Addendum

MAI 2000 and MAI 3000

Diagnostics Program Changes

BOSS/IX Level 7.3A

September 1987
006204-001

M6204A

PAGE STATUS

	Page Nunber	Effective Date
Title		September 1987
Page Status	iii/iv	September 1987
Table of Contents	v/vi	September 1987
Preface	vii/viii	September 1987
Change Overview	ix through x	September 1987
Section 1	1-1 through 1-4	September 1987
Section 2	2-1 through 2-10	September 1987
Section 3	3-1 through 3-8	September 1987

TABLE OF CONTENTS

PREFACE

This document is an addendum to the MAI 3000 Diagnostics and Error Log Manual, M6204, and describes the differences between BOSS/IX Level 7.2x and Level 7.3A diagnostics. The information contained herein is also applicable to the MAI 2000 diagnostics, when running BOSS/IX Level 7.3A.

Diagnostic Changes: 7.2x to 7.3A

The following reflects the changes and improvements that were made to the 7.3A Diagnostics with relation to the 7.2 product of Diagnostics.

-The 7.3A Diagnostics are installable on to a Winchester Disk on an MAI 2000 from tape.

-The Functional Select Test programs of the 7.3A Diagnostics now require that they be enabled by the SERVICE command

-All of the programs which were once in UTIL are now contained in EXEC.

-The ESD tape now contains both the ESD and DIA programs.

-7.3A Diagnostics now support MTS bootstrap.

-The directory size of the 7.3A Diagnostic partition was increased.

-The following changes appeared in EXEC:
  -Command line, entry improved.
  -System is sized for controllers when EXEC is loaded before prompt appears.
  -Tape commands are not supported on the floppy versions of EXEC.
  -The WHEREIS command was added for locating controllers.
  -Corrected Serial Number for special MAIBF number.
  -Corrected overlap problem when moving EXEC.
  -Displays the software copyright notice on system boot.
  -Both MAI 2000 and MAI 3000 programs on one Diagnostic tape.
  -a 'skip ' message is displayed on loading from a tape when passing a file that is not of the current system.
  -The MTS loader timeout was lengthened.
  -Added a delay to MCS skip file for -044 firmware workaround.
  -changed the name of the following commands: RESET to SRESET, FORMAT to DFMT, VERIFY to DVFY, ECC to PRINTECC.
  -Changed the serial number check to handle ASCII characters.
  -Modified the SSN display for compatibility with future MDS.

-The following changes appeared in WDC:
  -Renamed to DISK.
  -The sizing routine was improved to better size the system for all controllers and disks.
  -The commands LUNIT. INQUIRE and SETBLOCK were added.
  -Corrected Stop/Start test for operation with a Maxtor 85.
  -Modified 'begin' table for universal tape.

-The following changes appeared in WDCFS:
  -Renamed to DISKFS.
  -The sizing routine was improved to better size the system for all controllers and disks.
  -The commands LUNIT, INQUIRE and SETBLOCK were added.
  -The commands and test support for XEBEC controllers was removed.
  -Added detection for invalid entry for BLOCK and RETRY commands.
  -Modified 'begin' table for universal tape.

-The following changes appeared in DUTIL:
 -The sizing routine mas improved to better size the system for all
  controllers and disks.
 -The commands LUNIT, INQUIRE and SETBLOCK were added.
 -Added detection for invalid entry of decimal parameters.
 -Changed BFI to match LARL for WDC firmware.
 -Enhanced IBLOCK to provide feedback of GET/PUT.
 -Added warning that entry of defects by sector should be avoided
  except for Rodime manufacturing defects.
 -Modified 'begin' table for universal tape.
 -Corrected edit bug in format option 1.

-The following changes appeared in 4WAY:
 -Corrected to properly handle the Console on an B-Way port.
 -Fixed test 15 so that the terminals are not left in the XOFF
  condition.

-The following changes appeared in EWFS:
 -Corrected timeout for large word counts for REMOTE.

-The following changes appeared in FDFS:
 -Reallocated buffer space to accomodate larger EXEC

-The Floating Point Coprocessor Logic Test (FPCP) was removed from
 7.3A D iagnostics package.

-The following changes appeared in LAN:
 -Lengthened the timeout on the sender for the Master/Slave Test.

-The following changes appeared in MCS:
 -Corrected the IOPB processing buffer location.

-The following changes appeared in MTS:
 -Corrected the Bus Master Parity Error for proper operation on Eagle
  Phoenix.

-The following changes appeared in PIT:
 -Improved the operation of the INT_FACE command.

-The following changes appeared in SCC:
 -Tests 4-6 and 9 were changed to allow for running with the Cache
  in the MAI 3000.

-The following changes appeared in SIT:
 -Added support to test a line printer.
 -Removed all references to unsupported PMMU mask and FPCP task.
 -Enhanced the tape task to test multiple files and data patterns.
 -Corrected I/O drivers for MTS tape.
 -Fixed LAN sizing routine.
 -Corrected problem with word alignment for Eagle LP task.
 -Added support for console on any serial device.
 -Added check for CRT on device prior to changing console.
 -Corrected check routine for WD and MTS to issue SENSE after
  completing current command cycle.

Changes for Disk related diagnostics..... 7.3A*12 09/16/87

General to the programs <DISK, DISKFS and DUTIL)

The following changes were made to the Disk diagnostics, reflected in the 7312 and earlier diagnostic release.

This document contains three sections, 1) a summary of the changes to the disk diagnostics* 2) Further detail on new commands, and 3) examples.

1.0 CHANGE SUMMARY

General changes:

- Initialization of the diagnostics include sizing the system for attached controllers found in the system. Sizing includes identifying the disk drives attached to the controllers, and displaying results of the sizing

- A quick controller test is performed during diagnostic initialization and the results of the test are displayed with the sizing display.

- New Commands and functionality.
- New use of decimal command arguments for some commands
- Changed interrupt processing control
- Automatic test of overlap seek for single board controllers.
- Changed the process and arguments of some commands.
- Error messages for bad unit status were changed to indicate that the data is logical cylinder and head so that the data is not incorrectly assumed to be the actual physical cylinder and head. See the difference in Examples 1 and 2.

NEW DUTIL COMMANDS

LUNIT :	Select a logical unit 0 - 7. Updates the WDC to test and the unit on the WDC.
INQUIRE	To size and display disk subsystem at any time
SETBLOCK	To specify a decimal block number for next command as opposed to specifying cylinder,* head and sector.
GET n	To read a decimal block n. If no n, read the last block selected
PUT n	To write data to the disk to decimal block n. If no n, write to the last selected block. Data read by a previous GET is what is written. If no get preceeds the PUT, data written is what is in the write buffer.
MBLDCK n	To specify the maximum block to read for the IBLOCK command.
IBLOCK n	To specify the increment amount and begin a test loop. The starting block is specified by the GET command. the ending block is specified by the MBLDCK command. To do a get and put to the disk enter the following commands in sequence: MBLOCKi GET n, PUT, IBLOCK n.
EXTRAP n	To extrapolate (translate) a decimal block number. Use this command to translate any decimal block number, and display the translated values in decimal on the console for the blocks surrounding the block number.

COMMANDS CHANGES

BLOCK n — Translate and display data from decimal block n. Used to be hex block n. (DUTIL and WDAFS)

RETRY n — Specify the number of retries desired for the GET and PUT commands in decimal. Used to be specified in hex. (DUTIL and WDAFS)

DUTIL DECIMAL COMMAND SUMMARY

DUTIL commands which have decimal arguments are :

EXTRAP	GET	MBLOCK	BLOCK
RETRY	PUT	IBLDCK	

All decimal commands default to a decimal argument. If you want to enter a hex argument to these (and only these decimal) commands, enter the character "h" before the number to indicate a hex entry. This allows you for instance to "get" a hex block.

DUTIL commands which produce decimal output are :

- BLOCK (except for actual data at the block)
- TRANSLATE
- GET
- IBLOCK
- EXTRAPOLATE

DUTIL FORMATTING

Changed DUTIL formatting option 1 to format first with all defects entered, then do surface analysis. The analysis is followed by a reformat operation only if the surface test found additional defects.

DISK and WDAFS (DISKFS) changes

DISK.... Changed vector for interrupt testing to start at C0 hex.

DISKFS... removed old commands and all XEBEC controller support
The BLOCK and RETRY command arguments are now in decimal.

NOTES

SECTION 2

2.0 Detailed DUTIL Changes Description

NEW DUTIL FEATURES REVISION C2 and later

2.1. EXTRAPOLATION

PREVIOUS REVISIONS

When the DUTIL program executes a surface analysis, the results of the test are stored in a table. After the test, the results are "translated" into the form of CYLINDER, HEAD and BFI. This translated information is displayed in a table after a surface analysis.

When the surface analysis table is translated, or when the "CONVERT" command is used to translate, errors may be encountered when the controllers translate command is executed. The most common is the controller error 14, "no record found". When the translate command fails, DUTIL has to "extrapolate" to determine the actual physical location. by translating sectors surrounding the actual sector.

Revisions of DUTIL prior to C01, did not always extrapolate correctly, which would yield incorrect translated results. This would cause incorrect locations to be formatted out and cause the origional defects to actually move to a new sector when another surface analysis or "read only utility" is executed.

Two problems exist in earlier versions with table conversion

A. If the block before the block we want to extrapolate is the last sector of a track, the values returned for head and BFI will be incorrect if the latest firmware revisions are included on the WDC controller, either the two board or the single board controller. The symptom will be indicated as a returned BFI value of 9944. This is supposed to be a BFI value of 150, for the next head. In the worst case, the cylinder could be off by one cylinder as well.

This problem in other words only occurs when sector 0 of a track cannot be translated, and DUTIL uses extrapolation to determine the physical location.

B. If the physical block before the block we want to extrapolate was previously formatted out as a defect, then the extrapolated value will point to the previous defective sector, and not the actual one we wanted to extrapolate.

This problem only occurs when the sector after a defect previously formatted out cannot be translated and extrapolation is used.

Problem A only occurs when the latest firmware is used, when the formula for the BFI calculation was changed to :

BFI = 150 + n(576) Where n is the sector number, 0-16.

To know if you have the firmware which does not work right with the older DUTIL revisions, simply translate logical block li with the following commands in DUTIL :

```
SEEK  0
HEAD  0
SECTOR  1
TRANSLATE
```

BFI valued (returned in hex)	Problem A
2D6	will occur
2D8	will NOT occur

Note that extrapolation is only necessary when a surface analysis or "read only utility" is executed AFTER a format process which included disk defects. Thus problem B only occurs when defects are included in a previous disk format.

CURRENT C2 REVISION

The conversion problems A and B noted above are fixed.

The program first determines the difference between two blocks to obtain the BFI Offset. This offset will be either 578 or 576 (decimal), depending upon WDC controller firmware. The offset is determined by translating logical block 1. This offset is used to point to the actual bad block during extrapolation.

In the new version, the extrapolation code was changed to correct the known problems. The previous block is translated first if the actual block cannot be translated. If this block returns no error, then the next block after the actual block is translated. If no errors occur, then the difference between the two blocks is calculated. If the difference between the two blocks indicates only one block in between, then the actual block is located by adding the BFI Offset to the previous block.

Using the BFI Offset and this technique, problems A and B are fixed.

Extrapolation error handling

When errors occur with translating the previous and following blocks around the block we are trying to translate, the new extrapolation code will perform additional translating until two valid translations occur, one before the bad block, and one after. The new code will translate [block - 1], if error occurs, will translate [block -2], and then [block -3] if an error occurs. If all blocks -1 through -3 fail, the extrapolation is aborted, and the user is prompted that the logical block cannot be extrapolated.

The same code handles translating the blocks after the bad block that cannot be translated. First [block +1] is translated, then [block + 2] and [block + 3] if necessary until a good translation is obtained. If all blocks +1 to 3 fail, then the user is prompted that the logical block cannot be extrapolated.

When valid plus and minus block results are obtained, the results are analized. If the diffence is more than one block and less than 4. then all blocks in between must be included as defects for the one we are trying to extrapolate. This is because when more than one block is indicated, there is no way to tell which block is the bad block, and which is a defective sector previously formatted out. In this case, up to three extrapolated blocks will be indicated in the "extrapolate table" for one block we cannot translate. This will cause a "converted" or translated list to grow in length when extrapolation is used and additional errors occur.

The program assumes the arbitrary limits of plus and minus three blocks to limit the extrapolation process. Also if more than three defects exist in a row, then the defects may disappear if you simply reformat using the origional defect list. Also it is quite rare to have more than 3 defective Sectors in a row. When this occurs, and the program stops extrapolation and indicates that it cannot extrapolate. the disk should be reformatted using option 5 of the formatting menu, entering any or no new defects. This will remove soft errors associated with "no record found" errors, which may actually not be permanent disk defects. If the defect is permanent, it will reappear in a subsequent surface analysis performed by option 3 in the formatting menu.

2.2 EXTRAPOLATE command

Syntax : EXTRAPOL n Where n is the decimal block to translate

The EXTRAPOLate command is used to tranlate a DECIMAL block. A decimal block number is usually found in the ?.S. Error Log. Up to three blocks before and after the decimal block are used to calculate the actual CYLINDER, HEAD and BFI for the decimal block

If the block number is preceeded by the character "h", then the HEXIDECIMAL block is extrapolated. The actual block number indicated is NOT translated.

This command is useful when the BLOCK command fails, and you are interested in the translated data.

The EXTRAPOLATE command can be used to obtain the correct physical location of a block especially when the plus and minus blocks are not physically adjacent to the defective block. The translated data is used in a subsequent format operation.

The EXTRAPOLATE command will cause the display in example 3. The results of the extrapolation process are displayed as to which blocks were used, and the actual and "additional" blocks. If there are no "additional" blocks displayed* then the plus and minus blocks were translated successfully. Otherwise, additional blocks are displayed for previously formatted defects or when errors occurred with the plus or minus blocks.

When EXTRAPOLate is used to translate a block to be included in another formatting process, THE ACTUAL AND ALL ADDITIONAL BLOCKS SHOULD BE ENTERED AS ADDITIONAL DEFECTS.

Extrapolation examples are included.

2.3 GET and PUT Functions

You can use the GET (read block) and PUT (write block) commands to access the disk by decimal block number.

You can specify the number of software retries you wand to do for the GET and PUT functions. Up to 255 retries can be done for each get, to allow easy put functions after a good get. The put is not done until after successful get, when gets and puts are done over a given range of blocks.

You can defeat the normal disk retry logic and ECC corrections performed by the disk controller if so desired.

The GET and PUT command set allows the disk to be read in different ways not provided in the read only utility functions in the DUTIL menu mode.

2.3.1 GET Command

GET [n]	Where n is the DECIMAL block to read. If the block number is preceeded by the character "h" or "'", then the block number indicated is in HEX.

This command will read the specified block indicated, and will place the data read into the write buffer, to prepare for the next PUT. To display the data that was read, enter the command "WDISPLAY" to display the write buffer. (WDIS for short). If retries are in force, by entry of the RETRY comamnd, then the block is read until no error is found or the retry count is exhausted.

2.3.2 PUT Command

PUT [n]	Where OPTIONAL "n" is the decimal block number to write the data which is contained in the write buffer. If the block number is preceeded by the character "h" or ""', then the block number indicated is in HEX. When no block number is included, data is written to the last block selected.

>> PUT command with no argument.

The PUT command is intended to write the data that is in the write buffer to the block specified in the last GET command. If the GET command produced an error, then the PUT will not be executed when the command is entered.

The PUT command can also write the data to the last block indicated by the BLOCK or SETBLOCK commands. Care should be used when combining the GET n, BLOCK n and PUT commands. To ensure that the same data is written to the same block, never enter the BLOCK command after the GET command to change the block number prior to the PUT command. Refer to section 3.0.

>> PUT command with an argument.

The PUT command with the block number specified cannot be used with the IBLOCK command. The block specified with the PUT should only be used when you want the data at the sector to change, unless the GET and PUT block numbers are the same.

The PUT command with an argument allows you to "patch" a block or blocks with data read from another block, or with any data pattern available with the "CREATE" command.

2.4 MBLOCK Command

MBLOCK n — Where n is the maximum DECIMAL block to read for the IBLOCK command. If the block number is preceeded by the character "h" or "/" , then the block number indicated is in HEX.

When no argument is entered, the Highest Sector Available (HSA) is used. This HSA is not the O.S. HSA but the HSA of the entire disk, which includes the reserved diagnostic area.

The MBLOCK command is used to specify the LAST or MAXIMUM block to read by the GET function when a range of blocks are implemented by the IBLOCK command.

2.5 IBLOCK Command

IBLOCK n — Where n is the amount of blocks to read and to increment after each read. The command also causes the range of blocks specified by the "GET" and "MBLOCK" commands to be read, and written if the "PUT" command was entered before the "IBLOCK" command.

The IBLOCK command does two things. It specifies the amount of blocks to GET and PUT, and it causes a range of blocks to be read and written, when used with the MBLOCK, GET and PUT commands.

To specify a range of blocks to be used, enter the MBLOCK command to specify the last block to read, then the GET command to specify the starting block number, then the optional PUT command to specify a GET-PUT function, and finally the IBLOCK command to specify the amount to read each time, and the loop will be executed over the indicated range from GET block to MBLOCK block number.

With the GET command the disk can be read over a given range of blocks, reading a specified amount of blocks at a time. These commands also allow a disk to be "refreshed" by rewriting the data to the same block.

Examples of the "IBLOCK" command.

Read blocks 10 through and including block 159, 23 blocks at a time.

```
      MBLOCK 159              Specify the  last block to read

      GET 10                  Specify the first, and read it

      IBLOCK 23               Execute
```

Or read blocks 0 thru HSA one block at a time:

```
      MBLOCK                  No argument = HSA
      GET      0              Specify first block to read
      IBLOCK   1              Execute
```

2.6 SETBLOCK Command

SETBLOCK n — Where n is the decimal block to use for the next disk command. When n is not entered, the current decimal block number for the next disk operation is displayed.

If the block number is preceeded by the character "h" or "'" , then the block number indicated is in HEX.

The SETBLOCK command can preceed any disk operation commands such as GET or PUT or BLOCK, thus allowing these commands to be entered without an argument. This command can also be used before other DUTIL commands such as :

SEEK VERIFY READ WRITE TRANS

2.7 Combining with other commands

2.7.1 Combining GET, PUT and BLOCK commands

The BLOCK n command will translate and read the data at the decimal block n, and place the data in the READ buffer. To PUT the data back to the same block after the BLOCK command. you must enter the command "SWAP" to swap buffer pointers to the write buffer to prepare for the next "PUT" command to write it to the disk. Thus you can see that the GET n command is more suited to the GET/PUT function.

Due to the default size of the write and read buffers, and the maximum count of blocks available with the simple read command to the Winchester controller, the maximum number of sectors that can be read by the GET and BLOCK commands is limited to 256 sectors. Use of the PUT command imposes the same limitation.

2.7.2 Specifying different data to be written to a block.

The BLOCK n command can be used to specify different data to be written to a block by the PUT command, and can be confusing when the BLOCK and PUT are used. For example:

#SEC 1	Specify one sector reads.
GET 59	Read decimal block 59, data read is in the write buffer.
BLOCK 350	Read decimal block 350, place data in read buffer.
PUT	This writes the data from decimal block 59 to decimal block 350
PUT 60	This puts the data read from decimal block 59 to decimal block 60.

Another method is to use the "CREATE" command which creates different data patterns in the write buffer. The most probable data pattern to create is all zeros, which can be done with the following example.

```
#SEC  1        Specify the number  of sectors to write
CREATE 0       Create a block of zeros
PUT 498        Write the null block to decimal block number
               498.
```

2.8 Read and Write Retries

2.8.1 DUTIL Retries

If the GET fails, you may simply retry the operation by typing the commands in again, or you can specify the amount of times to retry, by using the "RETRY" command.

```
RETRY  n            Where n is the decimal number of
                    times to retry the operation when an
                    error is detected.
```

The number of retries specified will be used f or each read and write. This command allows up to 255 retries to be attempted before the command aborts. During the retry on error, the error display is suspended. After the retries are exhausted, the error display is returned to normal, and the operation is retried once more. The last error received will be displayed on the screen.

2.8.2 Controller retries

The controller will normally retry an operation if it detects an error. The retry logic in the controller works together with he ECC logic to make data corrections. The controller retry and ECC logic is normally enabled at power on, and is the default operation. This logic can also be disabled by controller command to allow software to detect soft errors, by the "NOECC" command. This command has always been available in DUTIL. The command "ECC" will re-enable the logic.

In the DUTIL read only utilities menu, you can easily enable and disable the controller retry logic with menu selections.

The surface analysis function in the formatting menu defaults to having the ECC and retry logic off, so that the test will find as many defects as possible.

2.9 Specifying the sector count

When you want to specify the number of sectors to get and put, the "#SECTORS" command can be used.

#SECTORS n	Where n is the number of blocks to get or put in hexidecimal. 1-ff hex.

By using this command a group of sectors can be read by the GET command, and written by the PUT command. Note that the IBLOCK command overrides the #SECTORS command argument.

For example:

#SECTORS 21	Read 21 hex blocks (33 decimal)
GET 57	Read block 57 to 89. (33 total)
PUT	Write the same data back.

#SECTORS 10	Read 16 blocks at a time
MBLOCK 58	Specify max block to read, 58
GET 50	Reads 16 blocks starting at 50
IBLOCK 5	Read 5 blocks at a time from block 50 to block 58. This causes two actual reads, one from blocks 50 to 54 (5 blocks). and the next from blocks 55 to 58 (4 blocks). The #SECTORS argument of 10 was overwritten by the IBLOCK 5 command.

Other examples are included in section 3.

3.0 DUTIL EXAMPLES

```
Prompt   command          description
------   -------          -----------
<dutil>  dec 170          Convert decimal 170 to hex
    HEX : AA   DEC: 170
<dutil>  block AA         Translate and display block AA
No simple I/O data transferred
    Status     Command
    CC          OF

Bad unit 0 status
     ERR CODE   Command    Sect addr   Sectors Left   Cyl    Hd    Sec
     14         OF         00000AA     0000           0001   02    00
```

Example of old may DUTIL used to translate a decimal block to the physical location.

EXAMPLE 1. OLD DUTIL TRANSLATION OF DECIMAL BLOCK

```
Prompt   command          description
------   -------          -----------
<dutil>seek 1             seek to cylinder 1. or select cyl 1
<dutil>head 2             select head 2
<dutil>sec 0              select sector 0
<dutil>trans              translate to get the physical location
No simple  I/O  data  transferred
     Status     Command  ,~
     CC          OF

Bad unit O  status
     ERROR   CMD   Block    Blocks Left  LOGICAL: CYL.  HEAD  SEC
     14      OF    OOOOOAA  0000         0001           02    00
```

This example shows the error in translating cylinder 1, head 2 and sector O. Because the translation fails, you can use the extrapolate command to determine the location.

EXAMPLE 2 USE OF TRANSLATE COMMAND and new error display

Prompt	command	description
<dutil>	extrap hAA	Extrapolate hex sector AA. from the Sector address in the error message.

```
DECIMAL BLOCK  :  170
Extrapolated values valid
          -3 : Not used
          -2 : Not used
          -1 :     CYL: 1      HEAD: 1      BFI: 9366

        Actual :   CYL: 1      HEAD: 2      BFI: 150

          +1 :     CYL: 1      HEAD: 2      BFI: 726
          +2 : Not used
          +3 : Not used
```

You can enter the block number to extrapolate in hex or decimal. The default is in decimal. To translate a hex block, preceed the block number with the character "h" or "'" to indicate hex.

EXAMPLE 3 USE OF EXTRAPOLATE COMMAND

```
 Prompt  command          description
 ------  -------          -----------
<dutil>  extrap 169       Translate decimal block 169 by extrapolation
DECIMAL BLOCK : 169
No simple I/O data transferred
    Status     Command
    CC         OF

Bad unit O  status
     ERROR   CMD   Block    Blocks Left  LOGICAL: CYL.  HEAD  SEC
     14      OF    OOOOOAA  0000         0001           02    00

Extrapolated values valid                                    Block xlated
          -3 : Not used                                      166
          -2 : Not used                                      167
          -1 :    CYL: 1     HEAD: 1        BFI: 8790       168

      Actual :    CYL: 1     HEAD: 1        BFI: 9366       169
  Additional :    CYL: 1     HEAD: 2        BFI: 150        170

          +1 : ERROR                                         170
          +2 :    CYL: 4     HEAD: 2        BFI: 726        171
          +3 : Not used                                      172

<dutil>extrap  171
DECIMAL  BLOCK  : 171
No simple I/O data transferred
    Status     Command
    CC         OF

Bad unit O  status
     ERROR   CMD   Block    Blocks Left  LOGICAL: CYL.  HEAD  SEC
     14      OF    OOOOOAA  0000         0001           02    00

Extrapolated values valid                                    Block xlated
          -3 : Not used                                      168
          -2 :    CYL: 1     HEAD: 1         BFI: 9366      169
          -1 : ERROR                                         170
      Actual :    CYL: 1     HEAD: 2         BFI: 150       170
  Additional :    CYL: 1     HEAD: 2         BFI: 726       171

         +1 :     CYL: 1     HEAD: 2         BFI: 1302      172
         +2 :     CYL: 1     HEAD: 2         BFI: 1878      173
         +3 : Not used                                      174
```

```
<dutil>extrap  172
DECIMAL  BLOCK  : 172
Extrapolated values valid                                    Block xlated
          -3 : Not used                                      169
          -2 : Not used                                      170
          -1 :    CYL: 1     HEAD: 2         BFU: 726        171

      Actual :    CYL: 1     HEAD: 2         BFI: 1302       172

          +1 :    CYL: 1     HEAD: 2         BFI: 1878       173
          +2 : Not used                                      174
          +3 : Not used                                      175
```

Note that block 170 was not used, because block 171 could be translated succsessfuly.

EXAMPLE 4 EXTRAPOLATE 169, 171 and 172 Examples

Example of the GET and PUT commands, over a given range and selectable increment amount.

Prompt	command	description
<dutil>	#sec 1	Specify 1 sector read to begin with.
<dutil>	mblock 500	Specify the last block to read as 500 decimal
<dutil>	get 68	Get (read) decimal block 68 the data is put into the-write buffer for the PUT command.

DECIMAL BLOCK : 68 RETRIES : O Block count : 1 Data : valid

<dutil>	put	Write the data back to block 68 immediately, and indicate to the “?block” command that we want to do a put after each get.
<dutil>	iblo 19	Increment block 19 decimal. This causes the following display, which gets 19 blocks at a time, up to and including block 500. The last group of blocks read will be the remainder of the blocks to be read, less than or equal to 19 blocks in this case. The argument of 19 overrides the command "#sec 1" to specify 19 sectors to be read at a time.

DECIMAL BLOCK : 68 RETRIES : 0 Block count : 19 Data : valid

Overwritten for each get. then the final one displayed:

DECIMAL BLOCK : 486 RETRIES : 0 Block count : 15 Data : valid

Example of a get and put over a small area of the disk with one get operation. Up to 255 blocks can be read with one;get. The amount of memory necessary is 255*512 + 30000hex = 328K. This mould be 2 Eagle memory boards.
The 30000 hex is the default address of the start of the read buffer

```
<dutil> #sec 20          Specify a 20 hex sector read (32 decimal)
<dutil0> get  5596       Get 32 sectors starting a  decimal block
                         5596
DECIMAL BLOCK  : 5596    RETRIES  : O    Block count : 32  Data valid
<dutil> put              Write the same data back.
<dutil> wdis 5000        To optionally display the write buffer
                         contents, the data that was written by the
                         put command.  Hit escape when enough data
                         pages are displayed
```

Example of putting null data at a specific block

```
<dutil> #sec 1           Write only one sector on the disk
<dutil> create O         Create null data in the write buffer
<dutil> put 6496         Write null data to decimal block 6496
```

Now compare the data just written.

```
<duitil> get 6496        Read the decimal block back

DECIMAL BLOCK    6496    RETRIES : 0    Block count :  1  Data  : valid

<dutil> compare          Compare data read with data written.
                         Note: The data written by the last PUT
                         was contained in the write buffer. The
                         GET command actually reads the data into
                         the read buffer, then swaps the buffer
                         pointers . AFTER the GET command, the data
                         written by: the previous PUT command is in
                         the read buffer, displayed by the "RDIS"
                         command, and the data read by the "get"
                         command is in the write buffer.
```

Compare data after a get and put

```
<dutil> #sec 2           Specify a two sector read
<dutil> get 9962         Get decimal block 9962 and 9963
<dutil> put              Write the data back
<dutil> get 9962         Read the data back
<dutil> compare          Compare data read back by the second
                         "get" command with the data written
                         by the put command.
```

```
Specify a read of a large range of blocks with no retry, or,
ECC detection and correction

<dutil> retry 0         Specify no retrys by DUTIL
<dutil> #sec 1          One sector  to start
<dutil> writeok         Specify it is ok to send the "noecc" command
<dutil> noecc           Turn off ecc and retry in the controller
<dutil> mblock 5000     Max block to read, 5000  decimal
<dutil> get 0           Read and start with block 0
DECIMAL BLOCK  : 0   RETRIES : 0    Block count:   1   Data    valid
<dutil> iblock 1         Read blocks 0-5000  1 block at a time.
                         with no ecc nor retry. The last block
                         read is displayed as:

DECIMAL BLOCK  : 5000   RETRIES : 0    Block count:   1   Data    valid

<dutil> *ecc             Turn retry and ECC back on
<dutil> readonly         Write protect the disk in software


Specify that you want to try to get the data at a specific block
and retry several times without having to type the command over
and over.
                                                                +
<dutil> #sec  1          One sector read

<dutil> retry 255        Retry up to 255 times if there is a read
                         error, to obtain  the data.
<dutil> get 6745         Read decimal block 6745, up to 255 times

 In this case if the retries are exhausted, the following would
 be displayed for example

 Bad unit 0  status
     ERROR     CKD   Block   Blocks left     LOGICAL: CYL.  HEAD   SEC
     14        08    hhhh    0000            xxxx           xx     xx

 DECIMAL BLOCK     6745  RETRIES : 255   Block count  : 1  Data  : INVALID

 While DUTIL is carrying out retries, the error display is disabled,
 and the last retry will cause the error display.   Then if you try to
 PUT data back that was read:

 <dutil> put
 "The PUT  command was not executed!"      This is displayed when the
 "get" command produced an error, and the "put" is not expected.

 Mow write null default in the block from the last example:

 <dutil> #sec  1         Specify one sector
 <dutil> put 6745        Put whatever is  in the write buffer into
                         block 6745 decimal

 <dutil>  create 0          Create a block of all zeros in  the write
                               buffer
 <dutil>  put 6745          Write the null data back
```

NOTES