open location and moves the pointer to the preceding memory location. The location and contents are typed out in hexadecimal.

(CR) (carriage return) closes the currently open location and returns to command input mode. The program will type out an "*".

Example:

```
*025 (CR)               OPEN location 25
1234 123 (LF)           change to 123, go to 26
ØØ26 45AB  (LF)         no change to ABF, go to 26
ØØ27 1564 ABF↑(CR)      change to ABF, go to 26
ØØ26 45AB FF (CR)       change to FF, Quit
```

Address Calculation Commands:

=nnnn Adds nnnn to Base and types out result. Converts relative addresses to absolute.

-nnnn Subtracts Base from nnnn. Converts absolute addresses to relative.

Program Control Commands

Tnnnn Trap. Sets a Breakpoint trap at relative location nnnn. The trap consists of two instructions.

```
SMB TRTN
JMP TRTN
```

These two instructions are inserted in the user program at the time that the "G" (GO) command is issued. When the trap is encountered, the routine TRTN will save the ACR and IXR, restore the two user instructions and return to command input mode. Once encountered the TRAP is removed and will not be set again until the user issues another 'T' command.

Gnnnn GO. Starts the user program running at relative location nnnn after inserting a TRAP (if previously called for by user) and restoring the ACR and IXR.

P Proceed. Starts the user program running from the last TRAP location. ACR and IXR are restored. The overflow and compare flops are lost.

Example:

```
1)  *T45 (CR)                            Set TRAP at 45
2)  *G25 (CR)                            Start at 25
3)  TRAP 65AD ACR 8ØØØ IXR ØØ37 Prog. Encounters TRAP
4)  *P (CR)                              Proceed from 45
```

At line 3 where the trap was encountered, the program types out the absolute Hexadecimal location of the trap and the contents of the ACR and the IXR.

NOTES:

1) Hexadecimal integers may consist of from 1 to n digits terminated by a non-numeric character. This character may be a comma, a carriage return or the next command letter (other than A,B,C,D, or E). If more than 4 digits are typed, only the last 4 will be taken as the desired hexadecimal integer.

EXAMPLES:

1) For Normal operation type "RWE (CR) ". This starts the test running so that it will return to command input mode when an error is encountered.

B) Looping on an error:
   1. After program has typed out an error and returning to command mode, Type "LE (CR) " to loop on the failed case to see if the error is solid. Type "LDE (CR) " to go into a scope loop with typeout disabled.
   2. After the error is corrected lift SSØ to interrupt the test. Type "D Rubout E (CR) " to check to make sure that it has been corrected.
   3. To Return to the full test, lift SSØ to interrupt and type "RWE (CR) " to proceed.

C) For an overnight Run type "RE (CR) ". The program will type out all errors and proceed. This will leave a record of any failures that may have occured. The program will stop the test automatically if more than 64 errors occur. This is done so as to prevent excessive wear on the Teletype in the case of a catastrophic failure.

SECTION 4
DETAILED DESCRIPTION OF TESTS AND ERROR MESSAGE

The impluse and the random FIFFT tests operate by supplying the AP-120B with 28-bit integers through the 32-bit integer, program control to DMA path of the AP-120B interface. An AP-120B micro-program converts these integers to floating point, rearranges the array in bit-reversed order, performs the forward Real FFT, again rearranges the array in bit-reversed order, performs the inverse FFT, and finally converts the array back to 28-bit integers. The test program then uses the 32-bit integer, DMA to program control, interface path to read the results back for comparison.

The first (impluse) test begins with the smallest FFT (8 points) and works its way up sequentially to the largest FFT size that is compatible with the memory configuration (8K Real FFT in 8K of MD). For each size FFT it slides an impluse (Ø4ØØ ØØØØ) across the data set. For the small FFT's (<256 points), the impulse is placed in sequence on every real and complex point of the input data set and the data set is tested. For the larger FFT's, the impulse is moved through the data set in such a manner as to skip larger and larger numbers of points as the size of the data set is increased while still testing real and complex points in an alternating fashion. This is done in order to allow the impulse test to complete in a finite length of time (typically less than 5 minutes for a maximum FFT size of 8K). Upon completion of the first test, the program

"END IMPULSE TEST."

The second portion of the test utilizes a very long-sequence, pseudo-random number generator to select the desired data set size and then to generate the set of 28-bit integers. These numbers are acted upon by the AP-120B in exactly the same fashion as in the first test. The test program then restarts the pseudo-random number generator and uses it to recreate the input data set for comparison against the AP-120B output. For every 64 successful random Forward/Inverse FFT's, the program types;

"F"

For a maximum Real FFT size of 8K (8K MD; 2K TM Roots of unity), the program sets an error limit of 4 in the least significant bits of the 28-bit mantissa for the impulse test. For the random FFT test the error limit is m+5 where m is the power of 2 such that $2^m = N$ (N is the number of complex data points).

If the deviation between the input and output is greater than the allowable error limit, then an error message is output in the following format.

```
LOC, SIZE, llll ssss EXP eeee eeee
ACT aaaa aaaa X,Y xxxx yyyy
```

```
where llll is the location in error,
      ssss is the size of the FFT under test
      eeee etc. is the expected result,
      aaaa etc. is the actual result,
  and xxxx yyyy are the restart parameters for the
           pseudo-random number generator.
```

If the short form typeout flag has been set by the user ('S' command), then the error typeout has the following format:

```
LOC,SIZE llll ssss aaaa aaaa.
```

Where the meanings are the same as those above.

SECTION 5
DIAGNOSTIC SUGGESTIONS

The most efficient field technique for isolating problems uncovered by FIFFT and no other diagnostic is to sequentially exchange boards until the problem either disappears, or changes in some fashion indicating that a part of the fault has been isolated. Lacking a set of spare boards, some progress can be made by examining the type of error. The smaller the size of FFT that can be made to fail, the better the chance of tracing the problem. It is sometimes worthwhile to try adjusting the power supply voltages (within ±5% limits) to try to make an intermittent problem more solid. Errors that seem to be address related (bit reversed pairs of points incorrect) can generally be traced to problems in Memory (MD, TM), or the addressing (ADDR, SPAD). Errors that seem to be data related can generally be traced to problems in the arithmetic or accumulators (FA, FM, DPAD). Errors that are voltage related are susceptible to a technique that involves setting the voltage at the border line between correct and incorrect behavior and then tracing the now intermittent bit or bits back in time using an oscilloscope until the first point at which a failure occurs is found. This technique can typically be extended past the board level to isolate the marginal chip in question.

(INTENTIONALLY BLANK)

SECTION 6
AP-120B HARDWARE DEBUG ROUTINES

6.1 DESCRIPTION OF ROUTINES

Incorporated into FIFFT are a set of assembly language routines that can be of general use in debugging micro-code programs using the AP-120B hardware in cases where the micro-program is too lengthy or complex to debug using APSIM and APDBG. The debug routines in FIFFT allow the user to start the AP-120B, set breakpoints and to examine and in some cases modify the contents of AP-120B internal registers and memories.

The routines all rely on the command input section of FIFFT (entry INPT) for their teletype input/output. They are started with the 'G' command and their results and input parameters can be examined and modified using the open (Onnnn) Command. These commands were described in Section 3.

For example, to deposit a value into MD, the user would first open the location LOC and set it to the desired MD address, he would then place the 3-word value into the first three words of VAL and start the DPMD routine running using the 'Gnnnn' command.

Example:

To set MD location 25 equal to 12, 123, 1234. Where LOC is at relative location 387, VAL is at 388, DPMD is at 3DC and the base address of FIFFT is 6800.

```
*.68ØØ  CR                          Set base address
*0387  CR                           Open 387
ØØØØ 25 LF                          Set LOC = 25
Ø388 ØØØØ 12  LF                    Set VAL = 12
Ø389 ØØØØ 123  LF                       VAL+1 = 123
Ø38A ØØØØ 1234  CR                      VAL+2 = 1234
*G3DC  CR                           Run DPMD
*
```

Description of Routines:

ENTRY POINT	FUNCTION	SECONDARY LOCATIONS
RUNIT	Runs the AP-120B starting at SADR with Breakpoint at BP. If a Breakpoint is not desired be sure that BP is outside of the user microcode.	SADR = Start BP = Breakpoint To be set on PSA

ENTRY POINT	FUNCTION	SECONDARY LOCATION
CNTU	Continue from a hardware Breakpoint with BP as the Breakpoint on PSA. See the notes below for the recommended useage of the hardware breakpoint.	BP = Breakpoint
DREG	Dumps the AP-120B interface registers on the TTY. The register contents are typed out in the following order: STATUS, SR, FN, LITES, WC, HMA, CTRL, APMA (LITES = PDR)	
DMADR	Dumps the internal registers of the AP-120B into RDS-500 memory starting at location DADRS in the following order: PSA, SPD, MA, TMA, DPA, SPFN, APSTATUS, DA	DADRS = first location into which registers are stored.
DSPAD	Dumps S-PAD into RDS-500 memory starting at SPAD.. Note: SP(Ø) will always return as a zero. The user should not attempt to continue from a breakpoint after using DSPAD.	SPAD. = first location into which SPAD registers are stored. STMP= contents of SPA
DDPAD	Dumps DATA PAD into memory starting at DPAD.. DPX occupies DPAD. to DPAD.+X'3F'. DPY occupies DPAD.+X '40' to DPAD.+X'FF'. Each DATA PAD register occupies the first three words of a 4-word block. This is done so that the hexadecimal displacementt from DPAD. To a given register is easier to calculate.	DPAD. = first location into which DATA PAD register are dumped. DTMP = contents of DPA

ENTRY POINT	FUNCTION	SECONDARY LOCATION
	Note: Be sure to examine DTMP (or DADRS+4 if DMADR has been called) in order to be able to convert the indices in the user micro-code into absolute DATA PAD addresses.	
EXMD	Examine MD locationx specified by LOC and place the contents in memory starting at VAL in the order: EXPONENT, HMANTISSA, LOWMANTISSA The contents are also typed out on the teletype in the above order.	LOC=location VAL=contents
EXPS	Examine the PS location specified by LOC, place the contents in VAL and type out the contents on the TTY.	LOC=location VAL=contents
DPMD	Deposit the contents of VAL (first three words) into the MD location specified by LOC.	LOC=location VAL=contents
DPPS	Deposit the contents of VAL (four words) into the PS location specified by LOC.	LOC=location VAL=contents

Notes on the use of the hardware panel and breakpoint.

1) Where does the AP stop on a breakpoint?

   a) With the BP set on PSA, the AP-120B will stop with PSA pointing to the next instruction.

      Thus breaking on a branch instruction and then examining PSA will show whether the branch condition was true or false (PSA = BP+DISP or PSA = BP+1)

   b) With the BP set on TMA the AP-120B will stop with PSA pointing to the second instruction following the one that set TMA = BP.

   c) With the BP set on MA the AP-120B will stop on either the next instruction or the second instruction after the one that set MA=BP depending on the state of the memory lock-out hardware.

      (next instruction if memory lockout, second instruction if no memory lockout)

      Thus the stopping point following an MA breakpoint will have a one instruction uncertainty.

2) Does the instruction on which the AP stops execute?

   a) Since SPFN is current, it will be set to the operation specified in the instruction that PSA is pointing to. The SP_{SPD} register will not be modified unless the user changes SPD. Thus if the user wishes to proceed from a breakpoint, he must do one of two things.

      1) Be sure that the instruction following the one on which the breakpoint is set does not do an S-PAD operation or does an S-PAD operation such as MOV 14,14 or MOV 14,15. (This last instruction is safe since it does not hurt to re-execute it).

      or 2) Be sure not be change SPD while the AP-120B is stopped. Changing SPD will cause the S-PAD operation to execute. It will be executed again if the user attempts to proceed.

      Note that a debug routine that examines all of S-PAD will perforce change SPD and thus make proceeding impossible, unless condition 1 above is satisfied.

   b) All other portions of the instruction that PSA is pointing to remain unexecuted and will execute correctly when the user steps or proceeds from the breakpoint.

3) What about MD timing and lockout on a breakpoint in the middle of an MD memory cycle?

   a) The hardware has been designed so that the AP can be stopped in the middle of a memory cycle. The hardware remembers where the memory timing was when the AP stopped so that the micro-processor can continue correctly from a breakpoint that occurs during a memory cycle.

   b) However, the user must not examine MD nor should there be any DMA transfers going to or from MD while the AP is stopped if the user wishes to proceed from the breakpoint.

   Thus, for example, it is possible to break in the tight-to-memory portions of the FFT and examine DATA PAD or the address registers (PSA, SPA etc) and then proceed. But it is not possible to proceed if the user or the host interface disturbs the memory timing by reading or writing MD.

4) Summary of rules for proceeding from a breakpoint.

   a) S-PAD. The instruction that PSA is pointing to, typically the instruction following the one on which the breakpoint is set, should be an SPAD NOP or an S-PAD instruction that can stand to be re-executed. OR, the user should not change SPD while the AP is stopped.

   b) MD

   If the breakpoint causes the AP to stop in the middle of a memory cycle. (PSA pointing to first or second instruction following SETMA, INCMA or DECMA), the user should not try to examine or modify MD.

5) What about stepping the AP?

   The same rules as for proceeding from a breakpoint apply to stepping the AP through a micro-program. The user can examine and modify any register or memory within the constraints mentioned in #4 a and b above.

6) What other pitfalls are there in the use of the virtual front panel?

   a) Note that the panel always examines SPFN not SP_{SPD}.

   Thus if the user wishes to see SP_{SPD} he must force $SPFN = SP_{SPD}$. This can most easily be done via the panel reset function which has the unhappy side effect of also clearing SP(Ø).

b) To examine TM, the user should first set TMA and then do a dummy panel operation (deposit TMA again for example) in order to enter the output of table memory into the table memory buffer register. He can then proceed to examine the addressed location using the appropriate panel functions.

c) MD

Setting MA from the panel initiates an MD memory read cycle. Depositing into MD from the panel initiates an MD memory write cycle.

Thus to write MD and then examine what was just written, the user must perform a deposit into MA operation (with the same address) to initiate a read cycle before using the examine MD commands.

d) Using the Increment field in the FN register.

DPA and TMA always increment after the EXAM or DEP operation is complete (remember that TMA is used to address program source memory for panel operations).

MA post-increments and initiates a new memory read cycle on an EXAM operation.

MA pre-increments on a DEP operation.

e) Starting the AP

The recommended starting procedure is as follows.

i) set the SR to the starting address and do a deposit into PSA

ii) set SR to the desired breakpoint and do a continue to start the AP-120B.

This proceedure has the significant advantage that it places the necessary breakpoint code into the users program should he need to debug his micro-program.

The panel START function can be used but the user should observe the following restriction on the first instruction executed by the AP-120B.

The first instruction should not branch or jump or modify PSA in any way other that to advance to the next instruction. The first instruction should not use the SPEC or IO fields. In fact the preferred first instruction is a NOP (all zeros).

APPENDIX A
Board Number vs. Board Name Glossary

200L	DPL	Data Pad Left - EXP and LMAN (left byte)
200R	DPR	Data Pad Right - HMAN and LMAN (right byte)
201	SPAD	SPAD Registers and ALU, MA Register and PS input Buffer.
202	MDREG	Main Data output Buffer Register and DPBS to/from PNLBS Buffers
203	FADD1	Floating Adder Board 1 Mantissa shift and ALU
204	FADD2	Floating Adder Mantissa normalize and round
205	FADD3	Floating Adder exponent arithmetic
206	FMULA	Floating Multiply Mantissa input register M1 and multiplier array A
207	FMULB	Floating Multiply Mantissa input register M2 and multiplier array B
208	FMULC	Floating Multiply exponent arithmetic and Mantissa partial product sum, normalize and round
209	TMREG	Table Memory output Buffer and 2's Complement logic for Cosine table unravel in FFT
210	CB1	Control Buffer, Board 1. Buffer/Decode Bits Ø to 13 and 23 to 31 of micro-instruction
211	CB2	Control Buffer Board 2. Timing for Main Data memory, APSTATUS Register, and Buffer/Decode Bits 32 to 63 of micro-instruction
212	ADDR	Next Instruction Address logic for micro-processor, Subroutine Return Stack, Table Memory Address and clock generation
213	MIREG	Main Data Input Buffer Register
214	EXPAN	AP-120B Panel (APSR, FN, LITES) and Buffer/Decode Bits 14 to 22 of micro-instruction
215	MDATA	Main Data Memory card 8K x 38 bits
216	PSRAM	Program source memory card 512 x 64 bits
217	TMROM	Table Memory ROM card 8K x 38 bits
228	RDSI/F	RDS 500 Interface
227	FMTE	Format Conversion card (Exponent).
226	FMTM	Format Conversion Card (Mantissa)

(INTENTIONALLY BLANK)

APPENDIX B

AP-120B BACKPLANE SIGNAL GLOSSARY

!ALCLK	Clock for A1 Register of FA
!A2CLK	Clock for A2 Register of FA
!CBCLK1	Clock for Bd. 210, CB-1
!CBCLK2	Clock for Bd. 211, BC-2
!DPLCLK	Clock for Bd. 200L DPAD
!DPLCLK*	Inverted clock for Bd. 200L DPAD
!DPRCLK	Clock for Bd. 200R DPAD
!DPRCLK*	Inverted clock for Bd. 200R DPAD
!FACLK2	Clock for Bd. 204, FADD2
!FACLK3	Clock for Bd. 205, FADD3
!FMCLKA	Clock for Bd. 206, FMULA
!FMCLKB	Clock for Bd. 207, FMULB
!FMCLKC	Clock for Bd. 208, FMULC
!IOCLK	Clock for Interface Bd
!M1CLK	Clock for M1 Register of FM
!M2CLK	Clock for M2 Register of FM
!MCLK	Clock for Main Data Memory Cards
!MDCLK*	Inverted Clock for MDREG Bd. 202
!MICLK	Clock for MIREG Bd. 213
!PNLCLK	Clock for Panel Logic on EXPAN, Bd. 214
!SPCLK	Clock for SPAD, Bd. 201
!T69	Clock delayed by 69ns for MD Timing Bd. 210
!TMCLK	Clock for TMREG Bd. 209
!WRT*	Low true write pulse used to write PS, and the Subrouting Return Stack

AP-120B BACKPLANE SIGNAL GLOSSARY - Continued

!WRTL*	Write pulse for DPL
!WRTR*	Write pulse for DPR
"GND	Extra Grounds not included in the standard set provided by the motherboard
A1CLKD*	A1 Clock Data (low true) causes A1CLK
A1EBSØ2* to A1EBS11*	A1 Exponent Bus
A1MBSØØ* to A1MBS27*	A1 Mantissa Bus
A2CLKD*	A2 Clock Data causes A2CLK
A2EBSØ2* to A2EBS11*	A2 Exponent Bus
A2MØØQ*	A2 Register Bit ØØ (sign bit)
A2MBSØØ* to A2MBS27*	A2 Mantissa Bus
ABORT*	Internal System Reset Line
BØCLK	Byte Ø Clock to FMT Bd.
B1CLK	Byte 1 Clock to FMT Bd.
B2CLK	Byte 2 Clock to FMT Bd.
B3CLK	Byte 3 Clock to FMT Bd.
B2IO	FMT Buffer to I/O Bus Enable
BH2HD	FMT Buffer High to Host Data Enable
BL2HD	FMT Buffer Low to Host Data Enable
BS2A1	A1BS to A1 input select line
BS2A2	A2BS to A2 input select line
BS2M1	M1BS to M1 select line
BS2M2	M2BS to M2 select line
BUF2CLK	FMT BUFFER #2 Clock

CAP2PNL*	Control Buffer AP to PNLBUS
CB2A1E*	Control Buffer to A1EBS Enable
CB2A2E*	Control Buffer to A2EBS Enable
CB2A2M*	Control Buffer to A2MBS Enable
CBCLKE*	Control Buffer Clock Enable

(BOTTOM HALF INTENTIONALLY BLANK)

DAØ to DA3	SPAD Destination Address Bits
DALD*	Device Address Load Enable
DEØ7 to DE11	Floating Adder
DEØ7* and DEØ8* DE1ØA and DE11A	Delta Exponent Bits for shift of mantissa of smaller argument
DECIMATE*	Bit-Reverse enable to SPAD source.
DMAØØ* to DMA15*	Direct Memory Address to MD from Host Interface
DMASAME	DMA Bank Address same as last bank
DP2A1E	DPAD to A1BS Enable
DP2A2E	DPAD to A2BS Enable
DP2DPE	DPAD to DPBS Enable
DP2M1E	DPAD to M1BS Enable
DP2M2E	DPAD to M2BS Enable
DPA2PNL*	DPAD Address to Panel Bus Enable
DPALD*A	DPAD Address Load Enable
DPBS2PSI*	DPBS to Program Source Input Select
DPE2PNL	DPBS Exponent to Panel Bus Enable
DPEBSØ2* to DPEBS11*	DPAD Exponent Bus Bits
DPH2PNL	DPBS HMAN to Panel Bus Enable
DPL2PNL	DPBS LMAN to Panel Bus Enable
DPMBSØØ* to DPMBS27*	DPAD Mantissa Bus
EOVCRY	FA Exponent overflow carry
EOVG*	FA Exponent overflow carry generated

EOVP*	FA Exponent overflow carry propagate
EX2PNL*	Exit to PNLBUS
EXIA	Exit Input Select A
EXIB	Exit Input Select B
EXP*	Exponent Write Select
FAA*	FA answer select A
FAB*	FA answer select B
FACIN*	Floating Adder Carry Input
FADD*	Floating Add microinstruction decode
FAEØØ* to FAE11*	Floating Adder Exponent output
FAMØØ* to FAM27*	Floating Adder Mantissa output
FANEG*	FA result negative
FAOVF*	FA result overflow
FASØ to FAS3	FA ALU mode select controls
FAUNF*	FA result underflow
FAZRO*	FA result = zero
FFTQ	FFT mode flag
FLØ7*A to FLØ9*A FL10* to FL11*	FA normalization shift count (Float number)
FLAGØ to FLAG3	Program selectable Flags
FMEØ2* to FME11*	Floating Multiplier Exponent output
FMMØØ* to FMM27*	FM Mantissa output
FMOVF*	FM result overflow
FMUL*	Floating Multiply micro-instruction decode
FMUL*A	Floating Multiply micro-instruction decode

FMUNF*	FM Result underflow
FN2HD*	Function Register to Host Data Enable
FRSGN*	FA Force Sign
FRSGNQ*	Force Sign Latch
FSCALE*	Floating Scale
FSCALEQ	Floating Scale Latch
FSCALEQ*	Floating Scale Latch
FSM(-1)* to FSM3Ø*	Floating Summer Mantissa bits (Connection from Stage 1 to State 2 of FA)
HDØØ to HD15	Host Data Bus
HD2DP	Host Data to DPBS Enable
HRSET*	Host Reset
I+HØ9	IO OR HOST Data Bit Ø9
I+H10	IO OR HOST Data Bit 1Ø
I+H13	IO OR HOST Data Bit 13
I+H14	IO OR HOST Data Bit 14
IFFTQ*	Inverse FFT Flag
IN	Decode of IO Input instruction
INTEN	Interrupt Enable
INTR*	Interrupt Request
INTRQ	Interrupt Request Latch
IOØØ* to IO39*	IO BUS
IOACK*	IO Acknowledge
IODRDY*	IO Data Ready

IODRDYQ	IO Data Ready Latch
IODRDYQ*	IO Data Ready Latch
IOSPMD*	IO Spin if MD Busy
LT2HD*	LITES to Host Data Enable
M1EBSØ2* to M1EBS11*	M1 Exponent Bus
M1MBSØØ* to M1MBS27*	M1 Mantissa Bus
M1RØØQ* to M1R27Q*	M1 Register outputs
M2EBSØ2* to M2EBS11*	M2 Exponent Bus
M2MBSØØ* to M2MBS27*	M2 Mantissa Bus
M2RØ2Q* to M2R27Q*	M2 Register outputs
MAØØ* to MA15*	Memory Address (MD)
MA2PNL	Memory Address to Panel Bus Enable
MACE*	Memory Address Count Enable
MAINC*	Memory Address Increment Select
MALD*	Memory Address Load Enable
MALD*A	Memory Address Load Enable
MAN*	Mantissa Write Select
MANOV	Mantissa overflow
MASAME	MA Bank same as last Bank
MDØ2* to MD39*	Main Data outputs
MD2A2	MD to A2BS Enable
MD2DP	MD o DPBS Enable
MD2M2	MD to M2BS Enable
MDCAØ	MD Cycle Acknowledge Ø (refresh)
MDCA1	MD Cycle Acknowledge 1 (Host interface)

MDCA2	MD Cycle Acknowledge 2
MDCA3	MD Cycle Acknowledge 3 (AP Internal)
MDCLKE*	MD Register Clock Enable
MDCR1Q*	MD Cycle Request 1
MDCR2*	MD Cycle Request 2
MDCR3*	MD Cycle Request 3
MDEXP	MD Exponent Write Enable
MDHMAN	MD high Mantissa Write Enable
MDIØ2 to MDI39	MD input bus
MDINA*	MD cycle initiate
MDLMAN	MD Low Mantissa Write Enable
MDWRT*	MD Write Enable
MDWRT3	MD Write Request 3
MIA	MD Input Select A
MIB	MD Input Select B
MICLKE*	MD Input Clock Enable
OUT*	IO OUT micro-instruction decode
OVFL*	Overflow status
PCYL1*	Panel cycle 1
PCYL2*	Panel cycle 2
PNLØØ* to PNL15*	Panel Bus
PNL2DP	Panel Bus to DPBS Enable
PNL2HOST*	Panel Bus to Host (Lites Load Enable)
PNL2MD*	Panel Bus to MD write request
PPAØ1* to PPA26* PPA27Q* to PPA3ØQ* PPA31* to PPA52*	FM Partial Product outputs of Array A

PPAUSE	Panel Pause from CB-1 (210)
PPB(-1)* to PPB24* PPB25Q* to PPB28Q* PPB29* to PPB	FM Partial Product outputs of ARRAY B
PSØØ* to PS63*	Program Source outputs
PSØ2PNL*	PS Word Ø to Panel Bus Enable
PSØWRT	PS Word Ø Write Strobe
PS12PNL*	PS Word 1 to PNL Bus Enable
PS1WRT	PS Word 1 Write Strobe
PS22PNL*	PS Word 2 to PNL Bus Enable
PS2WRT	PS Word 2 Write Strobe
PS32PNL*	PS Word 3 to PNL Bus Enable
PS3WRT	PS Word 3 write Strobe
PSAØ4* to PSA15*	Program Source Address
PSA2PNL	PSA to PNL Bus Enable
PSAAD	PSA Select A Data
PSABD	PSA Select B Data
PSACD	PSA Select C Data
PSACLKE*	PSA Clock Enable
PSAZRO	PSA = Zero, PS Disable
PSH2DP*	PS High to DPBS Enable
PSIØØ to PSI31	PS Input Bus
PSL2DP	PS Low to DPBS Enable
REFSYNC*	Refresh Sync
RUN*	AP-120B Running
S+C2*	Step OR Continue Cycle 2 (panel function)

SAØ to SA3	SPAD Source Address
SAMEX	DPX Read and Write Addresses equal
SAMEY	DPY Read and Write Addresses equal
SCØØ*	Sign bit out of FA Stage 1 mantissa scaler
SCIN	FA Scaler inhibit
SELA1A	Select A1 as larger input to FA
SELA2	Select A2 as larger input to FA
SFWE*	SPAD Function Write Enable
SHSØ	SPFN Shift Select Ø
SHS1	SPFN Shift Select 1
SIWE*	SPAD Input Write Enable
SM2TC*	Sign Magnitude to two's complement
SNSA	IO sense A
SP+DPØØ* to SP+DP15*	SPFN OR DPBS Bus
SP2ADDR	SPFN to Address (SP + DP Bus) select
SP2DP	SPFN to DPBS Enable
SP2PNL	SPFN to PNL Bus Enable
SPA2PNL*	SPAD Address to PNL Bus Enable
SPALD*	SPAD Destination Address Load Enable
SPCIN	SPFN Carry Input
SPFNØØ*	SPFN Sign Bit
SPFNCRY*	SPFN Carry Output
SPILD*	SPAD INPUT LOAD Enable
SPIN*	Micro-processor SPIN (hangs on current instruction)

SPIODØ	SPIN if IODRDY DATA=Ø
SPM	SPFN Mode
SPSØ to SPS3	SPFN ALU Controls
SPZED	SPFN = zero
SR2HD*	Switch Register to Host Data Enable
SR2PNL	Switch Register to Panel Bus Enable
SRACE*	Subroutine Return Address Count Enable
SRADEC*	Subroutine Return Address Decrement Select
SRAOVD*	Subroutine Return Address Overflow Data
SRSWE*	Subroutine Return Stack Write Enable
STA2PNL	AP STATUS to PNL Bus Enable
STALD*	APSTATUS Load Enable
TMØ2* to TM39*	Table Memory Outputs
TM2A1	TM to A1BS Enable
TM2DP	TM to DPBS Enable
TM2M1	TM to M1BS Enable
TMAØØ to TMA15	Table Memory Address
TMA2PNL	TMA to PNL Bus Enable
TMACE*	TMA Count Enable
TMADEC*	TMA Decrement Select
TMALD*	TMA Load Enable
TMALD*A	TMA Load Enable
TMINH	Table Memory Inhibit
TMNEG*	Table Memory Negate

TRUNC*	FA Truncate
TRUNCQ	FA Truncate Latch
TRUNCQ*	FA Truncate Latch
TSPIN	True Spin
UNFL*	Underflow Status
USECB*	Use Control Buffer Bits 48 to 63 as a Value
USEPSA*	Use PSAQ as Source for PSA
WRTEXP	MD Exponent Write Enable
WRTHM	HMAN Write Enable
WRTLM	LMAN Write Enable
XØ1 to XØ5 XØ2A to XØ5A	DPX Address
XECLKE*	DPX Exponent Clock Enable
XHMCLKE*	DPX HMAN Clock Enable
XIA	DPX Input Select A
XIB	DPX Input Select B
XLMCLKE*	DPX LMAN Clock Enable
YØ1 to YØ5 YØ2A to YØ5A	DPY Address
Y2A1	DPY to A1BS Select
Y2A2	DPY to A2BS Select
Y2DP	DPY to DPBS Select
Y2M1	DPY to M1BS Select
Y2M2	DPY to M2BS Select
YECLKE*	DPY Exponent Clock Enable
YHMCLKE*	DPY HMAN Clock Enable

AP-120B

APARTH

DIAGNOSTIC SOFTWARE MANUAL

Revision 1 1/5/76

SECTION 1
BRIEF DESCRIPTION: APARTH

1.1 BRIEF DESCRIPTION

This program exercises and verifies the accuracy of the arithmetic hardware in the AP-120B (FA, FM and S-PAD). In addition, due to the heavy use of Data Pad and S-Pad registers, the functioning of these registers is thoroughly tested.

The program utilizes a pseudo-random number generator to produce arguments for Data Pad and S-Pad, to select DPA and the Decimate shift count, and also to select combinations of Floating Adder and S-Pad operations, Data Pad Read and Write Indices, and S-Pad register addresses which are combined into micro-instructions for the AP-120B to execute.

The number (1 to 15) of randomly selected instructions is also selected by the Random Number Generator. A new user command ('N') has been provided, however, to override this selection and force the generation of simpler test cases in order to help simplify their interpretation.

After generating the data and instructions, the program loads them into AP-120B Data Pad, S-Pad, DPA, APSTATUS and Program Source and starts the AP-120B executing the test. The program then loads the data into a corresponding set of software simulation registers and calls on the simulation package (SPADS) to generate the expected results.

The program then checks all of S-Pad, Data Pad and the APSTATUS register against the predicted results. If any descrepancies are encountered, the expected and actual results are typed out along with the restart parameters for the Random Number Generator. To help facilitate the interpretation of the results, several new commands have been added to the FPS Teletype Control Routines. Refer to the 'N', '#', and 'F' commands in Section 3. Example error messages and their interpretation can be found in Section 4.

INTENTIONALLY BLANK

SECTION 2
RDS 500 OPERATING PROCEDURES

2.1 RAYTHEON RTOS

Start the program with:

```
:QU, APARTH
:EX
```

The program will indicate its readiness to accept user commands by typing an "*". Typical user response at this point is "RWE (CR) ". The program will then begin generating test cases and trying them on the hardware and software and will type an 'A' for every 512 test cases (approximately once every 30 seconds). The following section describes the full set of user commands.

Note:

This program requires that the FPS supplied package of Teletype Control Routines (INPT) and the simulation package (SPADS) be extended onto the Disk along with APARTH in order for the above Queing sequence to work.

INTENTIONALLY BLANK

SECTION 3
USER COMMANDS

3.1 USER COMMANDS

The test program responds to a set of single letter commands, some of which are to be followed by one or two Hexadecimal integers. A string of commands may be typed on a line terminated by a carriage return. With the exception of the 'E' (Execute) command, the commands can be typed in any order on the line. The 'E' command should be typed last since commands following the 'E' will not be seen by the command string interpreter. Typing a Control C "↑C" will cause the current line to be ignored. Sense Switch Ø is used to interrupt test program execution and bring control back to the command input portion of the test program. The "Rubout" function is more specialized than in RTOS Teletype input as it can be used to delete only certain commands. Table 3-1 summarizes the commands recognized by this test program.

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
Annnn	Input the starting address of another program in core to which control is to be transferred by the 'B' command.	Must be followed by a Hexadecimal integer specifying the desired absolute address.
B	Transfer program control to the address specified in the last preceding 'A' command.	
C	Type out the user input flags that have been set.	
D	Set the typeout Disable flag. Used for Scoping a hardware fault. Also disables the error detection.	
E	Execute the test program. Used to transfer control from the command input section of the program to the test section.	This is the last command on the line that will be seen by the Command String Interpreter.
Fnnnn	Type out the micro-instruction (Function) at PSA location nnnn. PSA location 4 corresponds to the first randomly selected micro-instruction The PSA value will be typed on a line followed by the micro-instruction represented as four hexidecimal numbers. At the end of the line line the user can type either a line-feed to cause the program to type out the next PSA value and micro-instruction or a carriage return to cause the program to return to command input mode.	Must be followed by a hexidecimal integer.

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
H	Set the unconditional Halt flag. Following the next 'E' command the program will execute one test case and return to command Input mode.	
I	Set the IO Reset flag. Test program Resets the AP-120B between each test case.	
L	Set the LOOP flag. Program loops on the last executed test case.	
Mnnnn	Define the AP-120B Memory Size.	Must be followed by a hexadecimal integer (2000=8K0.
Nnnnn	Select the Number of micro-instructions that are to constitute each test case. Once N is set, the Random Number Generater is inhibited from selecting the size of each Test case. This condition will persist until N is cleared by either the 'R' or Rubout command. The 'N' command is used to force the selection of simpler test cases in order to facilitate their interpretation in the case of a solid hardware failure.	Must be followed by a hexadecimal integer in the range 1 to F.
R	Reset all flags. Clears D,H,I,L,N,S and W.	
S	Set short form format for error typeout. Affects FIFFT only.	

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
W	Set Wait on error flag. If set, the test will return to command input mode after encountering and typing out an error.	
Xnnnn,nnnn	Reset the Random Number generator parameters. Used to recreate a test case from the parameters typed out in an error message.	Must be followed by two hexadecimal integers on the same line.
Rubout	Used to selectively clear one of the flags (D,H,I,L,N,S, or W). Rubout echoes as a slash "/".	Type the flag to be cleared followed by a rubout i.e., "LRubout" clears the Loop flag.
↑C	Used to delete an input command string if the "↑C" is typed before the Carriage Return.	

TABLE 3-1

USER COMMAND	FUNCTION	RESTRICTIONS
SSØ	Sense switch zero is used to cause the running or looping test to return to command input mode. If the program is executing correctly, lifting SSØ should cause it to type an asterisk on the teletype.	
SS1	Sense switch one is used to shorten the error typeout in APARTH. It is used to prevent the full error message from being typed our. Only the first error encountered and the restart parameter line will be typed out.	
GR2	RDS-500 General Register 2 is used to select the hardware trigger location. This feature is provided in order to give the technician a convenient means for locating micro-instruction sequences when scoping a hardware problem.	
#nnnn	Type out argument number nnnn. The argument will be typed out as a 16-bit hexidecimal integer. A line feed can be used to cause the program to type out the next sequential argument. A carriage return brings the program back to command input mode. The following table summarizes the corresponce between argument numbers and the various input and output contents of Data Pad and S-Pad.	Must be followed by a hexidecimal integer.

APARTH '#' COMMAND
Argument Number to Register
Correspondence Table

a) Input Arguments

#∅ to #F	SP(∅ to F)
#1∅	DPX (-4)
#13	DPX (-3)
#16	DPX (-2)
#19	DPX (-1)
#1C	DPX (∅)
#1F	DPX (1)
#22	DPX (2)
#25	DPX (3)
#28	DPY (-4)
#2B	DPY (-3)
#2E	DPY (-2)
#31	DPY (-1)
#34	DPY (∅)
#37	DPY (1)
#3A	DPY (2)
#3D	DPY (3)

b) Results of Software simulation

#4∅ to #4F	SP(∅ to F)
#5∅	DPX (-4)
#53	DPX (-3)
#56	DPX (-2)
#59	DPX (-1)
#5C	DPX (∅)
#5F	DPX (1)
#62	DPX (2)
#65	DPX (3)
#68	DPY (-4)
#6B	DPY (-3)
#6E	DPY (-2)
#71	DPY (-1)
#74	DPY (0)
#77	DPY (1)
#7A	DPY (2)
#7D	DPY (3)

c) Result of AP-120B Hardware

#8Ø to #8F	SP(Ø to F)
#9Ø	DPX (-4)
#93	DPX (-3)
#96	DPX (-2)
#99	DPX (-1)
#9C	DPX (Ø)
#9F	DPX (1)
#A2	DPX (2)
#A5	DPX (3)
#A8	DPY (-4)
#AB	DPY (-3)
#AE	DPY (-2)
#B1	DPY (-1)
#B4	DPY (Ø)
#B7	DPY (1)
#BA	DPY (2)
#BD	DPY (3)

SPECIAL RDS 500 DEBUG COMMANDS

User Command Functions.

; Type Absolute address of Calling Routine. Used to locate the Calling program in the absence of a load map.

.nnnn Set Base address. The Debugger has a Base address register that allows the user to reference locations in his relocatable program using the output of the SYMII Assembler. Following the ";" command the user should compute the absolute beginning address of his program and input it to the debugger using the "." Command.

Example:

```
*; (CR)
6CAA
*.6CA5 (CR)
*
```

Explanation:

Call to Debugger was at 6CAA. This is the absolute address of the next instruction following the call. Assuming this instruction was at relative location 5, the user subtracts 5 from 6CAA and enters 6CA5 as the Base address. (Asterisks are typed by program)

Onnnn Open the location nnnn + Base. The debugger will type out the contents (in hexadecimal) of the opened location.

To modify this location, the user simply types the desired hexadecimal value followed by one of the three pointer movement commands.

To leave it unchanged and open contiguous locations or return to command input, the user simply types one of the three pointer movement commands (see below).

Pointer Movement Commands:

(LF) (line feed) closes the currently open location and advances the pointer to the next location in memory. The debugger will type out the address and then the contents of the next location.

↑(CR) (up arrow, carriage return) closes the currently open location and moves the pointer to the preceding memory location. The location and contents are typed out in hexadecimal.

(CR) (carriage return) closes the currently open location and returns to command input mode. The program will type out an "*".

Example:

*025 (CR)	OPEN location 25
1234 123 (LF)	change to 123, go to 26
ØØ26 45AB (LF)	no change, go to 27
ØØ27 1564 ABF↑(CR)	change to ABF, go to 26
ØØ26 45AB FF (CR)	change to FF, Quit

Address Calculation Commands:

=nnnn Adds nnnn to Base and types out result. Converts relative addresses to absolute.

-nnnn Subtracts Base from nnnn. Converts absolute addresses to relative.

Program Control Commands

Tnnnn Trap. Sets a Breakpoint trap at relative location nnnn. The trap consists of two instructions.

```
SMB TRTN
JMP TRTN
```

These two instructions are inserted in the user program at the time that the "G" (GO) command is issued. When the trap is encountered, the routine TRTN will save the ACR and IXR, restore the two user instructions and return to command input mode. Once encountered the TRAP is removed and will not be set again until the user issues another 'T' command.

Gnnnn GO. Starts the user program running at relative location nnnn after inserting a TRAP (if previously called for by user) and restoring the ACR and IXR.

P Proceed. Starts the user program running from the last TRAP location. ACR and IXR are restored. The overflow and compare flops are lost.

Example:

```
1)  *T45 (CR)                         Set TRAP at 45
2)  *G25 (CR)                         Start at 25
3)  TRAP 65AD ACR 8ØØØ IXR ØØ37       Prog. Encounters TRAP
4)  *P (CR)                           Proceed from 45
```

At line 3 where the trap was encountered, the program types out the absolute Hexadecimal location of the trap and the contents of the ACR and the IXR.

NOTES:

1) Hexadecimal integers may consist of from 1 to n digits terminated by a non-numeric character. This character may be a comma, a carriage return or the next command letter (other than A,B,C,D or E). If more than 4 digits are typed, only the last 4 will be taken as the desired hexadecimal integer.

EXAMPLES:

1) For Normal operation type "RWE (CR) ". This starts the test running so that it will return to command input mode when an error is encountered.

B) Looping on an error:
   1. After program has typed out an error and returning to command mode, Type "LE (CR) " to loop on the failed case to see if the error is solid. Type "LDE (CR) " to go into a scope loop with typeout disabled.
   2. After the error is corrected lift SSØ to interrupt the test. Type "D Rubout E (CR) " to check to make sure that it has been corrected.
   3. To Return to the full test, lift SSØ to interrupt and type "RWE (CR) " to proceed.

C) For an overnight Run type "RE (CR) ". The program will type out all errors and proceed. This will leave a record of any failures that may have occurred. The program will stop the test automatically if more than 64 errors occur. This is done so as to prevent excessive wear on the Teletype in the case of a catastrophic failure.

SECTION 4

APARTH Error Message and Error Message Interpretation

4.1 ERROR MESSAGE FORMAT

The full APARTH error message has the following format:

1) APSTATUS E eeee A aaaa
2) SPD ssss E eeee A aaaa
3) DPX dddd E eee eeee eeee A aaaa aaaa aaaa
4) DPY dddd E eeee eeee eeee A aaaa aaaa aaaa
5) N nnnn X,Y xxxx yyyy APSTATUS pppp DPA dddd

Where in general, eeee stands for the expected and aaa stands for the actual result. Line 1) will appear only if there is a discrepancy in the status register. Line 2) appears if there is a discrepancy in an S-Pad Register. In line 2), ssss is the address of the S-Pad Register in error. Line 2) could appear up to 16 times if every S-Pad Register were wrong. Line 3) appears if there is an error in a Data Pad X Register. Line 4) appears for an error in a Data Pad Y Register. In lines 3) and 4), dddd stands for the address of the Data Pad Register in error. Lines 3) and 4) could appear up to 8 times each if all Data Pad Registers were in error. Line 5) appears if any of lines 1) to 4) are present indicating an error. Line 5) contains: nnnn, the size of the test case; xxxx yyyy, the restart parameters for the random number generator; pppp the expected value for the APSTATUS Register (same as eeee in line 1 if line 1 appears); and dddd, the contents of DPA, the Data Pad base address register.

4.2 ERROR MESSAGE INTERPRETATION

4.2.1 Background Information on program operation. The program begins by using the Random Number Generator (GRN) to select the number (ENP) of functions that will constitute the test case. If the user has set the N flag (Bit 6 of STAT) the number of functions is set equal to the user determined parameter "EN". The program then uses the GRN to select input values for S-Pad (16 by 16-Bits) and Data Pad (8 by 38-Bits for DPX and also for DPY).

It then attacks the problem of generating the Random micro-instructions. These instructions are placed into a block following the location CODEP in the listing. In order to insure that the machine is in a known state prior to the execution of the selected instructions, the block of code begins with a four instruction header that is used to fill the Multiplier and Adder pipelines with zeros.

Header Micro-Code:

```
0000    0001    FADD ZERO, ZERO; FMUL TM, MD
        DA00
        0000
        1F00

0001    0001    FADD ZERO,ZERO; FMUL TM, MD
        DA00
        0000
        1F00

0002    0001    FADD ZERO, ZERO; FMUL TM, MD
        DA00
        0000
        1F00

0003    0003    LDAPS; DB ← DECIMATE Count
        8C00
        0400
        000n
```

Where n is set by the program to equal the selected value of the Decimate shift count. Starting at location 4 then the program generates ENP random instructions that consist of randomly selected Decimate, SOP, SH, SPS, SPD, FADD, XR, YR, XW, and YW Fields. The only restrictions being that SPEC, IO, and NOP operations are not generated in the S-PAD and FADD fields. On the second instruction following the first FADD (ie. at location 6), the DPY field is set to DPY(YW) ← FA. On the next instruction DPX is set to DPX(XW) ← FM. The A1, A2, and M1, M2 fields are set to DPY(YR), DPX(XR). Thus the FA and FM operands are the same, DPY → A1, M1 and DPX → A2, M2. This makes the generated operation sequences recursive in that the output of the arithmetic (FA, FM, S-PAD) can become an input argument for a later operation. After the desired number of instructions has been generated, a seven instruction trailer is appended to the code to flush any remaining results out of the FA and FM pipelines and to provide an instruction sequence to set SPFN=SP(SPD) so that all of S-PAD can be examined.

Trailer Micro-Code:

```
n+4     0001    FADD ZERO, ZERO; FMUL TM, MD
        DA00
        0000    XW, YW Randomly selected
        1F00    see below for DPX, DPY

n+5     0000    FMUL TM, MD; DPY (YW) ← FA
        0000
        2000    XW, YW Randomly selected
        1F00    See below for DPX

n+6     0003    HALT; DPX (XW) ← FM
        F000
        C000    XW Randomly selected
        0000
```

```
n+7     0000    NOP
        0000
        0000
        0000

n+8     0303    LDPNL; RSPFN
        9A00
        0000
        0000

n+9     0003    HALT
        F000
        0000
        0000

n+10    0000    NOP
        0000
        0000
        0000
```

Notes:

1) n is the number of randomly selected instructions.

2) At instruction n+4 and n+5 DPX will be DPX(XW) ← FM if there have been three preceding FMUL'S. DPY will be DPY(YW) ← FA if there have been two preceding FADD's.

The AP-120B is started executing this instruction sequence. It stops when it encounters the HALT at location n+6 (PSA will be pointing at n+7 when it is stopped). The program then reads the APSTATUS Register in order to capture the state of the machine as of the last operation. It then uses the Panel Continue function to cause the AP-120B to execute instructions n+8 and n+9. Instruction n+8 has the effect of setting SPFN equal to SP(SPD) so that the program can then proceed to examine S-PAD and Data Pad to optain the hardware reslts.

4.2.2 Error message interpretation. Note that since FM always goes to DPX and FA goes to DPY, errors on DPX are typically caused by the multiplier while errors in DPY are typically caused by the adder.

Typical error message interpretation procedes by using the 'F' command to type out the micro-instructions, examining the micro-instructions to find the last one that wrote into the register in error, counting back Ø, 2, or 3 micro-instructions (Ø if S-Pad 2 if FA, 3 if FM), examining that instruction to find the input registers to the function in question, and then using the '#' command to type out the input numbers. At this point all the information is available. Knowing the location of the failing instruction, the technician can probably go directly to scoping the problem. By setting GR2 to one less than the address of the micro-instruction that failed he will have a convenient pointer for locating it. If the problem is voltage sensitive, this is the prefered course of action. If the problem is solid,however, it may be worthwhile to attempt to carry out the arithmetic by hand, especially if in S-PAD, or FA, in order to be able to localize the problem further.

4.2.3 Examples. The following pages contain a selection of actual error type outs and their interpretation. Note that they are all presented as if the 'W' flag (Wait or error) was set i.e., the program types an asterisk following the error indicating its readiness to accept new commands. The use of the 'F' and '#' commands is illustrated. The AP-120B Instruction Summary and the APSTATUS Register Format are included at the end of the examples in order to help facilitate the interpretation of the micro-instructions.

Example #1   N=1

```
AAAAAAAAA
DPY 0000 E 0284 0479 5AB5 A 0384 0479 5AB1  Error at DPY(+2)
N0001 X,Y  62B0 7EC6 APSTATUS 4002 DPA 001E

UFO
0000 0001  DA00  0000  1F00      User asks for instructions starting at Ø.
0001 0001  DA00  0000  1F00
0002 0001  DA00  0000  1F00      Header Instructions, user types line feed
0003 0003  8C00  0400  0002         to go to next instruction.
0004 A5F8  B400  01FD  7900
0005 0001  DA00  0002  DF00      Randomly selected instruction
0006 0000  0000  2001  FF00      Trailer instructions
0007 0003  F000  C007  C000      FA  DPY(3)
0008 0000  0000  0000  0000
0009 0303  9A00  0000  0000      User typed carriage return to return to
000A 0003  F000  0000  0000         command input mode.
000B 0000  0000  0000  0000
```

Notes:

1) The error is in DPY Register Ø which corresponds to an index of +2 with respect to DPA of IE. Since the Data Pad indices are biased By 4, this would correspond to an assembled index of 6 in the micro-instructions.

2) Instruction 6 is the only one with a non-zero DPY field, but the YW index in that instruction is 7=DPY(3) thus the wrong result is not in the FA output.

3) The error in this case was traced to an addressing problem in Data Pad in which writing into one register occasionally destroyed the contents of another.

Example #2   N=2

```
APSTATUS E 0802 A 1002                          FZ instead of FN
DPY 000C E 0230 0EF9 8AIC A 0000 0000 0000      Error at DPY(-1)
N0002 X,Y 294C 16F4 APSTATUS 0802 DPA 000D
*F4
0004 DB42 3400 01FF 3900
0005 CDD4 1400 00C1 D900       FIX DPX(-1)
0006 0001 DA00 2006 5F00
0007 0000 0000 E001 7F00       FA →DPY(-1)

0008 0003 F000 C003 4000
*#19
0231                           contents of DPX(-1)
0F7C
C50E
*
```

Notes:

1) The user here has only asked to see the relevant instructions ignoring the header and the last 5 instructions of the trailer.

2) Observe that the error in the APSTATUS Register (FZ set instead of FN) is compatible with the descrepancy in the DPY Register.

3) The interpretation process starts at the end of the code block looking for a YW of 3=DPY(-1) with a 2 in a DPY field. This is first encountered at instruction #7. The user then counts back two instructions to find the operation that failed, FIX DPX(-1). He then uses the table given in Section 3 for the '#' to type out the input argument.

4) This problem was traced to a missing pullup Resistor on A1EBSØ3* that caused the FIX function to shift the wrong argument.

Example #3 N=9

```
SPD 0000 E 73CC A 73C4
N 0009 X,Y CF8B 68AE APSTATUS C906          ERROR in SP(Ø)

*F4
0004 EFIE B400 00C2 3900

0005 6F7F 3400 0154 5900
0006 7441 3400 2046 1900                    EQVL 1,Ø
0007 FD60 2400 E198 3900
0008 0AE5 B400 EI4A F900
0009 ACC6 3400 E15B 1900

000A B308 1400 E1CD D900
000B 4DAA B400 E09F 1900
000C 6CAC B400 E1D1 D900
000D 0001 DA00 E005 1F00
000E 0000 0000 E005 1F00
000F 0003 F000 C002 A000
*#0
FA18
*#1
3C01
*
```

Notes:

1) Working backwards from the end, instruction #6 is found to be the first one to use SP(Ø) as a destination. The input arguments are shown below the micro-code.

2) Examination of this case with an oscilloscope showed that instruction #6 was executing correctly and thus that the problem in fact was address related. Termination of the S-Pad address lines solved it.

```
Example #4    N determined by GRN.
*SA4E3,632HE

DPY 0016 E 03FF 07FF FFFF A 03FF 0400 010F       error at DPY(3)
N 000B X,Y A4E3 0632 APSTATUS 9100 DPA 0013

*F4
0004 0EE4 5400 0099 1900
0005 7966 B400 014B 1900
0006 6A48 7400 205C B900
0007 DF89 B400 E1CE D900
0008 D928 2400 E1A0 7900
0009 572D 3400 E1D2 9900                FEQV DPX(-2), DPY(Ø)
000A 598C 6400 E063 3900
000B E050 4400 E155 5900                FA →DPY(3)
000C EB72 3400 E0A7 1900
000D 7AF4 B400 E059 3900
000E 8ED6 B400 E06B F900
000F 0001 DA00 E007 3F00
0010 0000 0000 E007 1F00
0011 0003 F000 C004 C000
0012 0000 0000 0000 0000
0013 0303 9A00 0000 0000
0014 0003 F000 0000 0000
*#74
03D6      DPY(Ø) Software Result

0B9E
18F4
*#56      DPX(-2) Software Result
03FF
0800
0000
*
```

Notes:

1) Working backwards instruction E is the last to write DPY(3).

2) The operation if FEQV DPX(-2), DPY(Ø).

3) Observe that instruction 9 modified DPX(-2) and instruction 9 also modifies DPY (Ø) and that DPX(-2) and DPY(Ø) remain unmodified to the end. Thus the input arguments for instruction C were taken from the software results rather than from the input argument list.

4) This error turned out to be due to a noise problem in the floating adder that was interfering with the overflow detection.

```
Example #5                N selected by GRN.

DPX 0018 E 0000 0000 0000 A 01D3 0000 0000        ERROR in DPX(-3)
N 000B X,Y B67F 078E APSTATUS  9000 DPA 001B

*F4
0004 E644 2400 0169 F900
0005 5DA6 B400 00FB 1900
0006 5968 6400 20EC B900

0007 D989 B400 E13E D900

0008 DE08 2400 E1F0 7900
0009 66ED 3400 E102 9900
000A 742F 3400 E074 5900              FMUL DPX(-3), DPY(2)
000B 06D0 7400 E045 9900
000C 2BD2 3400 E077 3900              FM →DPX(-3)
000D B634 B400 E109 7900
000E D4F6 B400 E1FB 3900
000F 0001 DA00 E007 7F00

0010 0000 0000 E007 7F00
0011 0003 F000 C004 0000
*#13                                  Input argument for DPX(-3)
03D4

00AF

E5D1
*#7A
0000                                  Software result for DPY (2)
0000
0000
*
```

Notes:

1) Instruction A uses DPX(-3), DPY(2)
   Instruction D destroys DPX(-3)
   Thus the input argument for DPX(-3) was examined.
   Instruction 9 modifies DPY(2)
   Thus the software result table is used to display DPY(2).

2) Not all cases are so fortuitous. In some cases an argument can be completely masked by preceding and succeeding instructions. In these cases the user can try regenerating the case with N forced to the minimum number of instructions necessary to get up to the failing instruction. Thus in this example the user would set N to 7 in order to recreate the sequence up to instruction A. The following command sequence would have the desired effect.

```
*N7                                       User sets N to 7
*XB67F,78E                                User enters RNG Restart parameters
*HE
*F4

0004 E644 2400 0169 F900                  Regenerated Case with N=7
0005 5DA6 B400 00FB 1900
0006 5968 6400 20EC B900

0007 D989 B400 E13E D900
0008 DE08 2400 E1F0 7900
0009 66ED 3400 E102 9900
000A 742F 3400 E074 5900                  FMUL DPX(-3), DPY(2)
000B 0001 DA00 E000 1F00
000C 0000 0000 E000 7F00
000D 0003 F000 C005 A000                  FM →DPX(1)
*
```

Note;

The error should now appear in Register DPX(1) and that the failing instruction is now the last "Real" one to be executed. In extreme cases it may be necessary to actually go into the code buffer (using the APARTH listing) in order to modify XW and YW in some of the instructions following the failing on one in order to be able to see the input arguments. In general, the user should try to attack the simplest possible cases first (by forcing N to a small number) before attempting to work on the larger ones.

INTENTIONALLY BLANK

AP-120B Instruction Field Layout

0	1 2 3	4 5	6 7 8 9	10 11 12 13	14 15 16	17 18 19	20 21 22	23 24 25 26	27 28 29 30 31
D	SOP	SH	SPS	SPD	FADD	A1	A2	COND	DISP
S-Pad Group					Adder Group			Branch Group	
			SOP1			FADD1			
			SPEC OPER			I/O			

32 33	34 35	36 37 38	39 40 41	42 43 44	45 46 47	48 49 50	51	52 53	54 55	56 57	58 59	60 61	62 63
DPX	DPY	DPBS	XR	YR	XW	YW	FM	M1	M2	MI	MA	DPA	TMA
Data Pad Group							Multiply Group			Memory Group			
						VALUE							

AP-120B Instruction Summary

Unconditional Fields

Each of the following fields may be used in any given instruction word.

al

e Field Name

D	SOP	SOP1	SH	SPS	SPD	FADD	FADD1	A1	A2	Octal Code
NOP	SOP1	NOP	NOP	(S-Pad	(S-Pad	FADD1	NOP	NC	NC	Ø
&	SPEC	WRTEXP	L	Source	Dest.	FSUBR	FIX	FM	FA	1
	ADD	WRTHMN	RR	Reg.)	Reg.)	FSUB	FIXT	DPX	DPX	2
	SUB	WRTLMN	R			FADD	FSCALE	DPY	DPY	3
	MOV	NOP		(0-17)	(0-17)	FEQV	FSM2C	TM	MD	4
	AND	NOP				FAND	F2CSM	ZERO	ZERO	5
	OR	NOP				FOR	NOP	ZERO	MDPX	6
	EQV	NOP				IO	FABS	ZERO	EDPX	7
		CLR								10
		INC								11
		DEC								12
		COM								13
		LDSPNL								14
		LDSPE								15
		LDSPI								16
		LDSPT								17

al

e Field Name

COND	DISP	DPX	DPY	DPBS	XR	YR	XW	YW	FM	Octal Code
NOP	(Branch	NOP	NOP	ZERO	(DPX	(DPY	(DPX	(DPY	NOP	Ø
#	Displa-	DB	DB	INBS	Read	read	Write	Write	FMUL	1
BR	cement)	FA	FA	VALUE*	Index)	Index)	Index)	Index)		2
BINTRQ	(0-37)	FM	FM	DPX						3
BION				DPY	(0-7)	(0-7)	(0-7)	(0-7)		4
BIOZ				MD						5
BFPE				SPEN						6
RETURN				TM						7
BFEQ										10
BFNE										11
BFGE										12
BFGT										13
BEQ										14
BNE										15
BGE										16
BGT										17

Octal Code	Field Name					
	M1	M2	M1	MA	TMA	DPA
Ø	FM	FA	NOP	NOP	NOP	NOP
1	DPX	DPY	FA	INCMA	INCTMA	INCDPA
2	DPY	DPY	FM	DECMA	DECTMA	DECDPA
3	TM	MD	DB	SETMA	SETTMA	SETDPA

*This instruction uses a 16 bit VALUE (in bits 48-63 of this instruction). The YW, FM, M1, M2, M1, TMA, and DPA Fields are then disabled for this instruction word.

꞉ Fields

of the SPEC Fields may be used per instruction word. The S-PAD Fields (D, SOP, SOP1 ,SH, SPS, and SPD) are then disabled for this instruction.

al e FIELD NAME								Octal Code
SPEC	STEST	HOSTPNL	SETPSA	PSEVEN	PSODD	PS	SETEXIT	
STEST	BFLT	LIT	JMPA*	RPSOA*	RPS1A*	RPSLA*	NOP	Ø
HOSTPNL	BLT	LIT	JSRA*	RPS2A*	RPS3A*	RPSFA*	SETEXA*	1
SPMDA	BNC	LIT	JMP*	RPSØ*	RPS1*	RPSL*	NOP	2
NOP	BZC	LIT	JSR*	RPS2*	RPS3*	RPSF*	SETEX*	3
NOP	BDBN	NOP	JMPT	RPSØT	RPS1T	RPSLT	NOP	4
NOP	BDBZ	NOP	JSRT	RPS2T	RPS3T	RPSFT	SETEXT	5
NOP	BIFN	NOP	JMPP	NOP	NOP	RPSLP	NOP	6
NOP	BIFZ	NOP	JSRP	NOP	NOP	RPSFP	SETEXP	7
SETPSA	NOP	SWDB	NOP	WPSØA*	WPS1A*	LPSLA*	NOP	10
PSEVEN	NOP	SWDBE	NOP	WPS2A*	WPS3A*	LPSRA*	NOP	11
PSODD	NOP	SWDBH	NOP	WPSØ *	WPS1 *	LPSL*	NOP	12
PS	NOP	SWDBL	NOP	WPS2 *	WPS3 *	LPSR*	NOP	13
SETEXIT	BFLØ	NOP	NOP	WPSØT	WPS1T	LPSLT	NOP	14
NOP	BFL1	NOP	NOP	WPS2T	WPS3T	LPSRT	NOP	15
NOP	BFL2	NOP	NOP	NOP	NOP	LPSLP	NOP	16
NOP	BFL3	NOP	NOP	NOP	NOP	LPSRP	NOP	17

Fields

or the IO Fields may be used per instruction word. The Floating Adder Fields (FADD, FADD1, A1, and A2) are then disabled for this instruction word.

al e Field Names							Octal Code
IO	LDREG	RDREG	INOUT	SENSE	FLAG	CONTROL	
LDREG	NOP	RPSA	OUT	SNSA	SFLØ	HALT	Ø
RDREG	LDSPD	RSPD	SPNOUT	SPININ	SFL1	IORST	1
SPMDAV	LDMA	RMA	OUTDA	SNSADA	SFL2	INTEN	2
NOP	LDTMA	RTMA	SPOTDA	SPNADA	SFL3	INTA	3
INOUT	LDDPA	RDPA	IN	SNSB	CFLØ	REFR	4
SENSE	LDSP	RSPFN	SPININ	SPINB	CFL1	WRTEX	5
FLAG	LDAPS	RAPS	OUTDA	SNSBDA	CFL2	WRTHM	6
CONTROL	LDDA	RDA	SPINDA	SPNBDA	CFL3	WRTLM	7

*This instruction used a 16-bit integer VALUE (in bits 48-63 of the instruction word). The YW, FM, M1, M3, MI,MA, TMA, and DPA Fields are then disabled for this instruction word.

AP-120B INTERNAL STATUS REGISTER

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OVF	UNF	DIVZ	FZ	FN	Z	N	C			SRAO	IFFT	FFT	BIT REVERSE		

Bits	Mnemonic	Meaning
0	OVF	Set when the current adder or multiplier (FA or FM) has overflowed. Overflow occurs when an exponent value is increased above 511. The offending result is set to the signed maximum of value of $(1-2^{-27})*2^{511}$, which is roughly $6.7 * 10^{153}$. This bit remains on until cleared by the microprogram or host computer.
1	UNF	Set when the current adder or multiplier result (FA or FM) has underflowed. Underflow occurs when an exponent value is decreased below -512. The minimum legal magnitude which numbers can take without underflowing is roughly $3.7 * 10^{-155}$. The offending value is set to zero. This bit remains on until cleared by the microprogram or host computer.
2	DIVZ	A divide by zero has occurred. The result was set to the value of the dividend. This bit remains on until cleared by the microprogram or host computer.
3	FZ	Set when the current adder result (FA) is zero.
4	FN	Set when the current adder result (FA) is negative.
5	Z	Set when the current S-pad function (SPFN) is zero.
6	N	Set when the current S-pad function (SPFN) is negative.
7	C	S-Pad carry bit. If no S-Pad shift is specified, carry is the carry bit from the S-Pad ALU. If a shift is specified, carry is the last bit shifted off the end of the S-Pad result by the shift.
10	SRAO	Subroutine return stack overflow. Set if more than 16 levels of nested subrouting calls have occurred.

11	IFFT	Inverse FFT flag. When set in conjunction with the FFT flag, bit 12, causes roots of unity table references to be interpreted as positive angles.
12	FFT	FFT Flag. When set causes Table Memory addresses to be interpreted as negative angles referencing the roots of unity table contained in Table Memory.
13-15	BIT REVERSE	15-$\log_2 N$ Where N is the length of a complex data array to which the S-Pad address bit-reverse operator is being applied.

FLOATING POINT SYSTEMS, INC.
P.O. BOX 23489 PORTLAND, OR 97223 11000 S.W. 11TH STREET, BEAVERTON, OR 97005 (503) 641-3151 TLX: 360470 FLOATPOINT PT