

UDA50 KDA50

RA60 RA80/81/82

BAD BLK REPLACE UTIL
CZUDLA0

AH-FF57A-MC
1 OF 2 OCT 1985
COPYRIGHT © 1985

digital
MADE IN USA

The main body of the document is a large grid of 100 small, illegible panels, arranged in 10 rows and 10 columns. Each panel appears to contain some form of data or a small diagram, but the text is too small and faded to be readable. The panels are separated by thin white lines, creating a clear grid structure.

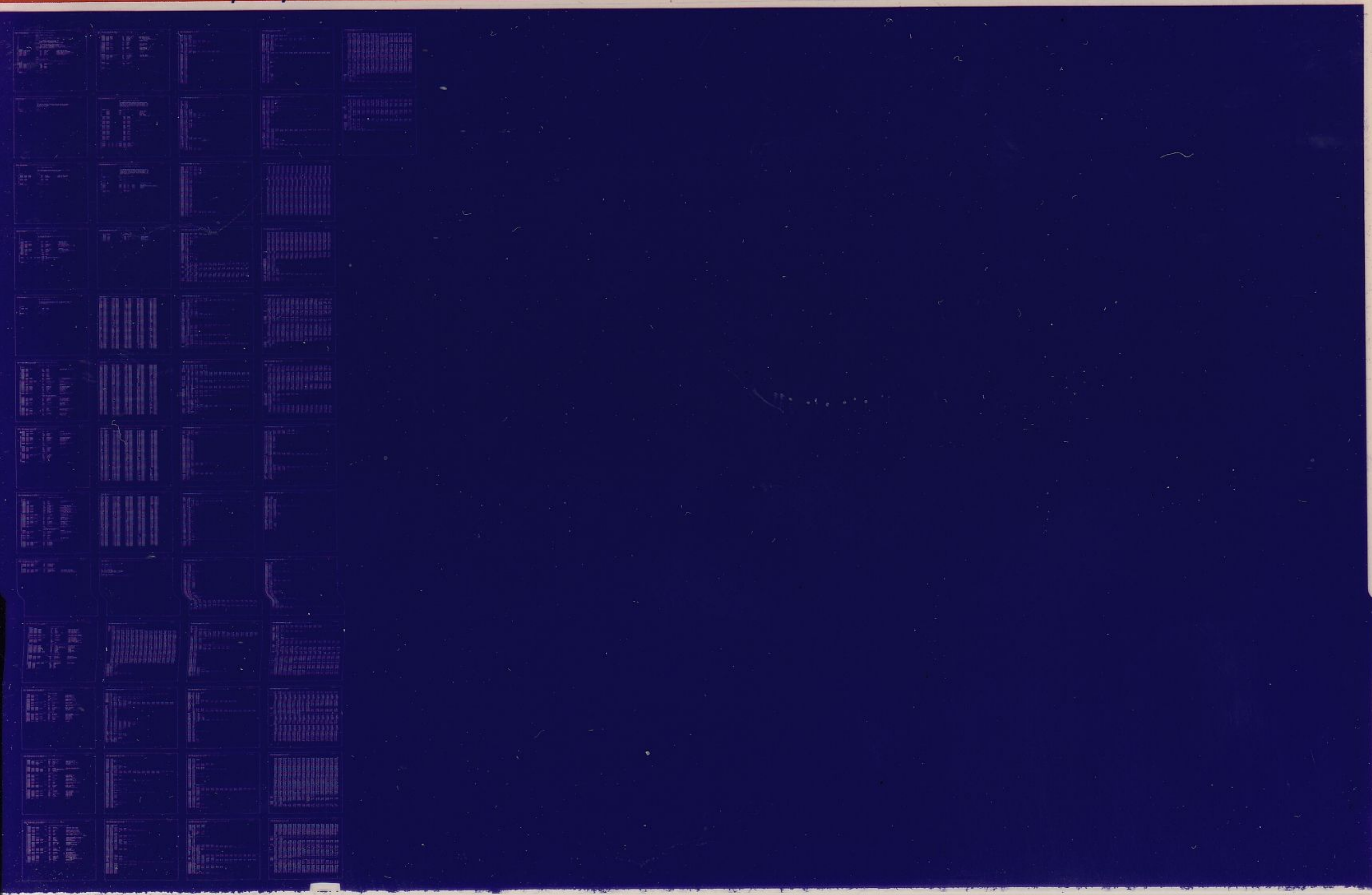
UDA50 KDA50

RA60 RA80/81/82

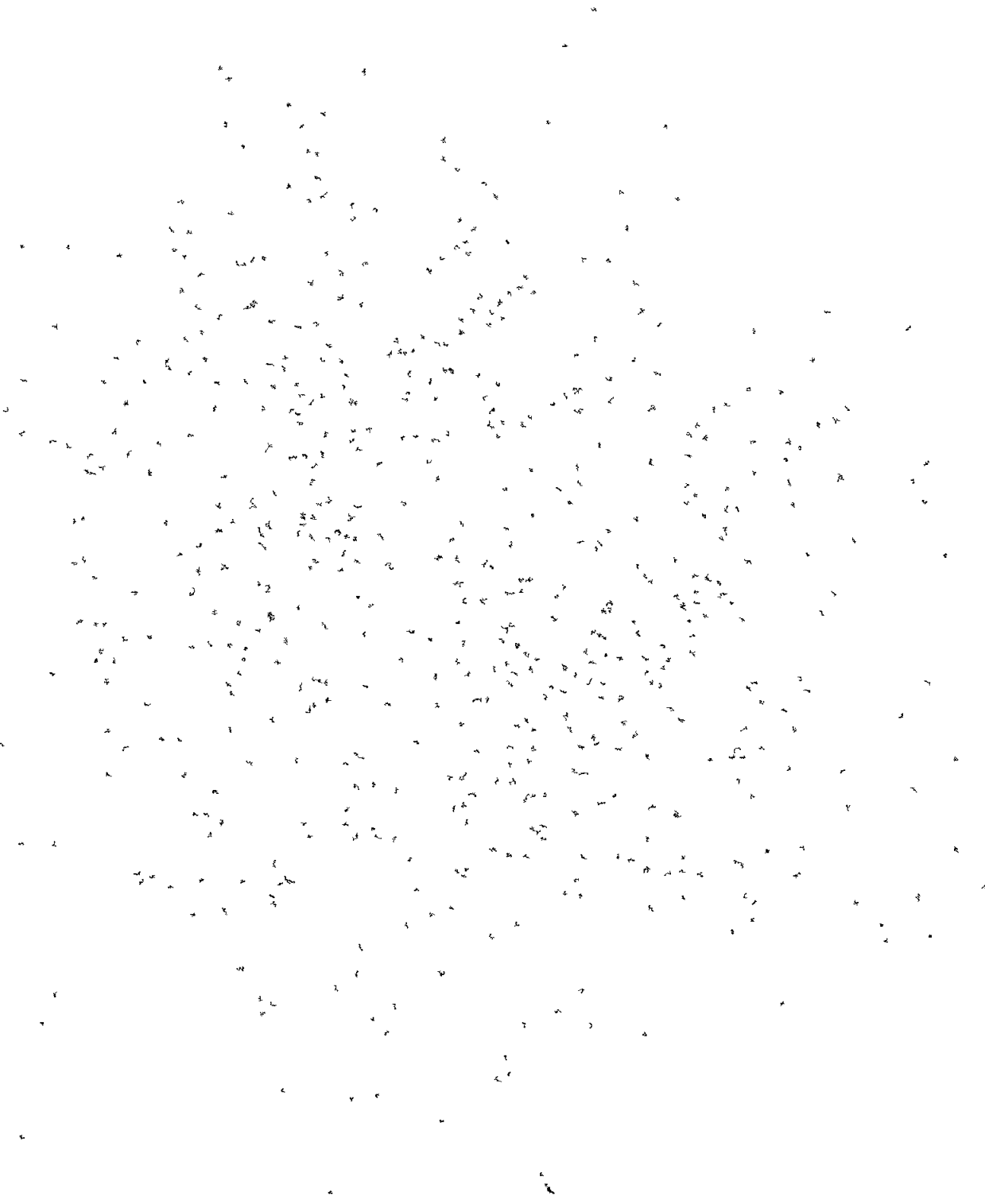
BAD BLK REPLACE UTIL
CZUDLA0

AH-FF57A-MC
2 OF 2 OCT 1985
COPYRIGHT© 1985

digital
MADE IN USA



C B H
A ? :



IDENTIFICATION

PRODUCT CODE: AC-FFS6A-YC
PRODUCT NAME: CZUDLAO BAD BLOCK REPLACEMENT UTILITY
PRODUCT DATE: 19 APR-1985
MAINTAINER: H. PAUL HOLSINGER
AUTHOR: H. PAUL HOLSINGER

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DECnet	IAS	VAX
DECsystem-10	MASSBUS	VMS
DECSYSTEM-20	PDP	VT
DECUS	PDT	Digital Logo
DECwriter	RSTS	

.REM 8

This software is furnished for the purpose of maintaining Digital Equipment Corporation RA series drives. It should only be used by persons authorized to perform service on these products. This software or any other copies thereof should not be provided or otherwise made available to any unauthorized person.

Even though this program is distributed along with diagnostics it is NOT (repeat not) intended for use as an RA drive diagnostic. This program assumes that the RA drive subsystem is not defective and that the user only intends to use this program for its intended purpose of finding and replacing bad blocks. Using this program on defective hardware could produce indeterminate results detrimental to the customer and Digital.

INDEX

1.0	INTRODUCTION
1.1	Hardware Requirements
1.2	Software Requirements
1.3	Memory Requirements
2.0	MODES OF OPERATION
2.1	Verify Mode
2.2	Automatic Mode
2.3	Manual Mode
3.0	BACKGROUND
4.0	DISCUSSION
4.1	Sample Run Times in Automatic Mode
4.2	Transient Error Table Discussion
4.2.1	Transient Error Table Overflow
4.3	System crash while running this program - WARNING
5.0	STARTING THE PROGRAM
6.0	QUESTIONS ASKED BY THE PROGRAM
6.1	Display Operator Help
6.2	Automatic or Manual Replacement
6.3	QUESTIONS ASKED IN AUTOMATIC REPLACEMENT MODE
6.3.1	Automatic Crash Recovery
6.3.2	Display Replacements As They Occur
6.3.2.1	Replacement Status Messages
6.4	QUESTIONS ASKED IN MANUAL REPLACEMENT MODE
6.4.1	Enter LBN to be Replaced (Decimal) or <CR> to Exit
6.4.2	Attempt Crash Recovery
6.5	QUESTIONS ASKED DURING EITHER REPLACEMENT MODE
6.5.1	Display ACT Replacement Descriptors
6.5.2	Enable Replacements
6.5.3	Enable Write With Forced Error Flag
7.0	"HARD" ERROR REPORTING
7.1	ERRORS REPORTED BY ERROR LOG PACKETS FROM THE CONTROLLER
7.1.1	Controller Error Packet Report
7.1.2	Host Memory Access Error Packet Report
7.1.3	SDI (Drive) Error Packet Report

- 7.2 STATUS/EVENT CODES
- 7.3 CONTROLLER INITIALIZATION ERRORS
 - 7.3.1 Memory Error Occurred While Accessing Controller Register
 - 7.3.2 Controller Resident Diagnostics Detected Failure
 - 7.3.3 Step Bit Error In SA Register During Initialization
 - 7.3.4 SA Register Did Not Zero After Step 3 Write
 - 7.3.5 SA Register Error During Initialization
 - 7.3.6 Controller Did Not Clear Ring Structure In Host Memory
- 7.4 CONTROLLER INITIALIZATION (BY SA REGISTER) ERROR CODES
- 7.5 HOST/CONTROLLER COMMUNICATION ERRORS
 - 7.5.1 No Interrupt Received For 30 Seconds
 - 7.5.2 Fatal Error Reported In SA Register
- 7.6 CONTROLLER COMMUNICATION ERROR CODES
- 7.7 UNIT INITIALIZATION ERRORS
 - 7.7.1 Failed Get Unit Status
 - 7.7.2 Failed Set Controller Characteristics
 - 7.7.3 Failed Online
- 7.8 PROGRAM INITIALIZATION ERRORS
 - 7.8.1 Failed Dynamic Memory Allocation For Selected Unit
- 7.9 MSCP COMMAND/RESPONSE ERRORS
 - 7.9.1 Failed READ
 - 7.9.2 Failed WRITE
 - 7.9.3 Failed ACCESS
 - 7.9.4 Failed REPLACE
 - 7.9.5 Command/Response reference number mismatch
 - 7.9.6 Fatal End Message status detected
 - 7.9.7 Failed Replacement verification
- 7.10 END MESSAGE FLAGS (Endmsg flags)
- 7.11 RCT READ/WRITE ERRORS
 - 7.11.1 Failed RCT READ
 - 7.11.2 Failed RCT WRITE
 - 7.11.3 Failed READ of all copies of RCT
 - 7.11.4 Failed WRITE of all copies of RCT
 - 7.11.5 No Null descriptor entry found in RCT
- 8.0 INFORMATIONAL MESSAGES
 - 8.1 Program Initialization
 - 8.2 Unit Initialization
 - 8.2.1 Zero RCT size detected -- BBR not supported by this drive
 - 8.2.2 Scanning RCT...
 - 8.3 End of Pass Information
 - 8.4 Elapsed Runtime message

1.0 INTRODUCTION

This program is a highly specialized utility for use in those cases where an RA drive has been diagnosed to have bad blocks on the media. These bad blocks may be causing ECC errors that can be isolated to a specific Logical Block Number(s) (LBN). Using the Digital Storage Architecture (DSA) Bad Block Replacement (BBR) algorithm, this program will replace those blocks that it either defines as bad (Automatic Replacement Mode) or will take an LBN address that the user supplies and replace only that block (Manual Mode). In either case, bad block(s) will be replaced, with no apparent loss of storage capacity to the customer.

WARNING WARNING WARNING WARNING WARNING WARNING WARNING

-----*

```

)      IN ORDER TO PROPERLY EXECUTE THIS PROGRAM IN AUTOMATIC OR MANUAL    )
)      MODE, THE FOLLOWING STEPS MUST BE FOLLOWED WITHOUT EXCEPTION.        )
)      UNPREDICTABLE RESULTS COULD RESULT IF NOT FOLLOWED.                  )
)                                                                              )
)                                                                              )
)1. CUSTOMER MUST BACKUP THE DATA FROM THE DRIVE(S) TO BE USED AND VERIFY  )
)   ITS CORRECTNESS.                                                         )
)                                                                              )
)2. EXECUTE THIS PROGRAM ON THE DRIVE(S). MULTIPLE PASSES ON EACH DRIVE, WHEN )
)   IN AUTOMATIC MODE, IS RECOMMENDED ---- DR>Start/pass:(Number of passes) )
)                                                                              )
)3. CUSTOMER MUST RESTORE THE BACKED UP DATA TO THOSE DRIVES FROM WHICH IT  )
)   WAS BACKED UP AND AGAIN VERIFY ITS CORRECTNESS.                          )
)                                                                              )
)-----)

```

1.1 Hardware Requirements

This program requires the following minimum hardware configuration:

- * PDP-11 Unibus or Q-bus processor
- * 28K words of memory (minimum)
- * Console terminal
- * XXDP+ load media containing this program
- * One or more UDA50-A or KDA50-Q subsystems.
- * One or more DSA disk drives.

- * A system clock - either type L or P - will be used to report runtime, and to detect controller MSCP response timeouts.

1.2 Software Requirements

This program is designed to run using XXDP+ V2.0 or greater. It has been assembled using the SVC35R MACRO library.

1.3 Memory Requirements

The utility occupies approximately 20k bytes of memory. Each drive configured for test will require about 1k bytes additional memory.

2.0 MODES OF OPERATION

This program will run in one of three modes.

2.1 Verify Mode

This allows the user to "scan" the media without the program taking any action on what it finds. In this way, you can get a "picture" of any problem found, without the program writing in any way on the media. This is extremely valuable, since the drive can be write protected and thus the lengthy Backup and Restore operations recommended need not be done. By getting a "look" at the condition of the media, you can then decide whether to commit to either

Automatic or Manual mode and do the backup and restore of the customer data. This mode can also be used to get the replacement descriptors displayed (By answering the question "Display Replacement Descriptors" with a yes) without the Backup and Restore operations. This way you can get a "look" at the total number of replacements quickly.

Execution in this mode is entered by:

- 1) WRITE PROTECT THE SUBJECT DRIVE --- AFTER THE DRIVE HAS BECOME READY. SPINNING UP AN RABO WITH IT WRITE PROTECTED WILL RESULT IN A FAULT, BECAUSE OF A WRITE FAIL DURING THE SPIN-UP WRITE TESTS.
- 2) BRING UP THE PROGRAM AND SPECIFY AUTOMATIC MODE
- 3) ANSWER THE QUESTION "ENABLE REPLACEMENTS" WITH A "N" (NO)

Verify Mode can also be entered by specifying Manual Mode with replacements disabled. In this way, you can "look" at the "status" of a particular LBN.

2.2 Automatic Mode

This is the mode used when you do not know what the bad block LBN(s) are, or do not have the bad block LBN address(es) available in decimal. Or, possibly you just want the program to run unattended, and take action on any blocks it finds bad. Normally you should use Manual Mode and enter the bad blocks, given that you know (from the error logger or other sources of information) what blocks are bad. Please make note of the Warning in Section 4.0, when running with large pass counts. Large pass counts in Automatic Mode are recommended, however please understand and follow that warning. Execution in this mode is entered by:

- 1) Insure the Customer has backed-up and verified the data from the subject drive
- 2) Specify Automatic replacements
- 3) Answer "Enable Replacements" with a Y (yes)
- 4) After the completion of the program, the customer can restore and verify the backed-up data.

2.3 Manual Mode

This is the preferred method for using the program and the most efficient. If you can obtain a decimal address of the bad block(s) you want replaced, this mode will take the information (LBN address) and without hesitation replace it. Operating System error loggers, should provide the LBN address of those blocks that generate ECC and Header errors. Sometimes you may have to "convert" the LBN address from Hexadecimal or Octal, as provided in the error log-report, to decimal so it can be used by this program. Providing the wrong decimal address of a bad block is not good. The program will replace any LBN you specify and if you specify the wrong one, you will have one additional replaced block on the media.

If you have no idea what the LBN addresses of the bad block(s) are, you can possibly run this program in Verify Mode and if Verify finds any bad blocks, it will report them to you in decimal. Taking these addresses and using them in Manual Mode, will result in the quickest method for replacing known bad blocks. However, the customers operating system has much more time to find bad blocks than does Verify Mode. Thus, you should use the error log for bad block determination whenever possible. Execution in this mode is entered by

- 1) Insure the Customer has backed-up and verified the data from the

- subject drive
- 2) Specify Manual replacements
 - 3) Answer "Enable Replacements" with a Y (yes)
 - 4) Provide the LBN address of the bad block in decimal, when asked for. A Carriage Return at this question will exit Manual mode.
 - 4) After the completion of the program, the customer can restore the backed-up data and verify it.

3.0 BACKGROUND

This program was primarily developed for use in those cases where a customer has an operating system that does not have the DSA Bad Block Replacement capability already in it. Several operating systems already have a method of doing DSA Bad Block Replacement (BBR) and would only rarely need the services of this program (VMS is one of many). However, there are cases where even with an operating system that has the BBR capability, the need to replace a specific LBN, or allow this program to search for bad blocks, may be appropriate.

In those cases where a customer is running an operating system that does not have the BBR capability, bad blocks may develop and show up as ECC errors in the system error log. When this occurs, this program can be used to replace those bad blocks and thus eliminate the logging of ECC errors related to specific bad block(s). It's a known fact that bad blocks can become worse with time. What may start out as a 6 symbol ECC error may eventually end up as an uncorrectable ECC error, if action is not taken to replace that block.

CAUTION

It is imperative that every effort be used to establish the fact that errors showing up in the error log are indeed related to a bad block, before attempting to use this program. If ECC errors showing up in an error log are the result of a hardware problem, this program (When used in Automatic Mode) could replace a considerable number of blocks that are indeed good. If this happens, and later you fix the problem causing the ECC errors, you should reformat the HDA/Pack. A description of this situation is given in RAB1-TT-12. The information in this Tech Tip applies to all RA series drives and would be helpful in this situation.

4.0 DISCUSSION

Normally, disk blocks that are defective generate ECC errors when read. These ECC errors, as seen from the error log, should all be logged against a particular LBN (or if multiple bad blocks, a specific number of LBN's). Thus, if you observe that the ECC error(s) seem to all occur at the same LBN address(es), you can start to assume that a bad block is the cause. This being the case, you can either provide this program with the LBN address(es) in decimal, and have them replaced in Manual Mode, or allow this program the opportunity to find the bad block(s) (Automatic Mode). IT IS RECOMMENDED that if you feel sure you have identified the bad blocks, and have the LBN addresses of them in decimal, you should use Manual Mode to have them replaced. Manual Mode is a quick efficient method of getting a block replaced and should be used to save time and effort, however, be sure you know what your doing and have the right information needed by this program.

In Automatic Mode, this program will run the specified number of passes on the drives you specify. It does not execute simultaneously on the drives you specify but works in a "serial" fashion, doing one drive at a time. One pass is one execution of the program on the drive(s) you specify. IT IS HIGHLY RECOMMENDED that when using this program in Automatic Mode that you allow multiple passes to be run. Running this program, in Automatic Mode, over a weekend or overnight would provide the most benefit.

4.1 Sample Run times in Automatic Mode

RA81 - About 9 Minutes
RA60 -- About 4.5 Minutes
RA80 -- About 3 Minutes

NOTE: This time is extended, when replacements occur. Thus, if you have many bad blocks, the program will take considerably longer.

4.2 In order to verify that an LBN indeed needs replacement, this program (And all other Operating Systems with BBR capability) tries to "verify" that the bad block report is not a transient error and in fact relates to a defect in the media. If all reports of Bad Blocks (Like ECC errors) caused replacement, without question, then a substantial possibility would exist that blocks with no defect would be replaced. As you know, many times errors are caused by the environment (Like static, electrical problems etc). Errors reported that are not related to a media defect should not be replaced. Thus, an attempt is made to verify that a bad block report can be duplicated and therefore related to a higher probability of a media defect being the cause. This is the way it happens in the most usual case:

- 1) This program uses the MSCP ACCESS command to do large Byte count transfers (Read) from the drive being tested. It starts with LBN 0 progresses to the highest user LBN on the drive. The ACCESS command moves data from the drive to the controller, where the data is checked for any errors. The data IS NOT actually moved host memory. This way we can read data at the fastest possible rate and thereby make it possible for more passes to be made across the media in the shortest possible amount of time.
- 2) If an error (Lets say an 8 symbol ECC error) is detected by the controller, a flag gets set called the Bad Block Reported flag. This flag is reported to this program by whats called an "end packet". This packet describes the success or failure of the ACCESS command that we gave the controller. The end packet also provides the LBN address that caused the error.
- 3) When this program sees this Bad Block Reported Flag, it takes the LBN address that caused the Bad Block reported flag to set and goes into a series of READ commands to the LBN that reported the bad block. A series of 32 reads are attempted. If a second Bad Block report occurs, the program terminates the testing and replaces the block.
- 4) If 32 re-READ's are accomplished without error or bad block report then the error is considered a transient and replacement is not attempted.

5) HOWEVER, if it is considered a transient error, the LBN address is logged in a memory table called the transient error table. If on a subsequent pass across the media, this same LBN address reports an error or Bad Block, the LBN is replaced without argument and without any further testing. In other words, if we have a transient error and then later this same LBN reports another error, it is replaced.

That is why running this program for multiple passes is beneficial. The transient error table is kept in memory and is kept updated as long as the program is running. If the program is terminated, the table is lost and any entries in it. The benefit of this table is that any marginal bad blocks that just does not show an error each time it is read can be replaced.

4.2.1 Transient Error Table Overflow

The Transient Error Table, that is kept in memory as long as this program is running, has the ability to keep track of 400 Transient errors for each drive being tested. This table should be of sufficient size for all cases where bad blocks that are only marginally bad need to be tracked. If this table overflows, a message is typed and the testing terminates. If this happens, you have more trouble than what this program can help with. This message is reported in the following manner:

Transient error table overflow;
ADDITIONAL MAINTENANCE REQUIRED.

This is true. If you log more than 400 transients, then the drive being tested may not have bad blocks causing the errors you see. Most likely you have a data path problem giving false indications of ECC errors or something like a worn-out spindle ground brush.

The recovery from this condition would be to first fix whatever is causing the high transient error rate then reformat the media. Most likely there are many replacements that were made on blocks that were in fact not bad. Reformatting will put the media back to a known state (Please read RA81-TT-12). You may want to run several passes of this program, after having reformatted, just to be sure that the formatter found all the bad spots.

4.3 System Crash while running this program -- WARNING

```
WARNING      WARNING      WARNING      WARNING      WARNING      WARNING
-----*-----*-----*-----*-----*-----*-----*
) If the system crashes (CPU failure, power fail etc) DURING )
) THE EXECUTION OF THIS PROGRAM (Either Manual or Automatic )
) Mode) IT IS REQUIRED that you run at least one pass of this )
) program after recovering from the crash condition. Abrupt )
) termination of this program could leave an incomplete replacement. )
) One quick pass of this program would allow for the completion of )
```

```
    } any incomplete replacements. Using the control "C" to terminate    }  
    } this program is not considered an abrupt termination and should    }  
    } not leave any incomplete replacements, although use of control    }  
    } "C" to terminate this program is not recommended.                }  
-----
```

5.0 STARTING THE PROGRAM

This section describes the hardware questions which must be answered for each Unit configured for testing. Each question is displayed along with the default value for the parameter. A carriage return (CR) selects the default value. An explanation of each question follows.

- 1) Boot the Diagnostic Media
- 2) After you have called for the running of this program, you will arrive at the diagnostic supervisor prompt - DR>. Typing "start/pass:(the number of passes you want run)" at the prompt will begin the execution of this program and ask you the following questions:
 - CHANGE HW (L) ? Yes - Answer "yes" to enable the selection of which drives you want to use and other operating parameters.
 - ♦ UNITS (D) 1 - This asks you HOW MANY DRIVES YOU WANT TO TEST. In Automatic Mode this diagnostic will execute on the number of drives you specify in a "serial" fashion (One drive at a time).
 - UNIT 0 - Unit 0 is NOT drive 0. This only refers to the first drive you want used and asks for certain information about this first drive to test.
 - BUS ADDRESS OF XDA (O) 172150 ? - UNIBUS ADDRESS OF THE CONTROLLER that the first drive is connected to is the first UDA50 default in Octal) "XDA" refers to the controller -- Like UDA or KDA.
 - VECTOR (O) 154 - VECTOR YOU WANT PROGRAMMED INTO THIS CONTROLLER. (RA drive controllers have software provided vectors (Take the default or provide the vector you want assigned).
 - BR LEVEL (D) 5 - TAKE THE DEFAULT, unless you have a good reason not to.
 - BUS BURST RATE (D) 63 ? - This defines the NUMBER OF LONG WORDS THAT WILL BE TRANSFERRED each time the controller becomes bus master. The default (63) should be good. Operating System Burst Rates must always equal 0 (zero).
 - DRIVE ♦ (D) 0 - This is the DRIVE NUMBER TO TEST, in decimal. Default is drive "0".

The above questions will be asked for each of the number of Units you specified to test. Once the above information has been defined, for each drive you wish to test, the following questions will be asked. The following questions are asked only once and apply to this run of the test.

6.0 QUESTIONS ASKED BY THE PROGRAM

6.1 DISPLAY OPERATOR HELP (L) N ?

IT IS HIGHLY RECOMMENDED THAT YOU READ THE OPERATOR HELP INFORMATION. Answering yes will cause this help file to be typed. For the Operator Help file to be printed, it must be resident on the diagnostic media that this program was obtained from. This information is not a part of the program but is a separate file that resides on the diagnostic media and will be typed when this question is answered with a "Y" (Yes)

6.2 AUTOMATIC OR MANUAL REPLACEMENT (A) A ?

The default response of "A" selects Automatic replacement mode; an "M" selects Manual replacement mode.

If AUTOMATIC REPLACEMENT MODE is selected, the entire user LBN area of each Unit will be processed, and this program will replace all blocks found to be bad. In this mode, up to eight Units can be processed in a "serial" fashion (One drive at a time).

If MANUAL REPLACEMENT MODE is selected, the operator must specify the desired block(s) to be replaced. In this mode, the LBN address supplied to the program must be in decimal (Not Octal, Hexadecimal etc). Only a single Unit can be processed during each program execution, in Manual Mode.

6.3 Questions Asked In Automatic Replacement Mode

Invoking Automatic Replacement Mode, the following Automatic replacement mode questions are asked.

6.3.1 AUTOMATIC CRASH RECOVERY (L) Y ?

This parameter determines whether crash recovery will be performed automatically, upon the detection of an incomplete replacement. Incomplete replacements can occur due to a "system crash" on an earlier run of this test or operating system replacement process. If automatic recovery is selected, the program will complete the replacement operation before processing the Unit. If automatic recovery is disabled, the Unit will be dropped upon the detection of an incomplete replacement. WE RECOMMEND TAKING THE DEFAULT YES ANSWER. In Manual mode this question is asked only upon detection of a need for crash recovery. When an incomplete replacement is detected, a message similar to the following will occur:

RCT ERROR DETECTED;

Crash occurred during previous replacement Phase 1.

Attempting Recovery...

LBN: 679864. Status: RBN: 13331. Operation: SEC

This example indicates that a crash occurred, sometime in the past, during a phase 1 replacement. Replacement at phase 1 will assign a replacement RBN, and attempt recovery. If this program shows a crash recovery, it would be wise to allow at least one pass of of this program, in Automatic Mode, to take place. No "Status" is listed. This would be normal for recovery, as the recovery process does not know the original reason replacement was requested.

6.3.2 DISPLAY REPLACEMENTS AS THEY OCCUR (L) Y ?

This parameter determines whether replacement operations are displayed as they occur. A response of "Y" will cause replacement operations to be displayed; a "N" will inhibit the display of replacement operations. In Manual replacement mode, display is permanently enabled, allowing the operator to verify desired LBN's have been replaced. If answered with a "Yes", as a replacement is made the following output will be provided on the console terminal.

6.3.2.1 Replacement Status Messages

Each time the program is started, it will type out the following information. This information is used during the report of a replacement operation (Shown Below). Understanding these "codes" will allow you to better understand the replacement operation that was performed and why replacement was requested.

LBN Status codes:

BBR - Bad Block reported (ECC Error)
ECH - Hard ECC Error encountered
HCE - Header Compare Error encountered
DST - Data Sync Timeout encountered
FER - Forced Error encountered.
WFE - Block written with FE (Forced Error) set

RBN Operation types:

PRI - Primary replacement for LBN.
SEC - Secondary replacement for LBN.
UNS - RBN marked unusable.

NOTE

+-----+
) Periods after a number indicate that the number)
) is a decimal number.)
+-----+

PRIMARY BAD BLOCK REPLACEMENT

The LBN was tested, and the BBR flag was set repeatedly. The LBN's Primary replacement RBN was used to store the LBN's data.

LBN: 23271. Status: BBR RBN: 456. Operation: PRI

SECONDARY BAD BLOCK REPLACEMENT

The LBN was tested, and the BBR flag was set repeatedly. The LBN's Primary replacement RBN is currently used by another LBN. A Secondary replacement RBN was used.

LBN: 23271. Status: BBR RBN: 455. Operation: SEC

RBN BECOMES UNUSABLE

Upon reading data from an RBN, an error resulted. This indicated that the RBN is bad and must be marked as unusable. Also, the customer data residing in the RBN must be moved to a new RBN. A Secondary replacement always results from this condition.

LBN: 23271. Status: RBN: 456. Operation: UNS
LBN: 23271. Status: BBR RBN: 455. Operation: SEC

LBN WITH AN UNCORRECTABLE ECC ERROR REPLACED

An LBN is read repeatedly with an uncorrectable ECC error (ECH). The indicated replacement operation is performed. If enabled, the LBN data is written with the Forced Error (FE) flag set, and the following information appears.

LBN: 23271. Status: WFE RBN: 456. Operation: SEC (or PRI)

If the question about writing uncorrectable ECC errors WITHOUT the Forced Error Flag is answered with a NO, the corrupted data is written to the RBN as is, and the following information appears:

LBN: 23271. Status: ECH RBN: 456. Operation: SEC (or PRI)

TRANSIENT ECC ERROR REPORTED

During the "scan" for bad blocks, the BBR flag was set for a particular LBN. Once this condition was noted, the program tries to verify the report of a bad block. If the bad block report can not be verified, the operator is notified of the LBN and NO replacement is attempted. If you do not feel good about an ECC error being reported and then the program not being able to verify the error, you can use Manual Mode to replace the reported LBN.

LBN: 23271. Status: BBR

NOTE

) Many times the occurrence of an ECC error that is not)
) repeatable can indicate a Data Path problem or other)
) conditions that are not related to bad spots.)
) If this condition occurs many times during a pass of)

```

) this program, I would start to look for a data path )
) problem or some other problem causing transient )
) errors. )
.....

```

6.4 Questions Asked In Manual Replacement Mode

Invoking Manual Replacement Mode, The following Manual Replacement Mode questions are asked.

6.4.1 ENTER LBN TO BE REPLACED (DECIMAL) OR <CR> TO EXIT (A) ?

This is asking for you to specify which LBN you wish to have replaced. BE SURE YOU ARE PREPARED TO PROVIDE THE LBN ADDRESS IN DECIMAL. Different Operating Systems error loggers display the LBN address in various radix's (Like Hex, Octal, Decimal etc) Be sure you know what your dealing with and make the proper conversions, if necessary.

The LBN entered is checked against the maximum number of user LBN's available (See section 7.12). If the limit check fails, the following message appears:

```
Limits: LO= 0; HI= ddddd.
```

HI would be the maximum LBN address that can be replaced on that drive. If the LBN passes the limit check, the replacement is made and the operation is displayed, as specified in the paragraphs below.

Once you have provided the LBN address for the replacement, and the replacement has occurred, this same question will be asked. If you have no more LBN's for replacement, just hit Carriage Return to exit.

6.4.2 Attempt Crash Recovery (L) Y ?

This question is only asked upon the detection of a previously incomplete replacement. Incomplete replacements can occur due to a "system crash" on an earlier run of this test or operating system replacement process. If you answer yes, the program will complete the replacement operation before processing the requested Manual Mode LBN replacement. If you answer no, the Unit will be dropped and the incomplete replacement will not be completed. We recommend a yes answer. The following is an example of this condition.

```
SCRUB1 SFT ERR 00061 ON UNIT 00 TST 001 SUB 000 PC:02506
RCT error detected
```

```
Crash occurred during previous replacement
Phase 1
```

```
Attempt crash recovery (L) Y ?
```


Attempting Recovery...

LBN: 679864. Status: RBN: 13331. Operation: SEC

This example indicates that a crash occurred, sometime in the past, during a phase 1 replacement. Replacement at phase 1 will assign a replacement RBN, and attempt recovery. If this program shows a crash recovery, it would be wise to allow at least one pass of of this program, in Automatic Mode, to take place. No 'Status' is listed. This would be normal for recovery, as the recovery process does not know the original reason replacement was requested.

6.5 Questions Asked During Either Replacement Mode

6.5.1 DISPLAY RCT REPLACEMENT DESCRIPTORS (L) N ?

This parameter affects program execution in both replacement modes. The Replacement Control Table (RCT) keeps a "log" of all LBN's that get replaced and the Replacement Block Number (RBN) that is used. In Automatic replacement mode, display selection will cause the updated Replacement Control Table (RCT) descriptors to be printed at the end of each pass, for each Unit processed. In Manual replacement mode, this question is asked twice: When the Unit is brought Online, the operator may display the existing RCT replacement descriptors. After all desired LBN's have been replaced, the operator may display the updated RCT replacement descriptors.

Remember that the period used after a number indicates that the number is displayed in decimal.

RBN: 455. is SECONDARY replacement for LBN: 23271.
RBN: 456. is PRIMARY replacement for LBN: 23272.
RBN: 457. is UNUSABLE

After having described the "status" of all replacements logged in the Replacement Control Table, the following summary is provided:

RCT DESCRIPTOR CONTENTS SUMMARY
Primary replacement blocks: ddd.
Secondary replacement blocks: ddd.
Unusable replacement blocks: ddd.
Total RBNs on media: dddddd.

Displaying this information will provide an indication of the TOTAL number of replacements that are "logged" in the RCT. These replacements may have been made by the Operating System (If the O/S has the capability of doing Bad Block Replacement), the formatter or possibly this program. Although there is no "specification" for how many replacements are too many, you should be aware of certain conditions that would indicate a problem. An RAB1 can accommodate 17,472 total replacements. A good working HDA can have a thousand primary replacements, however, the number of secondary replacements

should be small (Like on a ratio of about 100 primary's to 1 secondary or so). If you displayed something more than a thousand primary's or more than a few hundred secondaries, it could indicate that you may have a data path problem or had one in the past that was repaired. ECC errors being generated by a Data Path problem can cause a significant number of good blocks to be replaced before the data path problem can be repaired. If you see what may appear to be this condition, you should read RA81-TT-12. The Tech Tip applies to all RA drives, even though it is written as an RA81 Tech Tip. If this is the condition you sense, the tech tip will recommend that you reformat the HDA. Now that this program exists, it would be wise to not only reformat, to recover from this condition, but to run as many passes of this program as possible, after having reformatted. Any bad blocks that the formatter may not have found, may be found by this program and replaced.

6.5.2 ENABLE REPLACEMENTS (L) Y ?

This is a unique feature of this program. Disabling replacements (Answering with a NO) makes this program run in a similar fashion to the HSC50 "Verify" program. In other words, this program will "Scan" the media and report anything it finds but not do any replacements. DO NOT (REPEAT DO NOT) EXPECT THE SAME RESULTS FROM THIS PROGRAM WHEN COMPARED TO THE HSC50 "VERIFY" PROGRAM. THE HSC50 CAN LOWER THE "DRIVE THRESHOLD" WHERE THIS PROGRAM CAN NOT. Using it in this mode could possibly help you determine the "condition" of the media, without having any replacements occur. AGAIN -- THIS PROGRAM IS NOT (REPEAT NOT) INTENDED FOR USE AS A DIAGNOSTIC AND SHOULD NOT BE RUN ON DRIVES/CONTROLLERS THAT MAY HAVE PROBLEMS (Other than normal bad blocks and the like).

Using this program in this mode (Answering with a "NO") will disable all writes and therefore you can run it without the normal requirement for a back-up and restore. However, we can not be held responsible for any errors or subsystem problems that may create situations that cause this program (or subsystem) to do unexpected things.

WE HIGHLY RECOMMEND THAT YOU RUN IN THIS MODE (ANSWER = NO), WITH THE DRIVE(S) WRITE PROTECTED !!

What you can do is run with replacements disabled just to see if its worth your time to do the extensive Backup and Restore operations that are required when replacements are enabled. Remember you can not run this program with replacements enabled and the drives write protected. One pass may not find all the "marginal" bad spots. Thus we highly recommend that when running in the "disable replacement" mode (Answer=NO) that you allow the program to run several passes.

This mode can also be used to get a quick "look" at the number and types of replaced blocks that are "logged" in the Replacement Control table (RCT). By going into this mode, and answering the question "Display Replacement Descriptors" with a yes, you can get these typed without the Backup and Restore. When analyzing the descriptors that are typed, be sure you read and understand the discussion for the question "Display Replacement Descriptors"

(above).

6.5.3 ENABLE WRITE WITH FORCED ERROR FLAG (L) Y ?

This is not going to be easy to explain in 10 words or less. First you have to understand what a "Forced Error Flag" is. Many people have the wrong impression of what a "Forced Error" or "Forced Error Flag" is and what it does. Let me try to briefly explain. YOU NEED TO KNOW THIS.

When an uncorrectable ECC error is encountered, several attempts are made to read the data correctly (Using every means available). If these attempts fail, the block generating the Uncorrectable ECC error is assumed to be bad and in need of replacement. Thus, if you have a system that does "Dynamic Bad Block Replacement" (Like VMS, RSTS, RSX, IAS), or your running this program, the replacement process will take the uncorrectable (Corrupt) data and move it to a good replacement block (RBN). Now, we have a condition where we have corrupt data in a good block and if left this way the corrupt data would be read with no "indication" that the data is "corrupt". Therefore, in order to tell a user that the data was at one time uncorrectable and thus exists now as "corrupted" data (In a good RBN), the "Forced Error Flag" is attached to the block of "corrupted" data. When this block is read by a user, this flag (Inverted EDC character) is also read and is intended to inform the user that the requested data is "not reliable". THE FORCED ERROR FLAG IS NOT (REPEAT NOT) AN ERROR!! WHEN READ, IT WILL NOT MAKE AN ENTRY IN THE SYSTEM ERROR LOG. IT IS REPORTED AS A "STATUS" CODE 10 (OCTAL) IN A TRANSFER REQUEST "END PACKET" (This program calls the end packet status field the "Endmsg status", in several message types that can be displayed)

So why do we have this question. It seems that one or more Operating Systems have trouble dealing with, or reporting, the "Forced Error Flag". Also, some others do not have an intelligent way of reporting the flag to the user (Makes it appear as a "hardware error". Currently (March 1985) UNIX type systems (ULTRIX, Berkeley UNIX, AT&T UNIX etc) can "give up" (I assume that means crash) in certain situations when a "Forced Error Flag" is read. This being what we are told, we added this question to allow the user to disable the function that writes the forced error flag to an RBN with the uncorrectable data.

NOW LISTEN CAREFULLY.

If you answer the question "Enable write with Forced Error Flag" with a NO, you will put uncorrectable data into a good RBN (Just normal replacement). The "corrupted" data will read good, with NO indication of an error or the "corrupted data flag" (Forced error Flag). Therefore, the possibility of reading corrupted data with NO indication that it is corrupted. IF YOU FOLLOW THE RECOMMENDED PROCEDURE FOR RUNNING THIS PROGRAM AND DO THE BACKUP, AND THEN RESTORE THE BACKED-UP DATA AFTER RUNNING THIS PROGRAM, YOU WILL NOT HAVE A PROBLEM. Restoring the backed-up data, to a drive that had this question answered with a NO, should eliminate any problem condition for these systems. For those systems that do not

know how to handle the "Forced Error Flag", following the procedure of answering the question with a no and then doing a backed-up data restore is a REQUIREMENT WITHOUT EXCEPTION (repeat WITHOUT EXCEPTION)!!

7.0 "HARD" ERROR REPORTING

This program can display information from Error Log Packets and other sources of error information, that the controller/drive may send. This may not be what you think, due to the fact that THIS PROGRAM IS NOT (Repeat NOT) INTENDED TO BE USED ON RA DRIVES, OR CONTROLLERS, THAT ARE DEFECTIVE. You must be sure that you are trying to eliminate bad blocks, before using this program. This program will not give you all the details of an error that occurred (Like a Fault) but will give you the most important information AFTER HAVING GIVEN YOU THE ERROR INFORMATION IT WILL DROP THE DRIVE AND TERMINATE. This is for your protection. Doing replacements on a defective drive (or controller) can cause BIG TROUBLE. Since we can not control the use of the program, we must always assume the worst and try to prevent one from creating bigger problems from the misuse of this program.

If you see the following kinds of errors reported, you should discontinue the use of this program and use the other diagnostics, or system error logger, to better isolate any problems. If this program reports Hard Errors, like the following examples, their is always a possibility of a drive/controller generating a hard error that will cause an incomplete replacement. If you attempted to use this program, and noticed an error like the following examples, you should take action to better analyze the problem, using the other diagnostics. Once you feel that the subsystem is working properly, you should run at least one pass of this program, in automatic mode, to take care of any incomplete replacements and any other bad blocks.

The following examples list the possible errors reported by error log packets that will cause the program to terminate.

NOTE

```

+-----+
) The little "o" used for the Status/event, Host address and      )
) Drive Error Code means the information is typed in Octal. The )
) little "d" means decimal and the little "x" means Hexadecimal. )
+-----+

```

7.1 Errors Reported by Error Log Packets from the Controller

There are several sources of error information in an RA subsystem. One of the most common, is error log packets that the controller assembles and sends to the host. There are 5 types of error log packets. Three of these types can be reported by this program.

7.1.1 Controller Error reported

This error occurs when an Error log packet is received specifying that a Controller detected an error within itself. The

Unit being tested will be dropped. This type of error will be reported in the following manner:

Controller Error Reported:
Status/event: 00000

The Status/Event code will give you an indication of what the trouble may be. A list of the Status/Event codes and their meanings are given in section 7.2

7.1.2 Host Memory Access Error Reported

This error occurs when an Error log packet is received specifying that a Host Memory Access Error was detected. This type of error packet reports problems that the controller experienced "dealing" with host memory (Like while doing data transfers etc). This type of error will be reported in the following manner:

Host Memory Access Error reported;
Status/event: 00000
Host address: 00000

The host bus (Like UNIBUS) address is displayed and was the one being used at the time of the error. The Status/Event code will give you an indication of what the problem was. A list of the Status/Event codes and their meanings are given in section 7.2

7.1.3 SDI Error Reported

This error occurs when an Error log packet is received specifying that an SDI Error or Drive Detected Error was detected. If a Status/Event code of 353 (S/E Codes are in Octal) is reported, a valid Drive error code may be reported. The Drive Error Code is displayed along with the Drive dependent information in the packet, IN HEXADECIMAL. This type of error will be reported in the following manner.

SDI Error reported; LBN: dddddd.
Status/event: 00000
Drive error code: xxxxxx
Drive dependent information (hex): xx xx xx xx xx xx xx xx

The LBN is the Logical Block Number that was being accessed at the time the error was detected. The LBN is always reported by this program in decimal. The "Drive Dependent Information", and Drive Error Code, are typed in HEX because that's the way this information is provided in most of the drive service manuals. YOU CAN LOOK-UP THE DRIVE ERROR CODE IN THE DRIVE SERVICE MANUAL FOR THE MEANING OF THE REPORTED PROBLEM. Most of the drive service manuals (Like RA80 and RA81) provide some information on the meanings of the bytes displayed for the Drive Dependent information. This information is sometimes known as the "drive specific status bytes" of the "extended status". Re-stating this information as the byte number of the drive extended status, it would appear as:

Drive dependent information : 9 8 11 10 13 12 15 14

Where the byte number is the byte number of the drive specific status. As mentioned, many of the service manuals provide information on the meaning of these bytes.

7.2 STATUS/EVENT CODES

The following list shows the various Status/Event codes used in many of the messages displayed by this program. Certain deviations may exist for specific controllers. This is the "generic" list. Remember, this program reports Status/Event codes in Octal.

Octal	Hex	DESCRIPTION
0	0	Successful completion
1	1	Invalid command (high byte = byte offset of bnd command field)
2	2	Command aborted
3	3	Drive off line (unknown unit or Online to another controller)
4	4	Drive available
5	5	Media Format Error
6	6	Unit Write Protected
7	7	Compare error (on compare command or compare modifier)
10	8	Data Error (Or may have been written with "Forced Error Flag")
11	9	Host Buffer Access Error
12	A	Controller Error (Command Time Out/Retry Exceeded)
13	B	Drive Error
40	20	Spin down ignored (multi-unit drives only)
43	23	No volume mounted or drive run/stop switch out
45	25	Format Control Table unreadable - EDC Error
51	29	Odd transfer start address in MSCP packet
52	2A	SERDES overrun error (controller probably broken)
* 53	2B	SDI level two response timeout (Or maybe se : incomplete) ('53 also shows up with drive errors as a result NOT CAUSE)
100	40	Still Connected (multi-unit drives only)
103	43	Drive inoperative (UDA cannot communicate with drive)
105	45	Format Control Table unreadable - Invalid Sector Header
110	48	Header compare error (header not found and no revector)
111	49	Odd byte count in MSCP packet
112	4A	EDC error (SERDES broken or EDC written bad -- Controller)
* 113	4B	Invalid SDI level two response (unsuccessful or trash)
145	65	Format Control Table unreadable - Data Sync Timeout
150	68	Data Sync timeout (Data Sync field in sector)
151	69	UNIBUS nonexistent memory error
152	6A	Inconsistent controller state
153	6B	Positioner Error (headers consistent but not on cyl)
200	80	Duplicate Unit Numbers
203	83	Drive offline and duplicate unit numbers
211	89	UNIBUS parity error (UNIBUS read)(Host Memory Parity?)
* 213	8B	Lost read/write ready during or between transfers ('213 also shows up with drive errors as a result NOT CAUSE)
245	A5	Drive not 512 Byte format (16 bit format)
253	AB	Lost Drive clock during data/SDI transfer
305	C5	Drive not formatted or Format Control Table Corrupted

* 313	CB	Lost drive receiver ready during transfer (see 213 comment)
345	ES	ECC error on FCT read (media format - FCT unreadable)
750	EB	Uncorrectable ECC error (This uncorrectable data may be re-written to an RBN with the Forced Error Flag)
353	EB	Drive detected error (DRIVE HAD ERROR! -- Find drive error code)
400	100	Already Online (in response to ONLINE)
403	103	Drive offline. By field service or internal Diagnostics
410	108	One symbol ECC error
413	108	RTDS pulse/parity error (IS UDA M7486 UP TO REV ?)
450	128	Two symbol ECC error +-----+-----+
510	148	Three symbol ECC error) * NOTE: Codes 53,113,213 and 313 occur)
550	168	Four symbol ECC error) most often as the RESULT of a "Drive)
610	188	Five symbol ECC error) Error" (S/E Code 353 Octal). You)
650	1A8	Six symbol ECC error) should look for a problem in the)
710	1C8	Seven symbol ECC error) drive before believing that these)
750	1E8	Eight symbol ECC error) codes really represent the)
10006	1006	Drive software write protected description of your problem.)
20006	2006	Drive hardware write protected +-----+-----+

KEY: EDC = Error Detection Code (Written in each sector)
SDI = Standard Disk Interface (Drive Bus)
SERDES = SERIALizer/DESerializer (In Controller)
FCT = Format Control Table (Written on media surface)
ECC = Error Correction Code (Written in each sector)

7.3 CONTROLLER INITIALIZATION ERRORS

The following errors can be reported during controller initialization. These are basically "HARD" serious errors and usually indicate a very sick controller. This program, as with any operating system, must first initialize the controller, to prepare it for use. For example, in the UDA50 and KDA50 type controllers, this is a 4 step process. Errors that are detected during this initialization are reported via the following types of messages.

7.3.1 Controller Address Error Occurred While Accessing Controller Register

This error occurs when the program attempts to read a controller register and a memory access error occurred. This is most likely caused by the operator specifying an incorrect controller address. The specified Unit will be dropped from testing. This type of error will be reported in the following manner:

```
CONTROLLER ADDRESS ERROR OCCURRED WHILE ACCESSING CONTROLLER REGISTER
SA address: 000000
Check controller address.
```

The SA address displays the UNIBUS address of the controller register that was being accessed at the time of the error. This is most likely caused by the operator specifying an incorrect controller address. Unit is dropped from testing.

7.3.2 Controller Resident Diagnostics Detected Failure

This error indicates that the controller resident diagnostics reported an error during initialization. This type of error

will be reported in the following manner:

```
CONTROLLER RESIDENT DIAGNOSTICS DETECTED FAILURE;  
SA address: 000000  
SA contents: 000000
```

The SA register contains the detected error code. The specified Unit will be dropped from testing. Decoding information for the SA register contents (error code) is in section 7.4

7.3.3 Step Bit Error In SA Register During Initialization;

This error indicates that during the 4 step (phase) initialization of the controller a error was detected. This type of error will be reported in the following manner:

```
STEP BIT ERROR IN SA REGISTER DURING INITIALIZATION;  
SA address: 000000  
SA contents: 000000  
SA expected: 000000
```

The SA address is the UNIBUS address of the controller register that was being accessed, during the 4 step initialization of the controller, when the error was detected. The SA contents is what the controller register contained the displayed the error condition. SA expected is what this program expected, for correct results during that step in the initialization. Decoding information for the SA register contents (error code) is in section 7.4. The specified Unit will be dropped from testing.

7.3.4 SA Register Did Not Zero After Step 3 Write;

This error occurs when the SA register does not zero during Step 3 of controller initialization. This type of error will be reported in the following manner:

```
SA REGISTER DID NOT ZERO AFTER STEP 3 WRITE;  
SA address: 000000  
SA contents: 000000  
SA expected: 000000
```

The meanings of the SA address, SA contents and SA expected are the same as those for the previous error (Section 7.3.3) Decoding information for the SA register contents (error code) is in section 7.4. The specified Unit will be dropped from testing.

7.3.5 SA Register Error During Initialization.

This is the more common of the Controller Initialization Errors that can be reported. The SA register Error bit (Bit 15 should be set) was set during controller initialization. This type of error will be reported in the following manner:

```
SA REGISTER ERROR DURING INITIALIZATION.
```


SA address: 000000
 SA contents: 000000
 SA expected: 000000

The meanings of the SA address, SA contents and SA expected are the same as those for the previous error (Section 7.3.3) Decoding information for the SA register contents (error code) is in section 7.4. The specified Unit will be dropped from testing.

7.3.6 Controller Did Not Clear Ring Structure In Host Memory;

An error occurred during initialization of the Host communication area. At a certain point during host/controller initialization, the controller is responsible for clearing the host communication area (Rings) and this error results from a failure of the controller to perform this function (As detected by the host).

CONTROLLER DID NOT CLEAR RING STRUCTURE IN HOST MEMORY;
 Host address: 000000

The Host address is a host memory location in the host communication area that the controller was to have cleared. The problem could be a problem in the controller or possibly the controller had problems accessing the hosts memory locations of the communications area.

7.4 CONTROLLER INITIALIZATION (BY SA REGISTER) ERROR CODES

```

15          11 10          0
+-----+-----+-----+
|E)S)S)S)S) | INITIALIZATION | Formatted as would be represented when
|R)4)3)2)1) | ERROR CODE   | an error occurs during initialization.
+-----+-----+-----+
+-----+
|)-----)
| \-----) STEP BITS. Initialization is a four step process.
|          | These bits describe which initialization step the error
|          | occurred in.
|          |
|          | ERROR BIT. Qualifies the other bits of the register as
|-----) | containing valid information relating to an error that
|          | occurred during initialization.
    
```

UDA DURING INITIALIZATION
 SA REGISTER ERROR CODES

Octal	Error Description	Possible FRU
104000	Fatal Sequence Error	7485
104040	D processor ALU	7485
104041	D processor control rom parity error	7485
105102	D proc with no board #2 or RAM Parity error	7486
105105	D proc RAM Buffer Error	7486
105152	D proc SDI Error	7486
105153	D proc Write Mode Wrap SERDES Error	7486

105154	D proc Read Mode SERDES, RSGEN & ECC Error	7486
106040	U proc ALU Error	7485
106041	U proc Control Register Error	7485
106042	U proc DFAIL/Control Rom Parity/BD #1 tst cnt	7485
106047	U proc Constant PROM error with D proc running SDI test	7485
106055	Unexpected trap found, Abort Diagnostic	7485
106071	U proc constant PROM Error	7485
106072	U proc Control ROM Parity Error	7485
106200	Step 1 Data Error (MSB not set)	7485
107103	U proc RAM Parity Error	7486
107107	U proc RAM Buffer Error	7486
107115	Test Count was wrong (BD#2)	7486
112300	Step 2 Error	7485
122240	NPR Error	7485
122300	Step 3 Error	7485
142300	Step 4 Error	7485

7.5 HOST/CONTROLLER COMMUNICATION ERRORS

The following errors can be reported during normal program execution. These are basically "HARD" serious errors and usually indicate a very serious problem. Errors that are reported via the "Status/Address" (SA) register are reported via the following types of messages.

7.5.1 NO INTERRUPT RECEIVED FOR 30 SECONDS

This error occurs when an interrupt timeout occurs during wait for an MSCP end message. An end message indicated to the host that the controller has completed the command that it was given.

```
NO INTERRUPT RECEIVED FOR 30 SECONDS;
SA address: 000000
SA contents: 000000
```

The Host got tired of waiting for a response from the RA drive controller. If the end packet was not received by the host, the cause does not always necessarily mean that the controller was at fault. The SA address is the UNIBUS address of the controller Status/Address register that was being used at the time the problem was detected by the host. If the controller detected a problem, that accounted for the end packet not being delivered to the host. The Status/Address (SA) register contents should show an error code and the SA register error bit should be set (Bit 15). The specified Unit will be dropped from testing.

7.5.2 Fatal Error Reported In SA Register

This error can occur when the Error Bit is set in the SA register, during normal on-line use of the controller. The error bit (Bit 15) setting causes the controller to discontinue fetching commands from the host. The host will read the SA register, detecting that an error condition exists and display contents of the SA register in

this message.

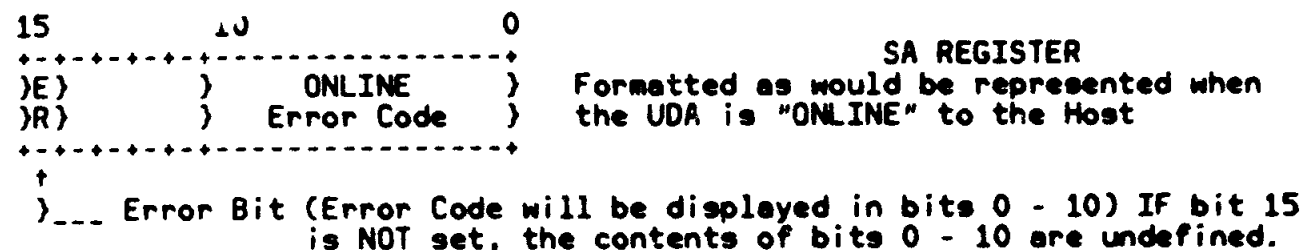
FATAL ERROR REPORTED IN SA REGISTER;

SA address: 000000
 SA contents: 000000

The SA register contents should have the error bit set (bit 15) and an error code. Section 7.6 has a list of the On-line error codes and a brief description of their meanings. Do not get these on-line error codes mixed-up with the initialization error codes that are described in section 7.4. The specified Unit will be dropped from testing.

7.6 CONTROLLER COMMUNICATION ERROR CODES

This section provides a list of the various controller error codes that can reside in the SA register, when the controller detects a hard error during on-line use of the controller. Do not get these confused with the SA register initialization error codes described in section 7.4. They are different



UDA ONLINE
 SA REGISTER ERROR CODES

octal	Error Description	FRU
100001	UNIBUS Packet Read Error	7485*
100002	UNIBUS Packet Write Error	7485*
100003	UDA ROM & RAM Parity Error	7485/7486
100004	UDA RAM Parity Error	7486
100005	UDA ROM Parity error	7485
100006	UNIBUS Ring Read Error	7485*
100007	UNIBUS Ring Write Error	7485*
100010	UNIBUS Interrupt Master Failure	7485
100011	Host Access Timeout Error	7485*
100012	Host Exceeded Command Limit	7485*
100013	UNIBUS Bus Master Failure	7486
100014	DM XFC Fatal Error	7486
100015	Hardware Timeout of Instruction Loop	7485*
100016	Invalid Virtual Ckt Identifier	7485*
100017	Interrupt Write Error on UNIBUS	7485*

* - denotes possible host CPU error.....

7.7 UNIT INITIALIZATION ERRORS

The following errors are reported when an error occurs during the initialization of a unit (Drive). These are basically "HARD" serious errors and usually indicate a problem either in a drive or communicating with a drive. Errors are reported via the "End Message" Status field (Endmsg status:). If the problem is severe, the specified unit (drive) will not continue being tested.

NOTE

```
-----  
) The meanings of the Endmsg status can be found by looking )  
) it up in the list of Status/Event codes provided in section )  
) 7.2 )  
-----
```

7.7.1 Failed Get Unit Status;

"Get Unit Status" is an MSCP command that is used by the host to get certain information about a drive connected to an RA drive controller. This error occurs when an attempt to "GET UNIT STATUS" fails during Unit initialization. This type of error will be reported in the following manner:

```
FAILED GET UNIT STATUS;  
Drive ddd is not accessible.  
Endmsg status: 000000
```

Drive ddd is the decimal drive logical unit address that the "Get Unit Status" was issued for. Possibly the drive port switches are not pushed in or the incorrect drive address was given to the program. Errors are reported via the "End Message" Status field (Endmsg status:) and the various Endmsg status codes can be found in the table of Status/Event codes in section 7.2

7.7 2 FAILED SET CONTROLLER CHARACTERISTICS;

"SET CONTROLLER CHARACTERISTICS" is an MSCP command that is used by the host to set certain controller characteristics (Like timeouts and to enable various kinds of messages). This error occurs when command completion status indicates a problem executing this command. This type of error will be reported in the following manner:

```
FAILED SET CONTROLLER CHARACTERISTICS;  
Host access timeout not disabled.  
Default timeout is 60 seconds.  
Error logging is not enabled.  
Endmsg status: 000000
```

The default host access timeout is retained, and error logging will not be enabled. Errors are reported via the "End Message" Status field (Endmsg status:) and the various Endmsg status codes can be found in the table of Status/Event codes in section 7.2. Processing of the specified unit (drive) will continue. However, if this

error message is seen, the user should terminate the test and try running it over. If the problem is again reported, the problem should be investigated using other diagnostics.

7.7.3 Failed ONLINE

"ONLINE" is an MSCP command that is used by the host to bring a unit (drive) Online to the controller. It also makes certain drive specific information available to the host and the drive to become usable. This error occurs when command completion status indicates a problem executing this command successfully. This type of error will be reported in the following manner:

```
Failed ONLINE;  
  Endmsg status: 000000
```

Errors are reported via the "End Message" Status field (Endmsg status:) and the various Endmsg status codes can be found in the table of Status/Event codes in section 7.2. Processing of the specified unit (drive) will not continue for the pass in progress.

7.8 PROGRAM INITIALIZATION ERRORS

The following sections list the possible errors reported during initialization of program tables, etc. These errors could indicate a possible "system" problem or programming error.

7.8.1 Failed Dynamic Memory Allocation For Selected Unit

This error occurs when the memory table allocation fails. Possibly insufficient memory exists on the system to support the testing of the specified number of drives. The error is reported in the following manner:

```
FAILED DYNAMIC MEMORY ALLOCATION FOR SELECTED UNITS;  
  RESTART PROGRAM AND SELECT FEWER UNITS.
```

7.9 MSCP COMMAND/RESPONSE ERRORS

The following sections list the possible errors reported that indicate the failure of an MSCP command issued by the host. Remember that a period after a number indicates that the number is provided in Decimal (Like the LBN below).

7.9.1 Failed READ

This error occurs when an MSCP "READ" command fails with an end message status (Endmsg status:) other than Success or Data Error. The error is reported in the following manner:

Failed READ; LBN: dddddd.
Endmsg flags: 000
Endmsg status: 000000

This error can be the result of many different conditions. The host issued a "READ" command to the controller being used for testing and then received an End Packet indicating that an error was detected while trying to execute that command. The LBN is the one the "READ" was attempting to access, at the time the error occurred, as reported by the End Packet. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The Unit (drive) being tested will be dropped from further testing.

7.9.2 Failed WRITE

This error occurs when an MSCP "WRITE" command fails with an end message status (Endmsg status:) other than Success or Data Error. The error is reported in the following manner:

Failed WRITE; LBN: dddddd.
Endmsg flags: 000
Endmsg status: 000000

This error can be the result of many different conditions. The host issued a "WRITE" command to the controller being used for testing and then received an End Packet indicating that an error was detected while trying to execute that command. The LBN is the one the "WRITE" was attempting to write at the time the error occurred, as reported by the End Packet. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The Unit (drive) being tested will be dropped from further testing.

7.9.3 Failed ACCESS

This error occurs when an MSCP "ACCESS" command fails with an end message status (Endmsg status:) other than Success or Data Error. PLEASE READ THIS COMPLETE ERROR DESCRIPTION. The error is reported in the following manner:

Failed ACCESS; LBN: dddddd.
Endmsg flags: 000
Endmsg status: 000000

This error can be the result of many different conditions. The host issued an "ACCESS" command to the controller being used for testing and then received an End Packet indicating that an error was detected while trying to execute that command. The LBN is the one the "ACCESS" was attempting to read at the time the error occurred, as reported by the End Packet. An ACCESS command is the MSCP command used by this program to read all the LBN's on the media. The ACCESS command causes a read operation, checks for any

error conditions (Bad Block), but does not transfer any data to the host memory. This purpose for this command is to verify that data can be read without error. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The Unit (drive) being tested will be dropped from further testing.

7.9.4 Failed REPLACE

This error occurs when an MSCP "REPLACE" command fails with an end message status (Endmsg status:) other than Success or Data Error. PLEASE READ THIS COMPLETE ERROR DESCRIPTION. The error is reported in the following manner:

```
Failed REPLACE; LBN: dddddd.  
Endmsg flags:    000  
Endmsg status:  000000
```

This error can be the result of many different conditions. The host issued a "REPLACE" command to the controller being used for testing and then received an End Packet indicating that an error was detected while trying to execute that command. The LBN is the one the "REPLACE" was attempting to use at the time the error occurred, as reported by the End Packet. A REPLACE command is the MSCP command used by programs that do bad block replacement to actually cause the bad LBN to be logically replaced. The execution of the REPLACE command is very involved and leaves little room for error. The header of the bad LBN is written with special codes and the replacement RBN is allocated to the bad LBN. IF THIS ERROR IS SEEN, WE RECOMMEND THAT THE MEDIA RE REFORMATTED USING THE FIELD FORMATTER PROGRAM. HOWEVER, THE FAILURE OF THIS COMMAND COULD BE DUE TO A DEFECT IN THE DRIVE, OTHER THAN MEDIA BAD BLOCKS. BE SURE THAT ANY OTHER CAUSES FOR A FAILURE LIKE THIS IS INVESTIGATED AND ANY APPROPRIATE REPAIRS MADE, BEFORE ATTEMPTING TO REFORMAT THE MEDIA. A FAILURE LIKE THIS ONE IS ONE OF THE GOOD REASONS FOR USING THE FORMATTER, HOWEVER, BE SURE NOT TO USE "RECONSTRUCT" MODE OF THE FORMATTER (SEE RA81-TT-19). The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The Unit (drive) being tested will be dropped from further testing.

7.9.5 Command/Response reference number mismatch

In Mass Storage Control Protocol (MSCP) commands that are issued to the controller are given a "unique" number. In this way, the host can distinguish this command, and any responses, from any other commands that may be issued. When a response to a command is received, the host attempts to associate the "unique" reference number to a command that has not yet received a response. If a response reference number can not be matched to a command with the same number (from commands that have not had responses yet) then this error occurs. This error is reported in the following manner:

```
Command/response reference number mismatch;  
Command ref: 000000  
Endmsg ref: 000000  
Endmsg flags: 000  
Endmsg status: 000000
```

The user should find that the Command Ref and Endmsg ref do not match. Why this would happen, is hard to say. The source for this kind of trouble, is not usually the drive being tested. Rather it is more likely that a controller or "system" problem exists. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. For this kind of error, you may find that the Endmsg flags and Endmsg status will not indicate an error. In this case, possibly a "system" type error would be more likely. The Unit (drive) being tested will be dropped from further testing.

7.9.6 Fatal End Message status detected

An "end message" is the means by which the controller tells the host about how the command was processed and whether any errors occurred while executing a command. This error is indicating that a fatal status was reported by an end message. The error is reported in the following manner:

```
Fatal end message status detected;  
Endmsg flags: 000  
Endmsg status: 000000
```

Since the error is reporting Fatal end message status, you should find the an "Endmsg status" code typed. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The drive being tested will be dropped.

7.9.7 Failed Replacement Verification

After the "REPLACE" command has been issued and completed with good results, the customers data must be written to the replacement block (RBN). At this point in time, the original bad LBN has been written with a code that will cause all references to the LBN address to be "revector" to the replacement block. Thus the host issues a "WRITE" command, with the compare modifier, and the customer data will end up in the replacement block. If this process reports a problem, this error message will result:

```
Failed Replacement verification; LBN: dddddd.  
Retry of replacement operation will be attempted.  
Endmsg flags: 000  
Endmsg status: 000000
```

The program assumes that the cause for the error is that the

replacement block is bad. This can happen. Thus, a retry of the replacement process takes place. This time another replacement block is used and the original one is marked unusable. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Endmsg flags are provided in section 7.10. You should look these up in the charts to get a possible cause for the error. The drive being tested will not be dropped and the replacement retry will be attempted.

7.10 END MESSAGE FLAGS (Endmsg flags)

FLAGS

Bit flags, collectively called end flags, used to report various conditions detected due to this command but not directly related to success or failure. The following flags are defined:

Bad Block Reported: (200 OCTAL -- Bit 7)

Set if one or more bad blocks were detected and the host is expected to perform bad block replacement. Indicates that the host should replace the bad block identified by the LBN:

Bad Blocks Unreported: (100 OCTAL -- Bit 6)

Set if one or more bad blocks were detected and not reported in the "first bad block" field of an End Message Packet.

If the "Bad Block Reported" flag is also set, this indicates that two or more bad blocks were detected, and the host (This program) should perform bad block replacement. In this case the "first bad block" (LBN:) field only reports the first bad block in the transfer. If this happens, the program should be run for multiple passes.

Error Log Generated: (40 OCTAL -- Bit 5)

Set if one or more error log messages were generated that refer to this command -- i.e., that contain this command's command reference number. This flag allows the host to save any outstanding command context that it wishes to include in the error report. The MSCP server must send the error log messages either before or shortly after it sends the end message containing this flag. Depending upon the type of error log report referenced by this flag, this program will determine whether to display the error log report or not. In most cases, the Disk Transfer Error Log report is used to report blocks that need replacement, thus reporting this error log report is somewhat meaningless, since the program will make a replacement due to having received this error log type.

7.11 RCT READ/WRITE ERRORS

The following sections list the possible errors reported when a READ or WRITE to the Replacement Control Table occurs. The RCT does not require special MSCP commands, just normal READ and WRITE commands with the LBN address such that the commands are going to reference blocks in the RCT. The error is reported only once for a particular RCT block. Depending upon the severity of the error

program execution may or may not continue. There must be at least one good copy of an RCT block for execution to continue. There are four copies of the RCT and at least one good one must be found to do replacements. If no good copies can be found, then reformatting the media must be accomplished. Reformatting for this kind of error is a good use of the formatter. However, be sure that a good copy of the Format Control Table (FCT) exists (See RAB1-TT-19).

Replacements can not be done on blocks within the RCT (That's why there are four copies). Thus, you may see RCT errors frequently and why they are not always fatal.

7.11.1 Failed RCT READ

Simply stated, a READ command was issued to a block in one of the copies of the RCT and an error was reported back for that command. This is not necessarily fatal, as there are four copies of the RCT and requirements are that at least one copy should have a good block for the one needed. The error is reported in the following manner:

```
Failed RCT READ;  
RCT copy: ddd  
Block no: ddd  
Endmsg status: 000000
```

The RCT copy defines which of the four copies were being referenced when the error was detected. Block no is the Block number (LBN) that was being referenced within the copy, when the error was detected. The Endmsg status should provide an indication of what the error condition was and can be interpreted from the list of Status/Event codes in section 7.2. Program execution will continue.

7.11.2 Failed RCT WRITE

Simply stated, a WRITE command was issued to a block in one of the copies of the RCT and an error was reported back for that command. This is not necessarily fatal, as there are four copies of the RCT and requirements are that at least one copy should have a good block for the one needed. The error is reported in the following manner:

```
Failed RCT WRITE;  
RCT copy: ddd  
Block no: ddd  
Endmsg status: 000000
```

The RCT copy defines which of the four copies were being referenced when the error was detected. Block no is the Block number (LBN) that was being referenced within the copy, when the error was detected. The Endmsg status should provide an indication of what the error condition was and can be interpreted from the list of Status/Event codes in section 7.2. Program execution will continue.

7.11.3 Failed READ of all copies of RCT

This is serious. This error occurs when the program detects a

failure during a READ of all copies of a particular RCT block. Replacements can not be done unless enough good RCT blocks exist to make-up a good RCT copy. Not only can this program not do replacements, but neither can any operating system that has the capability of doing replacements. The error is reported in the following manner:

```
Failed READ of all copies of RCT;  
Block no: ddd  
Endmsg status: 000000
```

The Block no is the block (LBN) that was being referenced. This same block exists in each of the RCT copies. Thus, references to this block in each of the copies resulted in an error. The Endmsg status should provide an indication of what the error condition was and can be interpreted from the list of Status/Event codes in section 7.2. The Unit (drive) is dropped from testing.

The appropriate recovery for this kind of error, should be to first make sure that the drive does not have a hardware problem causing data errors that are not related to the media. If your certain that the basic drive is functioning properly, the recovery would be to try reformatting the media. The Format Control Table (FCT) must be able to be used during the format (Read RA81-TT-19). Once this has been accomplished, you can re-run this program, to assure yourself that the drive is now operating properly. If the problem persists, after having reformatted, the HDA/Pack may need replacement.

7.11.4 Failed WRITE of all copies of RCT

This is serious. This error occurs when the program detects a failure during a WRITE of all copies of a particular RCT block. Replacements can not be done unless enough good RCT blocks exist to make-up a good RCT copy. Not only can this program not do replacements, but neither can any operating system that has the capability of doing replacements. The error is reported in the following manner:

```
Failed WRITE of all copies of RCT;  
Block no: ddd  
Endmsg status: 000000
```

The Block no is the block (LBN) that was being referenced. This same block exists in each of the RCT copies. Thus, references to this block in each of the copies resulted in an error. The Endmsg status should provide an indication of what the error condition was and can be interpreted from the list of Status/Event codes in section 7.2. The Unit (drive) is dropped from testing.

The appropriate recovery for this kind of error, should be to first make sure that the drive does not have a hardware problem causing data errors that are not related to the media. If your certain that the basic drive is functioning properly, the recovery would be to try reformatting the media. The Format Control Table (FCT) must be able to be used during the format (Read RA81-TT-19). Once this has been accomplished, you can re-run this program, to assure yourself

that the drive is now operating properly. If the problem persists, after having reformatted, the HDA/Pack may need replacement.

7.11.5 No Null descriptor entry found in RCT

This is serious. The Replacement Control Table (RCT) contains descriptors for 17,000 plus replacement blocks. If they are all used (No Null Descriptor) then something big is wrong. The descriptors could in fact all be used or possibly the RCT has been written with Garbage, which makes it appear that all replacement blocks have been used. Never-the-less, if this error occurs, you should read RA81-TT-12. The recovery will be to reformat the media. If the drive is operating normally, this should fix the RCT, as long as the Format Control Table (FCT) is good (See RA81-TT-19). You can re-run this program after having reformatted, just to be sure the problem has been corrected. The drive will be dropped form further testing. The error is reported in the following manner:

```
No Null descriptor entry found in RCT;  
RCT is probably corrupt.  
Reformat media.
```

8.0 INFORMATIONAL MESSAGES

The following sections describe the information which may be observed on the operator console during program execution.

8.1 Program Initialization

During program initialization, the following messages will appear on the console for the appropriate replacement mode:

1. MANUAL BAD BLOCK REPLACEMENT SELECTED.
2. AUTOMATIC BAD BLOCK REPLACEMENT SELECTED

8.2 Unit Initialization

Each Unit selected for test is brought Online, and the following information is displayed for that unit (Drive). Remember a unit is not necessarily the logical unit address of a drive.

```
INITIALIZATION INFORMATION FOR UNIT n  
Controller: xDA  
Address: 172150 (Or what you provided when starting the program)  
Drive no: d  
Drive type: RAnn  
Volume SN: dddd.  
Volume size: ddddd.
```

-- The controller will be a UDA50, KDA50 or whatever it finds

- Drive no: Is the Drive being initialized logical address
- Drive Type: Is the model of drive seen (Like RA80/81/60 etc)
- Volume SN: This is the Serial Number of the HDA or Pack. This information comes from the Format Control Table (FCT), which is read during the execution of the MSCP "ONLINE" command.
- Volume size: This is the number of user LBN's on the media. For example, the RA81 formatted for a VAX (16 bit) has 891072 (decimal) LBN's. Actually, starting with LBN zero, the last user LBN would be 891071.

8.2.1 Zero RCT size detected -- BBR not supported by this drive

This message is telling you that your running this program on a drive that does not support Bad Block replacement. The zero RCT size means that the drive has no "real" Replacement Control Table and replacements are not possible. If think that you have bad blocks on a drive like this, you should consult the appropriate service manual for action you need to take. The message is reported in the following manner:

```
Zero RCT size detected:
  BBR not supported by this drive.
  Endmsg status: 000004
```

The Endmsg status will most likely not tell you much (Unless something really out of the ordinary is happening). The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2. The Unit (drive) is dropped from testing.

8.2.2 Scanning the RCT...

Just after typing the Initialization information for a drive that is going to be tested (Above), the statement "Scanning the RCT..." is typed. This portion of the program, attempts to find enough good RCT blocks to account for a good complete copy. Any error that it finds doing this is displayed. These messages (Shown Below) are only status messages and DO NOT necessarily mean that the drive is defective. The status is shown in the following manner:

```
RCT copy 1, block 184 ( 891256.) Status/event: 000110
RCT copy 1, block 185 ( 891257.) Status/Event: 000153
RCT copy 2, block 184 ( 892021.) Status/event: 000350
```

These messages do not become critical until the same block number shows an error for each copy. In the above example Block 184 is bad in two copies. This is not fatal, since there are more than 2 copies in our example. You will get a hard error report, if all copies of a block are bad. The Endmsg status can be interpreted from the list of Status/Event codes in section 7.2.

8.3 END OF PASS INFORMATION

The operator is notified of End of Pass, for the drive being tested, by the following:

1. EOM reached. Elapsed runtime hh:mm:ss
The End of Media is reached in Automatic replacement mode when the highest LBN on the media has been tested.
2. EOF detected.
The End of File is reached in Manual replacement mode when the operator enters a null LBN for replacement.

When EOP is reached in either Automatic or Manual mode, the following pass summary is displayed for the drive being tested. This summary will also be displayed by the operator "PRINT" command at the diagnostic supervisor prompt.

REPLACEMENT INFORMATION FOR THIS PASS

```

BAD BLOCKS reported:   ddd.
ECC error detected:    ddd.
FORCED ERRORS detected: ddd.
FORCED ERRORS written: ddd.
PRIMARY replacement:   ddd.
SECONDARY replacements: ddd.
RBNs marked unusable:  ddd.
Total replacements made: ddd.

```

The entries should be self-explanatory, after all the discussion above. The Bottom line indicates the total number of replacements that were made, which should not exceed a few hundred, unless something more is wrong with the drive. The total secondary replacements should be on a ratio of about less than 10 to every 100 primary replacements. The number of Forced Errors written, indicates the number of Uncorrectable ECC (ECH) errors that were encountered. If you have any forced errors written, or forced errors detected, you should not fail to restore the customer data from the back-up that was made, before you started execution of this program. If excessive replacements were made, read RA81-TT-12 for a possible explanation and recovery procedure.

8.4 ELAPSED RUNTIME

In Automatic mode, the following runtime message is periodically displayed to indicate that execution is in progress:

```
LBN dddddd. Elapsed runtime is hh:mm:ss
```

The LBN listed indicates where on the media the program is currently testing; the elapsed runtime is zeroed when each Unit is selected for test.

```

1954      @
1955      ;*LAST REVISION 01-APR-1985
2496      .IDENT /01A/
2513      .SBTTL Program Header
2514      .ENABL AMA,ABS
2515      002000      =      2000
2516
2518      ;**
2519      ; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
2520      ; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.

```

```

2521      ;--
2522
2524
2525 002000      L$NAME::      ;DIAGNOSTIC NAME
      002000      132      .ASCII /Z/
      002001      125      .ASCII /U/
      002002      104      .ASCII /D/
      002003      114      .ASCII /L/
      002004      000      .BYTE 0
      002005      000      .BYTE 0
      002006      000      .BYTE 0
      002007      000      .BYTE 0
      002010      L$REV::      ;REVISION LEVEL
      002010      101      .ASCII /A/
      002011      L$DEPO::      ;0
      002011      060      .ASCII /O/
      002012      L$UNIT::      ;NUMBER OF UNITS
      002012      000001      .WORD T$PTHV
      002014      L$TIML::      ;LONGEST TEST TIME
      002014      000000      .WORD 0
      002016      L$HPCP::      ;POINTER TO H.W. QUES.
      002016      042146      .WORD L$HARD
      002020      L$SPCP::      ;POINTER TO S.W. QUES.
      002020      000000      .WORD 0
      002022      L$HPTP::      ;PTR. TO DEF. H.W. PTABLE
      002022      002132      .WORD L$HW
      002024      L$SPTP::      ;PTR. TO S.W. PTABLE
      002024      002146      .WORD L$SW
      002026      L$LADP::      ;DIAG. END ADDRESS
      002026      047476      .WORD L$LAST
      002030      L$STA::      ;RESERVED FOR APT STATS
      002030      000000      .WORD 0
      002032      L$CO::      .WORD 0
      002032      000000      .WORD 0
      002034      L$DTYP::      ;DIAGNOSTIC TYPE
      002034      000000      .WORD 0
      002036      L$APT::      ;APT EXPANSION
      002036      000000      .WORD 0
      002040      L$DTP::      ;PTR. TO DISPATCH TABLE
      002040      002124      .WORD L$DISPATCH
      002042      L$PRIO::      ;DIAGNOSTIC RUN PRIORITY
      002042      000340      .WORD PRI07
      002044      L$ENVI::      ;FLAGS DESCRIBE HOW IT WAS SETUP
      002044      000000      .WORD 0
      002046      L$EXP1::      ;EXPANSION WORD
      002046      000000      .WORD 0
      002050      L$MREV::      ;SVC REV AND EDIT #
      002050      003      .BYTE C$REVISION
      002051      003      .BYTE C$EDIT
      002052      L$EF::      ;DIAG. EVENT FLAGS
      002052      000000      .WORD 0
      002054      000000      .WORD 0
      002056      L$SPC::      .WORD 0
      002056      000000      .WORD 0
      002060      L$DEVP::      ; POINTER TO DEVICE TYPE LIST
      002060      004300      .WORD L$DVTYP
      002062      L$REPP::      ;PTR. TO REPORT CODE
  
```

Program Header

002062	035132		.WORD	L\$RPT	
002064		L\$EXP4::	.WORD	0	
002064	000000		.WORD	0	
002066		L\$EXP5::	.WORD	0	
002066	000000		.WORD	0	
002070		L\$AUT::	.WORD	0	;PTR. TO ADD UNIT CODE
002070	000000		.WORD	0	
002072		L\$DUT::	.WORD	L\$DU	;PTR. TO DROP UNIT CODE
002072	040146		.WORD	L\$DU	
002074		L\$LUN::	.WORD	0	.LUN FOR EXERCISERS TO FILL
002074	000000		.WORD	0	
002076		L\$DESP::	.WORD	L\$DESC	;PTR. TO DIAG. DESCRIPTION
002076	004324		.WORD	L\$DESC	
002100		L\$LGAD::	.WORD	L\$DESC	;GENERATE SPECIAL AUTOLOAD EMT
002100	104035		EMT	E\$LOAD	
002102		L\$ETP::	.WORD	L\$ERRTBL	;PTR. TO ERRTBL
002102	002166		.WORD	L\$ERRTBL	
002104		L\$ICP::	.WORD	L\$INIT	;PTR. TO INIT CODE
002104	035726		.WORD	L\$INIT	
002106		L\$CCP::	.WORD	L\$CLEAN	;PTR. TO CLEAN-UP CODE
002106	040124		.WORD	L\$CLEAN	
002110		L\$ACP::	.WORD	L\$AUTO	;PTR. TO AUTO CODE
002110	040122		.WORD	L\$AUTO	
002112		L\$PRT::	.WORD	L\$PROT	;PTR. TO PROTECT TABLE
002112	035720		.WORD	L\$PROT	
002114		L\$TEST::	.WORD	0	;TEST NUMBER
002114	000000		.WORD	0	
002116		L\$DLY::	.WORD	0	;DELAY COUNT
002116	000000		.WORD	0	
002120		L\$HIME::	.WORD	0	;PTR. TO HIGH MEM
002120	000000		.WORD	0	


```
1          .SBTTL Dispatch Table
2
3          ;**
4          ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
5          ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
6          ;--
7
8 002122 000002          .WORD 2
002124
002124 040314          L$DISPATCH:: .WORD T1
002126 040624          .WORD T2
9
```

```

1      .SBTTL  DEFAULT HARDWARE P-TABLE
2
3      ;**
4      ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
5      ; THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
6      ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
7      ; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
8      ;--
9
10     002130  000005      .WORD  L10000-L$HW/2
11     002132  000005      L$HW::
12     002132  172150      DFPTBL::
13     002132  172150      .WORD  172150      ; UNIBUS ADDRESS
14     002134  000154      .WORD  154      ; VECTOR ADDRESS
15     002136  000005      .WORD  5.      ; BR LEVEL
16     002140  000077      .WORD  63.     ; UNIBUS BURST RATE
17     002142  000000      .WORD  0.      ; LOGICAL DRIVE NUMBER
18
19     L10000:

```

```
1          .SBTTL  SOFTWARE P TABLE
2
3          ;**
4          ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
5          ; PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
6          ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
7          ; AT RUN TIME.
8          ;--
9
10         002144 000010          .WORD  L10001-L$SW/2
11         002146
12         002146          L$SW::
13         002150 000000          SFPTBL::
14         002152 000001          OPHELP::          .WORD  0          ; Set to print operator help
15         002154 000001          MANRPL::          .WORD  0          ; Set if manual replacement
16         002156 000000          AUTORC::          .WORD  1          ; Set for automatic crash recovery
17         002160 000001          DSPRPL::          .WORD  1          ; Set to display replacements
18         002162 000001          DSPRCT::          .WORD  0          ; If set print RCT descriptors
19         002164 000001          ENBRPL::          .WORD  1          ; Set to disable replacement
20
21         002166          ENBERRL::          .WORD  1          ; Set to enable Errorlog packets
22
23         002166          ENBWFE::          .WORD  1          ; Enable write FE
24
25         L10001:
```

CZUOLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 5
Global equates section

1
2
4
5
6
7
8
9

```
.SBTTL Global equates section

; **
; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
; ARE USED IN MORE THAN ONE TEST.
; **

;
; BIT DIFINITIONS
;
100000 BIT15== 100000
040000 BIT14== 40000
020000 BIT13== 20000
010000 BIT12== 10000
004000 BIT11== 4000
002000 BIT10== 2000
001000 BIT09== 1000
000400 BIT08== 400
000200 BIT07== 200
000100 BIT06== 100
000040 BIT05== 40
000020 BIT04== 20
000010 BIT03== 10
000004 BIT02== 4
000002 BIT01== 2
000001 BIT00== 1

;
001000 BIT9== BIT09
000400 BIT8== BIT08
000200 BIT7== BIT07
000100 BIT6== BIT06
000040 BIT5== BIT05
000020 BIT4== BIT04
000010 BIT3== BIT03
000004 BIT2== BIT02
000002 BIT1== BIT01
000001 BIT0== BIT00

;
; EVENT FLAG DEFINITIONS
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
;
;
; BIT POSITION IN SECOND STATUS WORD
000040 EF.START== 32. ; (100000) START COMMAND WAS ISSUED
000037 EF.RESTART== 31. ; (040000) RESTART COMMAND WAS ISSUED
000036 EF.CONTINUE== 30. ; (020000) CONTINUE COMMAND WAS ISSUED
000035 EF.NEW== 29. ; (010000) A NEW PASS HAS BEEN STARTED
000034 EF.PWR== 28. ; (004000) A POWER-FAIL/POWER UP OCCURRED

;
; PRIORITY LEVEL DEFINITIONS
;
000340 PRI07== 340
000300 PRI06== 300
000240 PRI05== 240
000200 PRI04== 200
000140 PRI03== 140
000100 PRI02== 100
```

000040	PRI01== 40
000000	PRI00== 0
	;
	;OPERATOR FLAG BITS
	;
000004	EVL== 4
000010	LOT== 10
000020	ADR== 20
000040	IDU== 40
000100	ISP== 100
000200	UAM== 200
000400	BOE== 400
001000	PNT== 1000
002000	PRI== 2000
004000	IXE== 4000
010000	IBE== 10000
020000	IER== 20000
040000	LOE== 40000
100000	HOE== 100000

```

1      .SBTTL  XDA bit definitions
2
3
4      ;XDA SA REGISTER UNIVERSAL READ BITS
5
6      100000  SA.ERR  = 100000  ;ERROR INDICATOR
7      040000  SA.S4   = 040000  ;STEP 4 STATUS BIT
8      020000  SA.S3   = 020000  ;STEP 3 STATUS BIT
9      010000  SA.S2   = 010000  ;STEP 2 STATUS BIT
10     004000  SA.S1   = 004000  ;STEP 1 STATUS BIT
11
12
13     ;XDA SA REGISTER ERROR STATUS BITS
14
15     003777  SA.ERC  = 003777  ;ERROR CODE
16
17
18     ;XDASA REGISTER STEP 1 SEND BITS
19
20     000177  SA.VEC  = 000177  ;INTERRUPT VECTOR (DIVIDED BY 4)
21     000200  SA.INT  = 000200  ;INTERRUPT ENABLE DURING INITIALIZATION
22     003400  SA.MSG  = 003400  ;MESSAGE RING LENGTH
23     034000  SA.CMD  = 034000  ;COMMAND RING LENGTH
24     040000  SA.WRP  = 040000  ;WRAP BIT
25     100000  SA.STP  = 100000  ;STEP - MUST ALWAYS BE WRITTEN A ONE
26
27     000400  SA.MS1  = 000400  ;LSB OF MESSAGE RING LENGTH
28     004000  SA.CM1  = 004000  ;LSB OF COMMAND RING LENGTH
29
30
31     ;XDASA REGISTER STEP 1 RESPONSE BITS
32
33     002000  SA.NV   = 002000  ;NON SETTABLE INTERRUPT VECTOR
34     001000  SA.QB   = 001000  ;22 BIT ADDRESS BUS
35     000400  SA.DI   = 000400  ;ENHANCED DIAGNOSTICS
36     000100  SA.MP   = 000100  ;MAPPING BIT
37     ;        000377  ;ALL BITS RESERVED
38
39
40     ;XDASA REGISTER STEP 2 SEND BITS
41
42     000001  SA.PRG  = 000001  ;ENABLE VAX UNIBUS ADAPTER PURGE INTERRUPT
43     ;        177776  ;LOW ORDER MESSAGE RING BYTE ADDRESS
44
45
46     ;XDASA REGISTER STEP 2 RESPONSE BITS
47
48     000007  SA.MSE  = 000007  ;MESSAGE RING LENGTH ECHO
49     000070  SA.CME  = 000070  ;COMMAND RING LENGTH ECHO
50     ;        000100  ;RESERVED
51     000200  SA.STE  = 000200  ;STEP ECHO
52     003400  SA.CTP  = 003400  ;CONTROLLER TYPE
53
54
55     ;XDASA REGISTER STEP 3 SEND BITS
56
57     ;        077777  ;HIGH ORDER MESSAGE RING BYTE ADDRESS
  
```

CZUDLAD BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 5-1
XDA bit definitions

58	100000	SA.TST = 100000	;PURGE POLE TEST ENABLE
59			
60			
61		;xDASA REGISTER STEP 3 RESPONSE BITS	
62			
63	000177	SA.VCE = 000177	;INTERRUPT VECTOR ECHO
64	000200	SA.INE = 000200	;INTERRUPT ENABLE ECHO
65	000400	SA.NVF = 000400	;VECTOR NOT PROGRAMMABLE
66		;	;RESERVED
67			
68			
69		;xDASA REGISTER STEP 4 SEND BITS	
70			
71	000001	SA.GO = 000001	;GO BIT TO START xDA FIRMWARE
72	000002	SA.LFC = 000002	;LAST FAILURE CODE REQUEST
73	000374	SA.BST = 000374	;BURST LEVEL
74			
75			
76		;xDASA REGISTER STEP 4 RESPONSE BITS	
77			
78	000017	SA.MCV = 000017	;xDA MICROCODE VERSION
79	000360	SA.CNT = 000360	;CONTROLLER TYPE
80		;	;RESERVED

```

1      .SBTTL  Host Communication area definitions
2
3      ;COMMAND/MESSAGE RING BIT DEFINITIONS
4
5      100000      RG.OWN  = 100000      ;SET WHEN xDA OWNS RING
6      040000      RG.FLG  = 040000      ;FLAG BIT
7
8
9      ;VIRTUAL CIRCUIT IDENTIFIERS
10
11     000000      MSCP    = 0           ;MSCP CIRCUIT
12     000001      LOG     = 1           ;LOG CIRCUIT
13     177777      DIAG    = -1          ;DIAGNOSTIC CIRCUIT
14     001000      DUP     = 1000        ;DIAGNOSTIC AND UTILITIES PROTOCOL
15
16
17     ;OFFSETS INTO HOST COMMUNICATIONS AREA WITH ONE DESCRIPTOR TO EACH RING
18     ;AND TWO PACKET
19
20     000004      HC.ISZ  = 4.          ;SIZE OF INTERRUPT INDICATOR WORDS
21     000004      HC.RSZ  = 4.          ;SIZE OF RING IN BYTES
22     000004      HC.ESZ  = 4.          ;SIZE OF ENVELOPE WORDS BEFORE PACKET
23     000060      HC.PSZ  = 48.         ;SIZE OF COMMAND AND MESSAGE PACKETS
24
25     000000      HC.INT  = 0.          ;INTERRUPT INDICATOR WORDS START
26
27     000004      HC.MSG  = HC.INT+HC.ISZ ;MESSAGE RING START
28     000006      HC.MCT  = HC.MSG+2.   ;MESSAGE RING CONTROL WORD
29
30     000010      HC.CMD  = HC.MSG+HC.RSZ ;COMMAND RING START
31     000012      HC.CCT  = HC.CMD+2.   ;COMMAND RING CONTROL WORDS
32
33     000014      HC.MEV  = HC.CMD+HC.RSZ ;MESSAGE ENVELOPE START
34     000020      HC.MPK  = HC.MEV+HC.ESZ ;MESSAGE PACKET START
35
36     000100      HC.CEV  = HC.MPK+HC.PSZ ;COMMAND ENVELOPE START
37     000104      HC.CPK  = HC.CEV+HC.ESZ ;COMMAND PACKET START
38
39     000164      HC.SIZ  = HC.CPK+HC.PSZ ;TOTAL SIZE OF HOST COMMUNICATION AREA

```



```
1      .SBTTL Host Communication area layout
2
3      :
4      :      HC.INT  }  INTERRUPT INDICATORS  }  4 BYTES
5      :      }
6      :      +-----+
7      :      :      HC.MSG  }  MESSAGE (RESPONSE) RING  }  4 BYTES
8      :      :      HC.M:1  }
9      :      :      +-----+
10     :      :      HC.C:1D  }  COMMAND RING  }  4 BYTES
11     :      :      HC.CC:1  }
12     :      :      +-----+
13     :      :      HC.MEV  }  MESSAGE (RESPONSE) ENVELOPE  }  4 BYTES
14     :      :      }
15     :      :      HC.MPK  }  MESSAGE (RESPONSE) PACKET  }  48 BYTES
16     :      :      }
17     :      :      }
18     :      :      }
19     :      :      }
20     :      :      +-----+
21     :      :      HC.CEV  }  COMMAND ENVELOPE  }  4 BYTES
22     :      :      }
23     :      :      HC.CPK  }  COMMAND PACKET  }  48 BYTES
24     :      :      }
25     :      :      }
26     :      :      }
27     :      :      }
28     :      :      +-----+
29     :
30     :      NOTE: BYTES ARE GIVEN IN DECIMAL
```

```
1      .SBTTL  Command packet opcodes definitions
2
3      000001  OP.ABO  = 1      ;ABORT COMMAND
4      000020  OP.ACC  = 20     ;ACCESS COMMAND
5      000010  OP.AVL  = 10     ;AVAILABLE COMMAND
6      000021  OP.CCD  = 21     ;COMPARE CONTROLLER DATA COMMAND
7      000040  OP.CMP  = 40     ;COMPARE HOST DATA COMMAND
8      000022  OP.ERS  = 22     ;ERASE COMMAND
9      000023  OP.FLU  = 23     ;FLUSH COMMAND
10     000002  OP.GCS  = 2      ;GET COMMAND STATUS COMMAND
11     000003  OP.GUS  = 3      ;GET UNIT STATUS CCOMMAND
12     000011  OP.ONL  = 11     ;ONLINE COMMAND
13     000041  OP.RD   = 41     ;READ COMMAND
14     000024  OP.RPL  = 24     ;REPLACE COMMAND
15     000004  OP.SCC  = 4      ;SET CONTROLLER CHARACTERISTICS COMMAND
16     000012  OP.SUC  = 12     ;SET UNIT CHARACTERISTICS COMMAND
17     000042  OP.WR   = 42     ;WRITE COMMAND
18     000030  OP.MRD  = 30     ;MAINTENANCE READ COMMAND
19     000031  OP.MWR  = 31     ;MAINTENANCE WRITE COMMAND
20     000200  OP.END  = 200    ;END PACKET FLAG
21     000007  OP.SEX  = 7      ;SERIOUS EXCEPTION END PACKET
22     000100  OP.AVA  = 100    ;AVAILABLE ATTENTION MESSAGE
23     000101  OP.DUP  = 101    ;DUPLICATE UNIT NUMBER ATTENTION MESSAGE
24     000102  OP.SHC  = 102    ;SHADOW COPY COMPLETE ATTENTION MESSAGE
25     000103  OP.RLC  = 103    ;RESET COMMAND LIMIT ATTENTION MESSAGE
26
27     000001  OP.GSS  = 1      ;DUP GET DUST STATUS
28     000002  OP.ESP  = 2      ;DUP EXECUTE SUPPLIED PROGRAM
29     000003  OP.ELP  = 3      ;DUP EXECUTE LOCAL PROGRAM
30     000004  OP.SSD  = 4      ;DUP SEND DUST DATA
31     000005  OP.RSD  = 5      ;DUP RECEIVE DUST DATA
32
33     ;NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING THE END
34     ;PACKET FLAG TO THE COMMAND OPCODE. FOR EXAMPLE, A READ COMMAND'S END PACKET
35     ;CONTAINS THE VALUE OP.RD+OP.END IN ITS OPCODE FIELD. THE INVALID COMMAND END
36     ;PACKET CONTAINS JUST THE END PACKET FLAG (I.E., OP.END) IN ITS OPCODE FIELD.
37     ;THE SERIOUS EXCEPTION END PACKET CONTAINS THE SUM OF THE END PACKET FLAG
38     ;PLUS THE SERIOUS EXCEPTION OPCODE SHOWN ABOVE (I.E., OP.SEX+OP.END) IN ITS
39     ;OPCODE FIELD.
40     ;
41     ;COMMAND OPCODE BITS 3 THROUGH 5 INDICATE THE COMMAND CLASS, WHICH IS ENCODED
42     ;AS FOLLOWS:
43     ; 000 IMMEDIATE COMMANDS
44     ; 001 SEQUENTIAL COMMANDS
45     ; 010 NON-SEQUENTIAL COMMANDS THAT DO NOT INCLUDE A BUFFER DESCRIPTOR
46     ; 100 NON-SEQUENTIAL COMMANDS THAT DO INCLUDE A BUFFER DESCRIPTOR
```

```

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
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

```

			:COMMAND MODIFIERS	
			;	= 020000
			MD.CMP	= 040000
	040000		MD.EXP	= 100000
	100000		MD.ERR	= 010000
	010000		MD.SCH	= 004000
	004000		MD.SCL	= 002000
	002000		MD.SEC	= 000100
	000100		MD.SER	= 000400
	000400		MD.SSH	= 000200
	000200		MD.SBN	= 000100
	000100		MD.WBV	= 000400
	000400		MD.SEQ	= 000020
	000020		MD.SPD	= 000001
	000001		MD.FEU	= 000001
	000001		MD.VOL	= 000002
	000002		MD.NXU	= 000001
	000001		MD.RIP	= 000001
	000001		MD.IMF	= 000002
	000002		MD.SWP	= 000004
	000004		MD.CWB	= 000010
	000010		MD.PRI	= 000001
	000001			
			:END PACKET FLAGS	
			EF.BBR	= 000200
	000200		EF.BBU	= 000100
	000100		EF.LOG	= 000040
	000040		EF.SEX	= 000020
	000020			
			:CONTROLLER FLAGS	
			CF.ATN	= 000200
	000200		CF.MSC	= 000100
	000100		CF.OTH	= 000040
	000040		CF.THS	= 000020
	000020		CF.SHD	= 000002
	000002		CF.576	= 000001
	000001			
			:UNIT FLAGS	
			UF.CMR	= 000001
	000001		UF.CMW	= 000002
	000002		UF.RPL	= 100000
	100000		UF.INA	= 040000
	040000		UF.SCH	= 004000
	004000		UF.SCL	= 002000
	002000		UF.WBN	= 000100
	000100		UF.WPH	= 020000
	020000		UF.WPS	= 001000
	001000		UF.576	= 000004
	000004			

```

;CLEAR SERIOUS EXCEPTION
;COMPARE
;EXPRESS REQUEST
;FORCE ERROR
;SUPPRESS CACHING (HIGH SPEED)
;SUPPRESS CACHING (LOW SPEED)
;SUPPRESS ERROR CORRECTION
;SUPPRESS ERROR RECOVERY
;SUPPRESS SHADOWING
;WRITE-BACK (NON-VOLATILE)
;WRITE BACK (VOLATILE)
;WRITE SHADOW SET ONE UNIT AT A TIME
;SPIN-DOWN
;FLUSH ENTIRE UNIT
;VOLATILE ONLY
;NEXT UNIT
;ALLOW SELF DESTRUCTION
;IGNORE MEDIA FORMAT ERROR
;SET WRITE PROTECT
;CLEAR WRITE-BACK DATA LOST
;PRIMARY REPLACEMENT BLOCK

;BAD BLOCK REPORTED
;BAD BLOCK UNREPORTED
;ERROR LOG GENERATED
;SERIOUS EXCEPTION

;ENABLE ATTENTION MESSAGES
;ENABLE MISCELLANEOUS ERROR LOG MESSAGES
;ENABLE OTHER HOST'S ERROR LOG MESSAGES
;ENABLE THIS HOST'S ERROR LOG MESSAGES
;SHADOWING
;576 BYTE SECTORS

;COMPARE READS
;COMPARE WRITES
;HOST INITIATED BAD BLOCK REPLACEMENT
;INACTIVE SHADOW SET UNIT
;SUPPRESS CACHING (HIGH SPEED)
;SUPPRESS CACHING (LOW SPEED)
;WRITE-BACK (NON-VOLATILE)
;WRITE PROTECT (HARDWARE)
;WRITE PROTECT (SOFTWARE OR VOLUME)
;576 BYTE SECTORS

```

```

1      .SRTTL  Command packet offsets
2
3      ;GENERIC COMMAND PACKET OFFSETS
4
5      000000      P.CRF   = 0.      ;COMMAND REFERENCE NUMBER
6      000004      P.UNIT  = 4.      ;UNIT NUMBER
7      000010      P.OPCD   = 8.      ;OPCODE
8      000012      P.MOD    = 10.     ;MODIFIERS
9      000014      P.BCNT   = 12.     ;BYTE COUNT
10     000020      P.BJFF   = 16.     ;BUFFER DESCRIPTOR
11     000020      P.UADR   = 16.     ;UNIBUS ADDRESS OF BUFFER DESCRIPTOR
12     000034      P.LBN    = 28.     ;LOGICAL BLOCK NUMBER
13
14
15     ;ABORT AND GET COMMAND STATUS COMMAND PACKET OFFSETS
16
17     000014      P.OTRF   = 12.     ;OUTSTANDING REFERENCE NUMBER
18
19
20     ;ONLINE AND SET UNIT CHARACTERISTICS COMMAND PACKET OFFSETS
21
22     000016      P.UNFL   = 14.     ;UNIT FLAGS
23     000020      P.HSTI   = 16.     ;HOST IDENTIFIER / RESERVED
24     000034      P.ELGF   = 28.     ;ERROR LOG FLAGS
25     000040      P.SHUN   = 32.     ;SHADOW UNIT
26     000042      P.CPSP   = 34.     ;COPY SPEED
27
28
29     ;REPLACE COMMAND PACKET OFFSETS
30
31     000014      P.RBN    = 12.     ;REPLACEMENT BLOCK NUMBER
32
33
34     ;SET CONTROLLER CHARACTERISTICS COMMAND PACKET OFFSETS
35
36     000014      P.VRSN   = 12.     ;MSCP VERSION
37     000016      P.CNTF   = 14.     ;CONTROLLER FLAGS
38     000020      P.HTMO   = 16.     ;HOST TIMEOUT
39     000022      P.USEF   = 18.     ;USE FRACTION
40     000024      P.TIME   = 20.     ;QUAD-WORD TIME AND DATE
41
42
43     ;MAINTENANCE READ AND MAINTENANCE WRITE COMMAND PACKET OFFSETS
44
45     000034      P.RGID   = 28.     ;REGION ID
46     000040      P.RGOF   = 32.     ;REGION OFFSET
47
48
49     ;EXECUTE SUPPLIED PROGRAM COMMAND PACKET OFFSETS
50
51     000024      P.DMDT   = 20.     ;DMDT TERMINAL ADDRESS (MAINT WRITE ONLY)
52     000034      P.OVRL   = 28.     ;BUFFER DESCRIPTOR FOR OVERLAYS

```

```

1      .S9TTL End packet offsets
2
3      ;GENERIC END PACKET OFFSETS
4
5      000000      P.CRF      = 0      ;COMMAND REFERENCE NUMBER
6      000004      P.UNIT     = 4.     ;UNIT NUMBER
7      000010      P.OPCD     = 8.     ;OPCODE (ALSO CALLED ENDCODE)
8      000011      P.FLGS     = 9.     ;END PACKET FLAGS
9      000012      P.STS      = 10.    ;STATUS
10     000014      P.BCNT     = 12.    ;BYTE COUNT
11     000034      P.F8BK     = 28.    ;FIRST BAD BLOCK
12
13
14     ;          Error log packet offsets
15
16     000000      L.CRF      == 0.     ; Command reference number
17     000010      L.FMT      == 8.     ; Format specifier
18     000011      L.FLGS     == 9.     ; Error log flags
19     000012      L.EVNT     == 10.    ; Event code
20     000030      L.BADR     == 24.    ; Host address
21     000050      L.HDCD     == 40.    ; Header code (LBN)
22     000054      L.SDI      == 44.    ; SDI information
23     000060      L.DS       == L.SDI+4 ; Drive status info
24
25     ;GET COMMAND STATUS END PACKET OFFSETS
26
27     000014      P.OTRF     = 12.    ;OUTSTANDING REFERENCE NUMBER
28     000020      P.CMST     = 16.    ;COMMAND STATUS
29
30
31     ;GET UNIT STATUS END PACKET OFFSETS
32
33     000014      P.MLUN     = 12.    ;MULTI-UNIT CODE
34     000016      P.UNFL     = 14.    ;UNIT FLAGS
35     000020      P.HSTI     = 16.    ;HOST IDENTIFIER
36     000024      P.UNTI     = 20.    ;UNIT IDENTIFIER
37     000034      P.MEDI     = 28.    ;MEDIA TYPE IDENTIFIER
38     000040      P.SHUN     = 32.    ;SHADOW UNIT
39     000042      P.SHST     = 34.    ;SHADOW STATUS
40     000044      P.TRKS     = 36.    ;TRACK SIZE
41     000046      P.GRPS     = 38.    ;GROUP SIZE
42     000050      P.CYLS     = 40.    ;CYLINDER SIZE
43     000054      P.RCTS     = 44.    ;RCT TABLE SIZE
44     000056      P.RBNS     = 46.    ;RBNS / TRACK
45     000057      P.RCTC     = 47.    ;RCT COPIES
46
47
48     ;ONLINE AND SET UNIT CHARACTERISTICS END PACKET AND AVAILABLE
49     ;ATTENTION MESSAGE OFFSETS
50
51     000014      P.MLUN     = 12.    ;MULTI-UNIT CODE
52     000016      P.UNFL     = 14.    ;UNIT FLAGS
53     000020      P.HSTI     = 16.    ;HOST IDENTIFIER
54     000024      P.UNTI     = 20.    ;UNIT IDENTIFIER
55     000032      P.MODEL    = P.UNTI+6 ; Model Identifier
56     000033      P.CLASS    = P.UNTI+7 ; Class Identifier
57     000034      P.MEDI     = 28.    ;MEDIA TYPE IDENTIFIER
  
```

End packet offsets

58	000040	P.SHUN = 32.	;SHADOW UNIT
59	000042	P.SHST = 34.	;SHADOW STATUS
60	000044	P.UNSZ = 36.	;UNIT SIZE
61	000050	P.VSER = 40.	;VOLUME SERIAL NUMBER
62			
63			
64		;SET CONTROLLER CHARACTERISTICS END PACKET OFFSETS	
65			
66	000014	P.VRSN = 12.	;MSCP VERSION
67	000016	P.CNTF = 14.	;CONTROLLER FLAGS
68	000020	P.CTMO = 16.	;CONTROLLER TIMEOUT
69	000022	P.CNCL = 18.	;CONTROLLER COMMAND LIMIT
70	000024	P.CNTI = 20.	;CONTROLLER ID
71			
72			
73		;GET DUST STATUS END PACKET OFFSETS	
74			
75	000014	P.DEXT = 12.	;EXTENSION FOR DOWNLINE LOADABLE PROGRAM
76	000017	P.DFLG = 15.	;FLAGS
77	000020	P.DPRG = 16.	;PROGRESS INDICATOR FOR REMOTE PROGRAM
78	000024	P.DTMO = 20.	;TIMEOUT

```

1      .SBTTL  Status and Event code definitions
2
3      000037  ST.MSK = 37      ;STATUS / EVENT CODE MASK
4      000040  ST.SUB = 40      ;SUB-CODE MULTIPLIER
5      000000  ST.SUC = 0       ;SUCCESS
6      000001  ST.CMD = 1       ;INVALID COMMAND
7      000002  ST.ABO = 2       ;COMMAND ABORTED
8      000003  ST.OFL = 3       ;UNIT-OFFLINE
9      000004  ST.AVL = 4       ;UNIT-AVAILABLE
10     000005  ST.MFE = 5       ;MEDIA FORMAT ERROR
11     000006  ST.WPR = 6       ;WRITE PROTECTED
12     000007  ST.CMP = 7       ;COMPARE ERROR
13     000010  ST.DAT = 10      ;DATA ERROR
14     000011  ST.HST = 11      ;HOST BUFFER ACCESS ERROR
15     000012  ST.CNT = 12      ;CONTROLLER ERROR
16     000013  ST.DRV = 13      ;DRIVE ERROR
17     000037  ST.DIA = 37      ;MESSAGE FROM AN INTERNAL DIAGNOSTIC
18     000400  ST.AOL = 400     ;ALREADY ON-LINE
19
20     ;DUP MESSAGE TYPES
21
22     010000  DU.QUE = 10000    ;QUESTION
23     020000  DU.DFL = 20000    ;DEFAULT QUESTION
24     030000  DU.INF = 30000    ;INFORMATION
25     040000  DU.TER = 40000    ;TERMINATOR
26     050000  DU.FTL = 50000    ;FATAL ERROR
27     060000  DU.SPC = 60000    ;SPECIAL
28
29     ;      Data error subcodes
30
31     000110  SB.HCE =      110  ; Header compare error
32     000150  SB.DST =      150  ; Data sync timeout
33     000350  SB.ECC =      350  ; Uncorrectable ECC
34

```

```

1      .SBTTL Controller table definitions
2
3      ;ONE TABLE WILL BE SET UP BY INITIALIZE SECTION FOR EACH xDA SELECTED
4      ;FOR TESTING. TABLES ARE CONTIGUOUS. THE END OF THE TABLES IS
5      ;MARKED BY A WORD OF ZEROS.
6      ;
7      ;THE FIRST TABLE IS POINTED TO BY THE CONTENTS OF CTLRTBL.
8      ;THE NUMBER OF TABLES IS CONTAINED IN CTRLRS.
9
10     000077      CT.UNT = 000077      ;LOGICAL UNIT NUMBER MASK
11     000777      CT.VEC = 000777      ;VECTOR ADDRESS MASK
12     007000      CT.BRL = 007000      ;BR LEVEL MASK
13
14     100000      CT.AVL = BIT15      ;SET WHEN NOT AVAILABLE FOR TESTING
15     000040      CT.U50 = BIT5      ;CONTROLLER IS xDA50 IF SET/xDA52 IF CLEARED
16     000020      CT.REQ = BIT4      ;BUFFER HAS BEEN GIVEN TO xDA FOR REQUEST
17                                     ;SET WHENEVER READ STUD DATA COMMAND GIVEN TO xDA
18     000010      CT.MSG = BIT3      ;MESSAGE RESPONSE RECEIVED
19                                     ;WHENEVER THIS BIT IS SET, CT.CMD IS CLEARED
20     000004      CT.CMD = BIT2      ;COMMAND ISSUED, WAITING FOR RESPONSE
21
22     000000      C.UADR = 0          ;UNIBUS ADDRESS OF xDAIP REGISTER
23     000002      C.VEC = 2          ;VECTOR ADDRESS/BR LEVEL
24     000004      C.BST = 4          ;BURST LEVEL
25     000006      C.DRVN = 6          ;DRIVE NUMBER TO TEST
26     000010      C.JSR = 10         ;INTERRUPT SERVICE ROUTINE FOR CONTROLLER
27     000012      C.JAD = 12         ;THESE TWO WORDS LOADED WITH [JSR RO xDASRV]
28     000014      C.FLG = 14         ;FLAGS
29     000016      C.HCOM = 16        ;BEGINNING ADRS OF HOST COMM AREA IN MEMORY
30     000020      C.DRVT = 20        ;POINTER TO DRIVE TABLE
31     000022      C.TO = 22          ;TIMEOUT COUNTER
32     000024      C.TOH = 24         ; (TWO WORDS)
33     000026      C.REF = 26         ;COMMAND REFERENCE NUMBER
34
35     000030      C.SIZE = 30        ;SIZE OF CONTROLLER TABLE IN BYTES
36

```



```

1
2           ;DRIVE TABLE DEFINITIONS
3           ;
4           ;ONE DRIVE TABLE WILL BE SET UP BY THE INITIALIZE SECTION FOR EACH
5           ;DRIVE SELECTED FOR TESTING.  EACH TABLE IS POINTED TO BY A
6           ;WORD IN THE CONTROLLER TABLE ON WHICH THE DRIVE EXISTS.
7
8           000077           DT.UNT      = 000077           ; LOGICAL UNIT NUMBER OF DRIVE
9
10          100000          DT.AVL      = BIT15           ; SET WHEN NOT AVAILABLE FOR TESTING
11          040000          D.IW       = BIT14           ; INITIAL WRITE
12          020000          D.DCY      = BIT13           ; DIAGNOSTIC CYLINDERS
13          010000          D.ECC      = BIT12           ; ECC CORRECTION ENABLED
14          004000          D.RO       = BIT11           ; READ ONLY
15          002000          D.WO       = BIT10           ; WRITE ONLY
16          001000          D.RET      = BIT9            ; RETRIES ENABLED
17          000400          D.CYL      = BIT8            ; START/END CYLINDERS SPECIFIED
18          000100          D.SEQ      = BIT6            ; SEQUENTIAL ACCESS
19          000040          D.BE       = BIT5            ; BEGIN/END BLOCKS USED
20          000020          D.TR       = BIT4            ; WHEN D.BE=0: 1 - TRACKS, 0 - GROUPS
21          000010          D.WC       = BIT3            ; WRITE CHECKS ENABLED
22          000004          D.WCA      = BIT2            ; ALWAYS WRITE CHECK
23          000002          D.DC       = BIT1            ; DATA COMPARES ENABLED
24          000001          D.DCA      = BIT0            ; ALWAYS DATA COMPARE
25
26          000000          D.PAT      = 0               ; DATA PATTERN NUMBER
27          000002          D.BB       = 2               ; BAD BLOCK COUNT
28          000004          D.BB01     = 4               ; BAD BLOCK 1
29          000010          D.BB02     = 10              ;
30          000014          D.BB03     = 14              ;
31          000020          D.BB04     = 20              ;
32          000024          D.BB05     = 24              ;
33          000030          D.BB06     = 30              ;
34          000034          D.BB07     = 34              ;
35          000040          D.BB08     = 40              ;
36          000044          D.BB09     = 44              ;
37          000050          D.BB10     = 50              ;
38          000054          D.BB11     = 54              ;
39          000060          D.BB12     = 60              ;
40          000064          D.BB13     = 64              ;
41          000070          D.BB14     = 70              ;
42          000074          D.BB15     = 74              ;
43          000100          D.BB16     = 100             ;
44

```

1				
2	000104	D.BEC = 104		;BEGIN/END SET COUNT
3	000106	D.BGN1 = 106		;BEGIN BLOCK 1
4	000112	D.END1 = 112		;END
5	000116	D.BGN2 = 116		;BEGIN BLOCK 2
6	000122	D.END2 = 122		;END
7	000126	D.BGN3 = 126		;BEGIN BLOCK 3
8	000132	D.END3 = 132		;END
9	000136	D.BGN4 = 136		;BEGIN BLOCK 4
10	000142	D.END4 = 142		;END
11	000146	D.BCYL = 146		;BEGIN CYLINDER
12	000152	D.ECYL = 152		;END CYLINDER
13	000156	D.SERN = 156		;DRIVE SERIAL NUMBER
14				
15	000164	D.SIZE = 164		;SIZE OF DRIVE TABLE IN BYTES

1
2
3
4
5
6
7

.SBTTL Miscellaneous definitions

000620
001000
000200
000200

CACHESIZE == 400.
BLKSIZ == 512.
GRPSIZ == 128.
ENTRYS == 128.

; Number entries in Caching table
; Number of bytes per block
; ACCESS group size
; Number entries per RCT descriptor

CZUDLA0 BBR Replacement Utility MACRU Y05.03c Friday 19-Apr-85 10:42 Page 18
 RCT Logical block 0 definitions

```

1          .SBTTL  RCT Logical block 0 definitions
2
3          000000      VOLSN  == 0          ;Volume serial number (LO WORD)
4          000010      RFLAGS == 10        ;Replacement flags
5          000014      LBNLO  == 14        ;LBN being replaced (LO WORD)
6          000020      RBNLO  == 20        ;RBN replacement (LO WORD)
7          000024      BADRBN == 24        ;Bad RBN (LO WORD)
8          000030      CACHID == 30        ;Cache ID (LO WORD)
9          000040      INCARN  == 40        ;Cache incarnation no. (LO WORD)
10         000044      TIMDAT  == 44        ;VAX time and date (LO WORD)
11
12
13         ;REPLACEMENT FLAG DEFINITIONS
14
15         000001      WBC     == BIT0      ;Write-Back caching
16         000002      VWP     == BIT1      ;Volume write protect
17         000200      FEF     == BIT7      ;Forced Error flag
18         020000      BRFB    == BIT13     ;Bad RBN flag
19         040000      P2      == BIT14     ;Phase 2 flag
20         100000      P1      == BIT15     ;Phase 1 flag
21

```

```

1      .SBTTL  Global data section
2
3      ;
4      ;      Program data areas
5 002166 L$ERRTBL::
      002166 000000  ERRRTYP::      .WORD  0
      002170 000000  ERRNBR::      .WORD  0
      002172 000000  ERRMSG::      .WORD  0
      002174 000000  ERRBLK::      .WORD  0
6
7 002176 000000  FFREE::      .WORD  0      ;FIRST FREE WORD IN MEMORY
8 002200 000000  FSIZE::      .WORD  0      ;SIZE OF FREE MEMORY IN WORDS
9 002202 000000  LOADFLG::    .WORD  0      ; Cleared by program load
10 002204 000000 RTFLAG::     .WORD  0      ; Runtime flag
11 002206 000000 CLKFLG::     .WORD  0      ; Set if KW11 present
12
13      ;      KW11 clock parameters
14
15 002210 000000  KW.CSR: .WORD  0      ;CSR OF CLOCK
16 002212 000000  KW.BRL: .WORD  0      ;BR LEVEL
17 002214 000000  KW.VEC: .WORD  0      ;VECTOR
18 002216 000000  KW.HZ:  .WORD  0      ;HERTZ (50. OR 60.)
19 002220 000000  KW.EL:  .WORD  0,0     ;ELAPSED TIME
20
21 002224 000000  NXMAD: .WORD  0      ;SET TO ALL ONES BY NON-EXISTANT ADDRESS
22 002226 177777  KTMEM: .WORD -1      ;SET TO ALL ONES IF NO KT EXISTS
23
24
25      ;      UUT tables and pointers
26
27 002230          CTLRLST::      .BLKW  10.     ; Controller address table list
28 002254          CACHLST::      .BLKW  10.     ; Bad block caching table list
29 002300 000000  TBLNDX::      .WORD  0      ; Table index
30 002302 000000  PTABLE::      .WORD  0      ; PTABLE address ptr
31 002304 000000  CTLRTBL::     .WORD  0      ; Address of current controller table
32 002306 000000  DRIVTBL::     .WORD  0      ; Address of current drive table
33 002310 000000  CACHTBL::     .WORD  0      ; Current caching table
34 002312 000000  RCTTBL::      .WORD  0      ; RCT bad block address table
35
36      ;      SA register parameter storage
37
38 002314 000000  SADDR::      .WORD  0      ; SA register address
39 002316 000000  SAVAL::      .WORD  0      ; Saved SA value
40 002320 000000  SAEXP::      .WORD  0      ; Expected SA value
41
42 002322 000000  IPVAL::      .WORD  0      ; Saved IP value
43

```

```

1          ;
2          ;       Communications tables and pointers
3
4 002324 000000 HCOM::      .WORD 0      ; Address of host communication area
5 002326 000000 CMDPCK::   .WORD 0      ; Address of command packet
6 002330 000000 MSGPCK::   .WORD 0      ; Address of message packet
7 002332 000000 INTRCV::   .WORD 0      ; Interrupt received flag
8
9 002334 001000 000000 BCNT::      .WORD BLKSIZ,0 ; Number of bytes to transfer
10 002340 042472 UADR::      .WORD DBUFF   ; contains UNIBUS buffer address
11
12          ;       Command packet parameter storage
13
14 002342 000000 CMDREF::   .WORD 0      ; Command reference
15 002344 000000 OPCODE::   .WORD 0      ; MSCP command
16 002346 000000 CMDMOD::   .WORD 0      ; Command modifiers
17
18          ;       End packet parameter storage
19
20 002350 000000 ENDREF::   .WORD 0      ; End packet reference
21 002352 000000 ECODE::    .WORD 0      ; MSCP endcode
22 002354 000000 EMSTAT::   .WORD 0      ; End msg status
23 002356 000000 FLAGS::    .WORD 0      ; End msg flags
24
25          ;       Error packet parameter storage
26
27 002360 000000 ERRREF::   .WORD 0      ; Error reference
28 002362 000000 EVENT::    .WORD 0      ; Event code
29 002364 000000 ERFLAG::   .WORD 0      ; Error msg flags
30 002366 000000 ERRFMT::   .WORD 0      ; Error format
31 002370 000000 EXTMSK::   .WORD 0      ; Extended error maskflags
32
33 002372 000000 ERLFMT::   .WORD 0      ; Error log format spec
34 002374 000000 000000 ERLLBN::   .WORD 0,0    ; Error log LBN
35 002400 000000 000000 HADDR::    .WORD 0,0    ; Error log host address
36 002404 000000 DCODE::    .WORD 0      ; Drive error code
37 002406          ERLDS::    .BLKW 4      ; Error log drive/status error info
38

```

```

1
2           ;          Drive parameter area
3
4 002416 000000 RCTSIJ:: .WORD 0 ; Number of blocks in the RCT (w/ pad)
5 002420 000000 TRKSIJ:: .WORD 0 ; Number of LBN'S per track
6 002422 000 RBNS:: .BYTE 0 ; Number of REPLACEMENT BLOCKS per track
7 002423 000 COPIES:: .BYTE 0 ; Number of copies of the RCT
8 002424 000000 RCTCOPY:: .WORD 0 ; Current RCT copy
9 002426 000000 RCTMASK:: .WORD 0 ; RCT bad copy mask
10 002430 000000 000000 UNSIZ:: .WORD 0,0 ; Number of blocks in of LBN space
11 ; RCT start
12 002434 000000 PRIMRY:: .WORD 0 ;primary replacement counter
13 002436 000000 SCNDRY:: .WORD 0 ;secondary replacement counter
14 002440 000000 UNUSE:: .WORD 0 ;unusable replacement counter
15
16 002442 000000 STATUS:: .WORD 0 ; Subroutine return value
17 002444 000000 TMPSTAT:: .WORD 0 ; Multi-READ/WRITE temp status
18 002446 000000 000000 LBN:: .WORD 0,0 ;LBN being replaced
19 002452 000000 000000 RBN:: .WORD 0,0 ;RBN replacement
20 002456 000000 000000 RCTLBN:: .WORD 0,0 ; RCT LBN
21 002462 000000 000000 MATRBN:: .WORD 0,0 ;RBN which matched LBN being replaced
22 002466 000000 RECOVR:: .WORD 0 ;recovery flag, Phase 1 or 2 recovery
23 ; no recovery=0, recovery=1
24 002470 000000 RDVALD:: .WORD 0 ; Valid data flag for READLBN
25 002472 000 VIBUFF:: .BYTE 0 ;Valid image flag.
26 002473 000 VBUFF1:: .BYTE 0 ;Valid buffer one flag.
27 002474 000 VBUFF2:: .BYTE 0 ;Valid buffer two flag.
28 002475 000 ERRCNT:: .BYTE 0 ;error count
29 .EVEN
30
31 002476 000000 RCTBLK:: .WORD 0 ;RCT logical block
32 002500 000000 MATBLK:: .WORD 0 ;RCT logical block for matched LBN
33 002502 000000 RCTOFF:: .WORD 0 ;RBN offset within RCT logical block
34 002504 000000 MATOFF:: .WORD 0 ;RBN offset within RCT logical block
35 ;for matched LBN
36 002506 000000 STROFF:: .WORD 0 ;starting offset within RCT block
37 002510 000000 DELTA:: .WORD 0 ;change in offset while for RBN search
38 002512 000 EMPTY:: .BYTE 0 ;empty flag
39 002513 000 RESCAN:: .BYTE 0 ;re-scan flag
40 002514 000 MATFLG:: .BYTE 0 ;match status,
41 ; 0=No match for LBN being replaced
42 ; 1=RBN in 'MATRBN' was match for LBN
43 ; being replaced
44 002515 000 SFLAG:: .BYTE 0 ;search status,
45 ; 0=Primary empty
46 ; 1=Secondary empty
47 ; 2=Full table (no match)
48 .EVEN
49
50
51           ;          LBNSTAT: LBN status word
52
53 000001 TF.FER == BIT0 ; FER
54 000002 TF.HCE == BIT1 ; HCE
55 000004 TF.DST == BIT2 ; DST
56 000010 TF.ECC == BIT3 ; ECC
57 000020 TF.SEX == BIT4 ; EF.SEX
    
```

58	000040	TF.LCG ==	BITS		; EF.LOG
59	000100	TF.BBU ==	BIT6		; EF.BBU
60	000200	TF.BBR -	BIT7		; EF.BBR
61	000400	TF.WFE ==	BIT8		; WFE
62	100000	TF.BAD ==	BIT15		; Set if image is inval'd
63					
64	002516	000000	LBNSTAT::	.WORD 0	; LBN status word
65	002520	000000	RPLSTAT::	.WORD 0	; Replacement status
66	002522	000000	LBNDX::	.WORD 0	; LBN descriptor index
67	002524	000000	RDSTAT::	.WORD 0	; READLBN read status
68	002526	000000	WRSTAT::	.WORD 0	; READLBN write status
69					
70	002530	000000	FERCNT::	.WORD 0	; FER count
71	002532	000000	HCECNT::	.WORD 0	; HCE count
72	002534	000000	DSTCNT::	.WORD 0	; DST error count
73	002536	000000	ECCCNT::	.WORD 0	; Uncorrectable ECC count
74	002540	000000	BBRCNT::	.WORD 0	; BBR count
75	002542	000000	WFECNT::	.WORD 0	; WRITE Forced error count
76					
77					
78	002544	000000	STEP::	.WORD 0	; Current STEP value
79	002546	000000	NXTSTEP::	.WORD 0	; Next STEP value
80	002550	000000	ERRSTEP::	.WORD 0	; Error STEP indicator
81	002552	000000	BASEMSG::	.WORD 0	; Error message address
82	002554	000000	INCRLBN::	.WORD 0	; LBN Increment
83	002556	000000	RDCNT::	.WORD 0	; Reread counter
84	002560	000000	PHASE::	.WORD 0	; Replacement phase
85					
86	002562	000000	000000	.WORD 0,0	; ACCESS group LBN
87	002566	000000	000001	.WORD 0,1	; ACCESS group byte count
88	002572	000200		.WORD GRPSIZ	; ACCESS block count
89	002574	000000	000000	.WORD 0,0	; ACCESS number
90	002600	000000	000000	.WORD 0,0	; Lowest LBN
91	002604	000000	000000	.WORD 0,0	; Highest LBN
92	002610	000000		.WORD 0	; Flag endless loop
93	002612	000000		.WORD 0	; Flags unit online
94	002614	000000		.WORD 0	; Bus type flag
95					; 0=UNIBUS, 1=QBUS
96	002616	000000	VSNPTR::	.WORD 0	; Ptr to VSN text
97	002620		VSNBUF::	.BLKB 24.	; VSN text
98					
99					


```
1          .SBTTL  Operator interface parameters
2
3 002650  000000      DEBUG:: .WORD  0          ; Debug flag
4 002652  000000      BACKUP:: .WORD  0          ; Backup flag
5
6 002654              REPLY:: .BLKB  12         ; Reply
7
8 002666              TTYIN:: .BLKW  12         ;TTY CHARACTER INPUT BUFFER
9 002712  000000      CMDIN:: .WORD  0          ;COMMAND CHARACTER INPUT BUFFER
10 002714  000000     OPTIN:: .WORD  0          ;OPTION CHARACTER INPUT BUFFER
11
12 002716              LINBUF:      .BLKB  128.    ; Print buffer
13
14
```

```

1          .SBTTL  SA register error parameters
2
3          ;      SA register error equates
4
5          000001      B.SADDR =      BIT0          ; Print SA address
6          000002      B.SAVAL =      BIT1          ; Print SA contents
7          000004      B.SAEXP =      BIT2          ; Print SA expected value
8
9          ;      SA register msg table
10
11         003116      SARMSG::
12         003116      003165      .WORD  XSADDR          ; "SA address:  xxxxxx"
13         003120      003136      .WORD  XSAVAL          ; "SA contents: xxxxxx"
14         003122      003214      .WORD  XSAEXP          ; "SA expected: xxxxxx"
15         003124      000000      .WORD  0              ; terminator
16
17         ;      SA register value table
18
19         003126      SARVAL::
20         003126      002314      .WORD  SADDR          ; SA address
21         003130      002316      .WORD  SAVAL          ; Actual value
22         003132      002320      .WORD  SAEXP          ; Expected value
23         003134      000000      .WORD  0              ; terminator
24
25         ;      SA register error extended text
26
27         .NLIST  BEX
28         003136      116      042      040  XSAVAL: .ASCIZ /N" SA contents: "016/
29         003165      116      042      040  XSADDR: .ASCIZ /N" SA address:  "016/
30         003214      116      042      040  XSAEXP: .ASCIZ /N" SA expected: "016/
31         .LIST  BEX
32         .EVEN
33

```

```

1          .SBTTL Drive model parameters
2
3          000000      M.UNKN      =      0
4          000001      M.RA80      =      1
5          000004      M.RA60      =      4
6          000005      M.RA81      =      5
7          000013      M.RA82      =     11.
8
9 003244 000000      MODEL:      .WORD  0      ; Unit model no.
10 003246 000000      MDLNDX:     .WORD  0      ; Index to model table
11
12 003250      MDLTBL:
13 003250 000000      .WORD  M.UNKN      ; "Unknown"
14 003252 000001      .WORD  M.RA80      ; "RA80"
15 003254 000004      .WORD  M.RA60      ; "RA60"
16 003256 000005      .WORD  M.RA81      ; "RA81"
17 003260 000013      .WORD  M.RA82      ; "RA82"
18
19          000012      MDLSIZ =      .-MDLTBL
20
21 003262      MDLTEXT:
22 003262 003274      .WORD  UNKNOWN      ; 0 "Unknown"
23 003264 003304      .WORD  RA80          ; 1 "RA80"
24 003266 003311      .WORD  RA60          ; 4 "RA60"
25 003270 003316      .WORD  RA81          ; 5 "RA81"
26 003272 003323      .WORD  RA82          ; 11 "RA82"
27
28          .NLIST BEX
29 003274      125      116      113 UNKNOWN:      .ASCIZ      /UNKNOWN/
30 003304      122      101      070 RA80:      .ASCIZ      /RA80/
31 003311      122      101      066 RA60:      .ASCIZ      /RA60/
32 003316      122      101      070 RA81:      .ASCIZ      /RA81/
33 003323      122      101      070 RA82:      .ASCIZ      /RA82/
34          .LIST BEX
35          .EVEN
36

```

```

1          .SBTTL  MSCP error message parameters
2
3          ;      MSCP error equates
4
5          000001      B.CMDRF =      BIT0          ; Print cmd ref number
6          000002      B.CODE  =      BIT1          ; Print opcode
7          000004      B.MOD   =      BIT2          ; Print cmd modifiers
8          000010      B.EMREF =      BIT3          ; Print end msg ref
9          000020      B.ECODE =      BIT4          ; Print endcode
10         000040      B.FLAGS =      BIT5          ; Print end flags
11         000100      B.ESAT  =      BIT6          ; Print end msg status
12
13         000140      B.MSCP  =      B.FLAGS+B.ESAT
14
15         ;      MSCP extended msg table
16
17         MSCPMSG::
18         003330      003370      .WORD  XCMDRF          ; "Command ref:  xxxxxx"
19         003332      003421      .WORD  XCODE          ; "MSCP command:  xxx"
20         003334      003454      .WORD  XMOD           ; "Cmd modifiers: xxxxxx"
21         003336      003506      .WORD  XENDRF         ; "Endmsg ref:   xxxxxx"
22         003340      003537      .WORD  XECODE         ; "MSCP endcode:  xxx"
23         003342      003572      .WORD  XFLAGS        ; "Endmsg flags:  xxx"
24         003344      003625      .WORD  XESTAT         ; "Endmsg status: xxxxxx"
25         003346      000000      .WORD  0              ; terminator
26
27         ;      MSCP extended value table
28
29         MSCPVAL::
30         003350      002342      .WORD  CMDREF         ; Command reference number
31         003352      002344      .WORD  OPCODE         ; MSCP command
32         003354      002346      .WORD  CMDMOD         ; Command modifiers
33         003356      002350      .WORD  ENDREF         ; End msg reference number
34         003360      002352      .WORD  ECODE         ; End code
35         003362      002356      .WORD  FLAGS         ; End msg flags
36         003364      002354      .WORD  EMSTAT        ; End msg status
37         003366      000000      .WORD  0              ; terminator
38
39         ;      MSCP extended error  :xt
40
41         .NLIST  BEX
42         003370      116      042      040  XCMDRF: .ASCIZ  /N" Command ref:  "016/
43         003421      116      042      040  XCODE:  .ASCIZ  /N" MSCP command:  "08/
44         003454      116      042      040  XMOD:   .ASCIZ  /N" Cmd modifiers: "016N/
45         003506      116      042      040  XENDRF: .ASCIZ  /N" Endmsg ref:   "016/
46         003537      116      042      040  XECODE: .ASCIZ  /N" MSCP endcode:  "08/
47         003572      116      042      040  XFLAGS: .ASCIZ  /N" Endmsg flags:  "08/
48         003625      116      042      040  XESTAT: .ASCII  /N/
49         003626      042      040      040  XSTAT:  .ASCIZ  /" Endmsg status: "016N2/
50         .LIST   BEX
51         .EVEN
52

```

RCT error message parameters

```

1      .SBTTL RCT error message parameters
2
3      ; RCT error equates
4
5      00000* B.BLOCK = BIT0 ; Print Block number
6      000002 B.COPY = BIT1 ; Print copy number
7      000004 B.PHASE = BIT2 ;
8
9      000003 B.RCT = B.BLOCK!B.COPY ; RCT error bits
10
11     ; RCT extended msg table
12
13     RCTMSG::
14     003660 003700 .WORD XBLOCK ; "RCT Block no: ddd"
15     003662 003732 .WORD XCOPY ; "RCT copy: ddd"
16     003664 003764 .WORD XPHASE ; "Replacement phase: ddd"
17     003666 000000 .WORD 0 ; terminator
18
19     ; RCT extended value table
20
21     RCTVAL::
22     003670 002476 .WORD RCTBLK ; "Block no: ddd"
23     003672 002424 .WORD RCTCOPY ; "RCT copy: ddd"
24     003674 002466 .WORD RECOVR ; "Replacement phase: ddd"
25     003676 000000 .WORD 0 ; terminator
26
27     .NLIST BEX
28     003700 116 042 040 XBLOCK: .ASCIZ /N" RCT block no: "D16"/
29     003732 116 042 040 XCOPY: .ASCIZ /N" RCT copy: "D16"/
30     003764 116 042 040 XPHASE: .ASCIZ /N" Replacement phase: "D16/
31     .LIST BEX
32     .EVEN

```

```

1          .SBTTL  Errorlog error message parameters
2
3          ;      Errorlog messages and equates
4
5          000001  B.EVNT =      BIT0          ; Print status/event
6          000002  B.HADDR =      BIT1          ; Print host address
7          000004  B.DCODE =      BIT2          ; Print drive error code
8          000010  B.DERR =      BIT3          ; Print drive error info
9
10         ;      Errorlog extended msg table
11
12         004020  ERLMSG::
13         004020  004034      .WORD  XEVENT          ; "Status/event: xxxxxx"
14         004022  004063      .WORD  XHADDR          ; "Host address: xxxxxx"
15         004024  000000      .WORD  0              ; terminator
16
17         ;      Errorlog extended value table
18
19         004026  ERLVAL::
20         004026  002362      .WORD  EVENT          ; Event code
21         004030  002400      .WORD  HADDR          ; Host address
22         004032  000000      .WORD  0              ; terminator
23
24         ;      Errorlog extended error text
25
26         .NLIST  BEX
27         004034      116          XEVENT: .ASCII  /N/
28         004035      042          040      XEVNT: .ASCIZ  "\" Status/event: "016\
29         004063      116          042          040      XHADDR: .ASCIZ  /N" Host address: "032/
30         004112      116          042          040      XDCODE: .ASCIZ  /N" Drive error code (hex): "H8/
31         004152      116          042          040      XDINFO: .ASCIZ  \N" Drive status/error info (hex): "\
32         004217      042          040          042      XDHEX: .ASCIZ  /" "H8/
33         .LIST   BEX
34         .EVEN
35

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr 85 10:42 Page 28
 Errorlog error message parameters

```

1
2      000000      NDX.RD =      0      ; Index to READ
3      000001      NDX.WR =      1      ; Index to WRITE
4      000002      NDX.ACC =     2      ; Index to ACCESS
5      000003      NDX.RPL =     3      ; Index to REPLACE
6
7 004226      MSCPNDX:
8 004226 000000      .WORD 0      ; Index to msg ptr
9
10 004230      MSCPTBL:
11 004230 004242      .WORD XREAD      ; "READ"
12 004232 004250      .WORD XWRITE     ; "WRITE"
13 004234 004257      .WORD XACCESS    ; "ACCESS"
14 004236 004267      .WORD XREPLACE   ; "REPLACE"
15 004240 000000      .WORD 0      ; Terminator
16
17
18 004242      122      105      101  XREAD:      .NLIST BEX
19 004250      127      122      111  XWRITE:     .ASCIZ /READ /
20 004257      101      103      103  XACCESS:    .ASCIZ /WRITE /
21 004267      122      105      120  XREPLACE:   .ASCIZ /ACCESS /
22
23
24

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr 85 10:42 Page 29
Global text section

```

1          .SBTTL  Global text section
2
3          ;**
4          ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
5          ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
6          ; MORE THAN ONE TEST.
7          ;-
8
9          ;NAMES OF DEVICES SUPPORTED BY PROGRAM
10         ;
11         L$DVTYP::
12         004300      114      157      147      .ASCIZ  *Logical disk drives*
13         004300      .EVEN
14
15
16         ; TEST DESCRIPTION
17         ;
18         L$DESC::
19         004324      102      141      144      .ASCIZ  /Bad Block Replacement Utility/
20         004324      .EVEN
21
22
23         ; FORMAT STATEMENTS USED IN PRINT CALLS
24         ;
25
26
27
28
29

```


Informational message text

```

1          .SBTTL  Informational message text
2
3          ;      Configuration information messages
4
5 004362    116    062    042  INF100: .ASCII  /N2"BAD BLOCK REPLACEMENT UTILITY."/
6 004424    116    062    042    .ASCII  /N2"WARNING: All drives configured for test"/
7 004500    116    042    115    .ASCIZ  /N"MUST have customer data backed-up."/
8
9 004546    116    042    114  INF101: .ASCIZ  /N"Line clock not available."/
10
11 004603    116    042    111  INF102: .ASCII  /N"It is recommended that you read the operator help"/
12 004667    116    042    151    .ASCIZ  /N"information before proceeding."/
13
14 004731    116    062    042  INF103: .ASCIZ  /N2"Restore customer data to drive after last pass."N/
15
16          ;      Initialization information messages
17
18 005016    116    064    042  INF110: .ASCII  /N4"INITIALIZATION INFORMATION FOR UNIT "D8/
19 005070    116    042    040    .ASCII  /N" Controller: "A/
20 005113    116    042    040    .ASCII  /N" Address:  "016/
21 005140    116    042    040    .ASCIZ  /N" Drive:    "D8/
22
23 005165    116    042    040  INF111: .ASCII  /N" Drive type: "A/
24 005210    116    042    040    .ASCII  /N" Volume SN: "A."/
25 005236    116    042    040    .ASCIZ  /N" Volume size: "D32"."N2/
26
27          ;      End of pass information messages
28
29 005271    116    062    042  INF200: .ASCIZ  /N2"REPLACEMENT INFORMATION FOR THIS PASS"/
30 005343    116    062    042  INF201: .ASCII  /N2" BAD BLOCKs reported:    "D16"."/
31 005414    116    042    040    .ASCII  /N" Header compare errors:    "D16"."/
32 005464    116    042    040    .ASCII  /N" Data sync timeout errors:  "D16"."/
33 005534    116    042    040    .ASCII  /N" Uncorrectable ECC errors:  "D16"."/
34 005604    116    042    040    .ASCII  /N" FORCED ERROR flags detected: "D16"."/
35 005654    116    042    040    .ASCIZ  /N" FORCED ERROR flags written: "D16"."/
36 005725    116    062    042  INF202: .ASCII  /N2" PRIMARY  replacements:    "D15"."/
37 005776    116    042    040    .ASCII  /N" SECONDARY replacements:    "D16"."/
38 006046    116    042    040    .ASCII  /N" RBN's marked UNUSABLE:    "D16"."/
39 006116    116    042    040    .ASCIZ  /N" Total blocks replaced:  "D16"."N/
40
41          ;      RCT descriptor information messages
42
43 006170    116    062    042  INF210: .ASCIZ  /N2"RCT DESCRIPTOR INFORMATION FOR UNIT "D8N/
44 006244    116    042    122  INF211: .ASCIZ  /N"RBN:"D28". is PRIMARY  replacement for LBN:"D28"."/
45 006332    116    042    122  INF212: .ASCIZ  /N"RBN:"D28". is SECONDARY replacement for LBN:"D28"."/
46 006420    116    042    122  INF213: .ASCIZ  /N"RBN "D28". is UNUSABLE"/
47
48 006452    116    062    042  INF220: .ASCIZ  /N2"RCT DESCRIPTOR SUMMARY FOR THIS UNIT"/
49 006523    116    042    040  INF221: .ASCII  /N" PRIMARY  replacement blocks: "D16"."/
50 006574    116    042    040    .ASCII  /N" SECONDARY replacement blocks: "D16"."/
51 006645    116    042    040    .ASCII  /N" UNUSABLE replacement blocks: "D16"."/
52 006716    116    042    040    .ASCIZ  /N" "D16". RBNs used out of "D16". RBNs on media."N/
53

```

```

1
2
3
4 007002      116      062      042  INF301: .ASCIZ  /N2"SCANNING RCT..."N/
5 007027      116      062      042  INF302: .ASCIZ  /N2"SCANNING LBN AREA..."N/
6 007061      116      062      042  INF310: .ASCIZ  /N2"MANUAL BAD BLOCK REPLACEMENT SELECTED."N/
7 007135      116      062      042  INF311: .ASCIZ  /N2" EOF detected.  "/
8 007164      116      062      042  INF320: .ASCIZ  /N2"AUTOMATIC BAD BLOCK REPLACEMENT SELECTED."/
9 007242      116      062      042  INF321: .ASCII  /N2" LBN Status codes:"/
10 007271     116      042      040      .ASCII  /N" BBR - Bad Block reported."/
11 007327     116      042      040      .ASCII  /N" ECH - Hard ECC Error encountered."/
12 007375     116      042      040      .ASCII  /N" HCE - Header Compare Error encountered."/
13 007451     116      042      040      .ASCII  /N" DST - Data Sync Timeout encountered."/
14 007522     116      042      040      .ASCII  /N" FER - Forced Error encountered."/
15 007566     116      042      040      .ASCIZ  /N" WFE - Block written with FE."/
16 007630     116      062      042  INF322: .ASCII  /N2" RBN Operation types:"/
17 007662     116      042      040      .ASCII  /N" PRI - Primary replacement for LBN."/
18 007731     116      042      040      .ASCII  /N" SEC - Secondary replacement for LBN."/
19 010002     116      042      040      .ASCIZ  /N" UNS - RBN marked Unusable."N/
20 010043     116      042      040  INF323: .ASCIZ  /N" LBN:"D28".  "/
21 010065     116      062      042  INF324: .ASCIZ  /N2" EOM reached.  "/
22
23
24
25 010114     116      042      040  INF401: .ASCIZ  /N" RCT copy "D16", block "D16" ("D28".)"/
26
27 010166     116      042      040  INF405: .ASCIZ  /N" LBN:"D28"."/
28 010206     042      040      040  INF406: .ASCIZ  /" Status: "A"/
29 010224     042      040      040  INF407: .ASCIZ  /" RBN:"D28"."/
30 010243     042      040      040  INF408: .ASCIZ  /" Operation: "A"/
31
32
33
34 010264     116      062      042  INF601: .ASCIZ  /N2"Attempting Recovery..."N/
35 010320     116      062      042  INF602: .ASCIZ  /N2"RCi Recovery failed; MEDIA INTEGRITY UNCERTAIN."N/
36 010405     116      062      042  INF603: .ASCIZ  /N2"Attempting to find another replacement block..."N/
37

```

```

1          .SBTTL  Error message text
2
3
4          ;      Program initialization messages
5
6 010472    120    122    117  HDR001: .ASCIZ /PROGRAM INITIALIZATION ERROR/
7 010527    116    042    106  ERM001: .ASCII /N"Failed dynamic storage allocation for Unit "D3";"/
8 010612    116    042    040    .ASCIZ /N" Restart program and select fewer Units."/
9
10         ;      Controller initialization messages
11
12 010667    103    117    116  HDR101: .ASCIZ /CONTROLLER INITIALIZATION ERROR/
13 010727    116    042    101  ERM101: .ASCII /N"Address error occured while accessing contoller register;"/
14 011023    116    042    040    .ASCIZ /N" Check contoller address."/
15 011061    116    042    105  ERM102: .ASCIZ /N"Error detected during controller initialization;"/
16 011145    116    042    103  ERM103: .ASCIZ /N"Controller did not clear ring structure in host memory;"/
17 011240    116    042    123  ERM104: .ASCIZ /N"SA register not zero after step 3 write;"/
18
19         ;      Unit initialization errors
20
21 011314    125    116    111  HDR201: .ASCIZ /UNIT INITIALIZATION ERROR/
22 011346    116    042    106  ERM201: .ASCII /N"Failed GET UNIT STATUS command;"/
23 011410    116    042    040    .ASCIZ /N" Drive "D8" is not accessable."/
24 011453    116    042    106  ERM202: .ASCII /N"Failed SET CONTROLLER CHARACTERISTICS command;"/
25 011534    116    042    040    .ASCII /N" Host access timeout not disabled."/
26 011602    116    042    040    .ASCII /N" Default timeout is 60 seconds."/
27 011645    116    042    040    .ASCIZ /N" Error log packets not enabled."/
28 011711    116    042    106  ERM203: .ASCIZ /N"Failed ONLINE command;"/
29 011743    116    042    132  ERM204: .ASCII /N"Zero RCT size detected;"/
30 011775    116    042    040    .ASCIZ /N" BBR not supported by this drive."/
31
32         ;      MSCP cmd/msg error messages
33
34 012043    115    123    103  HDR301: .ASCIZ /MSCP STATUS ERROR DETECTED/
35 012076    116    042    106  ERM300: .ASCIZ /N"Failed "A"command; LBN:"D28"."/
36 012137    116    042    106  ERM301: .ASCIZ /N"Failed READ command; LBN:"D28"."/
37 012202    116    042    106  ERM302: .ASCIZ /N"Failed WRITE command; LBN:"D28"."/
38 012246    116    042    106  ERM303: .ASCIZ /N"Failed ACCESS command; LBN:"D28"."/
39 012313    116    042    106  ERM304: .ASCIZ /N"Failed REPLACE command; LBN:"D28"."/
40 012361    116    042    106  ERM305: .ASCII /N"Failed replacement verification; LBN:"D28"."/
41 012437    116    042    040    .ASCIZ /N" Retry of replacement operation will be attempted."/
42 012526    116    042    103  ERM306: .ASCIZ \N"Command/message reference number mismatch; LBN:"D28"."\
43 012617    116    042    125  ERM307: .ASCIZ /N"Unrecognized endcode; LBN:"D28"."/
44

```

Error message text

```

1
2           ;      Controller communication error messages
3
4 012663    110    117    123 HDR401: .ASCIZ  \HOST/CONTROLLER COMMUNICATION ERROR\
5 012727    116    042    116 ERM401: .ASCIZ  /N"No interrupt received for 30 seconds;"/
6 013000    116    042    106 ERM402: .ASCIZ  /N"Fatal error reported in SA register;"/
7 013050    116    042    123 ERM403: .ASCIZ  /N"Serious exception reported;"/
8
9           ;      Program errors
10
11 013107    120    162    157 HDR501: .ASCIZ  /Program error/
12 013125    116    042    115 ERM501: .ASCIZ  /N"Media overrun detected; LBN:"D28"."/
13
14           ;      RCT read/write errors
15
16 013173    122    103    124 HDR601: .ASCIZ  /RCT ERROR DETECTED/
17 013216    116    042    106 ERM601: .ASCIZ  /N"Failed RCT READ;"/
18 013242    116    042    106 ERM602: .ASCIZ  /N"Failed RCT WRITE;"/
19 013267    116    042    106 ERM603: .ASCIZ  /N"Failed READ of all copies of RCT;"/
20 013334    116    042    106 ERM604: .ASCIZ  /N"Failed WRITE to all copies of RCT;"/
21 013402    116    042    103 ERM605: .ASCIZ  /N"Crash occured during previous replacement operation;"/
22 013472    116    042    116 ERM606: .ASCII  /N"No NULL descriptor entry found in RCT;"/
23 013543    116    042    040 .ASCIZ  /N" RCT is probably corrupt; REFORMAT MEDIA."/
24 013621    116    042    124 ERM607: .ASCII  /N"Transient error table overflow;"/
25 013663    116    042    040 .ASCIZ  /N" ADDITIONAL MAINTENANCE REQUIRED."/
26
27           ;      Error log messages
28
29 013731    105    122    122 HDR701: .ASCIZ  /ERROR LOG PACKET RECEIVED/
30 013763    116    042    125 ERM700: .ASCIZ  /N"Unrecognized Errorlog format;"/
31 014024    116    042    103 ERM701: .ASCIZ  /N"Controller Error reported;"/
32 014062    116    042    110 ERM702: .ASCIZ  /N"Host Memory Access Error reported;"/
33 014130    116    042    123 ERM703: .ASCIZ  /N"SDI Error reported;"/
34 014157    116    042    123 ERM704: .ASCIZ  /N"Small Disk Error reported;"/
35

```

```

1          .SBTTL Formatted text
2
3          ; Information text
4
5 014215    045    116    000 F.CRLF: .ASCIZ /*N/          ; FA0 New line
6 014220    045    124    000 F.TEXT: .ASCIZ /*T/          ; FA0 Text
7
8 014223    101    000          FMT.ASC: .ASCIZ /A/          ; FA0 .ASCIZ
9
10 014225   125    104    101 DEVNAME: .ASCIZ /UDA/          ; Controller name
11 014231   116    000          CRLF: .ASCIZ /N/
12 014233   042    040    040 RNTIM: .ASCIZ /" Elapsed runtime "D16":"/
13 014266   104    071    042 RNTIM1: .ASCIZ /D9":"/
14 014274   104    071    000 RNTIM2: .ASCIZ /D9/
15 014277   042    040    077 LMT28: .ASCIZ /" ?Exceeded 28 bit input limit"N/
16 014340   042    040    077 INVCHR: .ASCIZ /" ?Invalid character"N/
17 014367   116    042    114 OLIMIT: .ASCIZ /N"Limits: LO= 0; HI= "D32"."N/
18 014427   116    042    105 ERRME1: .ASCIZ /N"Error processing message string"N/
19 014473   116    042    115 MANDIS: .ASCIZ /N"Manual intervention is disabled"N/
20 014537   116    042    040 XRCT: .ASCIZ /N" RCT block number: "D16"."/
21 014575   116    042    040 XLBN: .ASCIZ /N" LBN:"D28"."/
22 014615   116    042    040 \RBN: .ASCIZ /N" RBN:"D28"."/
23 014635   116    042    040 XSTEP: .ASCIZ /N" Replacement STEP: "D8"."/
24          .EVEN
25
26          ; Unformatted text
27
28 014672    132    125    104 HLPFILE: .ASCIZ /ZUDLAO.HLP/
29
30 014705    105    156    164 TXT000: .ASCIZ /Enter LBN to be replaced (D) or <CR> to exit/
31 014762    122    102    116 TXT001: .ASCIZ /RBN /
32 014767    114    102    116 TXT002: .ASCIZ /LBN /
33 014774    105    156    164 TXT005: .ASCIZ /Enter starting group LBN (D)/
34 015031    101    164    164 TXT006: .ASCIZ /Attempt crash recovery/
35
36 015060    105    156    164 TX.LB1: .ASCIZ /Enter first LBN/
37 015100    105    156    164 TX.LB2: .ASCIZ /Enter last LBN/
38
39 015117    120    122    111 TX.PRI: .ASCIZ /PRI /
40 015124    123    105    103 TX.SEC: .ASCIZ /SEC /
41 015131    125    116    123 TX.UNS: .ASCIZ /UNS /
42 015136    106    105    122 TX.FER: .ASCIZ /FER /
43 015143    105    103    110 TX.ECH: .ASCIZ /ECH /
44 015150    102    102    122 TX.BBR: .ASCIZ /BBR /
45 015155    102    102    125 TX.BBU: .ASCIZ /BBU /
46 015162    123    105    130 TX.SEX: .ASCIZ /SEX /
47 015167    114    117    107 TX.LOG: .ASCIZ /LOG /
48 015174    040    040    040 TX.BLK: .ASCIZ / /
49 015201    127    106    105 TX.WFE: .ASCIZ /WFE /
50 015206    110    103    105 TX.HCE: .ASCIZ /HCE /
51 015213    104    123    124 TX.DST: .ASCIZ /DST /
52          .EVEN

```

1
2
3
4
5
6
7
8

.SBTTL Global error report section

; **
; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION. PRINTB
; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
; **

```

1      .SBTTL  Error processing routines
2
3      ;ERR001:  General case error routine
4      ;
5      ;          Prints basic error msg from BASEMSG with Drive number
6
7 015220  ERR001::
8
9 015220  010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
10
11 015222  013701  002304  MOV      CTLRTBL,R1          ; Load controller table ptr
12 015226  016546  000006  MOV      C.DRVN(R5),-(SP)      ;PUSH C.DRVN(R5) ON STACK
13 015232  013746  002552  MOV      BASEMSG,-(SP)      ;PUSH BASEMSG ON STACK
14 015236  004137  033254  JSR      R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
15 015242  000004  .WORD   ARG.CT          ;ARGUMENT COUNT * 2
16 015244  013746  002354  MOV      EMSTAT,-(SP)      ;PUSH EMSTAT ON STACK
17 015250  012746  003625  MOV      @XESTAT,-(SP)     ;PUSH @XESTAT ON STACK
18 015254  004137  033254  JSR      R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
19 015260  000004  .WORD   ARG.CT          ;ARGUMENT COUNT * 2
20
21 14
22 15 015262  012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
23 16 015264
24 015264  104423  L10002:  TRAP    C#MSG
25
26 ;ERR101:  Controller initialization error routine
27 ;
28 ;          Prints Basic msg in BASEMSG
29 ;          Prints extended SA register info
30
31 23 015266  ERR101::
32
33 24
34 25 015266  010146          MOV      R1,-(SP)          ;;PUSH R1 ON STACK
35 015270  010246          MOV      R2,-(SP)          ;;PUSH R2 ON STACK
36 015272  010346          MOV      R3,-(SP)          ;;PUSH R3 ON STACK
37
38 26
39 27 015274  013746  002552  MOV      BASEMSG,-(SP)      ;PUSH BASEMSG ON STACK
40 015300  004137  033254  JSR      R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
41 015304  000002  .WORD   ARG.CT          ;ARGUMENT COUNT * 2
42
43 28
44 29 015306  012702  003116  MOV      @SARMSG,R2          ; Load text table ptr
45 015312  012703  003126  MOV      @SARVAL,R3         ; Load value table ptr
46 015316  004737  016042  JSR      PC,CALR1          ; Print extended info
47
48 32
49 33 015322  012603          MOV      (SP)+,R3          ;;POP STACK INTO R3
50 015324  012602          MOV      (SP)+,R2          ;;POP STACK INTO R2
51 015326  012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
52
53 34
54 35 015330  000167          .WORD   J#JMP
55 015332  000000          .WORD   L10003-2-.
56
57 36
58 37 015334
59 015334  104423  L10003:  TRAP    C#MSG
60
61 38
    
```

```

1
2           ;ERR201: Unit initialization error routine
3           ;
4           ; Prints basic msg in BASEMSG
5           ; Prints extended end message info
6
7 015336    ERR201::
8
9 015336    010146    MOV     R1,-(SP)           ;;PUSH R1 ON STACK
015340    010246    MOV     R2,-(SP)           ;;PUSH R2 ON STACK
015342    010346    MOV     R3,-(SP)           ;;PUSH R3 ON STACK
10
11 015344    013746    002552    MOV     BASEMSG,-(SP)       ;PUSH BASEMSG ON STACK
015350    004137    033254    JSR     R1,LPNTB           ;CALL LPNTB PRINT ROUTINE
015354    000002                .WORD   ARG.CT             ;ARGUMENT COUNT * 2
12
13 015356    012702    003330    MOV     #MSCPMSG,R2       ; Load text table ptr
14 015362    012703    003350    MOV     #MSCPVAL,R3      ; Load value table ptr
15 015366    004737    016042    JSR     PC,CALR1         ; Print MSCP info
16
17 015372    012603    MOV     (SP)+,R3          ;;POP STACK INTO R3
015374    012602    MOV     (SP)+,R2          ;;POP STACK INTO R2
015376    012601    MOV     (SP)+,R1          ;;POP STACK INTO R1
18
19 015400    000167    .WORD   J$JMP
015402    000000    .WORD   L10004-2-.
20
21 015404                L10004:
015404    104423    TRAP   C$MSG
22
23           ;ERR300: General MSCP error routine
24           ;
25           ; Prints ERM300 with MSCP operation text ptr in MSCPNDX and LBN
26           ; Prints extended end message info
27
28 015406    ERR300::
29
30 015406    010146    MOV     R1,-(SP)           ;;PUSH R1 ON STACK
015410    010246    MOV     R2,-(SP)           ;;PUSH R2 ON STACK
015412    010346    MOV     R3,-(SP)           ;;PUSH R3 ON STACK
31
32 015414    013701    004226    MOV     MSCPNDX,R1       ; Load index
33 015420    042701    177774    BIC     #C3,R1           ; Clear illegal values
34 015424    006301                ASL     R1                ; Shift left
35
36 015426    013746    002450    MOV     LBN+2,-(SP)       ;PUSH LBN+2 ON STACK
015432    013746    002446    MOV     LBN,-(SP)        ;PUSH LBN ON STACK
015436    016146    004230    MOV     MSCPTBL(R1),-(SP) ;PUSH MSCPTBL(R1) ON STACK
015442    012746    012076    MOV     #ERM300,-(SP)    ;PUSH #ERM300 ON STACK
015446    004137    033254    JSR     R1,LPNTB         ;CALL LPNTB PRINT ROUTINE
015452    000010                .WORD   ARG.CT             ;ARGUMENT COUNT * 2
37
38 015454    012702    003330    MOV     #MSCPMSG,R2       ; Load text table ptr
39 015460    012703    003350    MOV     #MSCPVAL,R3      ; Load value table ptr
40 015464    004737    016042    JSR     PC,CALR1         ; Print MSCP info
41
42 015470    012603    MOV     (SP)+,R3          ;;POP STACK INTO R3
    
```



```

015472 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
015474 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
43
44 015476 000167      .WORD    J$JMP
015500 000000      .WORD    L10005-2-.
45
46 015502          L10005:
015502 104423      TRAP     C$MSG
47
48                ;ERR301: MSCP error routine
49                ;
50                ; Prints basic msg in BASEMSG with LBN in command packet
51                ; Prints extended end message info
52
53 015504          ERR301::
54
55 015504 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
015506 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
015510 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
56
57 015512 016401 000140      MOV      HC.CPK+P.LBN(R4),R1 ; Unload LBN (low)
58 015516 016402 000142      MOV      HC.CPK+P.LBN+2(R4),R2 ; Unload LBN (high)
59 015522 010246      MOV      R2,-(SP)      ;PUSH R2 ON STACK
015524 010146      MOV      R1,-(SP)      ;PUSH R1 ON STACK
015526 013746 002552      MOV      BASEMSG,-(SP) ;PUSH BASEMSG ON STACK
015532 004137 033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
015536 000006      .WORD    ARG.CT      ;ARGUMENT COUNT + 2
60
61 015540 012702 003330      MOV      #MSCPMSG,R2      ; Load text table ptr
62 015544 012703 003350      MOV      #MSCPVAL,R3      ; Load value table ptr
63 015550 004737 016042      JSR      PC,CALR1      ; Print MSCP info
64
65 015554 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
015556 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
015560 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
66
67 015562 000167      .WORD    J$JMP
015564 000000      .WORD    L10006-2-.
68
69 015566          L10006:
015566 104423      TRAP     C$MSG
70
71                ;ERR302: MSCP error routine
72                ;
73                ; Prints basic msg in BASEMSG with LBN for ACCESS error
74                ; Prints extended end message info
75
76 015570          ERR302::
77
78 015570 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
015572 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
015574 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
79
80 015576 013746 002450      MOV      LBN+2,-(SP)      ;PUSH LBN+2 ON STACK
015602 013746 002446      MOV      LBN,-(SP)      ;PUSH LBN ON STACK
015606 013746 002552      MOV      BASEMSG,-(SP) ;PUSH BASEMSG ON STACK
015612 004137 033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 37 2
 Error processing routines

```

015616 000006          .WORD  ARG.CT          ;ARGUMENT COUNT * 2
81
82 015620 012702 003330  MOV    #MSCPMSG,R2      ; Load text table ptr
83 015624 012703 003350  MOV    #MSCPVAL,R3     ; Load value table ptr
84 015630 004737 016042  JSR    PC,CALR1        ; Print MSCP info
85
86 015634 012603          MOV    (SP)+,R3        ;;POP STACK INTO R3
      015636 012602          MOV    (SP)+,R2        ;;POP STACK INTO R2
      015640 012601          MOV    (SP)+,R1        ;;POP STACK INTO R1
87
88 015642 000167          .WORD  J$JMP
      015644 000000          .WORD  L10007-2-.
89
90 015646          L10007: TRAP    C$MSG
      015646 104423
91

```

```

1
2      ;ERR401: Host/controller communication error routine
3      ;
4      ;      Prints basic msg in BASEMSG
5      ;      Prints extended end message info
6
7 015650      ERR401::
8
9 015650 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
015652 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
015654 010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
10
11 015656 013746 002552      MOV      BASEMSG,-(SP)      ;PUSH BASEMSG ON STACK
015662 004137 033254      JSR      R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
015666 000002      .WORD   ARG.CT          ;ARGUMENT COUNT * 2
12
13 015670 012702 003116      MOV      @SARMSG,R2          ; Load text table ptr
14 015674 012703 003126      MOV      @SARVAL,R3        ; Load value table ptr
15 015700 004737 016042      JSR      PC,CALR1          ; Print MSCP info
16
17 015704 012603      MOV      (SP)+,R3          ;;POP STACK INTO R3
015706 012602      MOV      (SP)+,R2          ;;POP STACK INTO R2
015710 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
18
19 015712 000167      .WORD   J$JMP
015714 000000      .WORD   L10010-2-.
20
21 015716      L10010:
015716 104423      TRAP    C$MSG
22
23      ;ERR601: RCT error routine
24      ;
25      ;      Prints basic msg in BASEMSG
26      ;      Prints extended RCT info
27
28 015720      ERR601::
29
30 015720 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
015722 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
015724 010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
31
32 015726 013746 002552      MOV      BASEMSG,-(SP)      ;PUSH BASEMSG ON STACK
015732 004137 033254      JSR      R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
015736 000002      .WORD   ARG.CT          ;ARGUMENT COUNT * 2
33
34 015740 012702 003660      MOV      @RCTMSG,R2        ; Load text table ptr
35 015744 012703 003670      MOV      @RCTVAL,R3        ; Load value table ptr
36 015750 004737 016042      JSR      PC,CALR1          ; Print RCT info
37
38 015754 012603      MOV      (SP)+,R3          ;;POP STACK INTO R3
015756 012602      MOV      (SP)+,R2          ;;POP STACK INTO R2
015760 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
39
40 015762 000167      .WORD   J$JMP
015764 000000      .WORD   L10011-2-.
41
42 015766      L10011:

```

Error processing routines

43 015766 104423

TRAP C#MSG

```

1
2           ;ERR701: Error Log packet error routine
3           ;
4           ; Prints basic msg in BASEMSG
5           ; Prints extended error log packet info
6
7 015770    ERR701::
8
9 015770    010146    MOV     R1,-(SP)           ;;PUSH R1 ON STACK
015772    010246    MOV     R2,-(SP)           ;;PUSH R2 ON STACK
015774    010346    MOV     R3,-(SP)           ;;PUSH R3 ON STACK
10
11 015776    013746    002552    MOV     BASEMSG,-(SP)       ;PUSH BASEMSG ON STACK
016002    004137    033254    JSR     R1,LPNTB           ;CALL LPNTB PRINT ROUTINE
016006    000002                .WORD   ARG.CT             ;ARGUMENT COUNT * 2
12 016010    013746    002362    MOV     EVENT,-(SP)       ;PUSH EVENT ON STACK
016014    012746    004034    MOV     @XEVENT,-(SP)    ;PUSH @XEVENT ON STACK
016020    004137    033254    JSR     R1,LPNTB           ;CALL LPNTB PRINT ROUTINE
016024    000004                .WORD   ARG.CT             ;ARGUMENT COUNT * 2
13
14 016026    012603    MOV     (SP)+,R3          ;;POP STACK INTO R3
016030    012602    MOV     (SP)+,R2          ;;POP STACK INTO R2
016032    012601    MOV     (SP)+,R1          ;;POP STACK INTO R1
15
16 016034    000167    .WORD   J$JMP
016036    000000    .WORD   L10012-2-.
17
18 016040    L10012:    TRAP   C$MSG
016040    104423
19

```

```

1      .SBTTL  Pre-programmed subroutines
2
3      ;CALR1: Pre-programmed routine 1
4      ;      Prints extended error info
5      ;
6      ;Inputs:
7      ;      EXTMSK  Extended message mask
8      ;      R2      Pointer to ASCIZ msg address table
9      ;      R3      Pointer to value address table
10
11 016042      CALR1:
12
13 016042      20$:
14 016042 032737 000001 002370      BIT      #1,EXTMSK      ; Test if bit set
15 016050 001406                      BEQ      30$          ; Branch if NO
16
17 016052 017346 000000      MOV      @R3,-(SP)      ;PUSH @R3 ON STACK
18 016056 011246      MOV      (R2),-(SP)      ;PUSH (R2) ON STACK
19 016060 004137 033264      JSR      R1,LPNTX      ;CALL LPNTX PRINT ROUTINE
20 016064 000004      .WORD    ARG.CT      ;ARGUMENT COUNT * 2
21
22 016066      30$:
23 016066 006237 002370      ASR      EXTMSK      ; Shift to next position
24 016072 062703 000002      ADD      #2,R3      ; Bump to next value
25 016076 062702 000002      ADD      #2,R2      ; Bump to next msg
26 016102 005712      TST      (R2)      ; Test for null msg
27 016104 001356      BNE      20$      ; Branch if NO
28
29 016106      40$:
30 016106 012746 014231      MOV      #CRLF,-(SP)      ;PUSH #CRLF ON STACK
31 016112 004137 033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
32 016116 000002      .WORD    ARG.CT      ;ARGUMENT COUNT * 2
33 016120 000207      RTS      PC      ; Return
34
35      ;Pre-programmed routine 2
36      ;
37      ;Not used
38
39 016122      CALR2:
40 016122 000207      RTS      PC      ; Return
41
42      ;Pre-programmed routine 3
43      ;
44      ;Not used
45
46 016124      CALR3:
47 016124 000207      RTS      PC      ; Return
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1          ;Pre-programmed routine 4
2          ;
3          ;Not used
4
5 016126   CALR4:
6 016126 000207      RTS      PC          ; Return
7
8          ;Pre-programmed routine 5
9          ;
10         ;Not used
11
12 016130   CALR5:
13 016130 000207      RTS      PC          ; Return
14
15         ;Pre-programmed routine 6
16         ;
17         ;Not used
18
19 016132   CALR6:
20 016132 000207      RTS      PC          ; Return
21
22
23         .SBTTL Error vector table
24
25         ; Pre-programmed error routine dispatch table
26
27 016134 016042      ERR.TB: .WORD CALR1      ; Print SA address
28 016136 016122      .WORD CALR2      ; Print SA value
29 016140 016124      .WORD CALR3      ; Print SA expected value
30 016142 016126      .WORD CALR4      ; Print MSCP parms
31 016144 016130      .WORD CALR5      ; Print LBN, RBN
32 016146 016132      .WORD CALR6      ; Not used
33 016150 016132      .WORD CALR6      ; Not used
34 016152 016132      .WORD CALR6      ; Not used
35
36         000010      ERR.SZ = <.-ERR.TB>/2
37
38         .SBTTL Interrupt service routines
39         ;
40         ;
41         ;
42         ;
43         ;
44         ;
45

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-83 10:42 Page 42
NXMI Nonexistant memory routine

```
1          .SBTTL NXMI   Nonexistant memory routine
2          ;NXMI
3          ;
4          ;Nonexistant memory service routine
5          ;
6          ;Inputs:
7          ;          NXMAD set to zero
8          ;Outputs:
9          ;          NXMAD set to ones if nonexistant trap occured.
10
11 016154   NXMI::
12 016154 012737 177777 002224   MOV     #-1,NXMAD
13 016162   L10013:
14 016162 000002   RTI
```


CZUDLA0 BBR Replacement Utility MACRC Y05.03c Friday, 19 Apr-85 10:42 Page 43
 XDASRV Interrupt service routine

```

1      .SBTTL  XDASRV  Interrupt service routine
2      ;XDASRV
3      ;
4      ;xDA interrupt service routine.  Marks xDA controller table that an
5      ;interrupt has been received.
6      ;
7      ;This routine is called by a [JSR R0,XDASRV] instruction from within
8      ;the controller table.  The PC stored in R0 is the address of the C.FLG
9      ;word in the controller table.  The STACK contains the saved contents
10     ;of R0 followed by the interrupted PC and PS.
11     ;
12     ;Inputs:
13     ;      R0      Address of C.FLG word in controller table
14     ;      STACK   Saved contents of R0
15     ;
16     ;Outputs:
17     ;      CT.CMD  Cleared and CT.MSG set in C.FLG word of controller table
18     ;      R0      Restored from STACK
19
20 016164  XDASRV::
21 016164  052710  000010      BIS      *CT.MSG,(R0)      ;SET CT.MSG
22 016170  012600              MOV      (SP)+,R0      ;;POP STACK INTO R0
23 016172  L10014:
24 016172  000002              RTI

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 44
KW11I Clock interrupt service routine

```
1          .SBTTL KW11I Clock interrupt service routine
2          ;KW11I
3          ;
4          ;Clock interrupt service routine
5
6 016174    KW11I::
7 016174    062737 000001 002220    ADD    #1,KW.EL          ;COUNT THE INTERRUPT
8 016202    005537 002222          ADC    KW.EL+2
9 016206    012777 00C105 163774    MOV    #KW.OUT,#KW.CSR  ;RESTART THE CLOCK
10 016214    L10015:
    016214    000002          RTI
11 016216    INTSRV::
12 016216    005237 002332          INC    INTRCV          ; FLAG INTERRUPT AS RECEIVED
13 016222    L10016:
    016222    000002          RTI
14
```

```

1      .SBTTL RESET   Reset xDA
2      ;RESET
3
4      ;Reset all xDA-50S in the controller tables
5
6      ;Inputs:
7      ;   CTLRLST Contains the address of the 1st controller table
8
9      ;Outputs:
10     ;   NONE
11
12
13     RESET:
14     016224 010246   MOV     R2,-(SP)      ;;PUSH R2 ON STACK
15     016226 010446   MOV     R4,-(SP)      ;;PUSH R4 ON STACK
16     016230 005037   CLR     NXMAD         ; CLEAR NON-EXISTANT MEMORY ADDRESS
17     016234 012737   177776 002300   MOV     #2,TBLNDX    ; Reset controller table index
18     016242 012746   000340   MOV     #PRI07,-(SP)
19     016246 012746   016154   MOV     #NXMI,-(SP)
20     016252 012746   000004   MOV     #4,-(SP)
21     016256 012746   000003   MOV     #3,-(SP)
22     016262 104437   TRAP   C$SVEC
23     016264 062706   000010   ADD     #10,SP
24
25     ;>>>>>>>>>BREAK BACK TO MONITOR<<<<<<<<<<<<
26     10$:
27     TRAP   C$BRK
28
29     20$:
30     ADD     #2,TBLNDX      ; Bump table index
31     MOV     TBLNDX,R2     ; Load index value
32     MOV     CTLRLST(R2),R4 ; Load address
33     BEQ     50$          ; Branch if done
34
35     30$:
36     MOV     R4,CTLRTBL    ; and save
37     CLR     @R4+         ; INITIALIZE xDA
38     TST     NXMAD         ; Test for address error
39     BEQ     10$          ; Branch if NO
40
41     50$:
42     MOV     #ERM102,BASEMSG ; "Memory error..."
43     MOV     #8,SADDR,EXTMSK ; Flags address only
44     TRAP   C$ERDF
45     .WORD  102
46     .WORD  HDR101
47     .WORD  ERR101
48     BR     10$          ; and continue
49
50     50$:
51     MOV     #4,R0
52     TRAP   C$CVEC
53     MOV     (SP)+,R4      ;;POP STACK INTO R4
54     MOV     (SP)+,R2      ;;POP STACK INTO R2
55     RTS     PC
56
57     40

```

```

1      .SBTTL RCTERR Process RCT block error
2      ;RCTERR
3      ;
4      ;This routine updates the RCT error table. The table contains one entry
5      ;for each RCT block. The entry is a bit field for up to 16 copies of each
6      ;RCT block. A new RCT error will set the corresponding copy bit in its entry.
7      ;
8      ;Inputs:
9      ;   RCTCOPY RCT copy count
10     ;   RCTTBL  Address of RCT error table
11     ;
12     ;Outputs:
13     ;   STATUS  Exit status
14     ;           1: New bad RCT block
15     ;           0: Bad RCT block not new
16     ;          -1: All copies of RCT block bad
17     ;
18     ;Calling Seq:
19     ;
20     ;   JSR     PC,RCTERR
21     ;   .WORD  <error exit>
22     ;
23
24 016370 RCTERR:
25
26 016370 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
27 016372 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
28 016374 113700 002424      MOVB   RCTCOPY,R0      ; Load number of RCT copies
29 016400 012737 000001 002426      MOV     #1,RCTMASK    ; Initialize copy mask
30 016406                10$:
31 016406 005300      DEC     R0            ; Decrement copy number
32 016410 001403      BEQ    20$           ; Branch if zero
33 016412 006337 002426      ASL    RCTMASK       ; Else, shift mask
34 016416 000773      BR     10$           ; And continue
35 016420                20$:
36 016420 013702 002312      MOV     RCTTBL,R2     ; Load table address
37 016424 013700 002476      MOV     RCTBLK,R0    ; Load RCT block no.
38 016430 006300      ASL    R0            ; Times two for index
39 016432 060002      ADD    R0,R2         ; Add to table start
40 016434 033712 002426      BIT    RCTMASK,(R2) ; Test if this block already bad
41 016440 001057      BNE    60$           ; Branch if YES
42 016442                30$:
43 016442 013704 002324      MOV     HCOM,R4       ; Load comm area ptr
44 016446 016446 000142      MOV     HC.CPK+P.LBN+2(R4),-(SP) ;PUSH HC.CPK+P.LBN+2(R4) ON STACK
45 016452 016446 000140      MOV     HC.CPK+P.LBN(R4),-(SP) ;PUSH HC.CPK+P.LBN(R4) ON STACK
46 016456 013746 002476      MOV     RCTBLK,-(SP) ;PUSH RCTBLK ON STACK
47 016462 013746 002424      MOV     RCTCOPY,-(SP) ;PUSH RCTCOPY ON STACK
48 016466 012746 010114      MOV     #INF401,-(SP) ;PUSH #INF401 ON STACK
49 016472 004137 033254      JSR    R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
50 016476 000012      .WORD  ARG.CT        ;ARGUMENT COUNT * 2
51 016500 013746 002354      MOV     EMSTAT,-(SP) ;PUSH EMSTAT ON STACK
52 016504 012746 004035      MOV     #XEVNT,-(SP) ;PUSH #XEVNT ON STACK
53 016510 004137 033254      JSR    R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
54 016514 000004      .WORD  ARG.CT        ;ARGUMENT COUNT * 2
55
56
57 016516 053712 002426      BIS    RCTMASK,(R2) ; Else, mark this block bad

```

CZUDLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 46-1
 RCTERR Process RCT block error

```

48 016522 005000          CLR      R0          ; Clear register
49 016524 113700 002423  MOVB     COPIES,R0    ; Load number of copies
50 016530 011202          MOV      (R2),R2     ; Load entry
51 016532 000241          CLC                ; Clear carry
52 016534          35$: ASR      R2          ; Shift LSB into Carry
53 016534 006202          BCC     50$         ; Branch if bit clear
54 016536 103011          DEC     R0          ; Decrement copy count
55 016540 005300          BNE     35$         ; Continue until zero
56 016542 001374
57
58          ; Fatal error exit
59 016544          40$: MOV      (SP)+,R2     ;; POP STACK INTO R2
60 016544 012602          MOV      (SP)+,R1     ;; POP STACK INTO R1
   016546 012601          MOV      @ (SP), (SP) ; Load fatal error return
61 016550 017616 000000  MOV      #-1,R0      ; Load fatal status
62 016554 012700 177777  RTS      PC          ; Return
63 016560 000207
64
65          ; Report new bad RCT block
66 016562          50$: MOV      (SP)+,R2     ;; POP STACK INTO R2
67 016562 012602          MOV      (SP)+,R1     ;; POP STACK INTO R1
   016564 012601          ADD     #2,(SP)      ; Bump past error return
68 016566 062716 000002  MOV      #1,R0      ; Report new bad RCT block
69 016572 012700 000001  RTS      PC          ; Return
70 016576 000207
71
72          ; Bad RCT block was not new
73 016600          60$: MOV      (SP)+,R2     ;; POP STACK INTO R2
74 016600 012602          MOV      (SP)+,R1     ;; POP STACK INTO R1
   016602 012601          ADD     #2,(SP)      ; Bump past error return
75 016604 062716 000002  MOV      #0,R0      ; Report bad RCT block not new
76 016610 012700 000000  RTS      PC          ; Return
77 016614 000207
78

```

```

1      .SBTTL ERRLOG Process error log packet
2      ;ERRLOG
3      ;
4      ;This routine processes all ERRORLOG msg packets. Errorlog types 0, 1, and 3
5      ;will cause a DEVICE FATAL error. Disk transfer errors will print msg.
6      ;
7      ;Inputs:
8      ;       HCOM      Host comm. area
9      ;Outputs:
10     ;       STATUS   Exit status
11     ;               0: Disk transfer errorlog packet (no msg)
12     ;               -1: Fatal errorlog packet (err msg)
13     ;
14     ;Calling Seq:
15     ;
16     ;       JSR      PC,ERRLOG
17     ;       .WORD   <error exit>
18     ;
19
20     016616      ERLPTBL:
21     016616 016734      .WORD   ERRLO      ; Controller error
22     016620 016772      .WORD   ERRL1     ; Memory access error
23     016622 017066      .WORD   ERRL2     ; Disk transfer error
24     016624 017224      .WORD   ERRL3     ; SDI error
25     016626 017462      .WORD   ERRL4     ; Small disk error
26
27     000005      ERLPSIZ =      <.-ERLPTBL>/2      ; maximum index
28
29     016630 000000 000000      ERLAST: .WORD   0,0      ; Last LBN
30
31     016634      ERRLOG:
32     016634 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
33
34     016636 013704 002324      MOV      HCOM,R4      ; Load comm area ptr
35     016642 116437 000030 002372      MOV     HC.MPK+L.FMT(R4),ERLFMT ; Save format specifier
36     016650 016437 000032 002362      MOV     HC.MPK+L.EVNT(R4),EVENT ; Save event code
37     016656 113701 002372      MOV     ERLFMT,R1     ; Load format spec
38     016662 122701 000005      CMPB    #ERLPSIZ,R1  ; Compare with max
39     016666 003017      BGT     10$          ; Branch if OK
40
41     ; Unrecognized errorlog format
42     016670      5$:
43     016670 012764 140000 000006      MOV     #RG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
44     016676 012737 000001 002370      MOV     #B.EVNT,EXTMSK      ; Load extended msg mask
45     015704 012737 013763 002552      MOV     #ERM700,BASEMSG     ; Load basic msg address
46     016712 104455      TRAP    C$ERDF
47     016714 001274      .WORD   700
48     016716 013731      .WORD   HDR701
49     016720 015770      .WORD   ERR701
50     016722 000137 017514      JMP     ERLEXIT      ; Exit
51
52     ; Process errorlog
53     016726      10$:
54     016726 006301      ASL     R1      ; Shift index
55     016730 000171 016616      JMP     @ERLPTBL(R1) ; Process errorlog

```

```

1
2
3 016734          ; ERRLO: Controller error
4 016734 012764 140000 000006 MOV    @RG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
5 016742 012737 000001 002370 MOV    @B.EVNT,EXTMSK          ; Load extended msg mask
6 016750 012737 014024 002552 MOV    @ERM701,BASEMSG        ; Load basic msg address
7 016756 104455 TRAP   C$ERDF
   016760 001275 .WORD  701
   016762 013731 .WORD  HDR701
   016764 015770 .WORD  ERR701
8 016766 000137 017514 JMP    ERLEXIT                ; Exit
9
10          ; ERR1: Memory access error
11 016772          ; ERR1:
12 016772 016437 000050 002400 MOV    HC.MPK+L.BADR(R4),HADDR; Save
13 017000 016437 000050 002402 MOV    HC.MPK+L.RADR+2(R4),HADDR+2; address
14
15 017006 012764 140000 000006 MOV    @RG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
16
17 017014 012737 000001 002370 MOV    @B.EVNT,EXTMSK          ; Load extended msg mask
18 017022 012737 014062 002552 MOV    @ERM702,BASEMSG        ; Load basic msg address
19 017030 104455 TRAP   C$ERDF
   017032 001276 .WORD  702
   017034 013731 .WORD  HDR701
   017036 015770 .WORD  ERR701
20 017040 013746 002402 MOV    HADDR+2,-(SP)          ;PUSH HADDR+2 ON STACK
   017044 013746 002400 MOV    HADDR,-(SP)           ;PUSH HADDR ON STACK
   017050 012746 004063 MOV    @XHADDR,-(SP)        ;PUSH @XHADDR ON STACK
   017054 004137 033264 JSR    R1,LPNTX             ;CALL LPNTX PRINT ROUTINE
   017060 000006 .WORD  ARG.CT              ;ARGUMENT COUNT * 2
21 017062 000137 017514 JMP    ERLEXIT                ; Exit
22

```

CZUOLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 49
 ERRLOG Process error log packet

```

1
2
3 017066          ; Disk transfer error
4 017066 016437 000070 002374  ; ERR2:
5 017074 016437 000072 002376  MOV HC.MPK+L.HDCD(R4),ERLLBN; Save
6 017102 012764 140000 000006  MOV HC.MPK+L.HDCD+2(R4),ERLLBN+2; LBN
7                                     MOV ARG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
8 017110 000436          BR 20$ ; Branch if NO
9 017112 023737 002374 016630  CMP ERLLBN,ERLAST ; Else, test last LBN (low)
10 017120 001004          BNE 10$ ; Branch if not same
11 017122 023737 002376 016632  CMP ERLLBN+2,ERLAST+2 ; Test LBN (high)
12 017130 001426          BEQ 20$ ; Branch if same
13 017132          10$:
14 017132 013737 002374 016630  MOV ERLLBN,ERLAST ; Save
15 017140 013737 002376 016632  MOV ERLLBN+2,ERLAST+2 ; Last LBN
16
17 017146 013746 002376          MOV ERLLBN+2,-(SP) ;PUSH ERLLBN+2 ON STACK
   017152 013746 002374          MOV ERLLBN,-(SP) ;PUSH ERLLBN ON STACK
   017156 012746 010166          MOV #INF405,-(SP) ;PUSH #INF405 ON STACK
   017162 004137 033264          JSR R1,LPNTX ;CALL LPNTX PRINT ROUTINE
   017166 000306          .WORD ARG.CT ;ARGUMENT COUNT * 2
18 017170 013746 002362          MOV EVENT,-(SP) ;PUSH EVENT ON STACK
   017174 012746 004035          MOV #XEVNT,-(SP) ;PUSH #XEVNT ON STACK
   017200 004137 033264          JSR R1,LPNTX ;CALL LPNTX PRINT ROUTINE
   017204 000004          .WORD ARG.CT ;ARGUMENT COUNT * 2
19 017206          20$:
20 017206 012601          MOV (SP)+,R1 ;;POP STACK INTO R1
21 017210 062716 000002          ADD #2,(SP) ; Bump past error return
22 017214 012737 000000 002442  MOV #0,STATUS ; Load success
23 017222 000207          RTS PC ; Return to caller
24

```



```

1
2
3 017224          ; SDI error
4 017224 016437 000070 002374 ; ERR3: MOV HC.MPK+L.HDCD(R4),ERLLBN; Save
5 017232 016437 000072 002376 ; MOV HC.MPK+L.HDCD+2(R4),ERLLBN+2; LBN
6 017240 062704 000100          ; ADD #HC.MPK+L.DS,R4 ; Add offset to status/error info
7 017244 012701 002406          ; MOV #ERLDS,R1 ; Load local address
8 017250 012702 000004          ; MOV #4,R2 ; Load counter
9 017254          10$:
10 017254 012421          ; MOV (R4)+,(R1)+ ; Move data
11 017256 005302          ; DEC R2 ; Decrement count
12 017260 001375          ; BNE 10$ ; continue until zero
13 017262          15$:
14 017262 013704 002324          ; MOV HCOM,R4 ; Reload comm area ptr
15 017266 012764 140000 000006 ; MOV #RG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
16
17 017274 012737 000001 002370 ; MOV #B.EVNT,EXTMSK ; Load extended msg mask
18 017302 012737 014130 002552 ; MOV #ERM703,BASEMSG ; Load basic msg address
19 017310 104455          ; TRAP C$ERDF
    017312 001277          ; .WORD 703
    017314 013731          ; .WORD HDR701
    017316 015770          ; .WORD ERR701
20
21 017320 013746 002376          ; MOV ERLLEN+2,-(SP) ; PUSH ERLLEN+2 ON STACK
    017324 013746 002374          ; MOV ERLLEN,-(SP) ; PUSH ERLLEN ON STACK
    017330 012746 014575          ; MOV #XLBN,-(SP) ; PUSH #XLBN ON STACK
    017334 004137 033264          ; JSR R1,LPNTX ; CALL LPNTX PRINT ROUTINE
    017340 000006          ; .WORD ARG.CT ; ARGUMENT COUNT * 2
22
23 017342 005737 003246          ; TST MDLNDX ; Test for known model
24 017346 001421          ; BEQ 30$ ; Branch if zero
25
26 017350 012701 000006          ; MOV #6,R1 ; Load drive code byte index
27 017354 122737 000004 003244 ; CMPB #M.RA60,MODEL ; Test for RA60
28 017362 001001          ; BNE 20$ ; Branch if NO
29 017364 005201          ; INC R1 ; Else, change index for RA60
30 017366          20$:
31 017366 116137 002406 002404 ; MOV# ERLDS(R1),DCODE ; Save drive error code
32 017374 013746 002404          ; MOV DCODE,-(SP) ; PUSH DCODE ON STACK
    017400 012746 004112          ; MOV #XDCODE,-(SP) ; PUSH #XDCODE ON STACK
    017404 004137 033264          ; JSR R1,LPNTX ; CALL LPNTX PRINT ROUTINE
    017410 000004          ; .WORD ARG.CT ; ARGUMENT COUNT * 2
33 017412          30$:
34 017412 012746 004152          ; MOV #XDINFO,-(SP) ; PUSH #XDINFO ON STACK
    017416 004137 033264          ; JSR R1,LPNTX ; CALL LPNTX PRINT ROUTINE
    017422 000002          ; .WORD ARG.CT ; ARGUMENT COUNT * 2
35 017424 012702 000010          ; MOV #8,R2 ; Load byte count
36 017430 012701 002406          ; MOV #ERLDS,R1 ; Load ptr to status info
37 017434          35$:
38 017434 112104          ; MOV# (R1)+,R4 ; Load status
39 017436 010446          ; MOV R4,-(SP) ; PUSH R4 ON STACK
    017440 012746 004217          ; MOV #XDHEX,-(SP) ; PUSH #XDHEX ON STACK
    017444 004137 033264          ; JSR R1,LPNTX ; CALL LPNTX PRINT ROUTINE
    017450 000004          ; .WORD ARG.CT ; ARGUMENT COUNT * 2
40 017452 005302          ; DEC R2 ; Decrement count
41 017454 001367          ; BNE 35$ ; continue until zero
42 017456 000137 017514          ; JMP ERLEXIT ; Exit

```



```

1
2
3 017462      ; Small disk error
4 017462 012764 140000 000006  ERRRL4:  MOV   #RG.OWN+RG.FLG,HC.MCT(R4); Return msg descriptor to controller
5 017470 012737 000001 002370      MOV   #B.EVNT,EXTMSK           ; Load extended msg mask
6 017476 012737 014157 002552      MOV   #ERM704,BASEMSG        ; Load basic msg address
7 017504 104455      TRAP  C#ERDF
   017506 001300      .WORD 704
   017510 013731      .WORD HDR701
   017512 015770      .WORD ERR701
8
9 017514      ERLEXIT:
10 017514 013704 002324      MOV   HCOM,R4                ; Reload comm area ptr
11 017520 005737 002650      TST   DEBUG                  ; Test for DEBUG
12 017524 001007      BNE   20$                    ; Branch if yes
13 017526      10$:
14 017526 012601      MOV   (SP)+,R1                ;;POP STACK INTO R1
15 017530 017616 000000      MOV   @B(SP),R1              ; Load fatal return
16 017534 012737 177777 002442      MOV   #-1,STATUS            ; Fatal status
17 017542 000207      RTS   PC                     ; and return
18 017544      20$:
19 017544 012601      MOV   (SP)+,R1                ;;POP STACK INTO R1
20 017546 062716 000002      ADD   #2,(SP)                ; Bump past error return
21 017552 012737 000000 002442      MOV   #0,STATUS             ; Load success
22 017560 000207      RTS   PC                     ; Return to caller
23

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 52
 CACHTST Bad block cache test

```

1      .SBTTL CACHTST Bad block cache test
2      ;CACHTST
3      ;
4      ;Inputs:
5      ;      LBN      Current LBN
6      ;      CACHTBL  Address of caching table
7      ;
8      ;Outputs:
9      ;      STATUS  Exit status
10     ;              1: LBN already in caching table
11     ;              0: LBN entered in caching table
12     ;              -1: Caching table full
13     ;
14     ;Calling Seq:
15     ;
16     ;      JSR      PC,CACHTST
17     ;      .WORD   <error exit>
18     ;
19
20     CACHTST:
21     017562      010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
22     017564      010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
23     017566      010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
24     017570      013702      002310    MOV      CACHTBL,R2       ; Load table start
25     017574      005000      CLR      R0                ; Clear cache entry count
26     017576      005762      000002    20$:    TST      2(R2)           ; Test for null
27     017602      100461      BMI     50$                ; Branch if YES
28     017604      023712      002446    CMP      LBN,(R2)         ; Compare low order LBN
29     017610      001007      BNE     30$                ; Branch if not same
30     017612      016201      000002    MOV      2(R2),R1         ; Else, load high order LBN
31     017616      042701      170000    BIC      #170000,R1       ; Clear flags
32     017622      020137      002450    CMP      R1,LBN+2        ; Compare high order LBN
33     017626      001431      BEQ     40$                ; Branch if same
34     017630      062702      000004    30$:    ADD      #4,R2            ; Bump to next entry
35     017634      005200      INC     R0                ; Increment count
36     017636      022700      000620    CMP      #CACHE SIZE,R0  ; Compare with max
37     017642      003355      BGT     20$                ; Branch if less
38     017644      012737      013621    35$:    MOV      #ERM607,BASEMSG  ; "transient error overflow..."
39     017652      012737      000000    MOV      #0,EXTMSK       ; No extended msg
40     017660      104455      TRAP   C$ERDF            ;
41     017662      001137      .WORD  607                ;
42     017664      013173      .WORD  HDR601             ;
43     017666      015720      .WORD  ERR601             ;
44     017670      012603      MOV     (SP)+,R3          ;;POP STACK INTO R3
45     017672      012602      MOV     (SP)+,R2          ;;POP STACK INTO R2
46     017674      012601      MOV     (SP)+,R1          ;;POP STACK INTO R1
47     017676      017616      000000    MOV     @SP,(SP)         ; Else, load error return
48     017702      012737      177777    MOV     #-1,STATUS       ; Set fatal status
49     017710      000207      RTS     PC                ; and return
50     017712      012712      177777    40$:    MOV     #-1,(R2)         ; Reset LBN (low)
51     017716      012762      177777    MOV     #-1,2(R2)        ; Reset LBN (high)

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 52-1
CACTST Bad block cache test

```
51 017724 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
    017726 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
    017730 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
52 017732 062716 000002  ADD      #2,(SP)      ; Bump past error return
53 017736 012737 000001 002442  MOV      #1,STATUS    ; Set replace status
54 017744 000207      RTS      PC           ; and return
55 017746      50+:
56 017746 013712 002446      MOV      LBN,(R2)     ; Save LBN
57 017752 013762 002450 000002  MOV      LBN+2,2(R2)  ;
58 017760 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
    017762 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
    017764 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
59 017766 062716 000002  ADD      #2,(SP)      ; Bump past error return
60 017772 012737 000000 002442  MOV      #0,STATUS    ; Set replace status
61 020000 000207      RTS      PC           ; and return
62
```

```

1      .SBTTL TYPSTAT Type replacement status
2      ;TYPSTAT
3      ;
4      ;THIS ROUTINE TYPES THE LBN, RBN REPLACEMENT INFORMATION TO THE TTY.
5      ;
6      ;INPUTS.
7      ;   SFLAG (Search status flag),
8      ;   MATFLG (Match status flag),
9      ;   LBN (Lo order word of LBN being replaced),
10     ;   LBN+2 (Hi order word of LBN being replaced),
11     ;   RBN (Lo order word of RBN replacement),
12     ;   RBN+2 (Hi order word of RBN replacement),
13     ;   MATRBN (Lo order word of RBN which matched the LBN being replaced),
14     ;   MATRBN+2 (Hi order word of RBN which matched the LBN being replaced)
15     ;
16     ;OUTPUTS:
17     ;   TYPE APPROPRIATE MESSAGE TO THE TTY
18     ;
19     ;   LBNSTAT: LBN status word
20     ;
21
22     TYPSTAT:
23     020002      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
24     020004      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
25
26     10$:
27     020006      105737      002514    TSTB     MATFLG      ; Test for LBN match
28     020012      001444      BEQ      20$         ; Branch if NO, else print msg
29
30     ;   Print UNUSABLE operation
31     15$:
32     020014      005237      002440    INC      UNUSE      ; Increment number of UNUSABLE RBNs
33     020020      013746      002450    MOV      LBN+2,-(SP) ;PUSH LBN+2 ON STACK
34     020024      013746      002446    MOV      LBN,-(SP)  ;PUSH LBN ON STACK
35     020030      012746      010166    MOV      @INF405,-(SP) ;PUSH @INF405 ON STACK
36     020034      004137      033254    JSR      R1,LPNTB   ;CALL LPNTB PRINT ROUTINE
37     020040      000006      .WORD    ARG.CT     ;ARGUMENT COUNT * 2
38     020042      012746      015174    MOV      @TX.BLK,-(SP) ;PUSH @TX.BLK ON STACK
39     020046      012746      010206    MOV      @INF406,-(SP) ;PUSH @INF406 ON STACK
40     020052      004137      033254    JSR      R1,LPNTB   ;CALL LPNTB PRINT ROUTINE
41     020056      000004      .WORD    ARG.CT     ;ARGUMENT COUNT * 2
42     020060      013746      002464    MOV      MATRBN+2,-(SP) ;PUSH MATRBN+2 ON STACK
43     020064      013746      002462    MOV      MATRBN,-(SP) ;PUSH MATRBN ON STACK
44     020070      012746      010224    MOV      @INF407,-(SP) ;PUSH @INF407 ON STACK
45     020074      004137      033254    JSR      R1,LPNTB   ;CALL LPNTB PRINT ROUTINE
46     020100      000006      .WORD    ARG.CT     ;ARGUMENT COUNT * 2
47     020102      012746      015131    MOV      @TX.UNS,-(SP) ;PUSH @TX.UNS ON STACK
48     020106      012746      010243    MOV      @INF408,-(SP) ;PUSH @INF408 ON STACK
49     020112      004137      033254    JSR      R1,LPNTB   ;CALL LPNTB PRINT ROUTINE
50     020116      000004      .WORD    ARG.CT     ;ARGUMENT COUNT * 2
51     020120      105037      002514    CLRB     MATFLG     ; Clear flag
52
53     20$:
54     020124      012702      000000    MOV      @0,R2      ; Zero index
55     020130      013701      002516    MOV      LBNSTAT,R1 ; Load status
56     020134      100422      BMI      30$         ; Branch if error status
57     020136      003111      BGT      50$         ; Else, branch if flag set
58
59
60
61
62

```

CZUDL BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 53 1
 TYPSTA: Type replacement status

```

43          ;          Print LBN with blank status
44 020140   ;          25$:
45 020140   013746 002450   MOV     LBN+2,-(SP)      ;PUSH LBN+2 ON STACK
    020144   013746 002446   MOV     LBN,-(SP)       ;PUSH LBN ON STACK
    020150   012746 010166   MOV     #INF405,-(SP)   ;PUSH #INF405 ON STACK
    020154   004137 033254   JSR    R1,LPNTB         ;CALL LPNTB PRINT ROUTINE
    020160   000006                .WORD  ARG.CT           ;ARGUMENT COUNT * 2
46 020162   012746 015174   MOV     #TX.BLK,-(SP)   ;PUSH #TX.BLK ON STACK
    020166   012746 010206   MOV     #INF406,-(SP)   ;PUSH #INF406 ON STACK
    020172   004137 033254   JSR    R1,LPNTB         ;CALL LPNTB PRINT ROUTINE
    020176   000004                .WORD  ARG.CT           ;ARGUMENT COUNT * 2
47 020200   000400                BR      30$             ; Exit
48
49          ;          Print LBN status code
50 020202   ;          30$:
51 020202   042701 177760   BIC     #C17,R1         ; Clear all but data codes
52 020206   001436                BEQ     40$             ; Branch if None set
53 020210   ;          32$:
54 020210   036201 020612   BIT     BITTBL(R2),R1   ; Test for bit set
55 020214   001426                BEQ     35$             ; Branch if bit clear
56
57 020216   013746 002450   MOV     LBN+2,-(SP)     ;PUSH LBN+2 ON STACK
    020222   013746 002446   MOV     LBN,-(SP)       ;PUSH LBN ON STACK
    020226   012746 010166   MOV     #INF405,-(SP)   ;PUSH #INF405 ON STACK
    020232   004137 033254   JSR    R1,LPNTB         ;CALL LPNTB PRINT ROUTINE
    020236   000006                .WORD  ARG.CT           ;ARGUMENT COUNT * 2
58 020240   016246 020624   MOV     STATBL(R2),-(SP) ;PUSH STATBL(R2) ON STACK
    020244   012746 010206   MOV     #INF406,-(SP)   ;PUSH #INF406 ON STACK
    020250   004137 033254   JSR    R1,LPNTB         ;CALL LPNTB PRINT ROUTINE
    020254   000004                .WORD  ARG.CT           ;ARGUMENT COUNT * 2
59 020256   005272 020636   INC     @CNTTBL(R2)     ; Increment counter
60 020262   046237 020612 002516   BIC     BITTBL(R2),LBNSTAT ; Clear so we won't print again
61 020270   000405                BR      40$             ; Exit loop
62 020272   ;          35$:
63 020272   062702 000002   ADD     #2,R2           ; Else, bump index
64 020276   005762 020624   TST    STATBL(R2)      ; Test address
65 020302   100342                BPL     32$             ; continue if not null
66

```

```

1
2
3      ; Test for Write w/FE
4 020304      032737      000400      002516      ; 40$:
5 020312      001451
6
7 020314      013746      002450      MOV      LBN+2,-(SP)      ;PUSH LBN+2 ON STACK
   020320      013746      002446      MOV      LBN,-(SP)      ;PUSH LBN ON STACK
   020324      012746      010166      MOV      #INF405,-(SP)      ;PUSH #INF405 ON STACK
   020330      004137      033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
   020334      000006      .WORD      ARG.CT      ;ARGUMENT COUNT * 2
8 020336      012746      015201      MOV      #TX.WFE,-(SP)      ;PUSH #TX.WFE ON STACK
   020342      012746      010206      MOV      #INF406,-(SP)      ;PUSH #INF406 ON STACK
   020346      004137      033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
   020352      000004      .WORD      ARG.CT      ;ARGUMENT COUNT * 2
9 020354      005237      002542      INC      WFECNT      ; Increment count
10 020360      000426      BR      60$      ; Type replacement info
11
12      ; Test for BBR
13 020362      032737      000200      002516      ; 50$:
14 020362      032737      000200      002516      BIT      #TF.BBR,LBNSTAT      ; Test for BBR
15 020370      001422      BEQ      60$      ; Branch if NO
16
17 020372      013746      002450      MOV      LBN+2,-(SP)      ;PUSH LBN+2 ON STACK
   020376      013746      002446      MOV      LBN,-(SP)      ;PUSH LBN ON STACK
   020402      012746      010166      MOV      #INF405,-(SP)      ;PUSH #INF405 ON STACK
   020406      004137      033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
   020412      000006      .WORD      ARG.CT      ;ARGUMENT COUNT * 2
18 020414      012746      015150      MOV      #TX.BBR,-(SP)      ;PUSH #TX.BBR ON STACK
   020420      012746      010206      MOV      #INF406,-(SP)      ;PUSH #INF406 ON STACK
   020424      004137      033254      JSR      R1,LPNTB      ;CALL LPNTB PRINT ROUTINE
   020430      000004      .WORD      ARG.CT      ;ARGUMENT COUNT * 2
19 020432      005237      002540      INC      BBRCNT      ; Increment count
20

```



```

1          ;      Print replacement operation
2 020436   ;      60$:
3 020436 005737 002520   TST    RPLSTAT      ; Test replacement status
4 020442 001455          BEQ    100$         ; Exit if none
5
6 020444 122737 000000 002515   CMPB   #0,SFLAG     ; Test for PRIMARY search status
7 020452 001023          BNE    80$         ; Branch if NO
8
9          ;      Print PRIMARY operation
10
11 020454 005237 002434   INC    PRIMRY       ; Increment PRIMARY RBN count
12 020460 013746 002454   MOV    RBN+2,-(SP)  ; PUSH RBN+2 ON STACK
   020464 013746 002452   MOV    RBN,-(SP)   ; PUSH RBN ON STACK
   020470 012746 010224   MOV    #INF407,-(SP) ; PUSH #INF407 ON STACK
   020474 004137 033254   JSR    R1,LPNTB    ; CALL LPNTB PRINT ROUTINE
   020500 000006          .WORD  ARG.CT      ; ARGUMENT COUNT * 2
13 020502 012746 015117   MOV    #TX.PRI,-(SP) ; PUSH #TX.PRI ON STACK
   020506 012746 010243   MOV    #INF408,-(SP) ; PUSH #INF408 ON STACK
   020512 004137 033254   JSR    R1,LPNTB    ; CALL LPNTB PRINT ROUTINE
   020516 000004          .WORD  ARG.CT      ; ARGUMENT COUNT * 2
14 020520 000426          BR     100$        ; and exit
15 020522          ;      80$:
16 020522 122737 000001 002515   CMPB   #1,SFLAG     ; Test for SECONDARY search status
17 020530 001022          BNE    100$        ; Branch if NO
18
19          ;      Print SCEONDARY operation
20
21 020532 005237 002436   INC    SCNDRY       ; Increment SECONDARY RBN count
22 020536 013746 002454   MOV    RBN+2,-(SP)  ; PUSH RBN+2 ON STACK
   020542 013746 002452   MOV    RBN,-(SP)   ; PUSH RBN ON STACK
   020546 012746 010224   MOV    #INF407,-(SP) ; PUSH #INF407 ON STACK
   020552 004137 033254   JSR    R1,LPNTB    ; CALL LPNTB PRINT ROUTINE
   020556 000006          .WORD  ARG.CT      ; ARGUMENT COUNT * 2
23 020560 012746 015124   MOV    #TX.SEC,-(SP) ; PUSH #TX.SEC ON STACK
   020564 012746 010243   MOV    #INF408,-(SP) ; PUSH #INF408 ON STACK
   020570 004137 033254   JSR    R1,LPNTB    ; CALL LPNTB PRINT ROUTINE
   020574 000004          .WORD  ARG.CT      ; ARGUMENT COUNT * 2
24 020576          ;      100$:
25 020576 012737 000000 002520   MOV    #0,RPLSTAT   ; Zero replacement status
26 020604 012601          MOV    (SP)+,R1     ;; POP STACK INTO R1
   020606 012602          MOV    (SP)+,R2     ;; POP STACK INTO R2
27 020610 000207          RTS    PC           ; RETURN
28
29
30 020612          BITTBL:
31 020612 000001          .WORD  TF.FER
32 020614 000002          .WORD  TF.HCE
33 020616 000004          .WORD  TF.DST
34 020620 000010          .WORD  TF.ECC
35 020622 177777          .WORD  -1
36
37 020624          STATBL:
38 020624 015136          .WORD  TX.FER      ; BIT0
39 020626 015206          .WORD  TX.HCE      ; BIT1
40 020630 015213          .WORD  TX.DST      ; BIT2
41 020632 015143          .WORD  TX.ECH      ; BIT3
42 020634 177777          .WORD  -1          ; Terminator

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr-85 10:42 Page 55-1
TYPSTAT Type replacement status

43
44
45
46
47
48
49
50

020636
020636 002530
020640 002532
020642 002534
020644 002536
020646 177777

CNTTBL:

.WORD	FERCNT	: FER count
.WORD	HCECNT	: HCE count
.WORD	DSTCNT	: DST count
.WORD	ECCCNT	: ECC count
.WORD	-1	: Terminator

1				.SBTTL RNTIME Print runtime	
2				;RNTIME	
3				:	
4				;PRINT RUNTIME	
5				:	
6				;INPUTS:	
7				; KW.EL - CONTAINS ELAPSED TIME	
8				; KW.HZ - HERTZ OF CLOCK	
9				;OUTPUTS:	
10				; IF CLOCK ON SYSTEM:	
11				; " RNTIME HH:MM:SS " PRINTED	
12				; IF NO CLOCK: ONE SPACE IS PRINTED	
13					
14	020650	005737	002210	RNTIME: TST KW.CSR	;CHECK IF A CLOCK PRESENT
15	020654	001470		BEQ RNTIMX	;BRANCH IF NOT
16	020656	010046		MOV R0,-(SP)	::PUSH R0 ON STACK
	020660	010346		MOV R3,-(SP)	::PUSH R3 ON STACK
	020662	010446		MOV R4,-(SP)	::PUSH R4 ON STACK
	020664	010546		MOV R5,-(SP)	::PUSH R5 ON STACK
17	020666	013703	002220	MOV KW.EL,R3	;GET ELAPSED TIME
18	020672	013704	002222	MOV KW.EL+2,R4	
19	020676	013700	002216	MOV KW.HZ,R0	;GET SPEED OF CLOCK
20	020702	004737	034436	JSR PC,DIVIDE	;COMPUTE SECONDS OF ELAPSED TIME
21	020706	012700	000074	MOV #60,R0	;NOW DIVIDE BY 60
22	020712	004737	034436	JSR PC,DIVIDE	; TO COMPUTE MINUTES
23	020716	010546		MOV R5,-(SP)	::PUSH R5 ON STACK
24	020720	004737	034436	JSR PC,DIVIDE	;DIVIDE BY 60 AGAIN
25	020724	010346		MOV R3,-(SP)	;PUSH R3 ON STACK
	020726	012746	014233	MOV #RNTIM,-(SP)	;PUSH #RNTIM ON STACK
	020732	004137	033302	JSR R1,LPNT	;CALL LPNT PRINT ROUTINE
	020736	000004		.WORD ARG.CT	;ARGUMENT COUNT * 2
26	020740	020527	000011	CMP R5,#9.	;IF MINUTES 9 OR LESS
27	020744	003004		BGT 1\$	
28	020746	112700	000060	MOVB #'0,R0	;STORE #'0 IN R0 AND
	020752	004737	033066	JSR PC,PRINTC	;PRINT THE CHARACTER.
29	020756			1\$:	
	020756	010546		MOV R5,-(SP)	;PUSH R5 ON STACK
	020760	012746	014266	MOV #RNTIM1,-(SP)	;PUSH #RNTIM1 ON STACK
	020764	004137	033302	JSR R1,LPNT	;CALL LPNT PRINT ROUTINE
	020770	000004		.WORD ARG.CT	;ARGUMENT COUNT * 2
30	020772	012605		MOV (SP)+,R5	::POP STACK INTO R5
31	020774	020527	000011	CMP R5,#9.	;IF 9 OR LESS
32	021000	003004		BGT 2\$	
33	021002	112700	000060	MOVB #'0,R0	;STORE #'0 IN R0 AND
	021006	004737	033066	JSR PC,PRINTC	;PRINT THE CHARACTER.
34	021012			2\$:	
	021012	010546		MOV R5,-(SP)	;PUSH R5 ON STACK
	021014	012746	014274	MOV #RNTIM2,-(SP)	;PUSH #RNTIM2 ON STACK
	021020	004137	033302	JSR R1,LPNT	;CALL LPNT PRINT ROUTINE
	021024	000004		.WORD ARG.CT	;ARGUMENT COUNT * 2
35	021026	012605		MOV (SP)+,R5	::POP STACK INTO R5
	021030	012604		MOV (SP)+,R4	::POP STACK INTO R4
	021032	012603		MOV (SP)+,R3	::POP STACK INTO R3
	021034	012600		MOV (SP)+,R0	::POP STACK INTO R0
36	021036			RNTIMX:	
	021036	112700	000040	MOVB #' ,R0	;STORE #' IN R0 AND
	021042	004737	033066	JSR PC,PRINTC	;PRINT THE CHARACTER.

CZUDLAC 'BR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 56-1
RNTIME 'rint runtime

37 021046 000207

RTS PC

```

1      .SBTTL  UNITINI Initialize Unit
2      ;
3      ;UNITINI:
4      ;       Perform the following initialization steps for the UUT:
5      ;
6      ;       Load controller table values
7      ;       Call XDAINT to initialize controller
8      ;       Issue GET UNIT STATUS command
9      ;       Issue SET CONTROLLER CHAR command
10     ;       Put drive ONLINE
11     ;       Allocate and clear RCT error table
12     ;       Read RCT block 0 and format Volume serial number
13     ;       Scan RCT using MULTI-READ
14     ;       Call STEP3 to test for crash during replacement
15     ;
16     ;Inputs:
17     ;       HCOM      Host communication area
18     ;       CTLRTBL  Controller table
19     ;
20     ;Outputs:
21     ;       STATUS   Exit status
22     ;               0: Disk transfer errorlog packet (no msg)
23     ;               -1: Fatal errorlog packet (err msg)
24     ;
25     ;Calling Seq:
26     ;
27     ;       JSR      PC,UNITINI
28     ;       .WORD   <error exit>
29     ;
30
31     UNITINI:
32     021050 013705 002304      MOV      CTLRTBL,R5      ; Load Table address
33     021054 011537 002314      MOV      (R5),SADDR     ; Save controller address
34     021060 01654E 000002      MOV      C.VEC(R5),-(SP) ;:PUSH C.VEC(R5) ON STACK
35     021064 011604              MOV      (SP),R4        ;GET BR LEVEL/VECTOR OF xDA AND
36     021066 042704 177000      BIC      #+C<CT.VEC>,R4  ;SAVE THE VECTOR ADDRESS.
37     021072 012600              MOV      (SP)+,R0       ;:POP STACK INTO R0
38     021074 042700 170777      BIC      #+C<CT.BRL>,R0  ;SAVE THE BR LEVEL AND
39     021100 000300              SWAB    R0              ;PUT IT INTO LO BYTE.
40     021102 010501              MOV      R5,R1          ;GET ADDRESS OF CONTROLLER TABLE AND
41     021104 062701 000010      ADD      #C.JSR,R1      ;USE THE TABLE ADDRESS OF THE 'JSR'
42                                     ;INSTRUCTION IN THE CONTROLLER TABLE. AS
43                                     ;THE INTERRUPT SERVICE ROUTINE.
44     021110 010046              MOV      R0,-(SP)
45     021112 010146              MOV      R1,-(SP)
46     021114 010446              MOV      R4,-(SP)
47     021116 012746 000003      MOV      #3,-(SP)
48     021122 104437              TRAP    C#SVEC
49     021124 062706 000010      ADD      #10,SP
50     021130 006204              ASR     R4              ;DIVIDE VECTOR ADDRESS BY 4 TO
51     021132 006204              ASR     R4              ;USE IN xDA INITIALIZE SUBROUTINE.
52     021134 004737 036670      JSR     PC,XDAINT      ;RING LENGTH, INTERRUPTS AND VECTOR ADDR
53     021140 001002              BNE    100#           ;BRANCH IF NO ERROR, ELSE
54     021142 000137 022172      JMP     900#           ; Error exit

```

```

1          ; Get UNIT status
2 021146          ; 100$:
3 021146 004737 030626 JSR PC.INTCPK ; INITIALIZE THE COMMAND PACKET
4          ; Setup to issue OP.GUS command
   021152 012700 000003 MOV #OP.GUS,R0 ; Load MSCP opcode
   021156 004737 030204 JSR PC.ISSUE ; Issue command to the xDA
   021162 022172 900$ BEQ 120$ ; Here if fatal error.
   021164 001423 ; Branch if command successful.

5
6 021166 013700 002354 MOV EMSTAT,R0 ; Reload endmsg status
7 021172 042700 177740 BIC #+CST.MSK,R0 ; Clear any subcodes
8 021176 022700 000004 CMP #ST.AVL,R0 ; See if unit is available
9 021202 001414 BEQ 120$ ; Branch if YES
10 021204          ; Else, report fatal error
11 021204 012737 011346 002552 MOV #ERM201,BASEMSG ; "Drive not accessible"
12 021212 012737 000:00 002370 MOV #B.ESAT,EXTMSK ; Print status
13 021220 104455 TRAP C$ERDF
   021222 000311 .WORD 201
   021224 011314 .WORD HDR201
   021226 015220 .WORD ERRO01
14 021230 000137 022172 JMP 900$ ; Error exit
15 021234          ; 120$:
16 021234 013704 002324 MOV HCOM,R4 ; Load comm area address
17 021240 016437 000064 002420 MOV HC.MPK+P.TRKS(R4),TRKSIZ ; Get the track size
18 021246 016437 000074 002416 MOV HC.MPK+P.RCTS(R4),RCTSIZ ; Number of blocks in the RCT
19 021254 003014 BGT 130$ ; Continue if RCT size > 0
20
21 021256 012737 011743 002552 MOV #ERM204,BASEMSG ; "Zero RCT size..."
22 021264 012737 000000 002370 MOV #0,EXTMSK ; No extended info
23 021272 104455 TRAP C$ERDF
   021274 000314 .WORD 204
   021276 011314 .WORD HDR201
   021300 015220 .WORD ERRO01
24 021302 000137 022172 JMP 900$ ; Error exit
25 021306          ; 130$:
26 021306 116437 000076 002422 MOVB HC.MPK+P.RBNS(R4),RBNS ; Number of RBN'S
27 021314 116437 000077 002423 MOVB HC.MPK+P.RCTC(R4),COPIES ; Number of copies of RBN's.
28
29          ; Set controller characteristics
30 021322          ; 200$:
31 021322 012764 000000 000124 MOV #0,HC.CPK+P.HTMO(R4) ; Disable host access timeout
32 021330 012700 000000 MOV #0,R0 ; Assume no error log
33 021334 005737 002162 TST ENBERRL ; Test error log enabled
34 021340 001402 BEQ 220$ ; Branch if NO
35 021342 052700 000020 BIS #CF.THS,R0 ; Else, set error log enabled
36 021346          ; 220$:
37 021346 010064 000122 MOV R0,HC.CPK+P.CNTF(R4) ; Load characteristics
38          ; Setup to issue OP.SCC command
   021352 012700 000004 MOV #OP.SCC,R0 ; Load MSCP opcode
   021356 004737 030204 JSR PC.ISSUE ; Issue command to the xDA
   021362 022172 900$ BEQ 300$ ; Here if fatal error.
   021364 001412 ; Branch if command successful.
39          ; REPORT ERROR
40 021366 012737 011453 002552 MOV #ERM202,BASEMSG ; Failed disable host timeout..."
41 021374 012737 000100 002370 MOV #B.ESAT,EXTMSK ; Print status
42 021402 104456 TRAP C$ERHRD
   021404 000312 .WORD 202
    
```

021406 011314
021410 015220

.WORD HDR201
.WORD ERRO01

43

```

1
2
3 021412          ; Put the drive ONLINE
4          300$:
5          021412 012700 000011      MOV    #OP.ONL,R0      ; Setup to issue OP.ONL command
6          021416 004737 030204      JSR    PC,ISSUE        ; Load MSCP opcode
7          021422 022172              900$      ; Issue command to the xDA
8          021424 001423              BEQ    320$            ; Here if fatal error,
9          ; Branch if command successful.
10         5
11         6 021426 013700 002354      MOV    EMSTAT,R0      ; Reload emdmsg status
12         7 021432 042700 177740      BIC    #+CST.MSK,R0   ; Clear any subcodes
13         8 021436 022700 000400      CMP    #ST.AOL,R0     ; See if unit is already online
14         9 021442 001414              BEQ    320$            ; Branch if YES
15         ; Else, Report fatal error
16         10 021444 012737 011711 002552  MOV    #ERM203,BASEMSG ; "Drive not ONLINE"
17         11 021452 012737 000100 002370  MOV    #B.ESAT,EXTMSK ; Print status
18         12 021460 104455              TRAP   C$ERDF
19         13 021462 000313              .WORD 203
20         14 021464 011314              .WORD HDR201
21         15 021466 015220              .WORD ERRO01
22         16 021470 000137 022172      JMP    900$           ; Error exit
23         17 021474              320$:
24         18 021474 013704 002324      MOV    HCOM,R4        ; Load comm area address
25         19 021500 016437 000064 002430  MOV    HC.MPK+P.UNSZ(R4),UNSI ; Get LO order word and
26         20 021506 016437 000066 002432  MOV    HC.MPK+P.UNSZ+2(R4),UNSI ; HI order word of unit size
27         21 021514 116437 000052 003244  MOVB   HC.MPK+P.MODEL(R4),MODEL ; Save model type
28         22 021522 012700 000012      MOV    #MDLSIZ,R0     ; Load max model table index
29         23 021526              325$:
30         24 021526 123760 003244 003250  CMPB   MODEL,MDLTBL(R0) ; Compare with table contents
31         25 021534 001403              BEQ    330$            ; Branch if match
32         26 021536 162700 000002      SUB    #2,R0          ; Decrement index
33         27 021542 003371              BGT    325$            ; Continue until zero
34         28 021544              330$:
35         29 021544 010037 003246      MOV    R0,MDLNDX     ; Save model index
36         30
37         31          ; Allocate new RCT error table
38         32 021550              340$:
39         33 021550 013746 002176      MOV    FFREE,-(SP)    ;;PUSH FFREE ON STACK
40         34 021554 013746 002200      MOV    FSIZE,-(SP)   ;;PUSH FSIZE ON STACK
41         35 021560 013701 002416      MOV    RCTSIZ,R1     ; Load RCT size
42         36 021564 004737 040054      JSR    PC,ALOCM      ; Allocate error table
43         37 021570 010137 002312      MOV    R1,RCTTBL     ; Save table address
44         38 021574 012637 002200      MOV    (SP)+,FSIZE   ;;POP STACK INTO FSIZE
45         39 021600 012637 002176      MOV    (SP)+,FFREE   ;;POP STACK INTO FFREE
46         40
47         41          ; Clear RCT error table
48         42 021604 013700 002416      MOV    RCTSIZ,R0     ; Load count
49         43 021610              350$:
50         44 021610 005021              CLR    (R1)+         ; Clear location
51         45 021612 005300              DEC    R0            ; Decrement
52         46 021614 001375              BNE    350$          ; Continue until zero
53         47
54         48          ; Read RCT block 0 to obtain volume info
55         49 021616              360$:
56         50 021616 004737 030626      JSR    PC,INTCPK     ;INITIALIZE THE COMMAND PACKET

```



```

49 021622 012737 042472 002340      MOV    #DBUFF,UADR      ;GET BEGINNING ADDRESS OF DATA BUFFER
50 021630 012737 000000 002476      MOV    #0,RCTBLK       ;INIT THE RCT LOGICAL BLOCK TO 0 AND
51 021636 004737 030074              JSR    PC,INITRW       ;SOME READ/WRITE PARAMETERS.
52                                ; Setup to issue MULTRD command
    021642 004737 027224              JSR    PC,MULTRD       ; Issue command to the xDA
    021646 022172                      900$                 ; Here if fatal error,
53 021650              370$:          MOV    HCOM,R4         ; Load host comm area ptr
54 021650 013704 002324              MOV    CTLRTBL,R5     ; Load controller table ptr
55 021654 013705 002304              JSR    PC,GETVSN      ; Format volume serial number
56 021660 004737 032022
57
58 021664 016546 000006              MOV    C.DRVN(R5),-(SP) ;PUSH C.DRVN(R5) ON STACK
    021670 011546              MOV    (R5),-(SP)     ;PUSH (R5) ON STACK
    021672 012746 014225              MOV    #DEVNAM, -(SP) ;PUSH #DEVNAM ON STACK
    021676 013746 002074              MOV    L#LUN, -(SP)  ;PUSH L#LUN ON STACK
    021702 012746 005016              MOV    #INF110, -(SP) ;PUSH #INF110 ON STACK
    021706 004137 033254              JSR    R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    021712 000012              .WORD ARG.CT        ;ARGUMENT COUNT * 2
59
60 021714              385$:          MOV    MDLNDX,R0      ; Load model index
61 021714 013700 003246              MOV    UNSIZ+2, -(SP) ;PUSH UNSIZ+2 ON STACK
62 021720 013746 002432              MOV    UNSIZ, -(SP)  ;PUSH UNSIZ ON STACK
    021724 013746 002430              MOV    VSNPTR, -(SP) ;PUSH VSNPTR ON STACK
    021730 013746 002616              MOV    MDLTEXT(R0),-(SP) ;PUSH MDLTEXT(R0) ON STACK
    021734 016046 003262              MOV    #INF111, -(SP) ;PUSH #INF111 ON STACK
    021740 012746 005165              JSR    R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    021744 004137 033254              .WORD ARG.CT        ;ARGUMENT COUNT * 2
63

```

```

1          ;      Read all RCT blocks
2
3 021752          400$:
4 021752 013704 002324      MOV      HCOM,R4          ; Load host comm area address
5 021756 004737 030626      JSR      PC,INTCPK       ; Initialize command packet
6 021762 012737 042472 002340  MOV      @DBUFF,UADR     ; Load buffer address
7 021770 012737 000000 002476  MOV      @0,RCTBLK      ; Reset RCT logical block number
8 021776 012746 007002      MOV      @INF301,-(SP)   ; PUSH @INF301 ON STACK
   022002 004137 033254      JSR      R1,LPNTB       ; CALL LPNTB PRINT ROUTINE
   022006 000002      .WORD ARG.CT          ; ARGUMENT COUNT * 2
9 022010          420$:
10 022010 004737 030074      JSR      PC,INITRW      ; Initialize R/W parameters
11 022014 063764 002476 000140  ADD      RCTBLK,HC.CPK+P.LBN(R4) ; Add RCT logical block
12 022022 005564 000142      ADC      HC.CPK+P.LBN+2(R4) ; To start of RCT area
13          JSR      PC,MULTRD    ; Setup to issue MULTRD command
   022026 004737 027224      JSR      PC,MULTRD    ; Issue command to the xDA
   022032 022172      900$          ; Here if fatal error,
14 022034          430$:
15 022034 005237 002476      INC      RCTBLK         ; Increment to next block
16 022040 023737 002476 002416  CMP      RCTBLK,RCTSIZ  ; Test for all block sread
17 022046 002760      BLT      420$          ; Branch if NO
18

```

```

1
2 ; Call STEP3 to read RCT block 0 and process replacement
3 ; recovery if necessary.
4
5 022050 500$:
6 022050 004737 024260 JSR PC,STEP3 ; Call STEP3 to test for crash
7 022054 100446 BMI 900$ ; Branch if fatal error
8
9 022056 005737 002546 TST NXTSTEP ; Test for recovery
10 022062 001433 BEQ 550$ ; Branch if NO
11
12 022064 005737 002150 TST MANRPL ; Test for MANUAL mode
13 022070 001004 BNE 520$ ; Branch if YES
14
15 022072 005737 002152 TST AUTORC ; Test for automatic recovery
16 022076 001015 BNE 540$ ; Branch if yes
17 022100 000434 BR 900$ ; Else, error exit
18 022102 520$:
19 022102 012737 000001 002654 MOV #1,REPLY ; Else, set default reply
20 022110 104443 TRAP C$GMAN
    022112 000404 BR 10000$
    022114 002654 .WORD REPLY
    022116 000130 .WORD T$CODE
    022120 015031 .WORD TXT006
    022122 177777 .WORD -1
21 022124 10000$:
22 022130 005737 002654 TST REPLY ; Test replay
23 022132 001420 BEQ 900$ ; Branch if NO
24 022132 540$:
    022132 012746 010264 MOV #INF601,-(SP) ; PUSH #INF601 ON STACK
    022136 004137 033254 JSR R1,LPNTB ; CALL LPNTB PRINT ROUTINE
    022142 000002 .WORD ARG.CT ; ARGUMENT COUNT + 2
25 022144 004737 024156 JSR PC,REPLACE ; Process possible recovery
26 022150 022172 900$ ; Error exit
27 022152 550$:
28 022152 004737 023160 JSR PC,INIPARM ; Initialize program variables
29
30 ; Normal exit
31 022156 800$:
32 022156 062716 000002 ADD #2,(SP) ; Bump past error return
33 022162 012737 000000 002442 MOV #0,STATUS ; Load success
34 022170 000207 RTS PC ; and return
35
36 ; Error exit
37 022172 900$:
38 022172 017616 000000 MOV @(SP),(SP) ; Load error return
39 022176 012737 177777 002442 MOV #-1,STATUS ; Load failure
40 022204 000207 RTS PC ; and return
41
  
```

```

1      .SBTTL INISWP Initialize software parms
2      ;
3      ;INISWP:
4      ; Initialize the parameters below by prompting for
5      ; Operator input:
6      ;
7      ; OPHELP:      Display operator help?
8      ; MANRPL:      Automatic or Manual replacement?
9      ; AUTORC:      Automatic crash recovery?
10     ; DSPRPL:      Display replacements as they occur?
11     ; DSPRCT:      Display RCT replacement descriptors?
12     ; ENBRPL:      Enable Replacements?
13     ; ENBWFE:      Exhibit write with Forced Error flag?
14     ;
15
16     INISWP:
17     022206 012746 004603      MOV      #INF102, -(SP)      ;PUSH #INF102 ON STACK
18     022212 004137 033254      JSR      R1, LPNTB          ;CALL LPNTB PRINT ROUTINE
19     022216 000002              .WORD    ARG.CT            ;ARGUMENT COUNT * 2
20     022220 012701 022525      MOV      #SWMSG0, R1        ; Load Question address
21     022224 012702 002146      MOV      #OPHELP, R2        ; Load reply address
22     022230 012712 000000      MOV      #0, (R2)           ; Default is "NO"
23     022234 004737 023126      JSR      PC, GETLOG         ; Ask operator
24     022240 005737 002146      TST     OPHELP              ; Test response
25     022244 001402              BEQ     10$                 ; Branch if "NO"
26
27     5$:
28     022246 004737 032150      JSR      PC, DSPHELP        ; Display operator help
29
30     10$:
31     022252 012737 000000 002150  MOV      #0, MANRPL          ; Default is Auto replacemet
32     022260 012701 022553      MOV      #SWMSG1, R1        ; Load Question address
33     022264 012702 002654      MOV      #REPLY, R2         ; Load reply address
34     022270 012712 000101      MOV      #'A, (R2)          ; Default is "AUTOMATIC"
35     022274 004737 023070      JSR      PC, GETCHR         ; Ask operator
36     022300 122737 000115 002654  CMPB    #'M, REPLY          ; Test for "M"
37     022306 001004              BNE     20$                 ; Branch if NO
38     022310 012737 000001 002150  MOV      #1, MANRPL         ; Else, set Manual replacement
39     022316 000430              BR      50$                 ; and continue
40
41     20$:
42     022320 012701 022613      MOV      #SWMSG2, R1        ; Load Question address
43     022324 012702 002152      MOV      #AUTORC, R2        ; Load reply address
44     022330 012712 000001      MOV      #1, (R2)           ; Default is "YES"
45     022334 004737 023126      JSR      PC, GETLOG         ; Ask operator
46
47     30$:
48     022340 012701 022644      MOV      #SWMSG3, R1        ; Load Question address
49     022344 012702 002154      MOV      #DSPRPL, R2        ; Load reply address
50     022350 012712 000001      MOV      #1, (R2)           ; Default is "YES"
51     022354 004737 023126      JSR      PC, GETLOG         ; Ask operator
52
53     40$:
54     022360 012701 022707      MOV      #SWMSG4, R1        ; Load Question address
55     022364 012702 002156      MOV      #DSPRCT, R2        ; Load reply address
56     022370 012712 000000      MOV      #0, (R2)           ; Default is "NO"
57     022374 004737 023126      JSR      PC, GETLOG         ; Ask operator
58
59     50$:
60     022400 012701 022753      MOV      #SWMSG5, R1        ; Load Question address
61     022404 012702 002160      MOV      #ENBRPL, R2        ; Load reply address
62     022410 012712 000001      MOV      #1, (R2)           ; Default is "YES"
63     022414 004737 023126      JSR      PC, GETLOG         ; Ask operator

```

```

56 022420
57 022420 000410
58 022422 012701 022777
59 022426 012702 002162
60 022432 012712 000001
61 022436 004737 023126
62 022442
63 022442 012701 023024
64 022446 012702 002164
65 022452 012712 000001
66 022456 004737 023126
67 022462
68 022462 000207
69
70
71 022464 110 141 163 BACKMSG: .ASCIZ \Have you backed-up customer data\
72 022525 104 151 163 SWMSG0: .ASCIZ \Display operator Help\
73 022553 101 165 164 SWMSG1: .ASCIZ \Automatic or Manual replacement\
74 022613 101 165 164 SWMSG2: .ASCIZ \Automatic crash recovery\
75 022644 104 151 163 SWMSG3: .ASCIZ \Display replacements as they occur\
76 022707 104 151 163 SWMSG4: .ASCIZ \Display RCT replacement descriptors\
77 022753 105 156 141 SWMSG5: .ASCIZ \Enable replacements\
78 022777 105 156 141 SWMSG6: .ASCIZ \Enable error logging\
79 023024 105 156 141 SWMSG7: .ASCIZ \Enable write with Forced Error flag\
80 .LIST BEX
81 .EVEN
82

```

```

60$: BR 70$ ; Bypass this one
MOV #SWMSG6,R1 ; Load Question address
MOV #ENBERRL,R2 ; Load reply address
MOV #1,(R2) ; Default is "YES"
JSR PC,GETLOG ; Ask operator

70$: MOV #SWMSG7,R1 ; Load Question address
MOV #ENBWFE,R2 ; Load reply address
MOV #1,(R2) ; Default is "YES"
JSR PC,GETLOG ; Ask operator

100$: RTS PC ; Return to caller

```

CZUDLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 63
 GETCHR Get character from Operator

```

1      .SBTTL  GETCHR  Get character from Operator
2      ;
3      ;GETCHR:
4      ; Gets an alphanumeric character input from operator.
5      ;Inputs:
6      ;   R1      .ASCIZ Prompt msg address
7      ;   R2      Reply address
8      ;
9      ;Outputs:
10     ; Prints prompt; response is placed in reply address.
11     ;
12     ;Calling Seq:
13     ;
14     ;   MOV      @PROMPT,R1
15     ;   MOV      @REPLY,R2
16     ;   JSR      PC,GETCHR
17     ;
18
19
20 023070      GETCHR:
21 023070      010137  023110      MOV      R1,CHRMSG      ; Save message address
22 023074      010237  023104      MOV      R2,CHRRPLY     ; Save reply address
27 023100
28 023100      104443
   023102      000406
   023104      023104      .WORD    CHRRPLY
   023106      000152      .WORD    T$CODE
   023110      023110      .WORD    CHRMSG
   023112      177777      .WORD    -1
   023114      000001      .WORD    T$LOLIM
   023116      000001      .WORD    T$HILIM
   023120
29 023120      10001$:
30 023120      117700  177760      20$:      MOVB    @CHRRPLY,R0      ; Set return code
31 023124      000207      RTS      PC              ; and return
32
33

```

```

1      .SBTTL  GETLOG  Get Logical from Operator
2      ;
3      ;GETLOG:
4      ; Gets an logical (Y/N) reply from operator.
5      ;Inputs:
6      ;   R1      .ASCIZ Prompt msg address
7      ;   R2      Reply address
8      ;
9      ;Outputs:
10     ; Prints prompt; response is placed in reply address.
11     ;
12     ;Calling Seq:
13     ;
14     ;   MOV      @PROMPT,R1
15     ;   MOV      @REPLY,R2
16     ;   JSR      PC,GETLOG
17     ;
18
19 023126      GETLOG:
20 023126 010137 023146      MOV      R1,LOGMSG      ; Save message address
21 023132 010237 023142      MOV      R2,LOGRPLY     ; Save reply address
26 023136
27 023136 104443      10$:      TRAP      C$GMAN
27 023140 000404      BR        10002$
27 023142 023142      .WORD     LOGRPLY
27 023144 000130      .WORD     T$CODE
27 023146 023146      .WORD     LOGMSG
27 023150 177777      .WORD     -1
27 023152
28 023152      10002$:
29 023152 117700 177764      20$:      MOVB     @LOGRPLY,R0   ; Set return code
30 023156 000207      RTS      PC             ; and return
31
32

```

```

1      .SBTTL  INIPARM Initialize program parameters
2      ;
3      ;INIPARM:
4      ; Initialize program variables.
5      ;Inputs:
6      ; None.
7      ;Outputs:
8      ; Initialize program variables.
9      ;
10     ;Calling Seq:
11     ;
12     ; JSR    PC,INIPARM
13     ;
14
15     023160 INIPARM:
16     023160 012737 000000 002574      MOV    #0,ACCESS      ; Initialize
17     023166 012737 000000 002576      MOV    #0,ACCESS+2    ; ACCESS count
18     023174 012737 000000 002562      MOV    #0,GLBN        ; Initialize starting
19     023202 012737 000000 002564      MOV    #0,GLBN+2      ; group LBN
20     023210 012737 000000 002600      MOV    #0,MINLBN      ; Reset
21     023216 012737 000000 002602      MOV    #0,MINLBN+2    ; Minimum
22     023224 013737 002430 002604      MOV    UNSIZ,MAXLBN    ; Reset
23     023232 013737 002432 002606      MOV    UNSIZ+2,MAXLBN+2 ; Maximum
24
25     023240 012737 000000 002566      MOV    #0,GBCNT       ; Group byte count
26     023246 012737 000001 002570      MOV    #1,GBCNT+2     ; is 128. x 512.
27
28     023254 012737 000000 002220      MOV    #0,KW.EL       ; Clear
29     023262 012737 000000 002222      MOV    #0,KW.EL+2     ; Elapsed time
30     023270 012737 000000 002466      MOV    #0,RECOVR      ; Reset recovery flag
31     023276 012737 000000 002434      MOV    #0,PRIMARY     ; Zero PRIMARY RBN count
32     023304 012737 000000 002436      MOV    #0,SCNDRY     ; Zero SECONDARY RBN count
33     023312 012737 000000 002440      MOV    #0,UNUSE       ; Zero UNUSABLE RBN count
34
35     023320 012737 000000 002530      MOV    #0,FERCNT      ; Clear FORCED ERROR count
36     023326 012737 000000 002532      MOV    #0,HCECNT      ; Clear HEADER COMP ERROR count
37     023334 012737 000000 002534      MOV    #0,DSTCNT      ; Clear DATA SYNC TMO count
38     023342 012737 000000 002536      MOV    #0,ECCCNT      ; Clear ECC ERROR count
39     023350 012737 000000 002540      MOV    #0,BBRCNT      ; Clear BBR count
40     023356 012737 000000 002542      MOV    #0,WFE CNT     ; Clear WRITE FE count
41
42     023364 012737 000000 002516      MOV    #0,LBNSTAT     ; Clear LBN status
43     023372 012737 000000 002520      MOV    #0,RPLSTAT     ; Clear Replacement status
44     023400 012737 000000 002342      MOV    #0,CMDREF      ; Zero cmd ref number
45
46     023406 000207      RTS    PC              ; Return to caller
47

```



```

1      .SBTTL GETLBN Get LBN from operator
2      ;GETLBN
3      ;
4      ;Inputs:
5      ;   R1 =   Address of operator prompt
6      ;   R2 =   Address of receiving long word
7      ;
8      ;Outputs:
9      ;   Operator is prompted for LBN. Reply is checked
10     ;   against UNSIZ maximum.
11     ;   LBN is converted to a 28-bit value by CNV28.
12     ;
13     ;Calling seq:
14     ;
15     ;   MOV    #PROMPT,R1
16     ;   MOV    #LBN,R4
17     ;   JSR    PC,GETLBN
18     ;
19
20     GETLBN:
21     023410 010146  MOV    R1,-(SP)           ;;PUSH R1 ON STACK
22     023412 010246  MOV    R2,-(SP)           ;;PUSH R2 ON STACK
23     023414 010346  MOV    R3,-(SP)           ;;PUSH R3 ON STACK
24     023416 010446  MOV    R4,-(SP)           ;;PUSH R4 ON STACK
25     023420 010546  MOV    R5,-(SP)           ;;PUSH R5 ON STACK
26     023422 010137 023456  MOV    R1,MSGADR         ; Save message address
27
28     10$:
29     023426 012746 014231  MOV    #CRLF,-(SP)       ;PUSH #CRLF ON STACK
30     023432 004137 033254  JSR    R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
31     023436 000002          .WORD  ARG.CT            ;ARGUMENT COUNT * 2
32     023440 012737 000000 002666  MOV    #0,TTYIN         ; Flag default response
33
34     20$:
35     023446          TRAP    C#GMAN
36     023450          BR     10003$
37     023452          .WORD  TTYIN
38     023454          .WORD  T#CODE
39     023456          .WORD  MSGADR
40     023460          .WORD  177777
41     023462          .WORD  T#LOLIM
42     023464          .WORD  T#HILIM
43
44     10003$:
45     023466          TST    TTYIN           ; Test response
46     023472          BEQ    40$           ; Branch if null
47     023474          JSR    PC,CNV28      ;CONVERT ASCIZ TO DECIMAL
48     023500          BCS    10$           ;BRANCH IF ERROR
49     023502          MOV    UNSIZ,R0      ;STUFF IN LO ORDER JNIT SIZE AND
50     023506          MOV    UNSIZ+2,R1    ;HI ORDER UNIT SIZE TO GET
51     023512          SUB    #1,R0        ;RCT STARTING LBN.
52     023516          SBC    R1           ;DECREMENT TO GET LAST HOST LBN AND
53     023520          CMP    2(R4),R1     ;SUBTRACT CARRY BIT.
54     023524          BGT    30$           ;SEE IF HI ORDER LBN WAS EXCEEDED.
55     023526          BLT    35$           ;BRANCH IF SO, ELSE
56     023530          CMP    (R4),R0     ;WITHIN LIMITS.
57     023532          BLOS  35$           ;SEE IF LO ORDER LBN WAS EXCEEDED.
58     023534          ;BRANCH IF NOT, ELSE
59
60     30$:
61          .PRINT 'LIMITS LO= 0, HI= "D32".'
```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr 95 10:42 Page 66 1
 GETLBN Get LBN from operator

```

49 023534 010146      MOV      R1,(SP)          ;PUSH R1 ON STACK
   023536 010046      MOV      R0,-(SP)        ;PUSH R0 ON STACK
   023540 012746 014367  MOV      #OLIMIT,-(SP)   ;PUSH #OLIMIT ON STACK
   023544 004137 033244  JSR      R1,LPNTF        ;CALL LPNTF PRINT ROUTINE
   023550 000006      .WORD   ARG.CT          ;ARGUMENT COUNT * 2
50 023552 000725      BR       10$            ;TRY AGAIN
51 023554                    35$:
52 023554 012737 000001 002666  MOV      #1,TTYIN        ; Set SUCCESS
53
54 023562                    40$:
55 023562 012605      MOV      (SP)+,R5        ;;POP STACK INTO R5
   023564 012604      MOV      (SP)+,R4        ;;POP STACK INTO R4
   023566 012603      MOV      (SP)+,R3        ;;POP STACK INTO R3
   023570 012602      MOV      (SP)+,R2        ;;POP STACK INTO R2
   023572 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
56 023574 013737 002666 002442  MOV      TTYIN,STATUS    ; Set condition codes for caller
57 023602 000207      RTS      PC              ; Return to caller
58

```

```

1      .SBTTL READLBN Read suspect LBN
2      ;READLBN
3      ;
4      ;Reread suspect LBN 32 times to determine if really bad
5      ;and retrieve best data.
6      ;
7      ;Inputs:
8      ;   HCOM      Host communications area
9      ;
10     ;Outputs:
11     ;   LBNSTAT  Status of read:
12     ;
13     ;       TF.FER = FORCED ERROR
14     ;       TF.HCE = HEADER COMP ERROR
15     ;       TF.DST = DATA SYNC TMO
16     ;       TF.ECC = ECC ERROR
17     ;       TF.SEX = SERIOUS EXCEPTION
18     ;       TF.LOG = ERRORLOG SENT
19     ;       TF.BBU = BAD BLOCK UNREPORTED
20     ;       TF.BBR = BAD BLOCK REPORTED
21     ;       TF.BAD = DATA IS BAD
22     ;
23     ;       STATUS  0 or 1: success
24     ;                -1: Fatal error
25     ;
26     ;Calling seq:
27     ;
28     ;   JSR      PC,READLBN
29     ;   .WORD   <error exit>
30     ;
31
32 023604 READLBN:
33 023604 013704 002324      MOV      HCOM,R4      ; Load comm ptr
34 023610 012737 000040 002556  MOV      #32.,RDCNT  ; Initialize READ count
35 023616      100$:
36 023616 012737 000000 002346  MOV      #0,CMDMOD   ; Zero cmd modifiers
37 023624 012737 000000 002516  MOV      #0,LBNSTAT  ; Zero LBN status
38 023632      110$:
39 023632 004737 024772      JSR      PC,STEP4    ; Read LBN data
40 023636 100517      BMI      900$        ; Branch if fatal error
41 023640 153737 002356 002516  BISB    FLAGS,LBNSTAT ; Set end msg flags
42 023646 005737 002354      TST     EMSTAT       ; Test engmsg status
43 023652 001464      BEQ     200$        ; Branch if SUCCESS
44 023654      120$:
45 023654 013700 002354      MOV     EMSTAT,R0    ; Load endmsg status
46 023660 052737 100000 002516  BIS     #TF.BAD,LBNSTAT ; Set LBN data bad
47
48 023666 022700 000010      CMP     #ST.DAT,R0   ; Test for Forced Error
49 023672 001004      BNE     122$        ; Branch if NO
50 023674 052737 000001 002516  BIS     #TF.FER,LBNSTAT ; Else, set FER status
51 023702 000450      BR      200$        ; And write to RCT
52 023704      122$:
53 023704 022700 000110      CMP     #SB.HCE,R0   ; Test for Header Compare Error
54 023710 001004      BNE     124$        ; Branch if NO
55 023712 052737 000002 002516  BIS     #TF.HCE,LBNSTAT ; Else, set HCEstatus
56 023720 000415      BR      130$        ; And continue
57 023722      124$:

```

58	023722	022700	000150		CMP	#SB.DST,RO	; Test for Data Sync Timeout
59	023726	001004			BNE	126#	; Branch if NO
60	023730	052737	000004	002516	BIS	#TF.DST,LBNSTAT	; Else, set DST status
61	023736	000406			BR	130#	; And continue
62	023740			126#:			
63	023740	022700	000350		CMP	#SB.ECC,RO	; Test for Uncorrectable ECC
64	023744	001003			BNE	130#	; Branch if NO
65	023746	052737	000010	002516	BIS	#TF.ECC,LBNSTAT	; Else, set ECC status
66	023754			130#:			
67	023754	005337	002556		DEC	RDCNT	; Decrement count
68	023760	001316			BNE	100#	; and continue until zero
69	023762			150#:			
70	023762	033737	000010	002516	BIT	TF.ECC,LBNSTAT	; Test for uncorrectable ECC
71	023770	001403			BEQ	180#	; Branch if NO
72	023772	012737	000100	002346	MOV	#MD.SEC,CMDMOD	; Else, suppress ECC
73	024000			180#:			
74	024000	004737	024772		JSR	PC,STEP4	; Read LBN
75	024004	100434			BMI	900#	; Branch if fatal error
76	024006	005737	002160		TST	ENBRPL	; Test for replacement enabled
77	024012	001420			BEQ	300#	; BRanch if NO
78	024014	004737	025136		JSR	PC,STEP5	; Write LBN data
79	024020	100426			BMI	900#	; Branch if fatal error
80	024022	000414			BR	300#	; Exit
81	024024			200#:			
82	024024	005737	002160		TST	ENBRPL	; Test for replacement enabled
83	024030	001403			BEQ	210#	; BRanch if NO
84	024032	004737	025136		JSR	PC,STEP5	; Else, record data in RCT
85	024036	100417			BMI	900#	; Branch if fatal error
86	024040			210#:			
87	024040	005737	002516		TST	LBNSTAT	; Test status
88	024044	001003			BNE	300#	; Brnach if non-zero
89	024046			220#:			
90	024046	005337	002556		DEC	RDCNT	; Decrement count
91	024052	001261			BNE	100#	; continue if not zero
92	024054			300#:			
93	024054	062716	000002		ADD	#2,(SP)	; Bump past error return
94	024060	013737	002516	002442	MOV	LBNSTAT,STATUS	; Set cond codes
95	024066	042737	100000	002442	BIC	#TF.BAD,STATUS	; Clear MSB
96	024074	000207			RTS	PC	; Return to caller
97	024076			900#:			
98	024076	017616	000000		MOV	#(SP),(SP)	; Load error return
99	024102	012737	177777	002442	MOV	#-1,STATUS	; Load fatal error
100	024110	000207			RTS	PC	; Return to caller
101							
103							

```

1      .SBTTL REPLACE Replacement routine
2      ;REPLACE
3      ;
4      ;This is the global bad block replacement routine, used by all tests.
5      ;It performs all steps used in the modified replacement algorithm
6      ;designed for this diagnostic. The routine will only execute one step
7      ;on each call, with the next step in the algorithm placed in NXTSTEP,
8      ;based upon program logic.
9      ;
10     ;Entries:
11     ;   REPLACE - Step to execute in NXTSTEP
12     ;   STEPx  - Step to execute in STEP
13     ;Exits:
14     ;   PC+2   - Fatal error
15     ;
16     ;   NXTSTEP = x: Continue with STEPx
17     ;             = 0: Replacement complete
18     ;             = -1: Replacement step error
19     ;
20     ;Steps:
21     ;   REPLACE - General purpose entry.
22     ;
23     ;   STEP3  - Read RCT 0 and test for recovery in progress.
24     ;   STEP4  - Read LBN selected for replacement.
25     ;   STEP5  - Write LBN image to RCT block 1.
26     ;   STEP6  - Update RCT block 0 to indicate PHASE 1.
27     ;   STEP9  - Search for available RBN for replacement.
28     ;   STEP10 - Update RCT block 0 to indicate PHASE 2.
29     ;   STEP11 - Update RCT replacement descriptor.
30     ;   STEP12 - Issue REPLACE command and write LBN data.
31     ;   STEP13 - Update RCT block 0 to indicate replacement complete.
32     ;   STEP15 - Update RCT descriptor to indicate usage.
33     ;   STEP16 - Write LBN image to replaced LBN.
34     ;   STEP17 - Update RCT block 0 to indicate replacement complete.
35     ;
36     ;
37
38     024112
39     024112 024250
40     024114 024250
41     024116 024250
42     024120 024260
43     024122 024772
44     024124 025136
45     024126 025204
46     024130 024250
47     024132 024250
48     024134 025350
49     024136 025460
50     024140 025610
51     024142 026132
52     024144 026530
53     024146 024250
54     024150 026640
55     024152 026760
56     024154 027132
57

```

STOPTBL:	.WORD	STEPFTL	; Step 0 entry
	.WORD	STEPFTL	; Step 1 entry
	.WORD	STEPFTL	; Step 2 entry
	.WORD	STEP3	; Step 3 entry
	.WORD	STEP4	; Step 4 entry
	.WORD	STEP5	; Step 5 entry
	.WORD	STEP6	; Step 6 entry
	.WORD	STEPFTL	; Step 7 entry
	.WORD	STEPFTL	; Step 8 entry
	.WORD	STEP9	; Step 9 entry
	.WORD	STEP10	; Step 10 entry
	.WORD	STEP11	; Step 11 entry
	.WORD	STEP12	; Step 12 entry
	.WORD	STEP13	; Step 13 entry
	.WORD	STEPFTL	; Step 14 entry
	.WORD	STEP15	; Step 15 entry
	.WORD	STEP16	; Step 16 entry
	.WORD	STEP17	; Step 17 entry

CZUDLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 68-1
 REPLACE Replacement routine

```

58          000044          STEPSIZ =          .-STEPTBL          ; Number of table entries
59
60 024156          REPLACE:
61
62 024156          10$:
63 024156 013700 002546          MOV          NXTSTEP,R0          ; Load next STEP
64 024162 010037 002544          MOV          R0,STEP          ; Save as current STEP
65 024166 006300          ASL          R0          ; STEP . 2 is offset
66 024170 003416          BLE          STEPERR          ; Error if STEP <= 0
67 024172 022700 000044          CMP          #STEPSIZ,R0          ; Compare to table max
68 024176 003413          BLE          STEPERR          ; Error if STEP >= STEPSIZ
69
70 024200 004770 024112          JSR          PC,@STEPTBL(R0)          ; Else, JSR to STEP entry
71 024204 005737 002546          TST          NXTSTEP          ; Test return code
72 024210 003362          BGT          10$          ; Branch if next STEP > zero
73 024212 100405          BMI          STEPERR          ; Else, error if STEP < zero
74
75          ; Normal return
76 024214          20$:
77 024214 062716 000002          ADD          #2,(SP)          ; Bump past error return
78 024220 013700 002546          MOV          NXTSTEP,R0          ; Set cond codes for caller
79 024224 000207          RTS          PC          ; and return
80
81          ; Fatal error return for replacement STEPx.
82 024226          STEPERR:
83 024226 012737 177777 002546          MOV          #-1,NXTSTEP          ; Load fatal status
84 024234 017616 000000          MOV          @2(SP),(SP)          ; Load error return
85 024240 013737 002546 002442          MOV          NXTSTEP,STATUS          ; Set cond codes
86 024246 000207          RTS          PC          ; Return to caller
87
88          ; Illegal STEP
89 024250          STEPFTL:
90 024250 012737 177777 002546          MOV          #-1,NXTSTEP          ; Load fatal error
91 024256 000207          RTS          PC          ; and return
92

```

```

1      .SBTTL  STEP3  Replacement step 3
2      ;STEP3
3      ;
4      ;Read RCT block 0.  Test for PHASE 1 or PHASE 2 replacement in progress.
5      ;If P1 set, control transferred to STEP9.  If P2 set, control transferred
6      ;to STEP11.
7      ;
8      ;Inputs:
9      ;      R4 - Address of host comm area
10     ;
11     ;Outputs:
12     ;      VIBUFF - Set if LBN data is valid
13     ;      TBUFF - Image of RCT block 0
14     ;
15     ;Externals:
16     ;      INTCPK, INITRW, MULTRD, COPY
17     ;
18     ;Exits:
19     ;      EXIT3 - DEFAULT next STEP is 4
20     ;              PHASE 1 recover STEP is 9
21     ;              PHASE 2 recover STEP is 11.
22     ;      STEPERR - Fatal error exit
23     ;
24     ;
25     STEP3:
26     024260 013704 002324      MOV      HCOM,R4      ; Load host comm area address
27     024264 004737 030626      JSR      PC,INTCPK   ; INITIALIZE THE COMMAND PACKET
28     024270 012737 042472 002340  MOV      #DBUFF,UADR ; GET BEGINNING ADDRESS OF DATA BUFFER
29     024276 012737 000000 002476  MOV      #0,RCTBLK  ; INIT THE RCT LOGICAL BLOCK TO 0 AND
30     024304 004737 030074      JSR      PC,INITRW   ; SOME READ/WRITE PARAMETERS.
31     ; Setup to issue MULTRD command
32     024310 004737 027224      JSR      PC,MULTRD  ; Issue command to the xDA
33     024314 024250      STEPFTL ; Here if fatal error.
34     1$:
35     024316 004537 034276      JSR      R5,COPY    ; COPY RCT LOGICAL BLOCK 0 DATA FROM
36     024322 042472      .WORD   DBUFF      ; DATA BUFFER TO
37     024324 043472      .WORD   TBUFF     ; TEMPORARY BUFFER FOR LATER USE.
38     024326 013701 002340      MOV      UADR,R1   ; GET THE BEGINNING OF UNIBUS ADDRESS

```

```

1
2
3
4 024332 005037 002466 CLR RECOVR ; Assume no recovery needed
5 024336 032761 100000 000010 BIT #P1,RFLAGS(R1) ; Test for Phase 1 crash
6 024344 001404 BEQ 25$ ; Branch if NO
7 024346 012737 000001 002466 MOV #1,RECOVR ; Else, set for PHASE 1
8 024354 000415 BR 3$ ; and continue
9 024356
10 024356 032761 040000 000010 25$: BIT #P2,RFLAGS(R1) ; Test for Phase 2 crash
11 024364 001404 BEQ 30$ ; Branch if NO
12 024366 012737 000002 002466 MOV #2,RECOVR ; Else, set for PHASE 2
13 024374 000405 BR 3$ ; and continue
14 024376
15 024376 012737 000000 002546 30$: MOV #0,NXTSTEP ; Set done
16 024404 000137 024770 JMP EXIT3 ; and exit
17 024410
18 024410 012737 000000 002516 3$: MOV #0,LBNSTAT ; Zero LBN status
19 024416 112737 000001 002472 MOVB #1,VIBUFF ; ASSUME IMAGE DATA WAS VALID.
20 024424 016137 000014 002446 MOV LBNLO(R1),LBN ; RESTORE LO WORD OF LBN AND
21 024432 016137 000016 002450 MOV LBNLO+2(R1),LBN+2 ; HI WORD OF LBN BEIN; REPLACED.
22 024440 032761 000200 000010 BIT #FEF,RFLAGS(R1) ; SEE IF FORCED ERROR FLAG IS SET.
23 024446 001405 BEQ 5$ ; BRANCH IF NOT, ELSE
24 024450 105037 002472 CLRB VIBUFF ; INDICATE IMAGE DATA IS NOT VALID
25 024454 052737 100001 002516 BIS #TF.BAD!TF.FER,LBNSTAT ; Set forced error indicator
26 024462
27 024462 016137 000020 002452 5$: MOV RBNLO(R1),RBN ; RESTORE LO WORD OF RBN AND
28 024470 016137 000022 002454 MOV RBNLO+2(R1),RBN+2 ; HI WORD OF THE RBN REPLACEMENT.
29 024476
30 024476 012737 000004 002370 50$: MOV #B.PHASE,EXTMSK ; Print phase
31 024504 012737 013402 002552 MOV #ERM605,BASEMSG ; "CRASH occurred during replacement..."
32 024512 104457 TRAP C$ERSOFT
   024514 001135 .WORD 605
   024516 013173 .WORD HDR601
   024520 015720 .WORD ERR601
33

```



```

1
2 024522          55$:
3 024522 012737 000000 002520      MOV    #0,RPLSTAT      ; Zero replacement status
4 024530 105037 002514          CLR    MATFLG          ; ASSUME NO MATCH FOR LBN
5 024534 032761 020000 000010      BIT    #BRF,RFLAGS(R1) ; SEE IF BAD RBN FLAG IS SET,
6 024542 001450          BEQ    7$              ; BRANCH IF NOT, ELSE
7 024544 112737 000001 002514      MOV    #1,MATFLG      ; SET THE MATCHED LBN FLAG.
8 024552 016137 000024 002462      MOV    BADRBN(R1),MATRBN ; RESTORE MATCHED RBN LO ORDER WORD AND
9 024560 016137 000026 002464      MOV    BADRBN+2(R1),MATRBN+2 ; RBN HI ORDER WORD.
10
11 024566 105037 002515          CLR    SFLAG          ; ASSUME PRIMARY EMPTY SEARCH STATUS
12 024572 010446          MOV    R4,-(SP)        ; PUSH R4 ON STACK
    024574 010546          MOV    R5,-(SP)        ; PUSH R5 ON STACK
13 024576 013703 002446          MOV    LBN,R3         ; GET LO WORD AND
14 024602 013704 002450          MOV    LBN+2,R4       ; HI WORD OF LBN BEING REPLACED.
15 024606 013700 002420          MOV    TRKSIZ,R0      ; GET THE TRACK SIZE,
16 024612 004737 034436          JSR    PC,DIVIDE      ; DIVIDE TO FIND THE TRACK
    17                          ; OF LBN BEING REPLACED
18 024616 012605          MOV    (SP)+,R5       ; POP STACK INTO R5
    024620 012604          MOV    (SP)+,R4       ; POP STACK INTO R4
19 024622 113700 002422          MOV    RBNS,R0        ; GET NUMBER OF RBNS PER TRACK,
20 024626 010046          MOV    R0,-(SP)      ; PUT THE MULTIPLIER ON THE STACK
21 024630 010346          MOV    R3,-(SP)      ; PUT THE MULTIPLICAND ON THE STACK
22 024632 004737 034542          JSR    PC,MULT        ; CALL THE MULTIPLY ROUTINE
23 024636 022637 002452          CMP    (SP)+,RBN      ; SEE IF LO ORDER RBN'S ARE SAME,
24 024642 001402          BEQ    59$           ; Branch if yes
25 024644 005726          TST   (SP)+          ; Else, adjust stack
26 024646 000403          BR    6$             ; And continue
27 024650          59$:
28 024650 022637 002454          CMP    (SP)+,RBN+2    ; SEE IF HI ORDER RBN'S ARE SAME,
29 024654 001403          BEQ    7$             ; BRANCH IF SO, ELSE
30 024656          6$:
31 024656 112737 000001 002515      MOV    #1,SFLAG      ; SET SECONDARY EMPTY SEARCH STATUS.

```

```

1
2 024664          7$:
3 024664 012737 044472 002340      MOV    #IBUFF,UADR      ;GET BEGINNING ADDRESS OF IMAGE BUFFER
4 024672 012737 000001 002476      MOV    #1,RCTBLK      ;INIT THE RCT LOGICAL BLOCK TO 1 AND
5 024700 004737 030074              JSR    PC,INITRW      ;SOME READ/WRITE PARAMETERS.
6 024704 063764 002476 000140      ADD    RCTBLK,HC.CPK+P.LBN(R4) ;ADD RCT LOGICAL BLOCK TO STARTING
7 024712 005564 000142              ADC    HC.CPK+P.LBN+2(R4) ;HOST LBN OF THE RCT AREA.
8
   024716 004737 027224              JSR    PC,MULTRD      ; Setup to issue MULTRD command
   024722 024250                      STEPFTL              ; Issue command to the xDA
   024724 001405                      BEQ    9$            ; Here if fatal error,
9 024726          8$:
10 024726 105037 002472              CLRB   VIBUFF        ;INDICATE IMAGE DATA IS NOT VALID
11 024732 052737 100000 002516      BIS    #TF.BAD,LBNSTAT ; Set BAD LBN status
12 024740          9$:
13 024740 032761 100000 000010      BIT    #P1,RFLAGS(R1) ;SEE IF CRASH OCCURED IN PHASE 1.
14 024746 001405                      BEQ    10$           ; BRANCH IF NOT, ELSE
15 024750 012737 000011 002546      MOV    #9.,NXTSTEP   ; Next STEP for PHASE 1 recovery is 9
16 024756 000137 024770              JMP    EXIT3         ; and exit
17 024762          10$:
18 024762 012737 000013 002546      MOV    #11.,NXTSTEP  ; Next STEP is 11.
19 024770          EXIT3:
20 024770 000207              RTS    PC            ; Return to caller
21

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr-85 10:42 Page 73
 STEP4 Replacement step 4

```

1      .SBTTL STEP4 Replacement step 4
2      ;STEP4
3      ;
4      ;Read LBN to be replaced.
5      ;
6      ;Inputs:
7      ;   R4 - Address of host comm area
8      ;   LBN - Address of LBN to read
9      ;
10     ;Outputs:
11     ;   IBUFF - Contains LBN image read
12     ;   VIBUFF - Set if LBN data is valid
13     ;
14     ;Externals:
15     ;   INITRW, CLRBUF, ISSUE
16     ;
17     ;Exits:
18     ;   LBNERR - LBN error exit
19     ;   STEPERR - Fatal error exit
20     ;
21
22     STEP4:
23     024772      MOV      HCOM,R4          ; Load host comm area address
24     024776      MOVVB   #1,VIBUFF      ; Assume image data is valid
25     025004      MOV      #IBUFF,UADR   ; GET BEGINNING ADDRESS OF IMAGE BUFFER
26     025012      JSR      PC,INITRW     ; INIT SOME READ/WRITE PARAMETERS AND
27     025016      MOV      LBN,HC.CPK+P.LBN(R4) ; GET THE REPLACED LBN ADDRESS.
28     025024      MOV      LBN+2,HC.CPK+P.LBN+2(R4)
29     025032      JSR      PC,CLRBUF     ; CLEAR THE BUFFER BEFORE READING
30                                     ; Setup to issue OP.RD command
31     025036      MOV      #OP.RD,RO     ; Load MSCP opcode
32     025042      JSR      PC,ISSUE      ; Issue command to the xDA
33     025046      STEPFTL ; Here if fatal error,
34     025050      BEQ      EXIT4        ; Branch if command successful.
35
36     20$:
37     025052      MOVVB   #0,VIBUFF      ; Set image data not valid
38     025056      MOV      EMSTAT,RO    ; Reload endmsg status
39     025060      BIC      #+CST.MSK,RO ; Clear any subcodes
40     025064      CMP      #ST.DAT,RO   ; Test for Data error
41     025070      BEQ      EXIT4        ; Branch if YES
42
43     025076      MOV      #ERM301,BASEMSG ; "Failed READ..."
44     025104      MOV      #B.MSCP,EXTMSK ; Flag all MSCP fields
45     025112      TRAP    C+ERHRD
46     025114      .WORD   301
47     025116      .WORD   HDR301
48     025120      .WORD   ERR301
49     025122      JMP      STEPFTL      ; Fatal error exit
50
51     EXIT4:
52     025126      MOV      #5,NXTSTEP    ; Next STEP is 5.
53     025134      RTS      PC           ; Return to caller
54
55

```

```

1      .SBTTL  STEPS  Replacement step 5
2      ;STEPS
3      ;
4      ;Write LBN image to RCT block 1.
5      ;
6      ;Inputs:
7      ;      R4 - Address of host comm area
8      ;      UADR - Contains address of LBN data to write
9      ;
10     ;Outputs:
11     ;      LBN data written to RCT block 1
12     ;
13     ;Externals:
14     ;      INITRW, MULTWR
15     ;
16     ;Exits:
17     ;      EXIT4 - Successful exit; NXTSTEP is 5.
18     ;      STEPERR - Fatal Error exit
19     ;
20
21     STEPS:
22     025136 013704 002324      MOV      HCOM,R4      ; Load host comm area address
23     025142 012737 000001 002476  MOV      #1,RCTBLK   ; INIT THE RCT LOGICAL BLOCK TO 1 AND
24     025150 004737 030074      JSR      PC,INITRW   ; SOME READ/WRITE PARAMETERS.
25     025154 063764 002476 000140  ADD      RCTBLK,HC.CPK+P.LBN(R4) ; ADD RCT LOGICAL BLOCK TO STARTING
26     025162 005564 000142      ADC      HC.CPK+P.LBN+2(R4) ; HOST LBN OF THE RCT AREA.
27
28     025166 004737 027514      JSR      PC,MULTWR   ; Setup to issue MULTWR command
29     025172 024250      STEPFTL           ; Issue command to the xDA
30
31     EXITS:
32     025174 012737 000006 002546  MOV      #6,NXTSTEP  ; Next step is 6.
33     025202 000207      RTS      PC          ; Return

```

```

1      .SBTTL STEP6 Replacement step 6
2      ;STEP6
3      ;
4      ;Read RCT block 0 into working buffer. Update flags to indicate PHASE 1
5      ;replacement in progress. Rewrite RCT block 0. If MULTIWRITE fails,
6      ;control is transferred to STEP17.
7      ;
8      ;Inputs:
9      ;   R4 - Address of host comm area
10     ;   VIBUFF - Set if LBN data is valid
11     ;
12     ;Outputs:
13     ;   TBUFF - Contains RCT block 0 image
14     ;
15     ;Externals:
16     ;   INITRW, MULTRD, MULTWR, COPY
17     ;
18     ;Exits:
19     ;   RCT MULTIREAD error exit, next STEP is 17
20     ;   STEPERR - Fatal error exit
21     ;
22     ;
23     STEP6:
24     025204      013704      002324      MOV      HCOM,R4      ; Load host comm area address
25     025210      012737      042472      002340      MOV      @DBUFF,UADR  ;GET BEGINNING ADDRESS OF DATA BUFFER
26     025216      012737      000000      002476      MOV      @0,RCTBLK   ;INIT THE RCT LOGICAL BLOCK TO 0 AND
27     025224      004737      030074      JSR      PC,INITRW   ;SOME READ/WRITE PARAMETERS.
28     ;
29     025230      004737      027224      JSR      PC,MULTRD   ; Setup to issue MULTRD command
30     025234      024250      STEPFTL   ; Issue command to the xDA
31     ; Here if fatal error.
32     1$:
33     025236      004537      034276      JSR      R5,COPY     ;COPY RCT LOGICAL BLOCK 0 DATA FROM
34     025242      042472      .WORD    DBUFF       ;DATA BUFFER TO
35     025244      043472      .WORD    TBUFF       ;TEMPORARY BUFFER FOR LATER USE.
36     ;
37     025246      013701      002340      MOV      UADR,R1     ;GET THE BEGINNING OF UNIBUS ADDRESS
38     025252      013761      002446      000014      MOV      LBN,LBNLO(R1) ;STORE LO WORD OF LBN AND
39     025260      013761      002450      000016      MOV      LBN+2,LBNLO+2(R1) ;HI WORD OF LBN BEING REPLACED.
40     025266      012761      100000      000010      MOV      @P1,RFLAGS(R1) ;SET P1 FLAG IN REPLACEMENT FLAG WORD
41     025274      005737      002516      TST      LBNSTAT     ; Test for Valid LBN DATA
42     025300      100003      BPL      3$          ;BRANCH IF SO, ELSE
43     025302      052761      000200      000010      BIS      @FEF,RFLAGS(R1) ;SET FORCED ERROR FLAG.
44     ;
45     025310      012737      000000      002476      3$:
46     025316      004737      030074      MOV      @0,RCTBLK   ;INIT THE RCT LOGICAL BLOCK TO 0 AND
47     ;
48     025322      004737      027514      JSR      PC,INITRW   ;SOME READ/WRITE PARAMETERS.
49     025326      024250      STEPFTL   ; Setup to issue MULTWR command
50     ; Issue command to the xDA
51     ; Here if fatal error.
52     4$:
53     025330      004537      034276      JSR      R5,COPY     ;COPY RCT LOGICAL BLOCK 0 DATA FROM
54     025334      042472      .WORD    DBUFF       ;DATA BUFFER TO
55     025336      043472      .WORD    TBUFF       ;TEMPORARY BUFFER FOR LATER USE.
56     ;
57     025340      012737      000011      002546      EXIT6:
58     025346      000207      MOV      @9.,NXTSTEP ; Load next STEP is 9.
59     RTS      PC      ; Return

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr 85 10:42 Page 76
 STEP9 Replacement step 9

```

1      .SBTTL  STEP9  Replacement step 9
2      ;STEP9
3      ;
4      ;Search for available RBN for replacement.
5      ;
6      ;Inputs:
7      ;      R4 - Address of host comm area
8      ;      DBUFF - Scratch buffer
9      ;
10     ;Outputs:
11     ;      RBN - Replacement RBN
12     ;      SFLAG - Search status
13     ;      MATRBN - Match RBN
14     ;      MATFLG - Set if LBNs matched
15     ;
16     ;Externals:
17     ;      SEARCH
18     ;
19     ;Exits:
20     ;      STEP16 - Search error exit
21     ;      STEPERR - Fatal error exit
22     ;
23
24 025350 STEP9:
25 025350 013704 002324      MOV      HCOM,R4      ; Load host comm area address
26 025354 004737 030626      JSR      PC,INTCPK    ; INITIALIZE THE COMMAND PACKET
27 025360 012737 042472 002340  MOV      @DBUFF,UADR  ; GET BEGINNING ADDRESS OF DATA BUFFER
28 025366 004737 031400      JSR      PC,SEARCH    ; CALL SEARCH ROUTINE
29 025372 100004              BPL      1$           ; Branch if unsuccessful
30 025374 012737 177777 002546  MOV      @-1,NXTSTEP  ; Set fatal error
31 025402 000207              RTS      PC           ; and return
32 025404
33 025404 123727 002515 000002  1$:      CMPB     SFLAG,@2    ; SEE IF RCT TABLE WAS FULL.
34 025412 001016              BNE      EXIT9       ; BRANCH IF NOT, ELSE
35 025414
36 025414 012737 013472 002552  10$:    MOV      @ERM606,BASEMSG ; "Full RCT table detected..."
37 025422 012737 000000 002370  MOV      @0,EXTMSK    ; Zero error mask
38 025430 104456              TRAP     C$ERHRD
39 025432 001136              .WORD   606
40 025434 013173              .WORD   HDR601
41 025436 015720              .WORD   ERR601
42 025440 012737 000020 002546  MOV      @16.,NXTSTEP ; Next STEP is 16.
43 025446 000207              RTS      PC           ; Return
44 025450
45 025450 012737 000012 002546  EXIT9:  MOV      @10.,NXTSTEP ; Next STEP is 10.
46 025456 000207              RTS      PC           ; Return

```

STEP10 Replacement step 10

```

1      .SBTTL STEP10 Replacement step 10
2      ;STEP10
3      ;
4      ;Update RCT block 0 from working buffer to indicate PHASE 2 of
5      ;replacement.
6      ;
7      ;Inputs:
8      ;      R4 - Address of host comm area
9      ;      TBUFF - Working copy of RCT block 0
10     ;
11     ;      RBN - Replacement RBN
12     ;      SFLAG - Search status
13     ;      MATRBN - Match RBN
14     ;      MATFLG - Set if LBNs matched
15     ;
16     ;Outputs:
17     ;      RCT block 0 RBN - Replacement RBN
18     ;      RCT block 0 BADRBN - Bad RBN (if any)
19     ;
20     ;Externals:
21     ;      INITRW, MULTWR, COPY
22     ;
23     ;Exits:
24     ;      STEP11 - Successful exit
25     ;      STEP16 - RCT MULTIWRITE error exit
26     ;      STEPERR - Fatal error exit
27     ;
28
29     STEP10:
30     025460      004537      034276      JSR      R5,COPY      ;COPY RCT LOGICAL BLOCK 0 DATA FROM
31     025464      043472              .WORD    TBUFF      ;TEMPORARY BUFFER TO
32     025466      042472              .WORD    DBUFF      ;DATA BUFFER.
33
34     025470      013704      002324      MOV      HCOM,R4      ; Load host comm area address
35     025474      013701      002340      MOV      UADR,R1      ;GET THE BEGINNING OF UNIBUS ADDRESS
36     025500      013761      002452      000020    MOV      RBN,RBNLO(R1) ;STORE LO WORD OF RBN AND
37     025506      013761      002454      000022    MOV      RBN+2,RBNLO+2(R1) ;HI WORD OF THE RBN REPLACEMENT.
38     025514      052761      040000      000010    BIS      #P2,RFLAGS(R1) ;SET PHASE 2 FLAG IN REPLACEMENT WORD
39     025522      105737      002514      TSTB    MATFLG      ;SEE IF LBN'S MATCHED DURING SEARCH.
40     025526      001411              BEQ      1$          ;BRANCH IF NOT, ELSE
41     025530      052761      020000      000010    BIS      #BRF,RFLAGS(R1) ;SET BAD RBN FLAG IN REPLACEMENT WORD.
42     025536      013761      002462      000024    MOV      MATRBN,BADRBN(R1) ;STORE LO WORD OF RBN AND
43     025544      013761      002464      000026    MOV      MATRBN+2,BADRBN+2(R1) ;HI WORD OF THE BAD RBN.
44     025552
45     025552      042761      100000      000010    BIC      #P1,RFLAGS(R1) ;CLEAR PHASE 1 FLAG IN REPLACEMENT WORD.
46     025560      012737      000000      002476    MOV      #0,RCTBLK    ;INIT THE RCT LOGICAL BLOCK TO 0 AND
47     025566      004737      030074      JSR      PC,INITRW    ;SOME READ/WRITE PARAMETERS.
48
49     025572      004737      027514      JSR      PC,MULTWR    ; Setup to issue MULTWR command
50     025576      024250              STEPFTL  ; Issue command to the xDA
51     025600              EXIT10:  ; Here if fatal error.
52     025600      012737      000013      002546    MOV      #11.,NXTSTEP ; Next STEP is 11.
53     025606      000207              RTS      PC          ; Return

```

STEP11 Replacement step 11

```

1      .SRTTL STEP11 Replacement step 11
2      ;STEP11
3      ;
4      ;Compute RCT block number and offset for descriptor.
5      ;Read RCT, update descriptor for new RBN and rewrite.
6      ;Update RCT descriptor for bad RBN, if appropriate.
7      ;
8      ;Inputs:
9      ;      R4   Address of host comm area
10     ;      SFLAG - search status
11     ;      DBUFF - Scratch buffer
12     ;
13     ;Outputs:
14     ;
15     ;Externals:
16     ;      CALBLK, INITRW, MULTRD, MULTWR, COPY
17     ;
18     ;Exits:
19     ;      STEP12 - Successful exit
20     ;      STEP16 - RCT MULTIREAD error exit
21     ;      STEP15 - RCT MULTIWRITE error exit
22     ;      STEPERR - Fatal error exit
23     ;
24
25 025610 STEP11:
26 025610 013704 002324      MOV    HCOM,R4           ; Load host comm area address
27 025614 105037 002473      CLRB  VBUFF1           ; ASSUME BUFFER 1 DATA AND
28 025620 105037 002474      CLRB  VBUFF2           ; BUFFER 2 DATA ARE NOT VALID.
29 025624 013746 002454      MOV    RBN+2,-(SP)      ; ;PUSH RBN+2 ON STACK
30 025630 013746 002452      MOV    RBN,-(SP)       ; ;PUSH RBN ON STACK
31 025634 004737 031332      JSR    PC,CALBLK       ; CALCULATE THE RCT LOGICAL BLOCK AND
32                                ; THE RBN OFFSET WITHIN THE BLOCK.
33 025640 011600              MOV    (SP),R0          ; GET OFFSET WITHIN RCT LOGICAL BLOCK
34 025642 012637 002502      MOV    (SP)+,RCTOFF    ; ;POP STACK INTO RCTOFF
35 025646 062716 000002      ADD    #2,(SP)         ; OFFSET TO THE BEGINNING OF THE RCT
36                                ; DESCRIPTOR BLOCKS.
37 025652 012637 002476      MOV    (SP)+,RCTBLK    ; ;POP STACK INTO RCTBLK
38 025656 013701 002340      MOV    UADR,R1         ; GET BEGINNING ADDRESS OF BUFFER
39 025662 006300              ASL    R0              ; MULTIPLY OFFSET
40 025664 006300              ASL    R0              ; BY 4
41 025666 060001              ADD    R0,R1           ; FIND RBN POSITION IN DESCRIPTOR TABLE
42
43 025670 004737 030074      JSR    PC,INITRW       ; SOME READ/WRITE PARAMETERS
44 025674 063764 002476 000140  ADD    RCTBLK,HC.CPK+P.LBN(R4) ; ADD THE RCT LOGICAL BLOCK TO STARTING
45 025702 005564 000142      ADC    HC.CPK+P.LBN+2(R4) ; HOST LBN OF THE RCT AREA.
46                                ; Setup to issue MULTRD command
47 025706 004737 027224      JSR    PC,MULTRD       ; Issue command to the xDA
48 025712 024250              STEPFTL                ; Here if fatal error.
49
50 025714 112737 000001 002473 24:  MOVB  #1,VBUFF1         ; SET VALID BUFFER 1 DATA FLAG
51 025722 004537 034276      JSR    R5,COPY         ; COPY RCT LOGICAL BLOCK DATA FROM
52 025726 042472              .WORD DBUFF           ; DATA BUFFER TO
53 025730 045472              .WORD BUFF1          ; BUFFER ONE.
54
55 025732 013721 002446      MOV    LBN,(R1)+       ; STORE LO WORD LBN AND
56 025736 013711 002450      MOV    LBN+2,(R1)     ; HI WORD LBN IN DESCRIPTOR ENTRY,
57 025742 052711 020000      BIS    #BIT13,(R1)    ; ASSUME PRIMARY RBN IN REPLACEMENT CODE.

```



```

1      .SBTTL STEP12 Replacement step 12
2      ;STEP12
3      ;
4      ;Issue REPLACE command to controller. Rewrite LBN data using
5      ;WRITE-COMPARE modifier and FORCED ERROR if appropriate.
6      ;
7      ;Inputs:
8      ;   R4 - Address of host comm area
9      ;   LBN - LBN to replace
10     ;   RBN - Replacement RBN
11     ;   SFLAG - Search status
12     ;   VIBUFF - Valid LBN daya flag
13     ;   RCT block 1 - LBN image
14     ;
15     ;Outputs:
16     ;
17     ;Externals:
18     ;   INTCPK, INITRW, ISSUE
19     ;
20     ;Exits:
21     ;   STEP15 - REPLACE error exit
22     ;   STEP9  - LBN verification error exit
23     ;   STEP13 - Successful exit
24     ;   STEPERR - Fatal error exit
25     ;
26
27 026132 STEP12:
28 026132 013704 002324      MOV      HCOM,R4          ; Load host comm area address
29 026136 004737 030626      JSR      PC,INTCPK       ; INITIALIZE THE COMMAND PACKET
30 026142 013764 002452 000120  MOV      RBN,HC.CPK+P.RBN(R4) ; STUFF IN LO ORDER AND
31 026150 013764 002454 000122  MOV      RBN+2,HC.CPK+P.RBN+2(R4) ; HI ORDER RBN TO REPLACE LBN.
32 026156 013764 002446 000140  MOV      LBN,HC.CPK+P.LBN(R4) ; STUFF IN LO ORDER AND
33 026164 013764 002450 000142  MOV      LBN+2,HC.CPK+P.LBN+2(R4) ; HI ORDER LBN BEING REPLACED.
34 026172 105737 002515      TSTB     SFLAG           ; SEE IF THIS IS A PRIMARY REPLACEMENT
35 026176 001003              BNE      1$             ; BRANCH IF NOT, ELSE
36 026200 012737 000001 002346  MOV      #MD.PRI,CMDMOD ; PRIMARY REPLACEMENT MODIFIER IN PACKET
37 026206
38
39     026206 012700 000024      MOV      #OP.RPL,R0      ; Setup to issue OP.RPL command
40     026212 004737 030204      JSR      PC,ISSUE       ; Load MSCP opcode
41     026216 024250              STEPFTL ; Issue command to the xDA
42     026220 001414              BEQ      3$             ; Here if fatal error.
43                                     ; Branch if command successful.
44
45 026222 012737 012313 002552  MOV      #ERM304,BASEMSG ; "Failed REPLACE..."
46 026230 012737 000140 002370  MOV      #B.MSCP,EXTMSK ; Flag all MSCP fields
47 026236 104456              TRAP     C:ERHRD
48     026240 000460              .WORD   304
49     026242 012043              .WORD   HDR301
50     026244 015504              .WORD   ERR301
51 026246 000137 024250      JMP      STEPFTL        ; Fatal exit
52
53 026252
54 026252 012737 044472 002340  MOV      #IBUFF,UADR     ; GET BEGINNING ADDRESS OF IMAGE BUFFER
55 026260 012737 040000 002346  MOV      #MD.CMP,CMDMOD ; STUFF COMPARE MODIFIER IN PACKET
56
57 026266 005737 002516      TST      LBNSTAT        ; Test for valid data
58 026272 100011              BPL      4$             ; Branch if no
59 026274 005737 002164      TST      ENBWE          ; Test for WFE enabled

```

CZUDLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 79 1
 STEP12 Replacement step 12

```

51 026300 001406          BEQ      4$          ; Branch if NO
52
53 026302 052737 010000 002346    BIS      #MD.ERR,CMDMOD    ; Set FORCED ERROR modifier
54 026310 052737 000400 002516    BIS      #TF.WFE,LBNSTAT  ; and WFE status
55 026316          4$:
56 026316 004737 030074          JSR      PC,INITRW        ; INIT SOME READ/WRITE PARAMETERS AND
57 026322 013764 002446 000140    MOV      LBN,HC.CPK+P.LBN(R4) ; GET THE REPLACED LBN ADDRESS.
58 026330 013764 002450 000142    MOV      LBN+2,HC.CPK+P.LBN+2(R4)
59
   026336 012700 000042          MOV      #OP.WR,R0        ; Setup to issue OP.WR command
   026342 004737 030204          JSR      PC,ISSUE        ; Load MSCP opcode
   026346 024250          STEPFTL          ; Issue command to the xDA
   026350 001401          BEQ      5$          ; Here if fatal error,
   026352 003005          BGT      50$         ; Branch if command successful,
                                     ; Branch if command unsuccessful
60
61          ; Success; test for BBR
62 026354          5$:
63 026354 132737 000200 002356    BITB     #EF.BBR,FLAGS    ; Test for BBR
64 026362 001456          BEQ      EXIT12         ; Branch if not SET
65 026364 000451          BR       6$          ; Else, get new RBN
66
67          ; Possible error
68 026366          50$:
69 026366 013700 002354          MOV      EMSTAT,R0       ; Reload endmsg status
70 026372 042700 177740          BIC      #+CST.MSK,R0    ; Clear any subcodes
71 026376 022700 000010          CMP      #ST.DAT,R0     ; Test for Data error
72 026402 001414          BEQ      52$         ; Branch if YES
73
74 026404 012737 012202 002552    MOV      #ERM302,BASEMSG ; "Failed WRITE..."
75 026412 012737 000140 002370    MOV      #B.MSCP,EXTMSK ; Flag all MSCP fields
76 026420 104456          TRAP     C$ERHRD
   026422 000456          .WORD   302
   026424 012043          .WORD   HDR301
   026426 015504          .WORD   ERR301
77 026430 000137 024250          JMP      STEPFTL        ; Fatal error exit
78 026434          52$:
79 026434 005737 002516          TST      LBNSTAT        ; Test LBN image data valid
80 026440 100004          BPL      55$         ; Error if valid
81 026442 022737 000010 002354    CMP      #ST.DAT,EMSTAT  ; Test for FER
82 026450 001423          BEQ      EXIT12         ; OK if yes
83 026452          55$:
84 026452 012737 012361 002552    MOV      #ERM305,BASEMSG ; Report program error
85 026460 012737 000140 002370    MOV      #B.MSCP,EXTMSK ; "Failed verification..."
86 026466 104456          TRAP     C$ERHRD        ; Flag all MSCP fields
   026470 000461          .WORD   305
   026472 012043          .WORD   HDR301
   026474 015504          .WORD   ERR301
87 026476 000404          BR       6$          ; Branch if yes
88 026500          60$:
89 026500 012737 177777 002546    MOV      #-1,NXTSTEP     ; Else, Fatal error
90 026506 000207          RTS      PC            ; Return
91 026510          6$:
92 026510 012737 000011 002546    MOV      #9.,NXTSTEP    ; Next STEP is 9.
93 026516 000207          RTS      PC            ; and return
94 026520          EXIT12:
95 026520 012737 000015 002546    MOV      #13.,NXTSTEP   ; Next STEP is 13.
96 026526 000207          RTS      PC            ; Return

```


STEP13 Replacement step 13

```

1      .SBTTL STEP13 Replacement step 13
2      ;STEP13
3      ;
4      ;Update RCT block 0 to indicate that replacement is complete.
5      ;
6      ;Inputs:
7      ;      R4 - Address of host comm area
8      ;      TBUF  - Working copy of RCT block 0
9      ;      RECOVR - Replacement recovery flag
10     ;
11     ;Outputs:
12     ;
13     ;Externals:
14     ;      INTCPK, INITRW, MULTWR
15     ;
16     ;Exits:
17     ;      EXIT13 - Successful exit
18     ;      STEP17 - RCT MULTIWRITE error exit
19     ;      STEPERR - Fatal error exit
20     ;
21
22     STEP13:
23     026530      013704      002324      MOV      HCOM,R4          ; Load host comm area address
24     026534      004737      030626      JSR      PC,INTCPK       ; INITIALIZE THE COMMAND PACKET
25     026540      012701      043472      MOV      #TBUF,R1       ; GET ADDRESS OF TEMP buffer and
26     026544      010137      002340      MOV      R1,UADR        ; CHANGE THE UNIBUS ADDRESS FOR WRITE.
27     026550      062701      000010      ADD      #RFLAGS,R1     ; SET OFFSET TO REPLACEMENT FLAG WORD
28     026554      005021      CLR      (R1)+           ; CLEAR REPLACEMENT WORD (#4) AND
29     026556      005721      TST     (R1)+           ; SKIP WORD #5.
30     026560      012700      000006      MOV      <CACHID-LBNLO/2>,R0 ; GET NUMBER OF ENTRIES TO CLEAR
31     026564      005021      1$:    CLR      (R1)+           ; CLEAR THAT LOCATION
32     026566      005300      DEC     R0              ; DECR REG. AND SEE IF DONE CLEARING.
33     026570      001375      BNE     1$             ; BRANCH IF NOT, ELSE
34
35     026572      012737      000000      002476      MOV      #0,RCTBLK      ; INIT THE RCT LOGICAL BLOCK TO 0 AND
36     026600      004737      030074      JSR      PC,INITRW     ; SOME READ/WRITE PARAMETERS.
37
38     026604      004737      027511      JSR      PC,MULTWR     ; Setup to issue MULTWR command
39     026610      024250      STEPFTL ; Issue command to the xDA
40     ; Here if fatal error.
41
42     EXIT13:
43     026612      012737      000000      002546      MOV      #0,NXTSTEP    ; NO next STEP
44     026620      012737      000001      002520      MOV      #1,RPLSTAT    ; Set replacement true
45
46     026626      004737      020002      JSR      PC,TYPSTAT    ; Type replacement status
47     026632      105037      002515      CLRB    SFLAG          ; Clear search flag
48     026636      000207      RTS     PC              ; Return

```

```

1      .SBTTL STEP15 Replacement step 15
2      ;STEP15
3      ;
4      ;Rewrite updated RCT descriptor to indicate RBN usage.
5      ;
6      ;Inputs:
7      ;   R4 - Address of host comm area
8      ;   BUFF1 - RCT descriptor block
9      ;   BUFF2 - RCT descriptor block
10     ;   VBUFF2 - BUFF2 data is valid
11     ;   VBUFF1 - BUFF1 data is valid
12     ;
13     ;Outputs:
14     ;
15     ;Externals:
16     ;   INTCPK, INITRW, MULTWR
17     ;
18     ;Exits:
19     ;   STEP16 - Successful exit
20     ;   STEPERR - Fatal error exit
21     ;
22
23     STEP15:
24     026640 013704 002324      MOV      HCOM,R4          ; Load host comm area address
25     026644 004737 030626      JSR      PC,INTCPK       ; INITIALIZE THE COMMAND PACKET
26     026650 105737 002473      TSTB    VBUFF1          ; DATA VALID IN BUFFER 1 ?
27     026654 001415              BEQ      2$              ; BRANCH IF NO, ELSE
28     026656 012737 045472 002340  MOV      #BUFF1,UADR     ; GET BEGINNING ADDRESS OF BUFF1 BUFFER
29     026664 004737 030074      JSR      PC,INITRW      ; INIT SOME READ/WRITE PARAMETERS
30     026670 063764 002476 000140  ADD      RCTBLK,HC.CPK+P.LBN(R4) ; ADD THE RCT LOGICAL BLOCK TO STARTING
31     026676 005564 000142      ADC      HC.CPK+P.LBN+2(R4) ; HOST LBN OF THE RCT AREA.
32
33     026702 004737 027514      JSR      PC,MULTWR      ; Setup to issue MULTWR command
34     026706 024250      STEPFTL ; Issue command to the xDA
35
36     2$:
37     026710 105737 002474      TSTB    VBUFF2          ; DATA VALID IN BUFFER 2 ?
38     026714 001415              BEQ      EXIT15         ; BRANCH IF NO, ELSE
39     026716 012737 046472 002340  MOV      #BUFF2,UADR     ; GET BEGINNING ADDRESS OF BUFF2 BUFFER
40     026724 004737 030074      JSR      PC,INITRW      ; SOME READ/WRITE PARAMETERS
41     026730 063764 002500 000140  ADD      MATBLK,HC.CPK+P.LBN(R4) ; ADD THE RCT LOGICAL BLOCK TO STARTING
42     026736 005564 000142      ADC      HC.CPK+P.LBN+2(R4) ; HOST LBN OF THE RCT AREA.
43
44     026742 004737 027514      JSR      PC,MULTWR      ; Setup to issue MULTWR command
45     026746 024250      STEPFTL ; Issue command to the xDA
46
47     EXIT15:
48     026750 012737 000020 002546  MOV      #16.,NXTSTEP    ; Next STEP is 16.
49     026756 000207              RTS      PC              ; Return
50

```

```

1      .SBTTL STEP16 Replacement step 16
2      ;STEP16
3      ;
4      ;Write the saved LBN data to the original LBN with FORCED ERROR set,
5      ;if appropriate.
6      ;
7      ;Inputs:
8      ;   R4 - Address of host comm area
9      ;   LBN - LBN address
10     ;   IBUFF - LBN image
11     ;   VIBUFF - Image valid flag
12     ;
13     ;Outputs:
14     ;
15     ;Externals:
16     ;   INCPK, INITRW, ISSUE
17     ;
18     ;Exits:
19     ;   NXTSTEP = 0 - Successful exit
20     ;   STEPERR - Fatal error exit
21     ;
22
23     STEP16:
24     026760 013704 002324      MOV     HCOM,R4          ; Load host comm area address
25     026764 004737 030626      JSR     PC,INTCPK       ; INITIALIZE THE COMMAND PACKET
26     026770 012737 044472 002340  MOV     #IBUFF,UADR     ; GET BEGINNING ADDRESS OF IMAGE BUFFER
27
28     026776 005737 002516      TST     LBNSTAT        ; SEE IF LBN IMAGE DATA WAS VALID,
29     027002 100006              BPL     1$             ; BRANCH IF SO, ELSE
30     027004 005737 002164      TST     ENBWFE         ; Test for WFE enabled
31     027010 001403              BEQ     1$             ; Branch if NO
32
33     027012 052737 010000 002346  BIS     #MD.ERR,CMDMOD ; STUFF FORCED ERROR MODIFIER IN PACKET.
34     027020              1$:
35     027020 004737 030074      JSR     PC,INITRW      ; INIT SOME READ/WRITE PARAMETERS AND
36     027024 013764 002446 000140  MOV     LBN,HC.CPK+P.LBN(R4) ; GET THE REPLACED LBN ADDRESS.
37     027032 013764 002450 000142  MOV     LBN+2,HC.CPK+P.LBN+2(R4)
38
39     027040 012700 000042      MOV     #OP.WR,R0      ; Setup to issue OP.WR command
40     027044 004737 030204      JSR     PC,ISSUE       ; Load MSCP opcode
41     027050 024250              STEPFTL                ; Issue command to the xDA
42     027052 001423              BEQ     EXIT16         ; Here if fatal error,
43                                     ; Branch if command successful.
44
45     027054 013700 002354      MOV     EMSTAT,R0      ; Reload endmsg status
46     027060 042700 177740      BIC     #+CST.MSK,R0   ; Clear any subcodes
47     027064 022700 000010      CMP     #ST.DAT,R0     ; Test for Data error
48     027070 001414              BEQ     EXIT16         ; Branch if YES
49
50     027072 012737 012202 002552  MOV     #ERM302,BASEMSG ; "Failed WRITE..."
51     027100 012737 000140 002370  MOV     #B.MSCP,EXTMSK ; Flag all MSCP fields
52     027106 104456              TRAP    C$ERHRD
53     027110 000456              .WORD  302
54     027112 012043              .WORD  HDR301
55     027114 015504              .WORD  ERR301
56     027116 000137 024250      JMP     STEPFTL        ; Fatal error exit
57
58     EXIT16:
59
60     027122

```

51 027122 012737 000021 002546
52 027130 000207
53

0V #17.,NXTSTEP
RTS PC

; Next STEP is 17.
; Return


```

1      .SBTTL STEP17 Replacement step 17
2      ;STEP17
3      ;
4      ;Update RCT block 0 to indicate that replacement is complete.
5      ;
6      ;Inputs:
7      ;      R4 - Address of host comm area
8      ;      TBUFF - Working copy of RCT block 0
9      ;
10     ;Outputs:
11     ;
12     ;Externals:
13     ;      INCPK, INITRW, MULTWR
14     ;
15     ;Exits:
16     ;      EXIT17 - Successful exit
17     ;      STEPERR - Fatal error exit
18     ;
19
20     STEP17:
21     027132      013704      002324      MOV      HCOM,R4      ; Load host comm area address
22     027136      004737      030626      JSR      PC,INTCPK    ; INITIALIZE THE COMMAND PACKET
23     027142      012701      043472      MOV      #TBUFF,R1    ; GET ADDRESS OF TEMP BUFFER AND
24     027146      010137      002340      MOV      R1,UADR      ; CHANGE THE UNIBUS ADDRESS FOR WRITE.
25     027152      062701      000010      ADD      #RFLAGS,R1   ; SET OFFSET TO REPLACEMENT FLAG WORD
26     027156      005021      CLR      (R1)+         ; CLEAR REPLACEMENT WORD (#4) AND
27     027160      005721      TST     (R1)+         ; SKIP WORD #5.
28     027162      012700      000006      MOV      #<CACHID-LBNLO/2>,R0 ; GET NUMBER OF ENTRIES TO CLEAR
29     027166      005021      1$:     CLR      (R1)+         ; CLEAR THAT LOCATION
30     027170      005300      DEC     R0             ; DECR REG. AND SEE IF DONE CLEARING,
31     027172      001375      BNE     1$            ; BRANCH IF NOT, ELSE
32
33     027174      012737      000000      002476      MOV      #0,RCTBLK    ; INIT THE RCT LOGICAL BLOCK TO 0 AND
34     027202      004737      030074      JSR      PC,INITRW    ; SOME READ/WRITE PARAMETERS.
35
36     027206      004737      027514      JSR      PC,MULTWR    ; Setup to issue MULTWR command
37     027212      024250      STEPFTL  ; Issue command to the xDA
38     027214      012737      177777      002546      MOV      #-1,NXTSTEP ; Here if fatal error.
39     027222      000207      RTS     PC             ; Else, load success
40
41     ; Return
  
```

```

1      .SBTTL  MULTRD  RCT Multiread routine
2      ;MULTRD
3      ;
4      ;THIS ROUTINE WILL PERFORM MULTIPLE READS, IF A BLOCK IN THE RCT CANNOT
5      ;BE READ FROM THE 1ST COPY OF THE RCT.
6      ;
7      ;INPUTS:
8      ;   COPIES - NUMBER OF COPIES OF THE RCT AREA
9      ;   RCTSIZ - RCT SIZE (W/ PAD)
10     ;   R4 - ADDRESS OF HOST COMMUNICATION AREA
11     ;OUTPUTS:
12     ;   LOAD COMMAND PACKET WITH;
13     ;   COMPARE MODIFIER,
14     ;   OPC      - RETURN IF FATAL ERROR,
15     ;   PC+2    - RETURN IF COMMAND NOT SUCCESSFUL
16     ;
17     ;Calling Seq:
18     ;
19     ;   JSR      PC,MULTRD
20     ;   .WORD   <error exit>
21     ;
22
23     MULTRD:
24     027224 010146      MOV      R1,-(SP)          ;; PUSH R1 ON STACK
25     027226 112737 000001 002424  MOVB     #1,RCTCOPY      ; Start with copy 1
26     027234 105037 002475      CLRB     ERRCNT          ; Clear RCT error count
27     027240 005037 002444      CLR      TMPSTAT        ; Clear temp status
28     027244 013704 002324      MOV      HCOM,R4         ; Load host comm area address
29     027250
30     027250 012737 040000 002346 10$:    MOV      #MD.CMP,CMDMOD   ; STUFF COMPARE MODIFIER IN PACKET
31     027256 004737 034240      JSR      PC,CLRBUF       ; CLEAR THE BUFFER BEFORE READING
32
33     027262 012700 000041      MOV      #OP.RD,R0       ; Setup to issue OP.RD command
34     027266 004737 030204      JSR      PC,ISSUE        ; Load MSCP opcode
35     027272 027500          JSR      PC,ISSUE        ; Issue command to the xDA
36     027274 001450          BEQ      50$             ; Here if fatal error,
37
38
39     027276 013700 002354      BEQ      50$             ; Branch if command successful.
40
41     20$:
42     027276 013700 002354      MOV      EMSTAT,R0       ; Reload endmsg status
43     027302 042700 177740      BIC      #+CST.MSK,R0    ; Clear any subcodes
44     027306 022700 000010      CMP      #ST.DAY,R0     ; Test for Data error
45     027312 001416          BEQ      30$             ; Branch if YES
46
47     30$:
48     027314 012737 013216 002552  MOV      #ERM601,BASEMSG  ; "Failed RCT READ..."
49     027322 012737 000140 002370  MOV      #B.MSCP,EXTMSK  ; Flag all MSCP fields
50     027330 104456          TRAP     C$ERHRD
51     027332 001131          .WORD   601
52     027334 012043          .WORD   HDR301
53     027336 015504          .WORD   ERR301
54     027340 012737 177777 002442  MOV      #-1,STATUS      ; Set Fatal status
55     027346 000454          BR      70$             ; Fatal error exit
56
57     40$:
58     027350
59     027350 050037 002444      BIS      R0,TMPSTAT      ; Save status
60     027354 105237 002475      INCB     ERRCNT          ; Increment error count
61     027360 004737 016370      JSR      PC,RCTERR       ; Check this block bad before
62     027364 027446          60$                    ; Fatal error return
63     027366
64     027366 105237 002424      40$:    INCB     RCTCOPY        ; Increment copy

```

CZUDLAD BBR Replacement Utility MACRO Y05.03c Friday 19 Apr-85 10:42 Page 84 1
 MULTRD RCT Multiread routine

```

51 027372 123737 002423 002424      CMPB   COPIES,RCTCOPY      ; Test for last copy
52 027400 002406                      BLT    50$                 ; Branch if none left
53 027402 063764 002416 000140      ADD    RCTSIZ,HC.CPK+P.LBN(R4) ; Else, adjust LBN
54 027410 005564 000142      ADC    HC.CPK+P.LBN+2(R4)   ; To next copy of RCT
55 027414 000715                      BR     10$                 ; And write next copy
56
57                                     ; Normal exit
58 027416                      ; 50$:
59 027416 012737 000000 002346      MOV    @0,CMDMOD           ; Clear command modifiers
60 027424 013737 002444 002442      MOV    TMPSTAT,STATUS     ; Reload status
61 027432 012601                      MOV    (SP)+,R1           ;; POP STACK INTO R1
62 027434 062716 000002                      ADD    @2,(SP)            ; Bump past error return
63 027440 013700 002442      MOV    STATUS,R0          ; Set condition codes for caller
64 027444 000207                      RTS    PC                 ; and return
65 027446                      ; 60$:
66 027446 012737 013267 002552      MOV    @ERM603,BASEMSG    ; "Failed READ of all RCT copies..."
67 027454 012737 000000 002370      MOV    @0,EXTMSK         ; Flag RCT fields
68 027462 104455                      TRAP   C$ERDF
    027464 001133                      .WORD 603
    027466 013173                      .WORD HDR601
    027470 015720                      .WORD ERR601
69 027472 012737 177777 002442      MOV    @-1,STATUS        ; Set cond. codes
70
71                                     ; Fatal Error Exit
72 027500                      ; 70$:
73 027500 012601                      MOV    (SP)+,R1           ;; POP STACK INTO R1
74 027502 017616 000000      MOV    @0(SP),(SP)       ; Load error return
75 027506 013700 002442      MOV    STATUS,R0          ; Set condition codes for caller
76 027512 000207                      RTS    PC                 ; and return
77

```

```

1      .SBTTL MULTWR RCT Multiwrite routine
2      ;MULTWR
3      ;
4      ;THIS ROUTINE WILL PERFORM MULTIPLE WRITES ON ALL COPIES OF A LOGICAL BLOCK
5      ;IN THE RCT AREA.
6      ;
7      ;INPUTS:
8      ;   COPIES - NUMBER OF COPIES OF THE RCT AREA
9      ;   RCTSIZ - RCT SIZE (W/ PAD)
10     ;   R4 - ADDRESS OF HOST COMMUNICATION AREA
11     ;OUTPUTS:
12     ;   LOAD COMMAND PACKET WITH;
13     ;   COMPARE MODIFIER,
14     ;   @PC - RETURN IF FATAL ERROR,
15     ;   PC+2- RETURN IF COMMAND NOT SUCCESSFUL
16     ;
17     ;Calling Seq:
18     ;
19     ;   JSR   PC,MULTWR
20     ;   .WORD <error exit>
21     ;
22
23     MULTWR:
24     027514 010146      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
25     027516 105037 002475  CLRB   ERRCNT          ;INIT THE ERROR COUNT
26     027522 005037 002444  CLR    TMPSTAT        ; Reset temp status
27     027526 112737 000001 002424  MOVB  #1,RCTCOPY      ; Start with first copy
28     027534 013704 002324  MOV    HCOM,R4        ; Load host comm area address
29     027540
30     027540 012737 040000 002346 10$:  MOV   #MD.CMP,CMDMOD   ; STUFF COMPARE MODIFIER IN PACKET
31     027546 012700 000042      MOV   #OP.WR,R0        ; Setup to issue OP.WR command
32     027552 004737 030204      JSR   PC,ISSUE        ; Load MSCP opcode
33     027556 030060              70$   ; Issue command to the xDA
34     027560 001472              BEQ   45$             ; Here if fatal error,
35                                     ; Branch if command successful.
36
37     20$:  MOV   EMSTAT,R0          ; Reload endmsg status
38     027562 013700 002354      BIC   #+CST.MSK,R0    ; Clear any subcodes
39     027566 042700 177740      CMP   #ST.DAT,R0     ; Test for Data error
40     027614 104456              BEQ   25$             ; Branch if YES
41     027616 001132
42     027620 012043
43     027622 015504
44     027624 012737 177777 002442  MOV   #ERM602,BASEMSG ; "Failed RCT WRITE..."
45     027632 000512              MOV   #B.MSCP,EXTMSK ; Flag all MSCP fields
46     027634 050037 002444      TRAP  C$ERHRD
47     027640 105237 002475      .WORD 602
48     027644 004737 016370      .WORD HDR301
49     027650 030026              .WORD ERR301
50     027652 052737 010000 002346  MOV   #-1,STATUS     ; Set Fatal status
51                                     BR    70$             ; Fatal error exit
52
53     25$:  BIS   R0,TMPSTAT        ; Save status
54     027634 050037 002444      INCB  ERRCNT         ; COUNT THE NUMBER ERRORS DETECTED
55     027640 105237 002475      JSR   PC,RCTERR     ; Test if new bad block
56     027644 004737 016370      60$   ; Fatal error return
57
58     30$:  BIS   #MD.ERR,CMDMOD   ; STUFF FORCED ERROR MODIFIER IN PACKET
59     027652 052737 010000 002346  ; Setup to issue OP.WR command
60

```

MULTWR RCT Multiwrite routine

```

027660 012700 000042      MOV    #OP.WR,R0      ; Load MSCP opcode
027664 004737 030204      JSR    PC,ISSUE      ; Issue commar.d to the xDA
027670 030060              70$                ; Here if fatal error.
J27672 001425              BEQ    45$           ; Branch if command successful.
51 027674              35$:
52 027674 013700 002354      MOV    EMSTAT,R0     ; Reload emdmsg status
53 027700 042700 177740      BIC    #+CST.MSK,R0  ; Clear any subcodes
54 027704 022700 000010      CMP    #ST.DAT,R0   ; Test for Data error
55 027710 001416              BEQ    45$           ; Branch if YES
56 027712              40$:
57 027712 012737 013242 002552  MOV    #ERM602,BASEMSG ; "Failed RCT WRITE..."
58 027720 012737 000140 002370  MOV    #B.MSCP,EXTMSK ; Flag all MSCP fields
59 027726 104456              TRAP   C#ERHRD
    027730 001132              .WORD 602
    027732 012043              .WORD HDR301
    027734 015504              .WORD ERR301
60 027736 012737 177777 002442  MOV    #-1,STATUS    ; Set Fatal status
61 027744 000445              B      70$           ; Fatal error exit
62 027746              45$:
63 027746 105237 002424              INCB   RCTCOPY       ; Increment copy
64 027752 123737 002423 002424  CMPB   COPIES,RCTCOPY ; Test for last copy
65 027760 002406              BLT    50$           ; Branch if none left
66 027762 063764 002416 000140  ADD    RCTSIZ,HC.CPK+P.LBN(R4) ; Else, adjust LBN
67 027770 005564 000142      ADC    HC.CPK+P.LBN+2(R4) ; To next copy of RCT
68 027774 000661              BR     10$           ; And write next copy
69
70              ; Normal exit
71 027776              50$:
72 027776 012737 000000 002346  MOV    #0,CMMOD      ; Clear command modifiers
73 030004 013737 002444 002442  MOV    TMPSTAT,STATUS ; Reload status
74 030012 012601              MOV    (SP)+,R1      ; POP STACK INTO R1
75 030014 062716 000002      ADD    #2,(SP)       ; Bump past error return
76 030020 013700 002442      MOV    STATUS,R0     ; Set condition codes for caller
77 030024 000207              RTS    PC            ; and return
78 030026              60$:
79 030026 012737 013334 002552  MOV    #ERM604,BASEMSG ; "Failed WRITE to all copies..."
80 030034 012737 000000 002370  MOV    #0,EXTMSK    ; Flag RCT fields
81 030042 104455              TRAP   C#ERDF
    030044 001134              .WORD 604
    030046 013173              .WORD HDR601
    030050 015720              .WORD ERR601
82 030052 012737 177777 002442  MOV    #-1,STATUS    ; Set cond. codes
83
84              ; Fatal Error Exit
85 030060              70$:
86 030060 012601              MOV    (SP)+,R1      ; POP STACK INTO R1
87 030062 017616 000000      MOV    @8(SP),(SP)   ; Load error return
88 030066 013700 002442      MOV    STATUS,R0     ; Set condition codes for caller
89 030072 000207              RTS    PC            ; and return
90

```

INITRW Initialize command packet for R/W

```

1      .SBTTL INITRW Initialize command packet for R/W
2      ;INITRW
3      ;
4      ;INITIALIZE SOME GENERAL PARAMETERS IN THE COMMAND PACKET FOR A
5      ;READ OR WRITE COMMAND.
6      ;
7      ;INPUTS:
8      ;   UADR - UNIBUS ADDRESS IN MEMORY,
9      ;   BCNT - BYTE COUNT,
10     ;   UNSIZ - LO ORDER WORD OF STARTING RCT LBN,
11     ;   UNSIZ+2 - HI ORDER WORD OF STARTING RCT LBN,
12     ;   R4 - ADDRESS OF HOST COMMUNICATION AREA
13     ;OUTPUTS:
14     ;   LOAD COMMAND PACKET WITH;
15     ;   UNIBUS ADDRESS,
16     ;   BYTE COUNT,
17     ;   STARTING RCT LBN
18     ;
19     ;Calling Seq:
20     ;
21     ;   JSR    PC,INITRW
22     ;
23
24 030074      INITRW:
25 030074 013704 002324      MOV    HCOM,R4          ; Load host comm area address
26 030100 013764 002340 000124  MOV    UADR,HC.CPK+P.UADR(R4) ;Load UNIBUS buffer address
27 030106 013764 002334 000120  MOV    BCNT,HC.CPK+P.BCNT(R4) ;STUFF low order BYTE COUNT
28 030114 013764 002336 000122  MOV    BCNT+2,HC.CPK+P.BCNT+2(R4) ;STUFF high order BYTE COUNT
29 030122 013764 002430 000140  MOV    UNSIZ,HC.CPK+P.LBN(R4) ;STUFF IN LO ORDER UNIT SIZE AND
30 030130 013764 002432 000142  MOV    UNSIZ+2,HC.CPK+P.LBN+2(R4) ;HI ORDER UNIT SIZE TO GET STARTING
31                                     ;LBN OF THE RCT AREA.
32 030136 000207      RTS    PC          ;RETURN
33

```

INITAC Initialize command packet for ACCESS

```

1      .SBTTL INITAC Initialize command packet for ACCESS
2      ;INITAC
3      ;
4      ;Initialize command packet parameters for an ACCESS command.
5      ;
6      ;Inputs:
7      ;      R4 - Address of host communications area
8      ;      UADR - Address of buffer
9      ;      GLBN - ACCESS group starting LBN
10     ;      GBCNT ACCESS group byte count
11     ;
12     ;Outputs:
13     ;      Load command packets with:
14     ;      Zero buffer descriptor
15     ;      Starting LBN
16     ;      Byte count
17     ;
18     ;Calling Seq:
19     ;
20     ;      JSR    PC,INITAC
21     ;
22
23 030140      INITAC:
24 030140 013704 002324      MOV    HCOM,R4          ; Load host comm area address
25 030144 012764 000000 000124      MOV    #0,HC.CPK+P.UADR(R4) ; Load UNIBUS buffer address
26 030152 013764 002566 000120      MOV    GBCNT,HC.CPK+P.BCNT(R4) ; Load low order group byte count
27 030160 013764 002570 000122      MOV    GBCNT+2,HC.CPK+P.BCNT+2(R4) ; Load high order group byte count
28 030166 013764 002562 000140      MOV    GLBN,HC.CPK+P.LBN(R4) ; Load group
29 030174 013764 002564 000142      MOV    GLBN+2,HC.CPK+P.LBN+2(R4) ; starting LBN address
30
31 030202 000207      RTS    PC          ; Return to caller
32

```

ISSUE Issue command

```

1      .SBTTL  ISSUE  Issue command
2      ;ISSUE
3      ;
4      ;ISSUE COMMAND TO xDA. CHECK THAT EXECUTION HAPPENED WITHOUT ERROR.
5      ;
6      ;INPUTS:
7      ;      RO      MSCP command code
8      ;      CTLRTBL Controller table address
9      ;
10     ;OUTPUTS:
11     ;      RO =    End message status
12     ;              0 = Success
13     ;              >0 = Error
14     ;              -1 = Fatal
15     ;
16     ;
17     ;Calling Seq:
18     ;
19     ;      MOV      #OP_CODE,RO
20     ;      JSR      PC,ISSUE
21     ;      .WORD    <error exit>
22     ;
23     ;
24     030204      ISSUE:
25     030204      010037  002344      MOV      RO,OPCODE      ; Save MSCP command
26     030210      013704  002324      MOV      HCOM,R4       ; Reload comm area ptr
27     030214      013705  002304      MOV      CTLRTBL,R5    ; Load table start address
28     030220      004737  030632      JSR      PC,BLDCMD     ; Build command packet
29     030224      004737  030744      JSR      PC,SNDCMD     ; Send command to XDA
30     030230
31     030230      004737  031072      JSR      PC,WAIT       ; Wait for message response
32     030234      030474      42$
33     030236      001100      BNE      41$           ; Error return
34     ;
35     ;      Unload end packet fields
36     030240      10$:
37     030240      013704  002324      MOV      HCOM,R4       ; Reload comm area ptr
38     030244      013705  002304      MOV      CTLRTBL,R5    ; Load table start address
39     030250      042765  000004  000014      BIC      #CT.CMD,C.FLG(R5) ; Clear command issued
40     030256      016437  000020  002350      MOV      HC.MPK+P.CRF(R4),ENDREF ; Save endmsg reference number
41     030264      016437  000032  002354      MOV      HC.MPK+P.STS(R4),EMSTAT ; Save endmsg status
42     030272      116437  000031  002356      MOV      HC.MPK+P.FLGS(R4),FLAGS ; Save endmsg flags
43     030300      116437  000030  002352      MOV      HC.MPK+P.OPCD(R4),ECODE ; Save endcode
44     ;
45     ;      Test command reference and end code
46     030306      20$:
47     030306      132737  000200  002352      BITB     #OP.END,ECODE  ; Test for end msg packet
48     030314      001004      BNE      25$           ; Branch if YES
49     030316      004737  016634      JSR      PC,ERRLOG     ; Else, process error log packet
50     030322      030614      50$
51     030324      000741      BR       5$           ; Error return
52     ;
53     ;      Process error log packet
54     030326      25$:
55     030326      122737  000207  002352      CMPB     #OP.END+OP.SEX,ECODE ; Test for serious exception
56     030334      001512      BEQ      44$           ; Branch if YES
57

```


ISSUE Issue command

```

58                                     ; Process end message packet
59 030336                               ; 30$:
60 030336 023737 002342 002350      CMP      CMDREF,ENDREF      ; Compare reference numbers
61 030344 001071                       BNE      43$                ; Branch if different
62
63 030346 013700 002344               MOV      OPCODE,RO        ; Reload MSCP command code
64 030352 052700 000200               BIS      #OP.END,RO       ; Set end msg bit
65 030356 120037 002352               CMPB    RO,F.CODE        ; and compare
66 030362 001010                       BNE      40$                ; Branch if different
67
68                                     ; SUCCESS exit
69 030364                               ; 35$:
70 030364 062716 000002               ADD      #2,(SP)          ; Else, bump past error return
71 030370 013737 002354 002442       MOV      EMSTAT,STATUS    ; Set success status
72 030376 013700 002442               MOV      STATUS,RO       ; Load status
73 030402 000207                       RTS      PC                ; and return
74
75                                     ; Process unrecognized end packet
76 030404                               ; 40$:
77 030404 012737 177777 002442       MOV      #-1,STATUS       ; Set fatal status
78 030412 012737 012617 002552       MOV      #ERM307,BASEMSG  ; "Unrecognized endcode"
79 030420 012737 000140 002370       MOV      #B.MSCP,EXTMSK  ; Flag all MSCP fields
80 030426 104455                       TRAP    C$ERDF
   030430 000463                       .WORD   307
   030432 012043                       .WORD   HDR301
   030434 015504                       .WORD   ERR301
81 030436 000466                       BR      50$                ; Exit
82
83                                     ; Process timeout
84 030440                               ; 41$:
85 030440 012737 177777 002442       MOV      #-1,STATUS       ; Set fatal status
86 030446 012737 012727 002552       MOV      #ERM401,BASEMSG  ; "No interrupt received"
87 030454 012737 000003 002370       MOV      #B.SADDR!B.SAVAL,EXTMSK; Flag all address and value
88 030462 104455                       TRAP    C$ERDF
   030464 000621                       .WORD   401
   030466 012663                       .WORD   HDR401
   030470 015650                       .WORD   ERR401
89 030472 000450                       BR      50$                ; Exit
90
91                                     ; Report fatal controller error
92 030474                               ; 42$:
93 030474 012737 177777 002442       MOV      #-1,STATUS       ; Set fatal status
94 030502 012737 013000 002552       MOV      #ERM402,BASEMSG  ; "Controller error reported..."
95 030510 012737 000003 002370       MOV      #B.SADDR!B.SAVAL,EXTMSK; Flag all address and value
96 030516 104455                       TRAP    C$ERDF
   030520 000622                       .WORD   402
   030522 012663                       .WORD   HDR401
   030524 015650                       .WORD   ERR401
97 030526 000432                       BR      50$                ; Exit
98
99                                     ; Report cmd reference error
100 030530                              ; 43$:
101 030530 012737 177777 002442       MOV      #-1,STATUS       ; Set fatal status
102 030536 012737 012526 002552       MOV      #ERM306,BASEMSG  ; "Command reference number error"
103 030544 012737 000151 002370       MOV      #B.CMDRF!B.EMREF!B.MSCP,EXTMSK ; Flag all MSCP fields
104 030552 104455                       TRAP    C$ERDF
   030554 000462                       .WORD   306

```

ISSUE Issue command

```

030556 012043      .WORD  HDR301
030560 015504      .WORD  ERR301
105
106                ;
107 030562          ;4$: Serious exception error
108 030562 012737 177777 002442  MOV    @-1,STATUS      ; Set fatal status
109 030570 012737 013050 002552  MOV    @ERM403,BASEMSG ; "Serious exception"
110 030576 012737 000140 002370  MOV    @B.MSCP,EXTMSK ; Flag all MSCP fields
111 030604 104455    TRAP   C$ERDF
      030606 000623    .WORD  403
      030610 012663    .WORD  HDR401
      030612 015650    .WORD  ERR401
112
113                ;
114 030614          ;50$: Error exit
115 030614 017616 000000  MOV    @ (SP), (SP)    ; Load error return
116 030620 013700 002442  MOV    STATUS,R0      ; Reload R0
117 030624 000207    RTS    PC                ; and return
118

```

```

1      .SBTTL BLDCMD Build command
2      ;BLDCMD
3      ;
4      ;BUILD A COMMAND IN COMMAND PACKET
5      ;
6      ;INPUTS:
7      ;   R0  COMMAND CODE
8      ;   R5  - CONTROLLER TABLE ADDRESS,
9      ;
10     ;OUTPUTS:
11     ;   R4  - ADDRESS OF HOST COMMUNICATION AREA,
12     ;   SET COMMAND PACKET SIZE,
13     ;   SET CIRCUIT IDENT,
14     ;   IF 3#=1, CLEAR REST OF COMMAND PACKET FIELDS, ELSE
15     ;       DON'T CLEAR REST OF COMMAND PACKET,
16     ;   SET OPCODE,
17     ;   SET UNIT NUMBER,
18     ;   CLR 3#
19     ;
20     ;Calling Seq:
21     ;   JSR   PC,INTCPK           ; Initialize packet/build command
22     ;or
23     ;   JSR   PC,BLDCMD         ; Build command
24     ;
25
29
30 030626      INTCPK:
31 030626      005237 030742      INC      3#           ;ENABLE COMMAND PACKET TO BE CLEARED
32 030632      BLDCMD:
33 030632      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
34 030634      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
35 030636      013705 002304      MOV      CTLRTBL,R5    ; Load table start address
36 030642      013704 002324      MOV      HCOM,R4      ; Load host comm area address
37 030646      010402      MOV      R4,R2        ;COPY TO R2
38 030650      062702 000100      ADD      #HC.CEV,R2    ;COMPUTE ADDRESS OF COMMAND ENVELOPE
39 030654      012722 000060      MOV      #HC.PSZ,(R2)+ ;LOAD PACKET SIZE
40 030660      012722 000000      MOV      #MSCP,(R2)+  ;LOAD MSCP CIRCUIT IDENTIFIER
41 030664      010122      MOV      R1,(R2)+     ;PUT IDENTIFIER INTO PACKET
42 030666      012701 000030      MOV      #<HC.PSZ>/2,R1 ;GET WORDS TO CLEAR
43 030672      013764 002344 000114      MOV      OPCODE,HC.CPK+P.OPCD(R4); Load command code
44 030700      013764 002346 000116      MOV      CMDMOD,HC.CPK+P.MOD(R4); Load command modifiers
45 030706      1#:
46 030706      005737 030742      TST      3#           ;INITIALIZE REST OF COMMAND PACKET ?
47 030712      001403      BEQ      2#           ;BRANCH IF NO
48 030714      005022      CLR      (R2)+        ;CLEAR PACKET
49 030716      005301      DEC      R1
50 030720      001372      BNE      1#
51 030722      016564 000006 000110      2#:      MOV      C.DRVN(R5),HC.CPK+P.UNIT(R4) ;LOAD UNIT NUMBER
52 030730      005037 030742      CLR      3#           ;DISABLE CLEAR FOR NEXT COMMAND
53 030734      012602      MOV      (SP)+,R2     ;;POP STACK INTO R2
54 030736      012601      MOV      (SP)+,R1     ;;POP STACK INTO R1
55 030740      000207      RTS      PC
56 030742      000000      3#:      .WORD 0           ;INIT PACKET FLAG GOES HERE
57
61

```

```

1      .SBTTL SNDCMD Send command
2      :SNDCMD
3      :
4      :SEND A COMMAND TO THE xDA.
5      :CLEAR THE RESPONSE PACKET. MARK BOTH PACKETS AVAILABLE TO THE
6      :xDA. SET COMMAND ISSUED BIT IN CONTROLLER TABLE AND INITIALIZE
7      :TIMEOUT COUNTER.
8      :
9      :INPUTS:
10     :      R5  CONTROLLER TABLE ADDRESS
11     :OUTPUTS:
12     :      R4  - ADDRESS OF HOST COMMUNICATION AREA
13     :
14     :Calling Seq:
15     :
16     :      JSR  PC,SNDCMD
17     :
18     :
19     :
20     SNDCMD:
21     030744 010046      MOV      R0,-(SP)          ;;PUSH R0 ON STACK
22     030746 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
23     030750 013705 002304  MOV      CTLRTBL,R5      ; Load table start address
24     030754 013704 002324  MOV      HCOM,R4         ; Load host comm area address
25     030760 005237 002342  INC      CMDREF          ; Increment command reference
26     030764 013764 002342 000104  MOV      CMDREF,HC.CPK+P.CRF(R4) ;PUT REFERENCE NUMBER IN PACKET
27     030772 012700 000014  MOV      @HC.MEV,R0      ;GET INDEX TO MESSAGE ENVELOPE
28     030776 060400      ADD      R4,R0           ;IN HOST COMMUNICATION AREA,
29     031000 010046      MOV      R0,-(SP)          ;;PUSH R0 ON STACK
30     031002 062716 000004  ADD      @HC.ESZ,(SP)     ;AND ADD ENVELOPE SIZE TO INDEX.
31     031006 012701 000032  MOV      @<HC.PSZ+HC.ESZ>/2,R1 ;SIZE OF MESSAGE PACKET
32     031012 005020      1$: CLR      (R0)+          ;CLEAR ENTIRE MESSAGE PACKET
33     031014 005301      DEC      R1
34     031016 001375      BNE     1$
35     031020 012600      MOV      (SP)+,R0         ;;POP STACK INTO R0
36     031022 010064 000004  MOV      R0,HC.MSG(R4)    ;INSERT THE MESSAGE PACKET ADDRESS
37     031026 012764 140000 000006  MOV      @RG.OWN+RG.FLG,HC.MCT(R4) ;AND MARK THE OWNER/FLAG BITS IN
38     :                                     ;THE MESSAGE RING.
39     031034 062700 000064  ADD      @<HC.PSZ+HC.ESZ>,R0 ;INDEX TO THE BEGINNING OF THE
40     031040 010064 000010  MOV      R0,HC.CMD(R4)    ;COMMAND PACKET, INSERT THE ADDRESS
41     031044 012764 100000 000012  MOV      @RG.OWN,HC.CCT(R4) ;AND MARK THE OWNER BIT IN THE
42     :                                     ;COMMAND RING.
43     031052 005775 000000  TST      @R5              ;TELL xDA TO START POLLING
44     031056 052765 000004 000014  BIS      @CT.CMD,C.FLG(R5) ;MARK COMMAND ISSUED
45     031064 012601      MOV      (SP)+,R1         ;;POP STACK INTO R1
46     031066 012600      MOV      (SP)+,R0         ;;POP STACK INTO R0
47     031070 000207      RTS      PC
  
```

WAIT Wait for xDA response

```

1      .SBTTL WAIT Wait for xDA response
2      ;WAIT
3      ;
4      ;WAIT FOR xDA TO RESPOND WITH A MESSAGE PACKET
5      ;
6      ;INPUTS:
7      ;      R5 - ADDRESS OF CONTROLLER TABLE
8      ;OUTPUTS:
9      ;      Z SET   IF NO ERROR,
10     ;      Z CLR   IF timeout
11     ;
12     ;Calling Seq:
13     ;
14     ;      JSR     PC, WAIT
15     ;      .WORD  <error exit>
16     ;      BNE    <cmd timeout>
17     ;
18
19     WAIT:
20     031072 010046      MOV     R0, -(SP)           ;; PUSH R0 ON STACK
21     031074 010146      MOV     R1, -(SP)           ;; PUSH R1 ON STACK
22     031076 012700 000036      MOV     #30., R0           ;; SET TIMEOUT VALUE OF 30 SECONDS.
23     031102 010501      MOV     R5, R1             ;; GET POINTER TO CONTROLLER TABLE AND
24     031104 062701 000022      ADD     #C.TO, R1          ;; INDEX TO TIMEOUT COUNTER SO THE
25     031110 004737 034334      JSR    PC, SETTO          ;; TIME CAN BE SAVED IN THE TABLE.
26     031114 013705 002304      MOV     CTLRTBL, R5       ;; Load table start address
27     031120 011500      1$:      MOV     (R5), R0           ;; GET ADDRESS OF xDAIP REGISTER
28     031122 032765 000010 000014      BIT     #CT.MSG.C.FLG(R5) ;; LOOK IF INTERRUPT OCCURRED
29     031130 001031      BNE    3$                ;; BRANCH IF SO
30     031132 016037 000002 002316      MOV     2(R0), SAVAL      ;; LOOK AT xDASA REGISTER
31     031140 001037      BNE    4$                ;; BRANCH IF ERROR CODE PRESENT
32
33     031142 000401      BR     10$               ; No console interrupt
34
35     031144 104422      TRAP   C$BRK            ;>>>>>>>>>BREAK BACK TO MONITOR<<<<<<<<<<<<
36     031146      10$:
37     031146 005737 002210      TST    KW.CSR            ;SEE IF A CLOCK ON SYSTEM
38     031152 001762      BEQ    1$
39     031154 023765 002222 000024      CMP    KW.EL+2, C.TO(R5) ;; CHECK IF TIMEOUT HAS HAPPENED
40     031162 101005      BHI    2$
41     031164 001355      BNE    1$
42     031166 023765 002220 000022      CMP    KW.EL, C.TO(R5)
43     031174 103751      BLO    1$
44
45     ; Controller timeout
46     031176      2$:
47     031176 012601      MOV     (SP)+, R1         ;; POP STACK INTO R1
48     031200 012600      MOV     (SP)+, R0         ;; POP STACK INTO R0
49     031202 062716 000002      ADD     #2, (SP)          ; Bump past error return
50     031206 012700 000001      MCV    #1, R0            ; Set timeout status
51     031212 000207      RTS     PC                ; Return to caller
52
53     ; Normal exit
54     031214      3$:
55     031214 042765 000010 000014      BIC    #CT.MSG.C.FLG(R5) ; Clear message received flag
56     031222 012601      MOV     (SP)+, R1         ;; POP STACK INTO R1

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 91-1
WAIT Wait for xDA response

```
031224 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
56 031226 062716 000002  ADD      #2,(SP)      ; Bump past error return
57 031232 012700 000000  MOV      #0,R0        ; Set success status
58 031236 000207      RTS      PC           ; Return to caller
59
60
61 031240          ; Controller error
62 031240 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
    031242 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
63 031244 017616 000000  MOV      @0,(SP),(SP) ; Load error return
64 031250 012700 177777  MOV      #-1,R0       ; Set Fatal error status
65 031254 000207      RTS      PC           ; Return to caller
66
```

```

1      .SBTTL  CALRBN  Calculate RBN
2      ;CALRBN
3      ;
4      ;THIS ROUTINE CALCULATES A PHYSICAL RBN VALUE, FROM THE RCT LOGICAL BLOCK AND
5      ;OFFSET WITHIN THE RCT LOGICAL BLOCK.
6      ;
7      ;INPUTS:
8      ;      RCTBLK (RCT logical block),
9      ;      RCTOFF (RBN offset within RCT logical block)
10     ;
11     ;OUTPUTS:
12     ;      RBN (lo order word of RBN),
13     ;      RBN+2 (hi order word of RBN)
14     ;
15     ;Calling seq:
16     ;
17     ;      JSR      PC,CALRBN
18     ;
19
20 031256 013737 002476 002452 CALRBN: MOV      RCTBLK,RBN      ;GET RCT LOGICAL BLOCK NUMBER.
21 031264 162737 000002 002452      SUB      #2,RBN      ;CALCULATE THE ACTUAL RBN.
22 031272 012746 000200          MOV      #ENTRYS,-(SP) ;PUT THE MULTIPLIER ON THE STACK
    031276 013746 002452          MOV      RBN,-(SP)   ;PUT THE MULTIPLICAND ON THE STACK
    031302 004737 034542          JSR      PC,MULT    ;CALL THE MULTIPLY ROUTINE
    031306 012637 002452          MOV      (SP)+,RBN   ;GET THE LSB'S OF THE PRODUCT
    031312 012637 002454          MOV      (SP)+,RBN+2 ;GET THE MSB'S OF THE PRODUCT
23 031316 063737 002502 002452      ADD      RCTOFF,RBN  ;ADD THE RCT LOGICAL BLOCK OFFSET AND
24 031324 005537 002454          ADC      RBN+2     ;ADD THE CARRY BIT.
25 031330 000207          RTS      PC      ;RETURN
    
```

```

1      .SBTTL CALBLK Calculate RCT block
2      ;CALBLK
3      ;
4      ;THIS ROUTINE CALCULATES THE RCT LOGICAL BLOCK AND THE OFFSET OF THE RBN
5      ;WITHIN THAT LOGICAL BLOCK, FROM THE PHYSICAL RBN VALUE.
6      ;
7      ;SP = STACK POINTER
8      ;
9      ;INPUTS:
10     ;      SP+2 (Hi order word of RBN),
11     ;      SP (Lo order word of RBN)
12     ;
13     ;OUTPUTS:
14     ;      R0 (contents destroyed),
15     ;      SP+2 (RCT logical block of RBN),
16     ;      SP (RBN offset within the RCT logical block)
17     ;
18     ;Calling seq:
19     ;
20     ;      JSR      PC, -LBLK
21     ;
22
23     031332      CALBLK:
24     031332      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
25     031334      MOV      R4,-(SP)          ;;PUSH R4 ON STACK
26     031336      MOV      R5,-(SP)          ;;PUSH R5 ON STACK
27     031340      MOV      10(SP),R3        ;GET LO ORDER WORD AND
28     031344      MOV      12(SP),R4        ;HI ORDER WORD OF PRIMARY RBN.
29     031350      MOV      vENTRYS,R0      ;GET # OF ENTRIES (128.) PER RCT BLOCK
30     031354      JSR      PC,DIVIDE        ;DIVIDE TO FIND THE APPROPRIATE
31     031360      MOV      R3,12(SP)        ;RCT LOGICAL BLOCK AND
32     031364      MOV      R5,10(SP)        ;THE RBN OFFSET WITHIN THE BLOCK.
33     031370      MOV      (SP)+,R5        ;;POP STACK INTO R5
34     031372      MOV      (SP)+,R4        ;;POP STACK INTO R4
35     031374      MOV      (SP)+,R3        ;;POP STACK INTO R3
36     031376      RTS      PC              ;RETURN

```


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
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50

```

.SBTTL SEARCH RCT search
;SEARCH
;
;THIS ROUTINE IMPLEMENTS THE RCT SEARCH ALGORITHM. IT SEARCHES THE RCT
;TO FIND AN AVAILABLE RBN FOR REPLACEMENT.
;
;INPUTS:
; LBN (Lo order word of LBN being replaced),
; LBN+2 (Hi order word of LBN being replaced),
; TRKSIZ (number of LBN'S per track),
; RBNS (number of REPLACEMENT BLOCKS per track),
;
;OUTPUTS:
; SFLAG (Search status flag),
; MATFLG (Match status flag),
; RBN (Lo order word of RBN replacement),
; RBN+2 (Hi order word of RBN replacement),
; MATRBN (Lo order word of RBN which matched the LBN being replaced),
; MATRBN+2 (Hi order word of RBN which matched the LBN being replaced)
;
;Calling seq:
;
; JSR PC,SEARCH
; BMI <error exit>
;
SEARCH:
CLRBLK MATFLG ;INIT THE MATCH FLAG,
CLRBLK EMPTY ;EMPTY STATUS FLAG AND
CLRBLK RESCAN ;RE-SCAN FLAG.
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV LBN,R3 ;GET LO WORD AND
MOV LBN+2,R4 ;HI WORD OF LBN BEING REPLACED.
MOV TRKSIZ,R0 ;GET THE TRACK SIZE,
JSR PC,DIVIDE ; Divide to find LBN track
MOVB RBNS,R0 ;GET NUMBER OF RBNS PER TRACK.
MOV R0,-(SP) ;PUT THE MULTIPLIER ON THE STACK
MOV R3,-(SP) ;PUT THE MULTIPLICAND ON THE STACK
JSR PC,MULT ;CALL THE MULTIPLY ROUTINE
JSR PC,CALBLK ;CALCULATE THE RCT LOGICAL BLOCK AND
;THE RBN OFFSET WITHIN THE BLOCK.
MOV (SP)+,STROFF ;;POP STACK INTO STROFF
MOV (SP)+,RCTBLK ;;POP STACK INTO RCTBLK
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R4 ;;POP STACK INTO R4
MOV (SP)+,R3 ;;POP STACK INTO R3
44 031500 062737 000002 002476 1$: ADD #2,RCTBLK ;OFFSET TO THE BEGINNING OF THE RCT
;DESCRIPTOR BLOCKS.
46 031506 004737 030074 2$: JSR PC,INITRW ;SOME READ/WRITE PARAMETERS
47 031512 063764 002476 000140 ADD RCTBLK,HC.CPK+P.LBN(R4) ;ADD THE RCT LOGICAL BLOCK TO STARTING
48 031520 005564 000142 ADC HC.CPK+P.LBN+2(R4) ;HOST LBN OF THE RCT AREA.
; Setup to issue MULTRD command
49 031524 004737 027224 JSR PC,MULTRD ; Issue command to the xDA
031530 032006 10$ ; Here if fatal error.
50 031532 3$:
    
```

SEARCH RCT search

51	031532	005037	002510			CLR	DELTA	;INIT OFFSET TO PRIMARY RBN
52	031536			4\$:				
53	031536	013701	002340			MOV	UADR,R1	;GET THE BEGINNING OF UNIBUS ADDRESS
54	031542	013700	002506			MOV	STROFF,RO	;GET STARTING OFFSET IN RCT BLOCK,
55	031546	063700	002510			ADD	DELTA,RO	;SEE IF OFFSET EXCEEDED LO BLOCK LIMIT.
56	031552	010037	002502			MOV	RO,RCTOFF	;STORE IT FOR LATER.
57	031556	002475				BLT	8\$;BRANCH IF SO, ELSE
58	031560	020027	000200			CMP	RO,#ENTRYS	;SEE IF OFFSET EXCEEDED HI BLOCK LIMIT.
59	031564	002072				BGE	8\$;BRANCH IF SO, ELSE
60	031566	006300				ASL	RO	;MULTPLY OFFSET
61	031570	006300				ASL	RO	;BY 4
62	031572	060001				ADD	RO,R1	; Find RBN position in descriptor table
63	031574	016100	000002			MOV	2(R1),RO	;GET HI ORDER LBN IN RBN DESCRIPTOR,
64	031600	042700	007777			BIC	#C<170000>,RO	;SEE IF THE RBN IS EMPTY OR A NULL.
65	031604	001006				BNE	5\$;BRANCH IF NOT EMPTY, ELSE
66	031606	113737	002512	002515		MOVB	EMPTY,SFLAG	;SET PRIMARY EMPTY SEARCH STATUS AND
67	031614	004737	031256			JSR	PC,CALRBN	;CALCULATE THE RBN.
68	031620	000475				BR	12\$	
69								
70	031622	042761	170000	000002	5\$:	BIC	#170000,2(R1)	;GET RID OF RBN REPLACEMENT CODE AND
71	031630	112737	000001	002512		MOVB	#1,EMPTY	;SET NOT PRIMARY EMPTY SEARCH STATUS.
72	031636	032700	020000			BIT	#BIT13,RO	;PRIMARY OR SECONDARY REPLACEMENT ?
73	031642	001422				BEQ	6\$;BRANCH IF NO, ELSE
74								;SEE IF LBN BEING REPLACED MATCHES
75	031644	021137	002446			CMP	(R1),LBN	;THE LBN IN THE LO WORD RBN DESCRIPTOR,
76	031650	001017				BNE	6\$;BRANCH IF NOT, ELSE
77	031652	026137	000002	002450		CMP	2(R1),LBN+2	;THE LBN IN THE HI WORD RBN DESCRIPTOR,
78	031660	001013				BNE	6\$;BRANCH IF NOT, ELSE
79	031662	112737	000001	002514		MOVB	#1,MATFLG	;SET THE MATCHED LBN FLAG.
80	031670	004737	031256			JSR	PC,CALRBN	;CALCULATE THE PHYSICAL RBN
81	031674	013737	002452	002462		MOV	RBN,MATRBN	;STORE MATCHED RBN LO ORDER WORD AND
82	031702	013737	002454	002464		MOV	RBN+2,MATRBN+2	;RBN HI ORDER WORD.
83								
84	031710	005700			6\$:	TST	RO	;IS THIS A NULL REPLACEMENT ?
85	031712	100017				BPL	8\$;BRANCH IF NO, ELSE
86	031714	105737	002513			TSTB	RESCAN	;SEE IF THIS IS A RE-SCAN OF THE RCT BLOCKS.
87	031720	001404				BEQ	7\$;BRANCH IF NOT, ELSE
88	031722	112737	000002	002515		MOVB	#2,SFLAG	;SET RCT FULL FLAG IN SEARCH STATUS AND
89	031730	000431				BR	12\$;EXIT.
90								
91	031732	112737	000001	002513	7\$:	MOVB	#1,RESCAN	;SET SEQUENTIAL RE-SCAN FLAG AND
92	031740	005037	002506			CLR	STROFF	;START SEQUENTIAL SEARCH THRU THE
93	031744	005037	002476			CLR	RCTBLK	;FIRST RCT LOGICAL BLOCK.
94	031750	000653				BR	1\$	
95								
96	031752	005437	002510		8\$:	NEG	DELTA	;SEE WHICH DIRECTION SEARCH IS GOING.
97	031756	100402				BMI	9\$;BRANCH IF REVERSE, ELSE
98	031760	005237	002510			INC	DELTA	;INCREMENT TO NEXT RBN
99								;DESCRIPTOR FORWARD.
100	031764	023727	002510	000200	9\$:	CMP	DELTA,#ENTRYS	;SEE IF STILL IN SAME RCT BLOCK.
101	031772	002661				BLT	4\$;BRANCH IF SO, ELSE
102	031774	005037	002506			CLR	STROFF	;START SEQUENTIAL SEARCH THRU THE
103	032000	005237	002476			INC	RCTBLK	;NEXT RCT LOGICAL BLOCK.
104	032004	000640				BR	2\$	
105	032006				10\$:			
106	032006	012700	177777			MOV	#-1,RO	; Set fatal error
107	032012				11\$:			

108	032012	000207		RTS	PC	; Return to caller
109	032014		124:			
110	032014	012700	000000	MOV	#0,R0	; Load success
111	032020	000207		RTS	PC	; and return

```

1      .SBTTL  GETVSN  Get volume Serial Number
2      ;GETVSN
3      ;
4      ;Retrieve and format volume serial number
5      ;
6      ;Inputs:
7      ;      UADR:  Address of buffer containing RCT block 0 image
8      ;      VSNBUF: Twenty-four byte ASCII buffer
9      ;
10     ;Outputs:
11     ;      VNSPTR: Address of .ASCIZ string containing VSN
12     ;
13     ;Externals:
14     ;      DIV10: Divide 4 byte positive binary number by 10.
15     ;
16     ;Calling seq:
17     ;
18     ;      JSR    PC.GETVSN
19     ;
20
21     GETVSN:
22     032022 010146      MOV    R1,-(SP)          ;; PUSH R1 ON STACK
23     032024 010246      MOV    R2,-(SP)          ;; PUSH R2 ON STACK
24     032026 010346      MOV    R3,-(SP)          ;; PUSH R3 ON STACK
25     032030 010446      MOV    R4,-(SP)          ;; PUSH R4 ON STACK
26     032032 010546      MOV    R5,-(SP)          ;; PUSH R5 ON STACK
27     032034 013700 002340  MOV    UADR,R0          ;GET THE BEGINNING OF UNIBUS ADDRESS
28     032040 012001      MOV    (R0)+,R1         ;PUT WORD 0,
29     032042 012002      MOV    (R0)+,R2         ;WORD 1,
30     032044 012003      MOV    (R0)+,R3         ;WORD 2 AND
31     032046 012004      MOV    (R0)+,R4         ;WORD 3 IN REGISTERS.
32     032050 012700 002620  MOV    #VSNBUF,R0      ; Load address of VSN text
33     032054 012705 000024  MOV    #20.,R5         ;GET NUMBER OF CHARACTERS
34
35     3$:
36     032060 112720 000040  MOVB   #' ,(R0)+       ;INITIALIZE STORAGE WITH SPACES
37     032064 005305      DEC    R5              ;DONE AREA YET ?
38     032066 001374      BNE   3$              ;BR IF NO
39
40     4$:
41     032070 005010      CLR   (R0)            ;INSERT NULL AS TERMINATOR
42
43     032072 004737 034474  JSR   PC,DIV10        ;CONVERT OCTAL TO DECIMAL AND
44     032076 062705 000060  ADD   #'0,R5          ;MAKE IT ASCIZ.
45
46     032102 110540      MOVE  R5,-(R0)
47     032104 010146      MOV   R1,-(SP)
48     032106 050216      BIS  R2,(SP)
49     032110 050316      BIS  R3,(SP)
50     032112 050426      BIS  R4,(SP)+
51     032114 001366      BNE  4$              ;BRANCH IF NOT DONE
52
53     032116 010037 002616  MOV   R0,VNSPTR      ; Save VSN pointer
54     032122 012605      MOV  (SP)+,R5        ;; POP STACK INTO R5
55     032124 012604      MOV  (SP)+,R4        ;; POP STACK INTO R4
56     032126 012603      MOV  (SP)+,R3        ;; POP STACK INTO R3
57     032130 012602      MOV  (SP)+,R2        ;; POP STACK INTO R2
58     032132 012601      MOV  (SP)+,R1        ;; POP STACK INTO R1
59     032134 000207      RTS   PC            ; Return to caller
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

CZUOLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 96
GTDRVT Get drive table pointer

```
1          SBTTL GTDRVT Get drive table pointer
2          ;GTDRVT
3          ;
4          ;GET DRIVE TABLE POINTER
5          ;
6          ;INPUTS:
7          ;      R1 - DRIVE NUMBER
8          ;      R5 - CONTROLLER TABLE ADDRESS
9          ;OUTPUTS:
10         ;      R4 - DRIVE TABLE ADDRESS
11         ;
12         ;
13 032136 010504          GTDRVT: MOV      R5,R4          ;GET CONTROLLER TABLE ADRESS
14 032140 062704 000020  ADD      #C.DRVT,R4      ;ADD OFFSET TO DRIVE TABLE ADDRESS
15 032144 011404          MOV      (R4),R4      ;GET ADDRESS OF TABLE
16 032146 000207          RTS      PC
17
```

```

1      .SBTTL  DSPHELP Display operator help
2      ;
3      ;
4      ;      SUBROUTINE TO DUMP DOS HELP FILE TO CONSOLE
5      ;
6      ;      INPUT:          HLPFILE:  .ASCIZ "FILNAM.EXT"
7      ;
8      ;      OUTPUT:         PRINTS 72 CHARACTER BUFFER TO CONSOLE
9      ;
10     ;      REGISTER USAGE: R1 =  POINTER TO LINE TEXT BUFFER
11     ;                          R2 =  TAB POSITION COUNTER
12     ;                          R4 =  CHARACTER FROM FILE
13     ;
14     ;      CALLING SEQ:    JSR      PC,DPSHELP
15     ;
16     ;      FUNCTION:      DSPHELP WILL OPEN, READ AND CLOSE THE SELETED FILE
17     ;                      USING THE DIAG SUPERVISOR MACROS.  CHARACTERS
18     ;                      ARE READ ONE AT A TIME AND TRANSFERRED TO THE
19     ;                      LINE BUFFER, WHICH IS DUMPED TO THE CONSOLE
20     ;                      WHEN FULL.
21     ;
22     032150      DSPHELP:
23     ;
24     032150      010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
25     032152      010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
26     032154      010446      MOV      R4,-(SP)          ;;PUSH R4 ON STACK
27     ;
28     ;      OPEN FILE AND PRINT NULL LINE
29     ;100$:
30     032156      012700      014672      MOV      #HLPFILE,R0
31     032162      104434      TRAP     C$OPNRD
32     032164      012701      002716      MOV      #LINBUF,R1          ;AND LOAD POINTER TO BUFFER
33     032170      112721      000015      MOVB     #CR,(R1)+          ;MOVE CARRAGE RETURN
34     032174      112721      000012      MOVB     #LF,(R1)+          ;AND LINE FEED TO BUFFER
35     032200      112704      000000      MOVB     #0,R4              ;FOLLOWED BY NULL
36     032204      004737      032270      JSR      PC,600$           ;PRINT BUFFER, RESET R1
37     ;
38     ;      GET NEXT CHARACTER FROM SUPERVISOR, AND TEST FOR END OF FILE
39     ;200$:
40     032210      104426      TRAP     C$GETB
41     032212      010004      MOV      R0,R4
42     032214      103004      BCC      400$
43     032216      001405      BEQ      425$              ;OR IF CHARACTER = ZERO
44     ;
45     ;      TEST AND PROCESS TAB CHARACTER
46     ;300$:
47     032220      004737      032270      JSR      PC,600$           ;PRINT SPACES FOR TAB
48     032224      000771      BR       200$             ;AND RETRIEVE NEXT CHARACTER
49     ;
50     ;      LAST CHAR OR END OF FILE
51     ;400$:
52     032226      112704      000000      MOVB     #0,R4              ;INSURE NULL CHAR
53     ;425$:
54     032232      004737      032270      JSR      PC,600$           ;EMPTY
55     032236      112721      000015      MOVB     #CR,(R1)+          ;OUTPUT CARRAGE RETURN
56     032242      112721      000012      MOVB     #LF,(R1)+          ;AND LINE FEED
57     032246      112704      000000      MOVB     #0,R4              ;FOLLOWED BY NULL

```

```

54 032252 004737 032270      JSR      PC,600$          ;PRINT
55
56          ;              CLOSE FILE AND RETURN TO MAIN
57          ;500$:
58 032256 104435      TRAP     C$CLOS
59 032260 012604      MOV      (SP)+,R4          ;;POP STACK INTO R4
      032262 012602      MOV      (SP)+,R2          ;;POP STACK INTO R2
      032264 012601      MOV      (SP)+,R1          ;;POP STACK INTO R1
60 032266 000207      RTS      PC              ;AND RETURN
61
62          ;              MOVE CHAR TO BUFFER, TEST FOR NULL CHAR OR END OF BUFFER
63 032270          ;600$:
64 032270 001415      BEQ      800$            ; Branch if null
65
66          ;              TEST FOR CONTROL CHARS, TAB ACTIVE OR RETURN
67          ;700$:
68 032272 120427 000040      CMPB     R4,#40          ;COMPARE CHAR WITH BLANK
69 032276 100010      BPL      750$          ;BRANCH IF CONTROL CHAR
70 032300 120427 000015      CMPB     R4,#15         ; Test for CR
71 032304 001006      BNE      780$          ; If not, ignore
72
73 032306 112721 000015      MOVB     #15,(R1)+      ; Else, load CR
74 032312 112721 000012      MOVB     #12,(R1)+      ; and LF
75 032316 000402      BR       800$          ; and print
76 032320          ;750$:
77 032320 110421      MOVB     R4,(R1)+      ;MOVE CHAR TO BUFFER
78 032322          ;780$:
79 032322 000207      RTS      PC              ;NO: JUST RETURN
80
81          ;              RESET TAB COUNTER, TAB ACTIVE FLAG AND RETURN
82          ;800$:
83 032324 112721 000000      MOVB     #0,(R1)+      ;MOVE CHAR TO BUFFER
84 032330 012701 002716      MOV      #LINBUF,R1    ;AND RESET POINTER
85 032334 010146      MOV      R1,-(SP)
      032336 012746 014220      MOV      #F.TEXT,-(SP)
      032342 012746 000002      MOV      #2,-(SP)
      032346 010600      MOV      SP,R0
      032350 104414      TRAP     C$PNTB
      032352 062706 000006      ADD      #6,SP
86 032356 012701 002716      MOV      #LINBUF,R1    ;AND RESET POINTER
87 032362 000207      RTS      PC              ;AND RETURN
88

```

```

1      .SBTTL DSPDSC Display RCT descriptors
2      ;DSPDSC
3      ;
4      ;Read and display RBN descriptors in RCT.
5      ;
6      ;Inputs:
7      ;      DBUFF Buffer for RCT blocks.
8      ;Outputs:
9      ;      Outputs replacement descriptors on console.
10     ;
11     ;Calling seq:
12     ;
13     ;      JSR PC,DEPDSC
14     ;
15
16
17     DSPDSC:
18     032364      013705 002304      MOV CTLRTBL,R5      ; Load table start address
19     032370      013704 002324      MOV HCOM,R4         ; Load host comm area address
20                                     ; "RCT descriptor info..."
21     032374      013746 002074      MOV L$LUN,-(SP)     ; PUSH L$LUN ON STACK
22     032400      012746 006170      MOV $INF210,-(SP)  ; PUSH $INF210 ON STACK
23     032404      004137 033254      JSR R1,LPNTB       ; CALL LPNTB PRINT ROUTINE
24     032410      000004      .WORD ARG.CT      ; ARGUMENT COUNT * 2
25
26     23 032412      004737 030626      JSR PC,INTCPK     ; INITIALIZE THE COMMAND PACKET,
27     24 032416      005037 002434      CLR PRIMRY        ; THE PRIMARY RBN COUNT,
28     25 032422      005037 002436      CLR SCNDRY        ; THE SECONDARY RBN COUNT,
29     26 032426      005037 002440      CLR UNUSE         ; THE UNUSABLE RBN COUNT,
30     27 032432      012737 042472 002340  MOV $DBUFF,UADR   ; GET BEGINNING ADDRESS OF DATA BUFFER
31     28 032440      012737 000001 002476  MOV $1,RCTBLK    ; THE RCT LOGICAL BLOCK TO 1 AND
32     29 032446
33     30 032446      005037 002502      CLR RCTOFF        ; THE OFFSET WITHIN THE RCT BLOCK.
34     31 032452      005237 002476      INC RCTBLK        ; INCREMENT RCT LOGICAL BLOCK AND
35     32 032456      023737 002476 002416  CMP RCTBLK,RCTSIZ ; SEE IF THE ENTIRE RCT WAS READ,
36     33 032464      002411      BLT 2$           ; BRANCH IF NOT, ELSE
37
38     35 032466      012737 013242 002552  MOV $ERM602,BASEMSG ; "No null entry..."
39     36 032474      104457      TRAP C$ERSOFT
40     032476      001132      .WORD 602
41     032500      013173      .WORD HDR601
42     032502      015220      .WORD ERRO01
43     37 032504      000137 033054      JMP 100$         ; BRANCH TO EXIT ON ERROR
44
45     38 032510
46     39 032510      004737 030074      JSR PC,INITRW    ; INIT SOME READ/WRITE PARAMETERS
47     40 032514      063764 002476 000140  ADD RCTBLK,HC.CPK+P.LBN(R4) ; ADD RCT LOGICAL BLOCK TO STARTING
48     41 032522      005564 000142      ADC HC.CPK+P.LBN+2(R4) ; HOST LBN OF THE RCT AREA.
49     42
50     032526      004737 027224      JSR PC,MULTRD    ; Setup to issue MULTRD command
51     032532      033054      100$           ; Issue command to the xDA
52                                     ; Here if fatal error.
53
54     43 032534
55     44 032534      013701 002340      MOV UADR,R1      ; GET THE BEGINNING OF UNIBUS ADDRESS
56     45 032540
57     46 032540      004737 031256      JSR PC,CALRBN    ; CALCULATE THE PHYSICAL RBN
58     47 032544      016103 000002      MOV 2(R1),R3     ; GET HI ORDER LBN IN RBN DESCRIPTOR.
59     48 032550      042703 007777      BIC $+C<170000>,R3 ; SEE IF THE RBN IS EMPTY OR A NULL.
60     49 032554      001521      BEQ 9$          ; BRANCH IF EMPTY,

```



```

50 032556 100033          BPL      6‡          ;BRANCH IF NOT NULL, ELSE
51                                  ;THIS IS MUST BE THE END, SO
52                                  ; "RCT descriptor summary..."
53 032560 013701 002434    MOV      PRIMARY,R1      ; Load primary
54 032564 063701 002436    ADD      SCNDRY,R1       ; Add secondary
55 032570 063701 002440    ADD      UNUSE,R1       ; Add unusable
56 032574 012746 006452    MOV      #INF220,-(SP)   ;PUSH #INF220 ON STACK
    032600 004137 033254    JSR      R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    032604 000002          .WORD    ARG.CT        ;ARGUMENT COUNT * 2
57 032606 013746 002422    MOV      RBNS,-(SP)     ;PUSH RBNS ON STACK
    032612 010146          MOV      R1,-(SP)       ;PUSH R1 ON STACK
    032614 013746 002440    MOV      UNUSE,-(SP)    ;PUSH UNUSE ON STACK
    032620 013746 002436    MOV      SCNDRY,-(SP)   ;PUSH SCNDRY ON STACK
    032624 013746 002434    MOV      PRIMARY,-(SP) ;PUSH PRIMARY ON STACK
    032630 012746 006523    MOV      #INF221,-(SP) ;PUSH #INF221 ON STACK
    032634 004137 033254    JSR      R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    032640 000014          .WORD    ARG.CT        ;ARGUMENT COUNT * 2
58
59 032642 000137 033042    JMP      90‡          ;BRANCH TO EXIT
60
61 032646 042761 170000 000002 6‡:  BIC      #170000,2(R1)  ;GET RID OF THE REPLACEMENT BLOCK CODE.
62                                  ;THE LBN CAN BE PRINTED LATER.
63 032654 020327 020000    CMP      R3,#BIT13      ;IS THIS A PRIMARY REPLACEMENT ?
64 032660 001017          BNE      7‡          ;BRANCH IF NO, ELSE
65 032662 005237 002434    INC      PRIMRY         ;UPDATE # OF PRIMARY REPLACEMENT BLOCKS
66                                  ; "RBN "D32". Primary replacement..."
67 032666 016146 000002    MOV      2(R1),-(SP)    ;PUSH 2(R1) ON STACK
    032672 011146          MOV      (R1),-(SP)    ;PUSH (R1) ON STACK
    032674 013746 002454    MOV      RBN+2,-(SP)   ;PUSH RBN+2 ON STACK
    032700 013746 002452    MOV      RBN,-(SP)     ;PUSH RBN ON STACK
    032704 012746 006244    MOV      #INF211,-(SP) ;PUSH #INF211 ON STACK
    032710 004137 033254    JSR      R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    032714 000012          .WORD    ARG.CT        ;ARGUMENT COUNT * 2
68 032716 000440          BR      9‡
69
70 032720 020327 030000    7‡:    CMP      R3,#BIT13!BIT12 ;IS THIS A SECONDARY REPLACEMENT ?
71 032724 001017          BNE      8‡          ;BRANCH IF NO, ELSE
72 032726 005237 002436    INC      SCNDRY        ;UPDATE # OF SECONDARY
73                                  ;REPLACEMENT BLOCKS
74                                  ; "RBN "D32". Secondary replacement..."
75 032732 016146 000002    MOV      2(R1),-(SP)    ;PUSH 2(R1) ON STACK
    032736 011146          MOV      (R1),-(SP)    ;PUSH (R1) ON STACK
    032740 013746 002454    MOV      RBN+2,-(SP)   ;PUSH RBN+2 ON STACK
    032744 013746 002452    MOV      RBN,-(SP)     ;PUSH RBN ON STACK
    032750 012746 006332    MOV      #INF212,-(SP) ;PUSH #INF212 ON STACK
    032754 004137 033254    JSR      R1,LPNTB       ;CALL LPNTB PRINT ROUTINE
    032760 000012          .WORD    ARG.CT        ;ARGUMENT COUNT * 2
76 032762 000416          BR      9‡
77
78 032764 020327 040000    8‡:    CMP      R3,#BIT14      ;IS THIS AN UNUSABLE REPLACEMENT ?
79 032770 103413          BLO      9‡          ;BRANCH IF NO, ELSE
80 032772 005237 002440    INC      UNUSE         ;UPDATE # OF UNUSABLE REPLACEMENT BLOCKS
81                                  ; "RBN "D32". Unusable"
82 032776 013746 002454    MOV      RBN+2,-(SP)   ;PUSH RBN+2 ON STACK
    033002 013746 002452    MOV      RBN,-(SP)     ;PUSH RBN ON STACK
    033006 012746 006420    MOV      #INF213,-(SP) ;PUSH #INF213 ON STACK
    033012 004137 033254    JSR      R1,LPNTB       ;CALL LPNTB PRINT ROUTINE

```

	033016	000006		.WORD	ARG.CT		;ARGUMENT COUNT * 2
83							
84	033020	022121					
85	033022	005237	002502	9\$:	CMP (R1)+,(R1)+		;UPDATE POINTER IN THE TABLE
86	033026	023727	002502		INC RCTOFF		;INCREMENT TO NEXT RBN DESCRIPTOR AND
87	033034	002641	000200		CMP RCTOFF,#ENTRYS		;SEE IF ALL ENTRIES DONE,
88	033036	000137	032446		BLT 5\$;BRANCH IF NOT, ELSE
89					JMP 1\$;CONTINUE TO READ NEXT BLOCK OF
90	033042			90\$:			;REPLACEMENT DESCRIPTORS.
91	033042	062716	000002		AD^ #2,(SP)		; Bump past error return
92	033046	012700	000000		MOV #0,R0		; Status is success
93	033052	000207			RTS PC		; Return to caller
94	033054			100\$:			
95	033054	017616	000000		MOV #8(SP),(SP)		; Load error return
96	033060	012700	177777		MOV #-1,R0		; Status is failure
97	033064	000207			RTS PC		; Return to caller
98							

PRINTC Print character

```

1      .SBTTL PRINTC Print character
2      ;PRINTC
3      ;
4      ;PRINT A CHARACTER
5      ;
6      ;CALL WITH MACRO PRINT
7
8 033066 110037 033242 PRINTC: MOVB    R0,TTYOUT      ;SAVE CHARACTER FOR TTY OUTPUT
9 033072 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
10 033074 012701 014220  MOV     #F.TEXT,R1      ;PICKUP FORMATTED ASCIZ STRING STATEMENT
11 033100 120027 000015  CMPB   RO,#CR          ;IF NOT A CARRIAGE RETURN, THEN
12 033104 001002          BNE     1$              ;PRINT SOME OTHER CHARACTER, ELSE
13 033106 012701 014215  MOV     #F.CRLF,R1      ;PICKUP FORMATTED ASCIZ STRING STATEMENT
14                                     ;GO PRINT CR-LF.
15 033112 004777 000232  1$:   JSR     PC,@PTYPE      ;PRINT THE ASCIZ STRING.
16 033116 012601          MOV     (SP)+,R1        ;;POP STACK INTO R1
17 033120 000207          RTS     PC
18 033122          PF:
   033122 012746 033242  MOV     #TTYOUT,-(SP)
   033126 010146          MOV     R1,-(SP)
   033130 012746 000002  MOV     #2,-(SP)
   033134 010600          MOV     SP,R0
   033136 104417          TRAP   C#PNTF
   033140 062706 000006  ADD     #6,SP
19 033144 000207          RTS     PC
20 033146          PB:
   033146 012746 033242  MOV     #TTYOUT,-(SP)
   033152 010146          MOV     R1,-(SP)
   033154 012746 000002  MOV     #2,-(SP)
   033160 010600          MOV     SP,R0
   033162 104414          TRAP   C#PNTB
   033164 062706 000006  ADD     #6,SP
21 033170 000207          RTS     PC
22 033172          PX:
   033172 012746 033242  MOV     #TTYOUT,-(SP)
   033176 010146          MOV     R1,-(SP)
   033200 012746 000002  MOV     #2,-(SP)
   033204 010600          MOV     SP,R0
   033206 104415          TRAP   C#PNTX
   033210 062706 000006  ADD     #6,SP
23 033214 000207          RTS     PC
24 033216          PS:
   033216 012746 033242  MOV     #TTYOUT,-(SP)
   033222 010146          MOV     R1,-(SP)
   033224 012746 000002  MOV     #2,-(SP)
   033230 010600          MOV     SP,R0
   033232 104416          TRAP   C#PNTS
   033234 062706 000006  ADD     #6,SP
25 033240 000207          RTS     PC
26
27 033242    000          TTYOUT: .BYTE 0      ;TTY OUTPUT BUFFER
28 033243    000          .BYTE 0      ;TERMINATOR FOR ASCIZ STRING
29          .EVEN

```

```

1          .SBTTL LPNTx Print formatted message
2          ;PRINT FORMATTED MESSAGE
3          ;
4          ;CALL WITH MACRO PNT, PNTF, PNTB, PNTX, OR PNTS
5
6 033244 012737 033122 033350 LPNTF: MOV  #PF,PTYPE
7 033252 000413                BR   LPNT
8
9 033254 012737 033146 033350 LPNTB: MOV  #PB,PTYPE
10 033262 000407                BR   LPNT
11
12 033264 012737 033172 033350 LPNTX: MOV  #PX,PTYPE
13 033272 000403                BR   LPNT
14
15 033274 012737 033216 033350 LPNTS: MOV  #PS,PTYPE
16
17 033302                LPNT:
18 033302 010246                MOV   R2,-(SP)           ;;PUSH R2 ON STACK
19 033304 010346                MOV   R3,-(SP)           ;;PUSH R3 ON STACK
20 033306 010446                MOV   R4,-(SP)           ;;PUSH R4 ON STACK
21 033310 010546                MOV   R5,-(SP)           ;;PUSH R5 ON STACK
22 033312 010604                ;   MOV   (R1)+,R2       ;GET ADDRESS OF ASCIZ STRING
23 033314 062704 000012        ;   MOV   SP,R4         ;COMPUTE ADDRESS OF 1ST ARGUMENT AND
24 033316 012402                ;   ADD   #12,R4        ;SAVE IT IN R4.
25 033318 012402                ;   MOV   (R4)+,R2     ; Load address of ASCII string
26 033320 010146                ;   MOV   R1,-(SP)     ;;PUSH R1 ON STACK
27 033322 004737 033352        ;   JSR   PC,OSTRNG    ;PRINT THE FORMATTED MESSAGE
28 033324 012600                ;   MOV   (SP)+,R0     ;;POP STACK INTO R0
29 033326 012605                ;   MOV   (SP)+,R5     ;;POP STACK INTO R5
30 033328 012604                ;   MOV   (SP)+,R4     ;;POP STACK INTO R4
31 033330 012603                ;   MOV   (SP)+,R3     ;;POP STACK INTO R3
32 033332 012602                ;   MOV   (SP)+,R2     ;;POP STACK INTO R2
33 033334 012601                ;   MOV   (SP)+,R1     ;;POP STACK INTO R1
34 033336 062006                ;   ADD   (R0)+,SP     ;ADJUST STACK POINTER OVER ARGUMENTS
35 033338 000110                ;   JMP   @R0          ;RETURN
36
37 033350 033122                PTYPE: .WORD  PF           ;PRINT TYPE
    
```

```

1      .SBTTL OSTRNG Formatted message output
2      ;OSTRNG
3      ;
4      ;OUTPUT A MESSAGE ACCORDING TO A FORMAT STRING
5      ;FORMAT OF THE ASCIZ STRING IS AS FOLLOWS:
6      ;
7      ;CHARACTERS ENCLOSED IN QUOTES ARE TO BE PRINTED AS THEY ARE.
8      ;
9      ;OTHERWISE CODE IS A SINGLE LETTER FOLLOWED BY AN OPTIONAL DECIMAL
10     ;NUMBER:
11     ; On - PRINT OCTAL NUMBER. n REPRESENTS SIZE OF BINARY NUMBER PASSED
12     ;       IN PARAMETER IN BITS. MAY BE IN RANGE 1 TO 32. IF n>16, TWO PARAMETER
13     ;       WORDS ARE USED, OTHERWISE ONLY ONE WORD. LEADING ZEROS ARE PRINTED.
14     ;       n IS ALWAYS SPECIFIED.
15     ; Dn - PRINT UNSIGNED DECIMAL NUMBER FROM n BIT PARAMETER. LEADING ZEROS
16     ;       ARE NOT PRINTED. A 16 BIT NUMBER EQUAL TO ZERO WILL PRINT "0".
17     ; Hn - PRINT HEX NUMBER FROM PARAMETER OF n BITS. IF n>16 TWO PARAMETERS
18     ;       ARE USED, OTHERWISE ONLY ONE PARAMETER. LEADING ZEROS ARE PRINTED.
19     ; Sn - PRINT n SPACES. n ASSUMED TO BE 1.
20     ; Nn - START NEW LINE (CR-LF SEQUENCE). n ASSUMED TO BE 1.
21     ; An - PRINT N ASCII CHARACTERS FROM PARAMETERS, n ASSUMED TO BE 1.
22     ;       n/2 PARAMETER WORDS USED.
23     ; Rn - EXECUTE ROUTINE @n. n MUST BE GIVEN AND DEFINED IN HOST PROGRAM.
24     ;
25     ;A NULL CHARACTER MEANS END OF MESSAGE. A NULL AS FIRST CHARACTER IN STRING
26     ;MUST BE IGNORED.
27     ;
28     ;INPUTS:
29     ; R2 - ADDRESS OF START OF FORMAT STRING
30     ; R4 - ADDRESS OF PARAMETERS
31     ;OUTPUTS:
32     ; R2 AND R4 UPDATED TO END OF STRING AND PARAMETERS
36
37 033352 112201 OSTRNG: MOVB (R2)+,R1 ;SEE IF TERMINATOR IN ASCIZ STRING.
38 033354 001422 BEQ OSTRE ;EXIT
39 033356 012700 033632 MOV @ERRC,R0 ;GET POINTER TO CHARACTER TABLE
40 033362 120110 NCONS: CMPB R1,(R0) ;COMPARE CHARACTER WITH TABLE ENTRY
41 033364 001410 BEQ NCONF ;BRANCH IF MATCH FOUND
42 033366 105720 TSTB (R0)+ ;INCREMENT POINTER
43 033370 001374 BNE NCONS ;CONTINUE SEARCH IF NOT END OF TABLE
44 033372 012746 014427 MOV @ERRME1,-(SP) ;PUSH @ERRME1 ON STACK
    033376 001157 033244 JSR R1,LPNTF ;CALL LPNTF PRINT ROUTINE
    033402 000002 .WORD ARG.CT ;ARGUMENT COUNT * 2
45 033404 000406 BR OSTRE
46 033406 162700 03363? NCONF: SUB @ERRC,R0 ;GET INCREMENT INTO TABLE
47 033410 006300 ASL R0 ;DOUBLE TO WORD COUNT
48 033414 004770 033644 JSR PC,@ERRD(R0) ;DISPATCH TO PRINT ROUTINE
49 033420 000754 BR OSTRNG ;GET NEXT
50 033422 000207 OSTRNG: RTS PC
    
```

```
1
2
3           ;CONTROL CHARACTER WAS A QUOTE, SO PRINT ALL CHARACTERS TO
4           ;THE NEXT QUOTE.
5 033424 112200          CON.QU: MOVB   (R2)+,R0          ;GET CHARACTER
6 033426 120027 000042  CMPB   RO,#'"          ;CHECK IF ENDING QUOTE
7 033432 001403          BEQ    CON.QX          ;IF SO, GO GET NEXT CONTROL CHARACTER
8 033434 004737 033066  JSR    PC,PRINTC        ;PRINT THE CHARACTER.
9 033440 000771          BR     CON.QU          ;CONTINUE PRINTING
10 033442 000207        CON.QX: RTS    PC
11
12           ;CONTROL CHARACTER WAS AN 'A', SO PRINT ASCII CHARACTERS FROM
13           ;PARAMETERS.
14
15 033444 004737 033664  CON.A:  JSR    PC,GETCNT        ;GET COUNT OF CHARACTERS
16 033450 012401          MOV    (R4)+,R1          ; GET STRING ADDRESS
17 033452 105711        CON.A1: TSTB   (R1)          ; TEST FOR NULL
18 033454 001404          BEQ    CON.A2          ; BRANCH IF YES
19 033456 112100          MOVB   (R1)+,R0          ;STORE (R1)+ IN R0 AND
20 033460 004737 033066  JSR    PC,PRINTC        ;PRINT THE CHARACTER.
21 033464 000772          BR     CON.A1          ; CONTINUE
22 033466 000207        CON.A2: RTS    PC          ;NOW GET NEXT CONTROL CHARACTER
23
24           ;CONTROL CHARACTER WAS A 'D', SO PRINT A DECIMAL NUMBER.
25 033470 012701 000012  CON.D:  MOV    #10.,R1        ;LOAD RADIX
26 033474 004737 033742  JSR    PC,PNTNUM        ;PRINT NUMBER
27 033500 000207        RTS    PC          ;NOW GET NEXT CONTROL CHARACTER
28
29           ;CONTROL CHARACTER WAS AN 'H', SO PRINT A HEX NUMBER.
30
31 033502 012701 000020  CON.H:  MOV    #16.,R1        ;LOAD RADIX
32 033506 004737 033742  JSR    PC,PNTNUM        ;PRINT NUMBER
33 033512 000207        RTS    PC          ;NOW GET NEXT CONTROL CHARACTER
```

```

1
2
3
4 033514 012701 000010      CON.O:  MOV    #8.,R1      ;LOAD RADIX
5 033520 004737 033742      JSR    PC,PNTNUM      ;PRINT NUMBER
6 033524 000207              RTS    PC              ;NOW GET NEXT CONTROL CHARACTER
7
8
9
10 033526 004737 033664     CON.N:  JSR    PC,GETCNT      ;GET COUNT
11 033532 000000              CON.N1:
12 033532 112700 000015     MOVB   #CR,R0          ;STORE #CR IN R0 AND
13 033536 004737 033086     JSR    PC,PRINTC      ;PRINT THE CHARACTER.
14 033542 005301              DEC    R1              ;COUNT THE SEQUENCES
15 033544 001372              BNE   CON.N1
16 033546 000207              RTS    PC              ;NOW GET NEXT CONTROL CHARACTER
17
18
19 033550 004737 033664     CON.R:  JSR    PC,GETCNT      ;GET ROUTINE NUMBER
20 033554 020127 000010     CMP    R1,#ERR.SZ     ;CHECK IF DEFINED ROUTINE NUMBER
21 033560 101004              BHI   CON.R1
22 033562 060101              ADD   R1,R1           ;DOUBLE COUNT TO GET WORD INDEX
23 033564 004771 016132     JSR    PC,#ERR.TB-2(R1);CALL ROUTINE
24 033570 000207              RTS    PC              ;NOW GET NEXT CONTROL CHARACTER
25
26 033572 000000              CON.R1:
27 033572 012746 014427     MOV    #ERRME1,(SP)   ;PUSH #ERRME1 ON STACK
28 033576 004137 033244     JSR    R1,LPNTF       ;CALL LPNTF PRINT ROUTINE
29 033602 000002              .WORD  ARG.CT         ;ARGUMENT COUNT * 2
30 033604 012601              MOV    (SP)+,R1       ;;POP STACK INTO R1
31 033606 000207              RTS    PC
32
33
34 033610 004737 033664     CON.S:  JSR    PC,GETCNT      ;GET COUNT
35 033614 000000              CON.S1:
36 033614 112700 000040     MOVB   #' ,R0          ;STORE #' IN R0 AND
37 033620 004737 033066     JSR    PC,PRINTC      ;PRINT THE CHARACTER.
38 033624 005301              DEC    R1              ;COUNT THE SPACES
39 033626 001372              BNE   CON.S1
40 033630 000207              RTS    PC              ;NOW GET NEXT CONTROL CHARACTER

```

```
1      .SBITL  FAO VECTOR TABLE
2      ;BUILD TWO TABLES
3      ;      FIRST CONTAINING CONTROL CHARACTERS
4      ;      SECOND CONTAINING ROUTINE ADDRESSES
5
22
23      ;      ERRC:  CONTROL CHARACTER TABLE
24
34
35      ERRC:
      033632      042      .BYTE  '"'
      033632      101      .BYTE  'A'
      033633      104      .BYTE  'D'
      033634      110      .BYTE  'H'
      033635      117      .BYTE  'O'
      033636      116      .BYTE  'N'
      033637      122      .BYTE  'R'
      033640      123      .BYTE  'S'
36      033642      000      .BYTE  0          ;FOLLOW WITH A NULL BYTE
37      .EVEN
38
39      ;      ERRD:  PROCESSING ROUTINE TABLE
40
50
51      ERRD:
      033644      033424  .WORD  CON.QU
      033644      033444  .WORD  CON.A
      033646      033470  .WORD  CON.D
      033650      033502  .WORD  CON.H
      033652      033514  .WORD  CON.O
      033654      033526  .WORD  CON.N
      033656      033550  .WORD  CON.R
      033660      033610  .WORD  CON.S
52
```



```

1      .SBTTL GETCNT GET CHARACTER COUNT
2      ;GETCNT
3      ;
4      ;GET COUNT IN NEXT CHARACTERS OF STRING POINTED TO BY R2.
5      ;NUMBER WILL BE IN DECIMAL. IF NO NUMBER, RETURN A
6      ;DEFAULT OF 1.
7      ;
8      ;INPUTS:
9      ;   R2 - POINTER TO ASCII STRING
10     ;
11     ;OUTPUTS:
12     ;   R1  NUMBER READ OR A ONE
13     ;   R2  - POINTING TO CHARACTER AFTER NUMBER
14     GETCNT:
15     033664 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
16     033666 005001      CLR      R1          ;;START WITH ZERO COUNT
17     033670 121227 000060 GETCNX: CMPB     (R2),#'0    ;;CHECK IF CHARACTER A DIGIT
18     033674 103415      BLO     GETCDN      ;;BRANCH IF LOWER THAN ZERO
19     033676 121227 000071      CMPB     (R2),#'9
20     033702 101012      BHI     GETCDN      ;;BRANCH IF HIGHER THAN NINE
21     033704 006301      ASL     R1          ;;MULTIPLY NUMBER BY 10
22     033706 010100      MOV     R1,R0      ;;SAVE 2N
23     033710 006301      ASL     R1          ;;COMPUTE 4N
24     033712 006301      ASL     R1          ;;COMPUTE 8N
25     033714 060001      ADD     R0,R1      ;;8N + 2N = 10N
26     033716 112200      MOVB   (R2)+,R0    ;;GET DIGIT FROM STING
27     033720 162700 000060      SUB     #'0,R0     ;;GET RID OF ASCII
28     033724 060001      ADD     R0,R1      ;;ADD TO NUMBER
29     033726 000760      BR     GETCNX     ;;GO TO NEXT CHARACTER
30     033730 005701      GFTCDN: TST    R1   ;;CHECK IF NUMBER IS ZERO
31     033732 001001      BNE    GETCXX     ;;IF ZERO, CHANGE
32     033734 005201      INC    R1        ;; TO DEFAULT OF ONE
33     033736 012600      GETCXX: MOV    (SP)+,R0 ;;POP STACK INTO R0
33     033740 000207      RTS     PC
  
```

```

1      .SBTTL PNTNUM PRINT NUMBER
2      ;PNTNUM
3      ;
4      ;PRINT A NUMBER
5      ;
6      ;INPUTS:
7      ;   R1 - RADIX OF NUMBER
8      ;   R2 - ASCII STRING TO COUNT OF BITS IN NUMBER
9      ;   R4 - POINTER TO NUMBER (LOW WORD)
10     ;OUTPUTS:
11     ;   NUMBER IS PRINTED. LEADING ZEROS ARE PRINTED EXCEPT FOR
12     ;   DECIMAL NUMBERS (LEFT JUSTIFIED).
13     ;   R0 - CONTENTS DESTROYED
14
15 033742 010100 PNTNUM: MOV R1,R0 ;SAVE RADIX
16 033744 004737 033664 JSR PC,GETCNT ;GET COUNT OF BITS
17 033750 PNTNUS:
18 033750 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
19 033752 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
20 033754 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
21 033756 012403 MOV (R4)+,R3 ;GET ONE PARAMETER WORD
22 033760 005005 CLR R5 ;CLEAR STORAGE FOR OTHER
23 033762 020127 000020 CMP R1,#16. ;MORE THAN 16 BITS IN NUMBER?
24 033766 003401 BLE 1$
25 033770 012405 MOV (R4)+,R5 ;YES, GET SECOND PARAMETER WORD
26 033772 1$:
27 033772 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
28 033774 010504 MOV R5,R4 ;PUT HIGH WORD IN R4
29 033776 012702 000020 MOV #16.,R2 ;COMPUTE BITS NOT WANTED
30 034002 160102 SUB R1,R2 ;BY SUBTRACTING BITS TO USE
31 034004 002002 BGE 2$ ;FROM 16.
32 034006 062702 000020 ADD #16.,R2 ;IF NEGATIVE, ADD 16 FOR FIRST WORD
33 034012 001414 2$: BEQ 6$ ;IF ZERO, NO BITS NEED BE CLEARED
34 034014 012705 100000 MOV #BIT15,R5 ;START MASK WITH SIGN BIT SET
35 034020 005302 3$: DEC R2 ;COUNT BITS IN MASK
36 034022 001402 BEQ 4$
37 034024 006205 ASR R5 ;SHIFT MORE BITS TO RIGHT
38 034026 000774 BR 3$
39 034030 020127 000020 4$: CMP R1,#16. ;MORE THAN 16 BITS IN NUMBER?
40 034034 003402 BLE 5$
41 034036 040504 BIC R5,R4 ;YES, CLEAR IN HIGH WORD
42 034040 000401 BR 6$
43 034042 040503 5$: BIC R5,R3 ;NO, CLEAR IN LOW WORD
44 034044 004737 034436 6$: JSR PC,DIVIDE ;DIVIDE BY RADIX IN R0
45 034050 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
46 034052 005202 INC R2 ;COUNT DIGITS ON STACK
47 034054 005703 TST R3 ;CHECK IF QUOTIENT IS ZERO
48 034056 001372 BNE 6$
49 034060 005704 TST R4
50 034062 001370 BNE 6$

```



```
1      .SBTTL CLRBUF CLEAR BUFFER
2      ;CLRBUF
3      ;
4      ;CLEAR A 512(10) BYTE BUFFER, STARTING AT THE UNIBUS ADDRESS CONTAINED IN
5      ;THE COMMAND PACKET OF THE HOST COMM AREA.
6      ;
7      ;INPUTS:
8      ;   R4 - ADDRESS OF HOST COMM AREA
9      ;OUTPUTS:
10     ;   512(10) BYTE BUFFER CLEARED
11
12 CLRBUF:
13     034240 010046      MOV     R0,-(SP)          ;;PUSH R0 ON STACK
14     034242 010146      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
15     034244 013704 002324  MOV     HCOM,R4          ; LOAD HOST COMM AREA ADDRESS
16     034250 016400 000124  MOV     HC.CPK+P.UADR(R4),R0 ;GET STARTING UNIBUS ADDRESS
17     034254 013701 002334  MOV     BCNT,R1          ;NUMBER OF BYTES TO CLEAR.
18     034260 006201      ASR     R1                ;MAKE R1 WORDS AND
19     034262 005020 1$:   CLR     (R0)+          ;CLEAR THE ENTIRE BUFFER.
20     034264 005301      DEC     R1
21     034266 001375      BNE    1$
22     034270 012601      MOV     (SP)+,R1          ;;POP STACK INTO R1
23     034272 012600      MOV     (SP)+,R0          ;;POP STACK INTO R0
24     034274 000207      RTS     PC
```

```

1      .SBTTL COPY COPY BUFFER
2      ;COPY
3      ;
4      ;COPY 512(10) BYTES OF DATA FROM ONE BUFFER TO ANOTHER BUFFER
5      ;
6      ;INPUTS:
7      ; (R5) - ADDRESS OF SOURCE BUFFER
8      ; 2(R5) - ADDRESS OF DESTINATION BUFFER
9      ;OUTPUTS:
10     ; 512(10) BYTES COPIED FROM 'SOURCE' BUFFER TO 'DESTINATION' BUFFER
11     ;
12     COPY:
13     034276 010046      MOV R0,-(SP)          ;;PUSH R0 ON STACK
14     034300 010146      MOV R1,-(SP)          ;;PUSH R1 ON STACK
15     034302 010246      MOV R2,-(SP)          ;;PUSH R2 ON STACK
16     034304 012500      MOV (R5)+,R0         ;GET STARTING ADDRESS OF SOURCE
17     034306 012501      MOV (R5)+,R1         ; AND DESTINATION BUFFERS.
18     034310 013702 002334  MOV BCNT,R2          ;NUMBER OF BYTES TO COPY
19     034314 006202      ASR R2                    ;MAKE R1 WORDS AND
20     034316 012021 1$:  MOV (R0)+,(R1)+        ;COPY HIS LOCATION TO OTHER BUFFER
21     034320 005302      DEC R2
22     034322 001375      BNE 1$
23     034324 012602      MOV (SP)+,R2         ;;POP STACK INTO R2
24     034326 012601      MOV (SP)+,R1         ;;POP STACK INTO R1
25     034330 012600      MOV (SP)+,R0         ;;POP STACK INTO R0
26     034332 000205      RTS R5

```

```

1      .SBTTL SETTO SET TIMEOUT
2      .SETTO
3      ;
4      ;SET TIMEOUT COUNTER TO SOME NUMBER OF SECONDS FROM CURRENT TIME.
5      ;
6      ;      R0 - NUMBER OF SECONDS FOR TIMEOUT
7      ;      R1 - ADDRESS WHERE TWO WORD TIME TO BE PUT
8      ;OUTPUTS:
9      ;      R0 - CONTENTS DESTROYED
10     ;      R1 - INCREMENTED BY 2
11     ;
12     ;COMPUTE CLOCK TICKS TIL TIMEOUT
13
14     034334      SETTO:
15     034334 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
16     034336 010346      MOV      R3,-(SP)          ;;PUSH R3 ON STACK
17     034340 005002      CLR      R2          ;CLEAR PRODUCT
18     034342 013703 002216  MOV      KW.HZ,R3      ;GET MULTIPLICAND
19     034346 006200      1$:      ASR      R0          ;SHIFT MULTIPLIER TO RIGHT
20     034350 103001      BCC     2$          ;IF A ONE BIT SHIFTED OUT
21     034352 060302      ADD     R3,R2      ;ADD MULTIPLICAND TO PRODUCT
22     034354 006303      2$:      ASL     R3          ;DOUBLE THE MULTIPLICAND
23     034356 005700      TST     R0
24     034360 001372      BNE     1$          ;CONTINUE UNTIL MULTIPLIER IS ZERO
25
26           ;GET CURRENT TIME
27     034362 013700 002220  3$:      MOV     KW.EL,R0          ;GET TIME
28     034366 013703 002222  MOV     KW.EL+2,R3
29     034372 020037 002220  CMP     R0,KW.EL      ;IF CHANGED DURING RETRIEVAL
30           BNE     3$          ;GET IT AGAIN
31
32           ;ADD TIME TIL TIMEOUT
33     034400 060200      ADD     R2,R0          ;ADD
34     034402 005503      ADC     R3
35
36           ;PUT RESULT IN STORAGE
37
38     034404 010021      MOV     R0,(R1)+
39     034406 010311      MOV     R3,(R1)
40     034410 012603      MOV     (SP)+,R3      ;;POP STACK INTO R3
41           034412 012602      MOV     (SP)+,R2      ;;POP STACK INTO R2
42     034414 000207      RTS     PC
    
```

```
1          .SBTTL CLOG COMPUTE BASE 2 LOG
2          ;CLOG
3          ;
4          ;COMPUTE LOGARITHMIC VALUE OF NUMBER TO BASE 2.
5          ;
6          ;INPUTS:
7          ;      RO - LOGARITHM TO BE CONVERTED
8          ;OUTPUTS:
9          ;      R1 - VALUE OF 2 RAISED TO POWER OF INPUT NUMBER
10         ;
11 034416 CLOG:
12 034416 010046      MOV      RO,-(SP)          ;;PUSH RO ON STACK
13 034420 005001      CLR      R1              ;SET UP ZERO START VALUE
14 034422 000261      SEC              ;WITH CARRY READY TO SHIFT IN
15 034424 006101      1$: ROL      R1          ;SHIFT TO LEFT
16 034426 005300      DEC      RO          ;UNTIL RO
17 034430 100375      BPL      1$          ;GOES NEGATIVE
18 034432 012600      MOV      (SP)+,RO      ;;POP STACK INTO RO
18 034434 000207      RTS      PC
```

```

1      .SBTTL DIVIDE DIVIDE NUMBER
2      ;DIVIDE
3      ;
4      ;DIVIDE A 32 BIT UNSIGNED NUMBER BY A 16 BIT UNSIGNED NUMBER.
5      ;REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
6      ;WILL NOT CHECK FOR DIVIDE BY ZERO.
7      ;
8      ;INPUTS:
9      ;   R3 - LOW 16 BITS OF DIVIDEND
10     ;   R4 - HIGH 16 BITS OF DIVIDEND
11     ;   R0 - DIVISOR
12     ;OUTPUTS:
13     ;   R3 - LOW 16 BITS OF QUOTIENT
14     ;   R4 - HIGH 16 BITS OF QUOTIENT
15     ;   R5 - REMAINDER
16
17 034436 DIVIDE:
18 034436 010246      MOV     R2,-(SP)           ;;PUSH R2 ON STACK
19 034440 012702      MOV     #32.,R2        ;SET UP SHIFT COUNT
19 034444 005005      CLR     R5                ;START WITH ZERO REMAINDER
20 034446 006303      1$:   ASL     R3                ;SHIFT LEFT INTO R5
21 034450 006104      ROL     R4
22 034452 006105      ROL     R5
23 034454 020005      CMP     R0,R5           ;WILL DIVISOR GO INTO REMAINDER
24 034456 101002      BHI     2$                ;ONLY SUBTRACT IF IT WILL
25 034460 160005      SUB     R0,R5           ;SUBTRACT DIVISOR
26 034462 005203      INC     R3                ;PUT A ONE INTO QUOTIENT
27 034464 005302      2$:   DEC     R2                ;COUNT THE SHIFTS
28 034466 001367      BNE     1$
29 034470 012602      MOV     (SP)+,R2        ;;POP STACK INTO R2
30 034472 000207      RTS     PC

```



```

1      .SBTTL DIV10  DIVIDE BY TEN
2      ;DIV10
3      ;
4      ;DIVIDE A 64 BIT UNSIGNED NUMBER BY A 10.
5      ;REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
6      ;WILL NOT CHECK FOR DIVIDE BY ZERO.
7      ;
8      ;INPUTS:
9      ;   R1 - LOW 16 BITS OF DIVIDEND
10     ;   R2 - NEXT 16 BITS OF DIVIDEND
11     ;   R3 - NEXT 16 BITS OF DIVIDEND
12     ;   R4 - HIGH 16 BITS OF DIVIDEND
13     ;OUTPUTS:
14     ;   R1 - QUOTIENT,
15     ;   R2 - QUOTIENT,
16     ;   R3 - QUOTIENT,
17     ;   R4 - QUOTIENT,
18     ;   R5 - REMAINDER
19
20 034474 DIV10: MOV     R0,-(SP)           ;;PUSH R0 ON STACK
    034474 010046 MOV     #64.,R0           ;;SET UP SHIFT COUNT
21 034476 012700 000100 CLR     R5               ;;START WITH ZERO REMAINDER
22 034502 005005 1#: ASL     R1
23 034504 006301 ROL     R2           ;;SHIFT LEFT INTO R5
24 034506 006102 ROL     R3
25 034510 006103 ROL     R4
26 034512 006104 ROL     R5
27 034514 006105 CMP     #10.,R5           ;;SILL DIVISOR GO INTO REMAINDER?
28 034516 022705 000012 BHI     2#           ;;ONLY SUBTRACT IF IT WILL
29 034522 101003 SUB     #10.,R5           ;;SUBTRACT DIVISOR
30 034524 162705 000012 INC     R1           ;;PUT A ONE INTO QUOTIENT
31 034530 005201 2#: DEC     R0           ;;COUNT THE SHIFTS
32 034532 005300 BNE     1#
33 034534 001363 MOV     (SP)+,R0         ;;POP STACK INTO R0
34 034536 012600 RTS     PC           ; <R4,R3,R2,R1> AND REMAINDER IN R5
35 034540 000207
  
```

CZUOLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 115
 MULT INTEGER MULTIPLY ROUTINE

```

1      .SBTIL MULT INTEGER MULTIPLY ROUTINE
2      ;MULT
3      ;
4      ;CALL
5      ;   MOV MULTIPLIER, (SP)
6      ;   MOV MULTIPLICAND, -(SP)
7      ;   JSR PC,MULT
8      ;   RETURN ;PRODUCT IS ON THE STACK
9      ;
10     ;   STACK PRODUCT
11     ;   -----
12     ;   TOP   LSB'S
13     ;   +2   MSB'S
14
15 034542 MULT:
    034542 010046 MOV R0, -(SP) ;: PUSH R0 ON STACK
    034544 010146 MOV R1, -(SP) ;: PUSH R1 ON STACK
    034546 010246 MOV R2, -(SP) ;: PUSH R2 ON STACK
16 034550 005046 CLR -(SP) ;CLEAR THE SIGN KEY
17 034552 016601 000012 MOV 12(SP), R1 ;GET THE MULTIPLICAND
18 034556 100002 BPL 1$ ;BR IF PLUS
19 034560 005216 INC (SP) ;SET THE SIGN KEY
20 034562 005401 NEG R1 ;MAKE THE MULTIPLICAND POSTIVE
21 034564 016602 000014 1$: MOV 14(SP), R2 ;GET THE MULTIPLIER
22 034570 100002 BPL 2$ ;BR IF PLUS
23 034572 005316 DEC (SP) ;UPDATE THE SIGN KEY
24 034574 005402 NEG R2 ;MAKE THE MULTIPLIER POSTIVE
25 034576 012746 000021 2$: MOV 17, -(SP) ;SET THE LOOP COUNT
26 034602 005000 CLR R0 ;SETUP FOR THE MULTIPLY LOOP
27 034604 103001 3$: BCC 4$ ;DON'T ADD IF MULTIPLICAND = 0
28 034606 060200 ADD R2, R0
29 034610 006000 4$: ROR R0 ;POSITION THE PARTIAL PRODUCT AND
30 034612 006001 ROR R1 ;THE MULTIPLICAND
31 034614 005316 DEC (SP) ;HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
32 034616 001372 BNE 3$ ;BR IF NO
33 034620 022616 CMP (SP)+, (SP) ;SHOULD PRODUCT BE NEGATIVE?
34 034622 001403 BEQ 5$ ;GO TO EXIT IF NO
35 034624 005400 NEG R0 ;YES--SO MAKE IT SO
36 034626 005401 NEG R1
37 034630 005600 SBC R0
38 034632 005726 5$: TST (SP)+ ;CLEAR SIGN INFO. OFF OF STACK
39 034634 010066 000012 MOV R0, 12(SP) ;PUT THE PRODUCT ON THE STACK (MSB'S)
40 034640 010.56 000010 MOV R1, 10(SP) ;LSB'S
41 034644 012.02 MOV (SP)+, R2 ;: POP STACK INTO R2
    034646 012601 MOV (SP)+, R1 ;: POP STACK INTO R1
    034650 012600 MOV (SP)+, R0 ;: POP STACK INTO R0
42 034652 000207 RTS PC

```

```

1      .SBTTL BLD28 BUILD 28-BIT NUMBER
2      ;BLD28
3      ;
4      ;BUILD DEFAULT 28-BIT NUMBER
5      ;
6      ;INPUT:
7      ; R4 - ADDRESS OF LO ORDER WORD OF DEFAULT NUMBER
8      ;OUTPUT:
9      ; TTYIN - ADDRESS OF ASCIZ STRING INPUT BUFFER
10
11 034654 BLD28:
    034654 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
    034656 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
    034660 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
    034662 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
    034664 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
12 034666 011403      MOV     (R4),R3      ;GET NUMBER
13 034670 016404 000002  MOV     2(R4),R4
14 034674 012700 000012  MOV     #10.,R0      ;DIVISOR IS 10.
15 034700 005001      CLR     R1           ;CLEAR CHARACTER COUNT
16 034702 004737 034436 1$: JSR     PC,DIVIDE
17 034706 062705 000060  ADD     #'0,R5      ;CONVERT REMAINDER TO ASCII CHARACTER
18 034712 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
19 034714 005201      INC     R1           ;COUNT THE CHARACTER
20 034716 010305      MOV     R3,R5      ;REPEAT UNTIL QUOTIENT IS ZERO
21 034720 050405      BIS     R4,R5
22 034722 001367      BNE     1$
23 034724 012700 002666  MOV     #TTYIN,R0    ;GET POINTER TO STRING
24 034730 2$:
    034730 012605      MOV     (SP)+,R5     ;;POP STACK INTO R5
25 034732 110520      MOVB   R5,(R0)+
26 034734 005301      DEC     R1
27 034736 001374      BNE     2$
28 034740 105020      CLRB   (R0)+        ;END WITH NULL
29 034742 012605      MOV     (SP)+,R5     ;;POP STACK INTO R5
    034744 012604      MOV     (SP)+,R4     ;;POP STACK INTO R4
    034746 012603      MOV     (SP)+,R3     ;;POP STACK INTO R3
    034750 012601      MOV     (SP)+,R1     ;;POP STACK INTO R1
    034752 012600      MOV     (SP)+,R0     ;;POP STACK INTO R0
30 034754 000207      RETURN
  
```

```

1      .SBTTL  CNV28  CONVERT ASCIZ TO 28-BIT BINARY
2      ;CNV28
3      ;
4      ;CONVERT ASCIZ STRING TO 28-BIT NUMBER
5      ;
6      ;INPUTS:
7      ;   TTYIN - ASCIZ STRING INPUT UP TO 9. CHARACTERS
8      ;   R4 - ADDRESS OF LO ORDER WORD OF STORAGE LOCATION
9      ;
10     ;OUTPUTS:
11     ;   IF STRING IS VALID NUMBER
12     ;     TWO WORDS AT R4 LOADED WITH NUMBER
13     ;     CARRY CLEAR
14     ;   IF STRING INVALID
15     ;     ERROR MESSAGE PRINTED
16     ;     CARRY SET
17     CNV28:
18     034756 010046      MOV     R0,-(SP)          ;;PUSH R0 ON STACK
19     034760 010146      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
20     034762 010246      MOV     R2,-(SP)          ;;PUSH R2 ON STACK
21     034764 010346      MOV     R3,-(SP)          ;;PUSH R3 ON STACK
22     18 034766 005000      CLR     R0              ;CLEAR LO ORDER WORD
23     19 034770 005001      CLR     R1              ;CLEAR HI ORDER WORD
24     20 034772 012702 002666  MOV     #TTYIN,R2        ;GET BEGINNING ADDRESS OF TYPED INPUT
25     21 034776 112203      1$:   MOVB   (R2)+,R3      ;GET A CHARACTER FROM ASCIZ LINE
26     22 035000 001443      BEQ     4$              ;IF NULL CHARACTER, ALL DONE
27     23 035002 162703 000060  SUB     #'0,R3          ;SUBTRACT CHARACTER 0
28     24 035006 100431      BMI     2$              ;
29     25 035010 022703 000011  CMP     #9.,R3          ;
30     26 035014 103426      BLO     2$              ;
31     27 035016 006300      ASL     R0              ;*2
32     28 035020 006101      ROL     R1              ;
33     29 035022 010146      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
34     30 035024 010046      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
35     31 035026 006300      ASL     R0              ;*4
36     32 035030 006101      ROL     R1              ;
37     33 035032 006300      ASL     R0              ;*8.
38     34 035034 006101      ROL     R1              ;
39     35 035036 062600      ADD     (SP)+,R0        ;*10.
40     36 035040 005501      ADC     R1              ;
41     37 035042 062601      ADD     (SP)+,R1        ;
42     38 035044 060300      ADD     R3,R0           ;ADD CURRENT DIGIT
43     39 035046 005501      ADC     R1              ;
44     40 035050 032701 170000  BIT     #170000,R1      ;CHECK SIZE OF NUMBER
45     41 035054 001750      BEQ     1$              ;MUST NOT BE MORE THAN 28 BITS
46     42 035056 012746 014277  MOV     #LMT28,-(SP)    ;PRINT ' ?EXCEEDED 28 BIT INPUT LIMIT'
47     43 035062 004137 033244  JSR     R1,LPNTF        ;PUSH #LMT28 ON STACK
48     44 035066 000002      .WORD  ARG.CT          ;CALL LPNTF PRINT ROUTINE
49     45 035070 000405      BR      3$              ;ARGUMENT COUNT * 2
50     46 035072 012746 014340  2$:   MOV     #INVCHR,-(SP) ;PUSH #INVCHR ON STACK
51     47 035076 004137 033244  JSR     R1,LPNTF        ;CALL LPNTF PRINT ROUTINE
52     48 035102 000002      .WORD  ARG.CT          ;ARGUMENT COUNT * 2
53     49 035104 000261      3$:   SEC                    ;SET CARRY FOR ERROR
54     50 035106 000404      BR      5$

```

```

48
49 035110 010014          4$:  MOV    R0,(R4)          ;MOVE LO ORDER WORD AND
50 035112 010164 000002    MOV    R1,2(R4)          ;HI ORDER WORD TO STORAGE AREA.
51 035116 000241          CLC                    ;CLEAR CARRY FOR NO ERROR
52 035120
   035120 012603          5$:  MOV    (SP)+,R3          ;;POP STACK INTO R3
   035122 012602          MOV    (SP)+,R2          ;;POP STACK INTO R2
   035124 012601          MOV    (SP)+,R1          ;;POP STACK INTO R1
   035126 012600          MOV    (SP)+,R0          ;;POP STACK INTO R0
53 035130 000207          RETURN
54          .SBTTL  REPORT CODING SECTION
55
57          ;**
58          ; THE REPORT CODING SECTION CONTAINS THE
59          ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
60          ;--
61
62 035132          L$RPT::
63
64 035132 013705 002304          MOV    CTLRTBL,R5          ; LOAD TABLE PTR
65 035136 013704 002324          MOV    HCOM,R4           ; LOAD COMM PTR
66 035142          10$:  MOV    #INF200,-(SP)       ; "REPLACEMENT INFORMATION..."
67 035142 012746 005271          JSR    R1,LPNTB          ;PUSH #INF200 ON STACK
68 035146 004137 033254          .WORD ARG.CT            ;CALL LPNTB PRINT ROUTINE
69 035152 000002          MOV    WFECNT,-(SP)       ;ARGUMENT COUNT * 2
70 035154 013746 002542          MOV    WFECNT,-(SP)       ;PUSH WFECNT ON STACK
71 035160 013746 002530          MOV    FERCNT,-(SP)       ;PUSH FERCNT ON STACK
72 035164 013746 002536          MOV    ECCCNT,-(SP)       ;PUSH ECCCNT ON STACK
73 035170 013746 002534          MOV    DSTCNT,-(SP)       ;PUSH DSTCNT ON STACK
74 035174 013746 002532          MOV    HCECNT,-(SP)       ;PUSH HCECNT ON STACK
75 035200 013746 002540          MOV    BBRCNT,-(SP)       ;PUSH BBRCNT ON STACK
76 035204 012746 005343          MOV    #INF201,-(SP)       ;PUSH #INF201 ON STACK
77 035210 004137 033254          JSR    R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
78 035214 000016          .WORD ARG.CT            ;ARGUMENT COUNT * 2
79
80 035216 013703 002434          MOV    PRIMARY,R3        ; LOAD PRIMARY COUNT
81 035222 063703 002436          ADD    SCNDRY,R3         ; ADD SECONDARY COUNT
82
83 035226 010346          MOV    R3,-(SP)          ;PUSH R3 ON STACK
84 035230 013746 002440          MOV    UNUSE,-(SP)        ;PUSH UNUSE ON STACK
85 035234 013746 002436          MOV    SCNDRY,-(SP)       ;PUSH SCNDRY ON STACK
86 035240 013746 002434          MOV    PRIMARY,-(SP)      ;PUSH PRIMARY ON STACK
87 035244 012746 005725          MOV    #INF202,-(SP)       ;PUSH #INF202 ON STACK
88 035250 004137 033254          JSR    R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
89 035254 000012          .WORD ARG.CT            ;ARGUMENT COUNT * 2
90
91 035256 005737 002650          TST    DEBUG              ; TEST DEBUG FLAG
92 035262 001007          BNE    100$              ; BRANCH IF YES
93
94 035264 012746 004731          MOV    #INF103,-(SP)      ;PUSH #INF103 ON STACK
95 035270 004137 033254          JSR    R1,LPNTB          ;CALL LPNTB PRINT ROUTINE
96 035274 000002          .WORD ARG.CT            ;ARGUMENT COUNT * 2
97
98 035276 000167          .WORD J$JMP
99 035300 000414          .WORD L10017-2-.

```

```

1
2           ;          DEBUG REPORTING
3 035302    100$:
4
5           ;          PRINT RCT ERROR TABLE
6
7 035302    013746    002312    MOV      RCTTBL,-(SP)          ;PUSH RCTTBL ON STACK
8 035306    012746    035470    MOV      #DBM01,-(SP)        ;PUSH #DBM01 ON STACK
9 035312    004137    033254    JSR     R1,LPNTB             ;CALL LPNTB PRINT ROUTINE
10 035316   000004                .WORD   ARG.CT              ;ARGUMENT COUNT * 2
11
12 035320    005003                CLR     R3
13 035322    013702    002312    MOV     RCTTBL,R2
14
15 035326    110$:
16 035326    011201                MOV     (R2),R1
17 035330    001411                BEQ    120$
18
19 035332    010146                MOV     R1,-(SP)             ;PUSH R1 ON STACK
20 035334    012746    000000    MOV     #0,-(SP)            ;PUSH #0 ON STACK
21 035340    010346                MOV     R3,-(SP)            ;PUSH R3 ON STACK
22 035342    012746    035556    MOV     #DBM02,-(SP)        ;PUSH #DBM02 ON STACK
23 035346    004137    033254    JSR     R1,LPNTB             ;CALL LPNTB PRINT ROUTINE
24 035352    000010                .WORD   ARG.CT              ;ARGUMENT COUNT * 2
25
26 035354    120$:
27 035354    062702    000002    ADD     #2,R2
28 035360    005203                INC    R3
29 035362    023703    002416    CMP     RCTSIZ,R3
30 035366    003357                BGT    110$
31
32           ;          PRINT CACHE TABLE CONTENTS
33
34 035370    013746    002310    MOV     CACHTBL,-(SP)       ;PUSH CACHTBL ON STACK
35 035374    012746    035601    MOV     #DBM03,-(SP)       ;PUSH #DBM03 ON STACK
36 035400    004137    033254    JSR     R1,LPNTB             ;CALL LPNTB PRINT ROUTINE
37 035404    000004                .WORD   ARG.CT              ;ARGUMENT COUNT * 2
38
39 035406    005003                CLR     R3
40 035410    013704    002310    MOV     CACHTBL,R4
41
42 035414    130$:
43 035414    016402    000002    MOV     2(R4),R2
44 035420    100413                BMI    140$
45
46 035422    011401                MOV     (R4),R1
47 035424    010246                MOV     R2,-(SP)             ;PUSH R2 ON STACK
48 035426    010146                MOV     R1,-(SP)             ;PUSH R1 ON STACK
49 035430    012746    000000    MOV     #0,-(SP)            ;PUSH #0 ON STACK
50 035434    010346                MOV     R3,-(SP)            ;PUSH R3 ON STACK
51 035436    012746    035667    MOV     #DBM04,-(SP)        ;PUSH #DBM04 ON STACK
52 035442    004137    033254    JSR     R1,LPNTB             ;CALL LPNTB PRINT ROUTINE
53 035446    000012                .WORD   ARG.CT              ;ARGUMENT COUNT * 2
54
55 035450    140$:
56 035450    062704    000004    ADD     #4,R4
57 035454    005203                INC    R3
58 035456    022703    000620    CMP     #CACHESIZE,R3
59 035462    003354                BGT    130$
60
61 035464    000167                .WORD   J$JMP
    
```

```
035466 000226 .WORD L10017-2-.
41
42 .NLIST BEX
43 035470 116 062 042 DBM01: .ASCII /N2"RCT ERROR TABLE: "016/
44 035521 116 062 042 .ASCIZ /N2" RCT BLOCK. CONTENTS"/
45 035556 116 042 040 DBM02: .ASCIZ /N" "D28". "016/
46
47 035601 116 062 042 DBM03: .ASCII /N2"CACHING TABLE: "016/
48 035630 116 062 042 .ASCIZ /N2" ENTRY NO. BLOCK NO. "/
49 035667 116 042 040 DBM04: .ASCIZ /N" "D28". "D28". "/
50 .EVEN
51
52 035716 L10017:
035716 104425 TRAP C#RPT
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
16

.SBTTL PROTECTION TABLE

;++
; THIS TABLE IS USED BY THE RUNTIME SERVICES
; TO PROTECT THE LOAD MEDIA.
:--

L\$PROT::

035720

035720 177777
035722 177777
035724 177777

-1
-1
-1

;OFFSET INTO P-TABLE FOR CSR ADDRESS
;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;OFFSET INTO P-TABLE FOR DRIVE NUMBER

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42

```
.SBTTL  INITIALIZATION SECTION

; **
; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
; AT THE BEGINNING OF EACH PASS.
; **

; *****
; RESET THE WORLD
; IF HERE FROM START, RESTART, OR NEW PASS COMMAND
; THEN
;     INITIALIZE LOGICAL UNIT NUMBER
; ENDIF
; SELECT A LOGICAL UNIT FOR TEST
; ESTABLISH FREE MEMORY
; INITIALIZE TABLES
; INITIALIZE KW11 CLOCK
; PRINT DRIVE ON-LINE
; PRINT NUMBER LOGICAL BLOCKS IN DRIVE
; END
; *****
I$SECTION          = 1

L$INIT::

; RESET SYSTEM
10$: TRAP C$RESET
    TST LOADFLG ; TEST FOR FIRST TIME THROUGH
    BNE 20$ ; BRANCH IF NO
    INC LOADFLG ; ELSE, SET FLAG
15$: ; RESET START OF FREE MEMORY
    TRAP C$MEM
    MOV R0,FFREE
    MOV @FFREE,FSIZE ; RESET SIZE OF FREE MEMORY
18$: ; READ BUS TYPE
    TRAP C$RDBU
    BCC 20$
    MOV #1,BUSFLAG ; SET FLAG FOR QBUS
    MOVB #'K,DEVNAME ; SET CONTROLLER "KDA"
```

```
000001
035726
035726 104433
035730 005737 002202
035734 001020
035736 005237 002202
035742 104431
035744 010037 002176
035750 017737 144222 002200
035756 104407
035760 103006
035762 012737 000001 002614
035770 112737 000113 014225
```

```

1          .SBTTL  INITIALIZE THE KW11
2
3 035776          20$:
5 035776 005037 002220      CLR      KW.EL          ;CLEAR ELAPSED TIME
6 036002 005037 002222      CLR      KW.EL+2
7 036006 012700 000114      MOV      @'L,RO
   036012 104462      TRAP     C$CLCK
8 036014 103414      BCS      25$
9 036016 012700 000120      MOV      @'P,RO
   036022 104462      TRAP     C$CLCK
10 036024 103410      BCS      25$
11 036026 005037 002210      CLR      KW.CSR          ;IF NEITHER, CLEAR CSR STORAGE WORD
12 036032 012746 004546      MOV      @INF101,-(SP)      ;PUSH @INF101 ON STACK
   036036 004137 033244      JSR      R1,LPNTF          ;CALL LPNTF PRINT ROUTINE
   036042 000002      .WORD     ARG.CT          ;ARGUMENT COUNT * 2
13 036044 000426      BR       30$
14 036046          25$:
15 036046 012037 002210      MOV      (RO)+,KW.CSR      ;STORE DATA RETURNED
16 036052 012037 002212      MOV      (RO)+,KW.BRL
17 036056 012037 002214      MOV      (RO)+,KW.VEC
18 036062 012037 002216      MOV      (RO)+,KW.HZ
19 036066 012746 000340      MOV      @PRI07,-(SP)
   036072 012746 016174      MOV      @KW11I,-(SP)
   036076 013746 002214      MOV      KW.VEC,-(SP)
   036102 012746 000003      MOV      @3,-(SP)
   036106 104437      TRAP     C$SVEC
   036110 062706 000010      ADD      @10,SP
20 036114 012777 000105 144066  MOV     @KW.OUT,@KW.CSR      ;START THE CLOCK
21
22 036122          30$:
23

```

```
1          .SBTTL  DETERMINE INITIALIZATION ENTRY
2
3          ;      TEST FOR OPERATOR "START"
4 036122   50$:   MOV     @EF.STA,RO
5 036122   012700 006040 TRAP   C$REFG
   036126   104447
6 036130   103412   BCS     100$
7
8          ;      TEST FOR OPERATOR "RESTART"
9 036132   60$:   MOV     @EF.RES,RO
10 036132  012700 000037 TRAP   C$REFG
   036136   104447
11 036140  103455   BCS     120$
12
13         ;      TEST FOR NEW PASS
14 036142   70$:   MOV     @EF.NEW,RO
15 036142  012700 000035 TRAP   C$REFG
   036146   104447
16 036150  103455   BCS     124$
17
18 036152  000137 036646   JMP     300$           ; ELSE, EXIT
19
```

```

1      .SBTTL OPERATOR "START"
2
3      ;OPERATOR "START" COMMAND
4      ;
5      ; INITIALIZE FREE MEMORY (FFREE,FSIZE)
6
7 036156      100$:
8 036156 012746 004362      MOV      #INF100,-(SP)      ;PUSH #INF100 ON STACK
   036162 004137 033244      JSR      R1,LPNTF          ;CALL LPNTF PRINT ROUTINE
   036166 000002              .WORD    ARG.CT          ;ARGUMENT COUNT * 2
9 036170 012701 022464      MOV      #BACKMSG,R1      ; LOAD QUESTION ADDRESS
10 036174 012702 002652     MOV      #BACKUP,R2       ; LOAD REPLY ADDRESS
11 036200 012712 000000     MOV      #0,(R2)         ; DEFAULT IS "NO"
12 036204 004737 023126     JSR      PC,GETLOG        ; ASK OPERATOR
13 036210 005737 002652     TST     BACKUP           ; TEST FOR BACKUP
14 036214 001001              BNE     105$             ; BRANCH IF YES
15
16 036216 104444              TRAP    C$DCLN
17 036220      105$:
18 036220 104431              TRAP    C$MEM
   036222 010037 002176      MOV     RO,FFREE
19 036226 017737 143744 002200  MOV     #FFREE,FSIZE      ; RESET SIZE OF FREE MEMORY
20
21 036234 012700 000010     MOV     #8.,RO           ; LOAD COUNTER
22 036240 012701 002230     MOV     #CTRLST,R1       ; LOAD CONTROLLER TABLE LIST
23 036244 012702 002254     MOV     #CACHLST,R2      ; LOAD CACHING TABLE LIST
24 036250      110$:
25 036250 005021              CLR     (R1)+            ; CLEAR CONTROLLER TABLE ENTRY
26 036252 005022              CLR     (R2)+            ; CLEAR CACHE TABLE ENTRY
27 036254 005300              DEC     RO               ; DECREMENT
28 036256 003374              BGT     110$            ; AND CONTINUE
29
30 036260 004737 040004     JSR     PC,HCOMM         ; ALLOCATE SPACE FOR HOST COMM AREA
31 036264 012701 000010     MOV     #8.,R1          ; LOAD SIZE
32 036270 004737 040054     JSR     PC,ALOCN        ; AND ALLOCATE
33

```

```

1          .SBTTL OPERATOR "RESTART"
2          ;
3          ;OPERATOR "RESTART"COMMAND
4          ;
5
6 036274   120$:
7 036274   104450   TRAP   C$MANI
8 036276   103002   BCC    124$
9 036300   004737   022206   JSR    PC,INISWP           ; INITIALIZE SW PARMS
10 036304   124$:
11 036304   005737   002150   TST   MANRPL             ; TEST FOR MANUAL MODE
12 036310   001403   BEQ   125$             ; BRANCH IF NO
13 036312   012737   000001   002012   MOV   #1,L$UNIT        ; ELSE, SET LAST UNIT TO ZERO
14 036320   125$:
15 036320   012737   177777   002074   MOV   #-1,L$LUN        ; INITIALIZE LOGICAL UNIT NUMBER
16 036326   012737   177776   002300   MOV   #-2,TBLNDX      ; INITIALIZE ADDRESS TABLE INDEX
17 036334   130$:
18 036334   005237   002074   TNC   L$LUN            ; INCREMENT TO NEXT LOGICAL UNIT
19 036340   062737   000002   002300   ADD  #2,TBLNDX        ; INCREMENT ADDRESS TABLE INDEX
20
21 036346   023737   002074   002012   CMP   L$LUN,L$UNIT     ; TEST IF LOGICAL UNITS ARE EXHAUSTED
22 036354   002134   BGE   300$            ; BRANCH IF YES
23
24 036356   013700   002074   MOV   L$LUN,R0
   036362   104442   TRAP  C$GPHRD
   036364   010037   002302   MOV   R0,PTABLE
25 036370   103361   BCC   130$
26

```

```

1
2
3 036372          ;
4 036372 013702 002300      ; 150$: ALLOCATE CONTROLLER TABLE
5 036376 016201 002230      MOV   TBLNDX,R2          ; LOAD TABLE INDEX
6 036402 001402          MOV   CTLRLST(R2),R1     ; LOAD TABLE ADDRESS
7 036404 000137 036334      BEQ   155$              ; BRANCH IF NO ADDRESS
8 036410          JMP   130$              ; ELSE, NEXT UNIT
9 036410 012701 000016      ; 155$: MOV   #<C.SIZE+4>/2,R1   ; LOAD TABLE SIZE * 2
10 036414 004737 040054      JSR   PC,ALOCHE         ; AND ALLOCATE
11 036420 010162 002230      MOV   R1,CTLRLST(R2)    ; SAVE TABLE ADDRESS
12 036424 013761 002324 000016  MOV   HCOM,C.HCOM(R1)   ; SAVE COMM AREA ADDRESS
13 036432 012761 000000 000020  MOV   #0,C.DRVT(R1)    ; ZERO DRIVE TABLE PTR
14
15          ;
16 036440          ; 160$: INITIALIZE CONTROLLER TABLE
17 036440 010137 002304      MOV   R1,CTLRTBL        ; SAVE CONTROLLER TABLE ADDRESS
18 036444 013703 002302      MOV   PTABLE,R3        ; LOAD PTABLE ADDRESS
19 036450 011321          MOV   (R3),(R1)+        ; STORE XDA IP ADDRESS
20 036452 016304 000004      MOV   H.BRL(R3),R4     ; GET THE BR LEVEL
21 036456 000304          SWAB  R4                ; SWAP TO HIGH BYTE,
22 036460 006104          ROL   R4                ; SHIFT ONE MORE TO LEFT,
23 036462 056304 000002      BIS   H.VEC(R3),R4     ; ADD VECTOR ADDRESS AND
24 036466 010421          MOV   R4,(R1)+        ; STORE IT IN THE CONTROLLER TABLE.
25 036470 016321 000006      MOV   H.BST(R3),(R1)+  ; STORE BURST RATE AND
26 036474 016321 000010      MOV   H.DRV(R3),(R1)+  ; DRIVE NUMBER TO THE CONTROLLER TABLE.
27 036500 012721 004037      MOV   #4037,(R1)+     ; STORE THE 'JSR R0' INSTRUCCION AND
28 036504 012721 016164      MOV   #XDASRV,(R1)+   ; STORE ISR ADDRESS
29
30 036510 013701 002304      MOV   CTLRTBL,R1       ; RELOAD TABLE START
31 036514 005061 000014      CLR   C.FLG(R1)        ; CLEAR FLAGS
32 036520 005061 000022      CLR   C.TO(R1)         ; CLEAR TIMEOUT (LOW)
33 036524 005061 000024      CLR   C.TOH(R1)        ; CLEAR TIMEOUT (HIGH)
34 036530 005061 000026      CLR   C.REF(R1)        ; CLEAR CMD REF
35

```

```
1
2 036534
3 036534 013705 002304
4 036540 016501 000020
5 036544 001006
6
7 036546 012701 000074
8 036552 004737 040054
9 036556 010165 000020
10
11
12 036562
13 036562 012700 000072
14 036566
15 036566 005021
16 036570 005300
17 036572 003375
18
```

;

170\$:

ALLOCATE DRIVE TABLE

MOV CTLRTBL,R5 ; LOAD CONTROLLER TABLE ADDRESS

MOV C.DRVT(R5),R1 ; LOAD DRIVE TABLE ADDRESS

BNE 180\$; BRANCH IF PRESENT

MOV *(<D.SIZE+4>/2,R1) ; LOAD DRIVE TABLE SIZE * 2

JSR PC,ALOC ; ALLOCATE

MOV R1,C.DRVT(R5) ; AND STORE OIN CONTROLLER TABLE

;

180\$:

INITIALIZE DRIVE TABLE

MOV *(<D.SIZE>/2,R0) ; LOAD DRIVE TABLE SIZE

185\$:

CLR (R1)+ ; CLEAR LOCATION

DEC R0 ; AND CONTINUE

BGT 185\$; UNTIL TABLE ENL

```

1
2
3 036574          ; ALLOCATE CACHING TABLE
4 036574 013702 002300 200$: MOV TBLNDX,R2 ; LOAD TABLE INDEX
5 036600 016201 002254  MOV CACHLST(R2),R1 ; LOAD TABLE ADDRESS
6 036604 001006      BNE 220$ ; BRANCH IF ADDRESS PRESENT
7
8 036606 012701 001442  MOV #<CACHESIZE+1>*2,R1 ; LOAD TABLE SIZE + 2
9 036612 004737 040054  JSR PC,ALOCM ; AND ALLOCATE
10 036616 010162 002254  MOV R1,CACHLST(R2) ; SAVE TABLE ADDRESS
11
12          ; INITIALIZE CACHE TABLE
13 036622          220$: MOV R1,CACHTBL ; SAVE CONTROLLER TABLE ADDRESS
14 036622 010137 002310  MOV #<CACHESIZE>*2,R0 ; LOAD TABLE SIZE
15 036626 012700 001440
16 036632          225$: MOV #-1,(R1)+ ; INITIALIZE ENTRY
17 036632 012721 177777  DEC R0 ; DECREMENT
18 036636 005300      BNE 225$ ; CONTINUE UNTIL ZERO
19 036640 001374
20
21 036642 000137 036334  JMP 130$ ; CONTINUE WITH NEXT UUT
22
23          ; EXIT INITIALIZATION
24 036646          300$: TST BACKUP ; TEST FLAG
25 036646 005737 002652  BNE 305$ ; BRANCH IF SET
26 036652 001001
27
28 036654 104444      TRAP C#DCLN
29 036656          305$: MOV #PRI00,R0 ; SET PRIORITY TO ZERO
30 036656 012700 000000  TRAP C#SPRI
31 036662 104441
32          I$SECTION = 0
33
34 036664 104432      TRAP C#EXIT
35 036666 001232      .WORD L10021-.

```



```

1      .SBTTL XDAINT INITIALIZE XDA
2      ;XDAINT
3      ;
4      ;FUNCTIONAL DESCRIPTION:
5      ;   SUBROUTINE TO INITIALIZE A XDA AND BRING IT ON-LINE.
6      ;   ALL STEPS ARE CHECKED. AN ERROR MESSAGE IS REPORTED IF ANY ERROR
7      ;   DETECTED.
8      ;
9      ;INPUTS:
10     ;   R5 - ADDRESS OF CONTROLLER TABLE.
11     ;   R4 - LENGTH, INTERRUPT AND VECTOR FIELDS TO SEND TO XDA
12     ;IMPLICIT INPUTS:
13     ;   HCOM - HOST COMMUNICATIONS AREA
14     ;   FFREE - FIRST FREE ADDRESS OF MEMORY. THIS ADDRESS IS GIVEN TO XDA
15     ;           AS START OF RING BUFFER.
16     ;   FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS.
17     ;OUTPUTS:
18     ;   R1 - SIZE OF RING BUFFER IN WORDS IF NO ERROR,
19     ;   R4 - ADDRESS OF XDAIP REGISTER IN XDA,
20     ;   R5 - UNCHANGED,
21     ;
22     ;   Z CLR  IF NO ERROR,
23     ;   Z SET  IF ANY ERROR REPORTED
24     ;
25     ;CHECK IF ENOUGH FREE MEMORY FOR RING BUFFER
26
27 036670 XDAINT:
28 036670 010346      MOV     R3,-(SP)           ;;PUSH R3 ON STACK
29 036672 010400      MOV     R4,R0           ;GET MESSAGE LENGTH
30 036674 000300      SWAB    R0
31 036676 042700 177770 BIC     #177770,R0
32 036702 004737 034416 JSR     PC,CLOG           ;COMPUTE LOGARITHMIC VALUE
33 036706 010102      MOV     R1,R2           ;SAVE RESULT IN R2
34 036710 010400      MOV     R4,R0           ;GET COMMAND LENGTH
35 036712 000300      SWAB    R0
36 036714 006000      ROR     R0
37 036716 006000      ROR     R0
38 036720 006000      ROR     R0
39 036722 042700 177770 BIC     #177770,R0
40 036726 004737 034416 JSR     PC,CLOG           ;COMPUTE LOGARITHMIC VALUE
41 036732 060201      ADD     R2,R1           ;ADD THE TWO RESULTS
42 036734 006301      ASL     R1           ;MULTIPLY BY 2 WORDS PER RING
43 036736 062701 000002 ADD     #<HC.ISZ>/2,R1   ;ADD SPACE FOR INTERRUPT INDICATORS
44 036742 020137 002200 CMP     R1,FSIZE         ;COMPARE WITH SIZE OF FREE MEMORY
45 036746 101402      BLOS   1$
46 036750 000137 040100 JMP     FMERR            ;FATAL ERROR IF NOT ENOUGH MEMORY
47
48     ;FILL HOST COMMUNICATION AREA WITH ALL ONES
49
50 036754 013702 002324 1$:  MOV     HCOM,R2         ; LOAD COMM AREA ADDRESS
51 036760 010103      MOV     R1,R3         ;GET SIZE OF RING BUFFER
52 036762 012722 177777 2$:  MOV     #-1,(R2)+      ;WRITE ONES TO BUFFER
53 036766 005303      DEC     R3           ;COUNT THE WORDS IN BUFFER
54 036770 003374      BGT     2$           ;LOOP UNTIL ENTIRE BUFFER WRITTEN
55
56     ;DO THE INITIALIZATION
57

```

```

58 036772 004737 037200      JSR    PC,XDAIST      ;DO FIRST THREE STEPS
59 036776 103475              BCS    9$             ;GET OUT IF MICROCODE REPORTED FAILURE
60 037000 012364 000002      MOV    (R3)+,2(R4)    ;WRITE NEXT WORD TO XDASA REGISTER
61 037004 012700 000310      MOV    #200.,R0      ;GET TRY COUNTER
62 037010                    3$:
63 037010 016402 000002      MOV    2(R4),R2      ;LOOK AT XDASA
64 037014 001420              BEQ    5$
65 037016 100015              BPL    4$
66
67 037020 010237 002316      MOV    R2,SAVAL      ; SAVE CONTENTS
68 037024 012737 011061 002552  MOV    #ERM102,BASEMSG ; "SA ERROR DURING STEP 3"
69 037032 012737 000003 002370  MOV    #B.SADDR!B.SAVAL,EXTMSK ; FLAG ADDRESS AND VALUE
70 037040 104455              TRAP   C#ERDF
    037042 000146              .WORD 102
    037044 010667              .WORD HDR101
    037046 015266              .WORD ERR101
71 037050 000450              BR     9$
72
73 037052 005300              4$: DEC    R0
74 037054 001355              BNE    3$
75 037056 010264 000002      5$: MOV    R2,2(R4)      ;WRITE 0 TO XDASA (PURGE)
76 037062 011402              MOV    (R4),R2      ;READ FROM XDAIP (POLL)
77 037064 004737 037606      JSR    PC,XDARSP     ;WAIT FOR STEP OR ERROR BIT
78 037070 103440              BCS    9$             ;GET OUT IF MICROCODE REPORTED FAILURE
79 037072 010146              MOV    R1,-(SP)      ;;PUSH R1 ON STACK
80 037074 004733              JSR    PC,@(R3)+     ;CALL LAST ROUTINE
81 037076 012601              MOV    (SP)+,R1     ;;POP STACK INTO R1
82
83                          ;CHECK HOST COMMUNICATION AREA FOR ALL ZEROS
84
85 037100 013702 002324      MOV    HCOM,R2      ; LOAD COMM AREA ADDRESS
86 037104 010103              MOV    R1,R3        ;GET SIZE OF RING BUFFER
87 037106 005722              6$: TST    (R2)+      ;CHECK WORD IN BUFFER
88 037110 001003              BNE    7$           ;GO TO ERROR REPORTER IF NOT ZERO
89 037112 005303              DEC    R3            ;COUNT THE WORDS IN BUFFER
90 037114 003374              BGT    6$           ;LOOP UNTIL ALL WORDS CHECKED
91 037116 000412              BR     8$
92 037120                    7$:
93 037120 016402 000002      MOV    2(R4),R2
94 037124 012737 011145 002552  MOV    #ERM103,BASEMSG ; "DID NOT CLEAR MEMORY"
95 037132 104455              TRAP   C#ERDF
    037134 000147              .WORD 103
    037136 010667              .WORD HDR101
    037140 015266              .WORD ERR101
96 037142 000413              BR     9$
97
98                          ;SEND GO BIT TO XDASA REGISTER TO END INITIALIZATION
99
100 037144 016500 000004      8$: MOV    C.BST(R5),R0 ;GET BURST VALUE
101 037150 006300              ASL    R0            ;SHIFT TO POSITION
102 037152 006300              ASL    R0
103 037154 052700 000001      BIS    #SA.GO,R0     ;SET THE GO BIT
104 037160 010064 000002      MOV    R0,2(R4)     ;SEND TO XDA
105 037164 012603              MOV    (SP)+,R3     ;;POP STACK INTO R3
106 037166 000244              CLZ
107 037170 000207              RTS    PC
108

```

109
110

;ERROR RETURN

111 037172
037172 012603
112 037174 000264
113 037176 000207

9*:

MOV (SP)+,R3
SEZ
RTS PC

;;POP STACK INTO R3
;SET Z TO INDICATE ERROR OCCURRED


```

49 037364 103414          BCS      4$          ;EXIT IF ERROR
50 037366 004733          JSR      PC,@(R3)+   ;CALL RESPONSE CHECKER FOR STEP
51 037370 103412          BCS      4$          ;GET OUT IF ERROR
52 037372 006337 040002  ASL      XDARSD     ;SHIFT TO NEXT STEP BIT
53 037376 032737 040000 040002 BIT      @SA.S4,XDARSD ;CHECK IF NOW AT STEP 4
54 037404 001003          BNE      3$          ;GET OUT IF SO
55 037406 012364 000002  MOV      (R3)+,2(R4) ;WRITE DATA TO XDASA REGISTER
56 037412 000762          BR       2$          ;STAY IN LOOP
57
58 037414 000241          3$:      CLC          ;CLEAR CARRY FOR NO ERROR INDICATION
59 037416          4$:
    037416 012601          MOV      (SP)+,R1    ;:POP STACK INTO R1
60 037420 000207          RTS      PC
61
62          ;DATA TO BE SENT AND RECEIVED BY XDA INITIALIZATION
63
64 037422 037440          INITBL: .WORD   RSP.S1      ;1ST WORD RESPONSE CHECK ROUTINE
65 037424 000000          SND.S1: .WORD   0          ;1ST WORD TO SEND TO XDASA
66 037426 037452          .WORD   RSP.S2      ;2ND WORD RESPONSE CHECK ROUTINE
67 037430 000000          SND.S2: .WORD   0          ;2ND WORD TO SEND TO XDASA
68 037432 037472          .WORD   RSP.S3      ;3RD WORD RESPONSE CHECK ROUTINE
69 037434 000000          SND.S3: .WORD   0          ;3RD WORD TO SEND TO XDASA
70 037436 037510          .WORD   RSP.S4      ;4TH WORD RESPONSE CHECK ROUTINE
71
72          ;
73          ; RESPONSE CHECK FOR FIRST WORD (STEP 1) FROM XDASA
74 037440          ; CHECK FOR PROPER CONTROLLER TYPE
75 037440 012701 004400  RSP.S1: MOV      @SA.S1+SA.DI,R1    ;SET STEP 1 & ENHANCED DIAGNOSTIC BITS
76 037444 042702 001140  BIC      @SA.QB+SA.MP+40,R2 ;IGNORE 22 BIT ADRS & MAPPING BITS
77 037450          10$:
78 037450 000434          BR       RSP.CK      ;NOW DO A RESPONSE CHECK
79
80          ;
81          ; RESPONSE CHECK FOR SECOND WORD (STEP 2) FROM XDASA
82 037452          ; CHECK FOR ECHO OF INTI AND VECTOR
83 037452 013701 037424  RSP.S2: MOV      SND.S1,R1      ;GET WORD SENT TO XDASA
84 037456 000301          SWAB    R1          ;GET HIGH 8 BITS
85 037460 042701 177400  BIC      @177400,R1
86 037464 052701 010000  BIS      @SA.S2,R1      ;SET STEP 2 BIT
87 037470 000424          BR       RSP.CK      ;NOW DO A RESPONSE CHECK
88
89          ;
90          ; RESPONSE CHECK FOR THIRD WORD (STEP 3) FROM XDASA
91 037472          ; CHECK FOR ECHO OF MESSAGE AND COMMAND RING LENGTHS
92 037472 013701 037424  RSP.S3: MOV      SND.S1,R1      ;GET WORD SENT TO XDASA
93 037476 042701 177400  BIC      @177400,R1      ;JUST LOW 8 BITS
94 037502 052701 020000  BIS      @SA.S3,R1      ;SET STEP 3 BIT
95 037506 000415          BR       RSP.CK      ;NOW DO A RESPONSE CHECK
96
97          ;
98          ; RESPONSE CHECK FOR FOURTH WORD (STEP 4) FROM XDASA
99 037510          ; CHECK FOR ECHO OF PURGE AND LFAIL BITS
100 037510 010201          RSP.S4: MOV      R2,R1      ;GET RESPONSE FROM XDA
101 037512 042701 137400  BIC      @+C<SA.S4+SA.CNT+SA.MCV>,R1 ;KEEP MICROCODE VERSION AND STEP 4
102 037516 032701 000360  BIT      @SA.CNT,R1     ;CHECK WITH CONTROLLER MODEL
103 037522 001404          BEQ     1$          ;IF ZERO, XDA50(M7161)
104 037524 042765 000040 000014  BIC      @CT.U50,C.FLG(R5) ;ELSE, XDA52(M7485)

```

```
105 037532 000403          BR      2$          ;AND BRANCH
106 037534          1$:
107 037534 052765 000040 000014  BIS     @C7.U50,C.FLG(R5)
108 037542          2$:
109
110          ; RESPONSE CHECK, COMPARE EXPECTED DATA IN R1 WITH ACTUAL DATA IN R2
111 037542          RSP.CK:
112 037542 020102          CMP     R1,R2          ;COMPARE THE DATA
113 037544 001417          BEQ     1$          ;EXIT IF COMPARED CORRECTLY
114
115 037546 010137 002520          MOV     R1,SAEXP          ; SAVE EXPECTED VALUE
116 037552 010237 002316          MOV     R2,SAVAL          ; SAVE ACTUAL CONTENTS
117 037556 012737 011061 002552  MOV     @ERM102,BASEMSG    ; "SA REGISTER ERROR"
118 037564 012737 000007 002370  MOV     @B.SADDR!B.SAVAL!B.SAEXP,EXTMSK; FLAG ALL FIELDS
119 037572 104455          TRAP   C$ERDF
      037574 000146          .WORD  102
      037576 010667          .WORD  HDR101
      037600 015266          .WORD  ERR101
120 037602 000261          SEC
121 037604          1$:
122 037604 000207          RTS     PC
```

```

1      .SBTTL XDARSP WAIT FOR XDA INITIALIZE RESPONSE
2      ;XDARSP
3      ;
4      ;WAIT FOR XDA TO RESPOND WITH DATA IN XDASA REGISTER.
5      ;EITHER STEP BIT FROM MASK IN LOCATION XDARSD OR ERROR BIT
6      ;WILL CAUSE A TERMINATION.
7      ;AN ERROR MESSAGE WILL BE PRINTED IF THE XDA DOES NOT RESPOND
8      ;IN 10 SECONDS OR IF ERROR SETS.
9      ;
10     ;INPUTS:
11     ;   XDASRD - MASK OF STEP BIT TO LOOK FOR
12     ;   R5 - ADDRESS OF CONTROLLER TABLE
13     ;   R4 - ADDRESS OF XDAIP REGISTER
14     ;OUTPUTS:
15     ;   ERROR MESSAGE IF TIME OUT ON RESPONSE OR ERROR BIT SETS
16     ;   R2 - DATA FROM XDASA REGISTER
17     ;   CARRY SET IF ERROR BIT SETS OR TIME OUT
18     ;
19     XDARSP:
20     037606 010146          MOV     R1,-(SP)           ;;PUSH R1 ON STACK
21     037610 013737 040002 002320  MOV     XDARSD,SAEXP      ; SAVE EXPECTED VALUE
22     037616 052737 100000 040002  BIS     #SA.ERR,XDARSD   ;SET ERROR BIT IN MASK WORD
23     037624 012700 000012          MOV     #10.,R0          ;SET UP FOR 10 SECOND TIMEOUT
24     037630 010501          MOV     R5,R1            ;POINT TO COUNTER IN CONTROLLER TABLE
25     037632 062701 000022          ADD     #C.TO,R1
26     037636 004737 034334          JSR    PC,SETTO
27     037642 012601          MOV     (SP)+,R1        ;;POP STACK INTO R1
28     037644          1$:      MOV     2(R4),R2          ; READ SA REGISTER
29     037650 010237 002316          MOV     R2,SAVAL        ; AND SAVE CONTENTS
30     037654 033702 040002          BIT     XDARSD,R2      ; LOOK AT ERROR AND STEP BIT
31     037660 001030          BNE     3$             ;BRANCH IF EITHER SET
32     037662 104422          TRAP   C$BRK           ;>>>>>>>>>BREAK BACK TO MONITOR<<<<<<<<<<<<
33     037664 005737 002210          TST    KW.CSR          ;SEE IF CLOCK ON SYSTEM
34     037670 001765          BEQ    1$
35     037672 023765 002222 000024  CMP     KW.EL+2,C.TO(R5) ;CHECK IF TIME OUT OCCURRED
36     037700 101005          BHI    2$
37     037702 001360          BNE    1$
38     037704 023765 002220 000022  CMP     KW.EL,C.TO(R5)
39     037712 103754          BLO    1$
40     037714          2$:      ; REPORT FATAL ERROR
41     037714 012737 011061 002552  MOV     #ERM102,BASEMSG ; "STEP BIT ERROR"
42     037722 012737 000007 002370  MOV     #B.SADDR!B.SAVAL!B.SAEXP,EXTMSK; FLAG ALL FIELDS
43     037730 104455          TRAP   C$ERDF
44     037732 000146          .WORD  102
45     037734 010667          .WORD  HDR101
46     037736 015266          .WORD  ERQ101
47     037740 000414          BR     4$
48     037742          ; 3$:      CHECK IF ERROR BIT SET
49     037742 005702          TST    R2              ; READ TEST CONTENTS
50     037744 100014          BPL    5$              ; EXIT IF ERROR NOT SET
51
52     037746 012737 011061 002552  MOV     #ERM102,BASEMSG ; "RESIDENT DIAG FAILURE"
53     037754 012737 000003 002370  MOV     #B.SADDR!B.SAVAL,EXTMSK ; FLAG ADDRESS AND VALUE
  
```

```
54 037762 104455          TRAP    C$ERDF
    037764 000146          .WORD  102
    037766 010667          .WORD  HDR101
    037770 015266          .WORD  ERR101
55 037772                4$:
56 037772 000261          SEC
57 037774 000207          RTS    PC
58
59                          ;NORMAL EXIT
60
61 037776 000241          5$:   CLC                ;CLEAR CARRY AS NO ERROR INDICATION
62 040000 000207          RTS    PC
63
64                          ;LOCATION FOR STEP BIT MASK
65
66 040002 000000          XDARSD: .WORD  0          ;LOAD BY CALLING ROUTINE
67
```



```
1 .SBTTL HCOMM INITIALIZE HOST COMM AREA
2 ;HCOMM
3 ;
4 ;ALLOCATES MEMORY FOR HOST COMMUNICATION AREA AND PACKET BUFFERS WITH ONE
5 ;DESCRIPTOR IN EACH RING. TO BE CALLED AFTER INITIALIZING
6 ;A CONTROLLER WITH SA.MSG=0 AND SA.CMD=0.
7 ;
8 ;INPUTS:
9 ; R5 - ADDRESS OF CONTROLLER TABLE
10 ;OUTPUTS:
11 ; CONTROLLER TABLE POINTING TO HOST COMMUNICATION AREA,
12 ; RING POINTERS TO PACKETS,
13 ; R4 - ADDRESS OF HOST COMMUNICATION AREA
14
15 040004 012701 000074 HCOMM: MOV #<HC.SIZ+4>/2,R1 ; LOAD COMM AREA SIZE + 2
16 040010 004737 040054 JSR PC,ALOCM ; ALLOCATE
17 040014 010104 MOV R1,R4 ; SAVE ADDRESS
18 040016 010437 002324 MOV R4,HCOM ; LOCALLY
19 040022 062701 000020 ADD #HC.MPK,R1 ; COMPUTE START OF MESSAGE PACKET
20 040026 010164 000004 MOV R1,HC.MSG(R4) ; PLACE IN RING
21 040032 010137 002330 MOV R1,MSGPCK ; AND SAVE LOCALLY
22 040036 062701 000064 ADD #<HC.CPK-HC.MPK>,R1 ; COMPUTE START OF COMMAND PACKET
23 040042 010164 000010 MOV R1,HC.CMD(R4) ; PLACE IN RING
24 040046 010137 002326 MOV R1,CMDPCK ; AND SAVE LOCALLY
25 040052 000207 RTS PC
26
```

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
32
33
34
35
36
37
38
39

```

.SBTTL MEMORY ALLOCATION ROUTINES
;ALOCM
;
;ALLOCATE A BLOCK OF FREE MEMORY. REPORT ERROR IF MEMORY EXH'USTED.
;
;INPUTS:
; R1 - NUMBER OF WORDS TO ALLOCATE
; FFREE - FIRST FREE WORD IN MEMORY
; FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS
;
;OUTPUTS:
; R1 - ADDRESS OF FIRST WORD OF ALLOCATED MEMORY
; FFREE - NEW FIRST FREE WORD IN MEMORY
; FSIZE - SIZE OF FREE MEMORY LEFT AFTER ALLOCATION
;SYSTEM FATAL ERROR WILL BE REPORTED IF NOT ENOUGH MEMORY AVAILABLE
;AND ENTIRE PROGRAM WILL BE STOPPED.

ALOCM:
MOV FFREE, -(SP) ;;PUSH FFREE ON STACK
SUB R1, FSIZE ;;REDUCE SIZE OF FREE MEMORY
BLT FMERR ;;REPORT ERROR IF NOT ENOUGH MEMORY
ADD R1, R1 ;;CHANGE WORDS TO BYTES
ADD R1, FFREE ;;CALCULATE NEW START OF FREE MEMORY
MOV (SP), R1 ;;PCP STACK INTO R1
RTS PC

;FMERR
;
;MEMORY ALLOCATION ERROR
;
;THIS ROUTINE PRINTS A SYSTEM FATAL ERROR AND EXITS THE TEST

FMERR:
MOV @ERM001, BASEMSG ; "MEMORY ALLOCATION ERROR"
TRAP C$ERSF
.WORD 1
.WORD HDR001
.WORD ERR001
TRAP C$DCLN

.EVEN

L10021:
TRAP C$INIT

```

```

040054 013746 002176
040060 160137 002200
040064 002405
040066 060101
040070 060137 002176
040074 012601
040076 000207

040100
040100 012737 010527 002552
040106 104454
040110 000001
040112 010472
040114 015220
040116 104444

040120
040120 104411

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 133
AUTODROP SECTION

```
1          .SBTTL AUTODROP SECTION
2
3          ;**
4          ; THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
5          ; THE "ADR" FLAG WAS SET THE UNIT(S) UNDER TEST ARE CHECKED TO
6          ; SEE IF THEY WILL RESPON. . THOSE THAT DON'T ARE IMMEDIATELY
7          ; DROPPED FROM TESTING.
8          ;--
9
10         L$AUTO::
11
12         L10022:
           TRAP      C$AUTO
           040122 104461
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

.SBTTL CLEANUP CODING SECTION

: THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
: AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
:--

L\$CLEAN::

040124	112700	000015	MOVB	@CR,RO	:STORE @CR IN RO AND
040130	004737	033066	JSR	PC,PRINTC	:PRINT THE CHARACTER
040134	004737	016224	JSR	PC,RESET	:RESET ALL UNITS
040140	104432		TRAP	C\$EXIT	
040142	000002		.WORD	L10023-.	
			.EVEN		
040144			L10023:		
040144	104412		TRAP	C\$CLEAN	

```

1          .SBTTL DROP UNIT SECTION
2
3          ;**
4          ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
5          ; TO NO LONGER BE TESTED.
6          ;--
7
8 040146    L$DU::
9
10 040146 005737 002442      TST     STATUS      ; TEST EXIT STATUS
11 040152 100410              BMI     20$          ; BRANCH IF FATAL
12 040154              10$:      ; "TEST COMPLETE."
13 040154 013746 002074      MOV     L$LUN, -(SP)  ; PUSH L$LUN ON STACK
14 040160 012746 040216      MOV     @CMPUUT, (SP) ; PUSH @CMPUUT ON STACK
15 040164 004137 033244      JSR    R1,LPNTF     ; CALL LPNTF PRINT ROUTINE
16 040170 000004              .WORD   ARG.CT       ; ARGUMENT COUNT * 2
17 040172 000407              BR      30$
18 040174              20$:      ; "DROPPED."
19 040174 013746 002074      MOV     L$LUN, -(SP)  ; PUSH L$LUN ON STACK
20 040200 012746 040254      MOVB   @DRPUUT, -(SP) ; PUSH @DRPUUT ON STACK
21 040204 004137 033244      JSR    R1,LPNTF     ; CALL LPNTF PRINT ROUTINE
22 040210 000004              .WORD   ARG.CT       ; ARGUMENT COUNT * 2
23 040212              30$:      ;
24 040212 000167              .WORD   J$JMP
25 040214 000066              .WORD   L10024-2-.
26
27          .NLIST BEX
28          042 CMPUUT: .ASCIZ /N2"UNIT "D8" TEST COMPLETE."N/
29          042 DRPUUT: .ASCIZ /N2'UNIT "D8" DROPPED."N/
30          .LIST BEX
31          .EVEN
32
33          L10024: TRAP C$DU
    
```

ADD UNIT SECTION

```
1          .SBTTL  ADD UNIT SECTION
2
3          ;**
4          ; THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
5          ; TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
6          ; TO THE TEST CYCLE.
7          ;--
8
9 040306    L$AU::
10
11 040306 000167    .WORD  J$JMP
12 040310 000000    .WORD  L10025-2-.
13
14          .EVEN
15 040312    L10025:
16 040312 104452    TRAP   C$AU
17
18
19
```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr-85 10:42 Page 137
 TEST1: MANUAL BAD BLOCK REPLACEMENT

```

1          .SBTTL TEST1: MANUAL BAD BLOCK REPLACEMENT
2
3          T1::
4
5 040314   005737   002150           TST      MANRPL           ; TEST FOR MANUAL REPLACEMENT
6 040320   001002                   BNE      10$             ; BRANCH IF YES
7 040322   104432           TRAP     C$EXIT
   040324   000276           .WORD    L10026-.
8 040326           10$:
9 040326   104450           TRAP     C$MANI
10 040330   103402          BCS      15$
11 040332   104432           TRAP     C$EXIT
   040334   000266           .WORD    L10026-.
12 040336           15$:
13 040336   012746   007061          MOV      #INF310,-(SP)   ; PUSH #INF310 ON STACK
   040342   004137   033244          JSR      R1,LPNTF        ; CALL LPNTF PRINT ROUTINE
   040346   000002           .WORD    ARG.CT         ; ARGUMENT COUNT * 2
14
15 040350   012737   000000   002074          MOV      #0,L$LUN       ; ZERO LUN
16 040356   013737   002230   002304          MOV      CTRLRLST,CTLRTBL ; LOAD TABLE ADDRESS
17 040364           20$:
18 040364   004737   021050          JSR      PC,UNITINI     ; INITIALIZE UNIT UNDER TEST
19 040370   040602           1200$          ; ERROR EXIT
20 040372           50$:
21 040372   012701   022707          MOV      #SWMSG4,R1     ; LOAD QUESTION ADDRESS
22 040376   012702   002654          MOV      #REPLY,R2     ; LOAD ANSWER ADDRESS
23 040402   012712   000000          MOV      #0,(R2)       ; DEFAULT IS "NO"
24 040406   004737   023126          JSR      PC,GETLOG      ; ASK OPERATOR
25 040412   105737   002654          TSTB     REPLY          ; TEST REPLY
26 040416   001403           BEQ      100$          ; BRANCH IF "NO"
27
28 040420   004737   032364          JSR      PC,DSPDSC      ; ELSE, DISPLAY DESCRIPTORS
29 040424   040602           1200$          ; ERROR EXIT
30
31          ; MANUAL BAD BLOCK REPLACEMENT.
32          ; FETCH LBN FROM OPERATOR; CHECK LIMITS.
33 040426           100$:
34 040426   012701   014705          MOV      #TXT000,R1     ; LOAD MESSAGE ADDRESS
35 040432   012704   002446          MOV      #LBN,R4       ; LOAD LBN ADDRESS
36 040436   004737   023410          JSR      PC,GETLBN      ; GET LBN FROM OPERATOR
37 040442   001425           BEQ      200$          ; EXIT IF TTYIN = 0
38 040444           110$:
39 040444   004737   023604          JSR      PC,READLBN     ; READ SUSPECT LBN
40 040450   040602           1200$          ; ERROR EXIT
41 040452   004737   017562          JSR      PC,CACHTST     ; UPDATE CACHING TABLE
42 040456   040602           1200$          ; ERROR EXIT
43
44          ; REPLACE LBN IF REPLACEMENT ENABLED.
45 040460           130$:
46 040460   005737   002160          TST      ENBRPL         ; TEST FOR REPLACEMENT ENABLED
47 040464   001004                   BNE      150$          ; BRANCH IF YES
48 040466   004737   020002          JSR      PC,TYPSTAT     ; TYPE REPLACEMENT INFO
49 040472   000137   040426          JMP      100$
50 040476           150$:
51 040476   012737   000006   002546          MOV      #6,NXTSTEP     ; ENTRY IS STEP6
52 040504   004737   024156          JSR      PC,REPLACE     ; REPLACE LBN
53 040510   040602           1200$          ; ERROR RETURN

```

CZUDL0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr-85 10:42 Page 137-1
 TEST1: MANUAL BAD BLOCK REPLACEMENT

```

54 040512 000137 040426          JMP      100$          ; CONTINUE
55 040516          200$:          MOV      @INF311,-(SP) ; PUSH @INF311 ON STACK
56 040516 012746 007135          JSR      R1,LPNTF     ; CALL LPNTF PRINT ROUTINE
    040522 004137 033244          .WORD   ARG.CT       ; ARGUMENT COUNT * 2
    040526 000002          TRAP    C$DRPT
57 040530 104424
58
59 040532 012701 022707          MOV      @SWMSG4,R1  ; LOAD QUESTION ADDRESS
60 040536 012702 002654          MOV      @REPLY,R2  ; LOAD ANSWER ADDRESS
61 040542 012712 000000          MOV      @0,(R2)    ; DEFAULT IS "NO"
62 040546 004737 023126          JSR      PC,GETLOG   ; ASK OPERATOR
63 040552 001403          BEQ     400$         ; BRANCH IF NO
64
65 040554 004737 032364          JSR      PC,DSPDSC   ; ELSE, DISPLAY DESCRIPTORS
66 040560 040602          1200$:          ; ERROR EXIT
67
68          ; NORMAL EXIT
69 040562          400$:          MOV      @0,STATUS   ; SET SUCCESS
70 040562 012737 000000 002442    MOV      L$LUN,RO
71 040570 013700 002074          TRAP    C$DODU
    040574 104451          TRAP    C$EXIT
72 040576 104432          .WORD   L10026-.
    040600 000022
73
74          ; ERROR EXIT
75 040602          1200$:          MOV      @-1,STATUS  ; SET FATAL STATUS
76 040602 012737 177777 002442    MOV      L$LUN,RO
77 040610 013700 002074          TRAP    C$DODU
    040614 104451          TRAP    C$EXIT
78 040616 104432          .WORD   L10026-.
    040620 000002
79
80          .EVEN
81
82 040622          L10026:        TRAP    C$ETST
    040622 104401
83

```


TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1      .SBTTL TEST2: AUTOMATIC BAD BLOCK REPLACEMENT
2
3      T2::
4
5      040624 005737 002150      TST      MANRPL      ; TEST FOR MANUAL REPLACEMENT
6      040630 001402      BEQ      10$      ; BRANCH IF NO
7      040632 104432      TRAP     C$EXIT
8      040634 001306      .WORD   L10027-.
9      040636      10$:      MOV      #INF320,-(SP)      ; PUSH #INF320 ON STACK
10     040642 012746 007164      JSR     R1,LPNTF      ; CALL LPNTF PRINT ROUTINE
11     040644 004137 033244      .WORD  ARG.CT      ; ARGUMENT COUNT * 2
12     040646 000002
13     10 040650 012746 007242      MOV      #INF321,-(SP)      ; PUSH #INF321 ON STACK
14     040652 004137 033254      JSR     R1,LPNTB      ; CALL LPNTB PRINT ROUTINE
15     040654 000002      .WORD  ARG.CT      ; ARGUMENT COUNT * 2
16     11 040662 012746 007630      MOV      #INF322,-(SP)      ; PUSH #INF322 ON STACK
17     040664 004137 033254      JSR     R1,LPNTB      ; CALL LPNTB PRINT ROUTINE
18     040666 000002      .WORD  ARG.CT      ; ARGUMENT COUNT * 2
19
20     12 040674 012737 177777 002074      MOV      #-1,L$LUN      ; INITIALIZE LUN
21     13 040702 012737 177776 002300      MOV      #-2,TBLNDX      ; INITIALIZE TABLE INDEX
22     14 040710      100$:      INC      L$LUN      ; INCREMENT LUN
23     15 040710 005237 002074      CMP      L$LUN,L$UNIT      ; TEST IF ALL UNITS DONE
24     16 040714 023737 002074 002012      BLT     105$      ; BRANCH IF NO
25     17 040722 002402      JMP     1200$      ; ELSE, FINISHED
26     18 040724 000137 042136
27     19 040730      105$:      ADD      #2,TBLNDX      ; INCREMENT TABLE INDEX
28     20 040730 062737 000002 002300      MOV      TBLNDX,R2      ; LOAD INDEX
29     21 040736 013702 002300      MOV      CTRLST(R2),CTRLTBL      ; LOAD ADDRESS
30     22 040742 016237 002230 002304      BEQ     100$      ; BRANCH IF NONE
31     23 040750 001757
32     24 040752      120$:      JSR     PC,UNITINI      ; INITIALIZE UNIT UNDER TEST
33     25 040752 004737 021050      1000$
34     26 040756 042104      ; ERROR EXIT
35
36     28
37     29      ; AUTOMATIC BAD BLOCK REPLACEMENT.
38     30      ; INITIALIZE TEST VARIABLES
39
40     31 040760      300$:      ; HEADER
41     32 040760 012746 007027      MOV      #INF302,-(SP)      ; PUSH #INF302 ON STACK
42     33 040764 004137 033254      JSR     R1,LPNTB      ; CALL LPNTB PRINT ROUTINE
43     34 040770 000002      .WORD  ARG.CT      ; ARGUMENT COUNT * 2
44
45     35 040772 104450      TRAP     C$MANI
46     36 040774 103033      BCC     305$
47
48     37 040776 005737 002650      TST     DEBUG      ; TEST DEBUG FLAG
49     38 041002 001430      BEQ     305$      ; BRANCH IF NOT SET
50
51     39
52     40      ; GET MINIMUM AND MAXIMUM LBN'S FOR DEBUG
53
54     41 041004      301$:      MOV      #0,MINLBN
55     42 041004 012737 000000 002600      MOV      #0,MINLBN+2
56     43 041012 012737 000000 002602      MOV      #TX.LB1,R1
57     44 041020 012701 015060      MOV      #MINLBN,R4
58     45 041024 012704 002600      JSR     PC,GETLBN
59     46 041030 004737 023410
60     47 041034      302$:      MOV      UNSIZ,MAXLBN
61     48 041034 013737 002430 002604

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 138-1
TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```
49 041042 013737 002432 002606      MOV    UNSIZ+2,MAXLBN+2
50 041050 012701 015100      MOV    @TX.LB2,R1
51 041054 012704 002604      MOV    @MAXLBN,R4
52 041060 004737 023410      JSR    PC,GETLBN
53
54                                     ;
55 041064                                     ; 305$: INITIALIZE STARTING GLBN
56 041064 013737 002600 002562      MOV    MINLBN,GLBN          ; LOAD INITIAL LBN (LOW)
57 041072 013737 002602 002564      MOV    MINLBN+2,GLBN+2     ; LOAD INITIAL LBN (HIGH)
58 041100 012737 000200 001000      MOV    @GRPSIZ,BLKSIZ     ; INITIALIZE FOR .8 BLOCK GROUPS
59 041106 000137 041630      JMP    600$               ; TEST FOR EOM BEFORE ACCESS
60
```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 139
 TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1
2           ; ACCESS NEXT GROUP OF BLOCKS
3 041112    ; 310$:
4 041112    104422    TRAP    C$BRK
5 041114    005037    002446    CLR    LBN                ; RESET TEST LBN (LOW)
6 041120    005037    002450    CLR    LBN+2            ; RESET TEST LBN (HIGH)
7 041124    005037    002516    CLR    LBNSTAT         ; RESET LBN STATUS
8 041130    005037    002520    CLR    RPLSTAT         ; RESET REPLACEMENT STATUS
9 041134    105037    002515    CLRB   SFLAG           ; RESET SEARCH FLAG
10
11 041140    012737    044472    002340    MOV    #IBUFF,UADR     ; LOAD IMAGE BUFFER ADDRESS
12 041146    004737    030140    JSR    PC,INITAC      ; INITIALIZE ACCESS PARAMETERS
13
14 041152    012737    000000    002346    MOV    #0,CMDMOD      ; CLEAR MODIFIERS
15                                     ; SETUP TO ISSUE OP.ACC COMMAND
16 041160    012700    000020    MOV    #OP.ACC,R0     ; LOAD MSCP OP CODE
17 041164    004737    030204    JSR    PC,ISSUE       ; ISSUE COMMAND TO THE XDA
18 041170    042104    BEQ    1000$          ; HERE IF FATAL ERROR,
19 041172    001520    BEQ    500$           ; BRANCH IF COMMAND SUCCESSFUL,
20
21           ; ERROR RETURN: CALCULATE ERROR LBN.
22           ; 320$:
23 041174    013701    002324    MOV    HCOM,R1        ; LOAD COMM AREA ADDRESS
24 041174    016103    000034    MOV    HC.MPK+P.BCNT(R1),R3 ; UNLOAD ACCESS
25 041200    016104    000036    MOV    HC.MPK+P.BCNT+2(R1),R4 ; BYTE COUNT
26 041210    012700    001000    MOV    #512.,R0      ; DIVISOR IS BYTES PER BLOCK
27 041214    004737    034436    JSR    PC,DIVIDE     ; DIVID
28 041220    060337    002562    ADD    R3,GLBN        ; ADVANCE LBN
29 041224    005537    002564    ADC    GLBN+2         ; TO BAD BLOCK
30 041230    013737    002562    002446    MOV    GLBN,LBN      ; TRANSFER GROUP START
31 041236    013737    002564    002450    MOV    GLBN+2,LBN+2  ; TO LBN
32
33           ; CHECK FOR DATA ERROR.
34           ; 350$:
35 041244    013700    002354    MOV    EMSTAT,R0     ; LOAD STATUS
36 041250    042700    177740    BIC    #CST.MSK,R0   ; CLEAR ANY SUBCODES
37 041254    022700    000010    CMP    #ST.DAT,R0   ; TEST FOR DATA ERROR
38 041260    001414    BEQ    400$          ; BRANCH IF YES
39
40           ; UNRECOGNIZED ERROR CODE
41           ; 380$:
42 041262    012737    012246    002552    MOV    #ERM303,BASEMSG ; "FAILED ACCESS..."
43 041270    012737    000140    002370    MOV    #B.MSCP,EXTMSK ; FLAG ALL FIELDS
44 041276    104455    TRAP    C$ERDF
45 041300    000457    .WORD   303
46 041302    012043    .WORD   HDR301
47 041304    015570    .WORD   ERR302
48 041306    000137    042104    JMP    1000$         ; EXIT TEST

```

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr 85 10:42 Page 140
 TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1
2
3 041312
4 041312 004737 023604
5 041316 042104
6 041320 001004
7
8 041322 004737 017562
9 041326 042104
10 041330 001407
11 041332
12 041332 032737 000200 002516
13 041340 001403
14 041342
15 041342 005737 002160
16 041346 001011
17 041350
18 041350 004737 020002
19 041354 062737 000001 002562
20 041362 005537 002564
21 041366 000137 041630
22
23
24 041372
25 041372 012737 000006 002546
26 041400 004737 024156
27 041404 042104
28 041406 032737 000400 002516
29 041414 001405
30 041416 062737 000001 002562
31 041424 005537 002564
32 041430
33 041430 000137 041630
34

```

```

; PROCESS DATA ERROR.
400$:
JSR PC,READLBN ; READ SUSPECT LBN
1000$ ; ERPOR RETURN
BNE 420$ ; BRANCH IF REPLACE

JSR PC,CACHTST ; UPDATE CACHING TABLE
1000$ ; ERROR EXIT
BEQ 440$ ; BRANCH IF NO REPLACE

420$:
BIT #TF.BBR,LBNSTAT ; TEST FOR BBR
BEQ 440$ ; BRANCH IF NO

425$:
TST ENBRPL ; TEST FOR REPLACEMENT ENABLED
BNE 450$ ; BRANCH IF YES

440$:
JSR PC,TYPSTAT ; ELSE, TYPE INFO
ADD #1,GLBN ; BUMP TO NEXT LBN
ADC GLBN+2 ; ADD CARRY
JMP 600$ ; CONTINUE WITH NEXT LBN

; REPLACE LBN.
450$:
MOV #6,NXTSTEP ; ENTRY IS STEP6
JSR PC,REPLACE ; CALL REPLACEMENT ALGORITHM
1000$ ; ERROR RETURN
BIT #TF.WFE,LBNSTAT ; TEST FOR WFE
BEQ 455$ ; BRANCH IF NO
ADD #1,GLBN ; ELSE, BUMP LBN
ADC GLBN+2 ; PAST FER

455$:
JMP 600$ ; CONTINUE

```

CZUDL00 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr 85 10:42 Page 141
 TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1
2
3 041434
4 041434 132737 000200 002356
5 041442 001007
6 041444 063737 002572 002562
7 041452 005537 002564
8 041456 000137 041630
9
10
11 041462
12 041462 013704 002324
13 041466 016437 000054 002446
14 041474 016437 000056 002450
15 041502 013737 002446 002562
16 041510 013737 002450 002564
17
18
19 041516
20 041516 004737 023604
21 041522 042104
22 041524 001004
23
24 041526 004737 017562
25 041532 042104
26 041534 001403
27 041536
28 041536 005737 002160
29 041542 001011
30 041544
31 041544 004737 020002
32 041550 062737 000001 002562
33 041556 005537 002564
34 041562 000137 041630
35
36
37 041566
38 041566 012737 000006 002546
39 041574 004737 024156
40 041600 042104
41 041602 032737 000400 002516
42 041610 001405
43 041612 062737 000001 002562
44 041620 005537 002564
45 041624
46 041624 000137 041630
47

```

; NORMAL RETURN; CHECK FLAGS
 ; 500\$:
 BITB #EF.BBR,FLAGS ; TEST FLAGS FOR BBR
 BNE 510\$; BRANCH IF YES
 ADD BLKCNT,GLBN ; ELSE, MOVE TO NEXT GROUP
 ADC GLBN+2 ; ADD CARRY
 JMP 600\$; CONTINUE WITH NEXT LBN

 ; PROCESS BBR.
 ; 510\$:
 MOV HCOM,R4 ; LOAD HOST COMM AREA ADDRESS
 MOV HC.MPK+P.FBBK(R4),LBN ; UNLOAD FIRST BAD BLOCK
 MOV HC.MPK+P.FBBK+2(R4),LBN+2
 MOV LBN,GLBN
 MOV LBN+2,GLBN+2

 ; REREAD LBN.
 ; 520\$:
 JSR PC,READLBN ; READ SUSPECT LBN
 1000\$; ERROR RETURN
 BNE 530\$; BRANCH IF REPLACE

 JSR PC,CACHTST ; UPDATE CACHING TABLE
 1000\$; ERROR RETURN
 BEQ 540\$; BRANCH IF NO REPLACE

 ; 530\$:
 TST ENBRPL ; TEST FOR REPLACEMENT ENABLED
 BNE 550\$; BRANCH IF YES

 ; 540\$:
 JSR PC,TYPSTAT ; ELSE, TYPE INFO
 ADD #1,GLBN ; BUMP TO NEXT LBN
 ADC GLBN+2 ; ADD CARRY
 JMP 600\$; CONTINUE WITH NEXT LBN

 ; REPLACE LBN.
 ; 550\$:
 MOV #6,NXTSTEP ; ENTRY IS STEP6
 JSR PC,REPLACE ; CALL REPLACEMENT ALGORITHM
 1000\$; ERROR RETURN
 BIT #TF.WFE,LBNSTAT ; TEST FOR WFE
 BEQ 555\$; BRANCH IF NO
 ADD #1,GLBN ; ELSE, BUMP LBN
 ADC GLBN+2 ; PAST FER

 ; 555\$:
 JMP 600\$; CONTINUE

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page 142
 TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1
2 ; TEST FOR END OF MEDIA; RESET BYTE COUNT
3 041630 ; 600$:
4 041630 013701 002604 MOV MAXLBN,R1 ; LOAD TOTAL LBN'S (LOW)
5 041634 013702 002606 MOV MAXLBN+2,R2 ; LOAD TOTAL LBN'S (HIGH)
6 041640 ; 605$:
7 041640 163702 002564 SUB GLBN+2,R2 ; SUBTRACT START LBN (HIGH)
8 041644 103466 BLO 800$ ; DONE IF GLBN+2 > UNSIZ+2
9 041646 101030 BHI 630$ ; CONTINUE IF GLBN+2 < UNSIZ+2
10
11 041650 163701 002562 SUB GLBN,R1 ; ELSE, SUBTRACT START LBN (LOW)
12 041654 101462 BLOS 800$ ; DONE IF GLBN >= UNSIZ
13 041656 ; 610$:
14 041656 000240 NOP ;
15 041660 022701 000200 CMP #GRPSIZ,R1 ; COMPARE REMAINDER WITH GROUP SIZE
16 041664 101421 BLOS 630$ ; BRANCH IF REMAINING >= GRPSIZ
17 041666 012737 000000 002566 MOV #0,GBCNT ; ELSE, RESET
18 041674 012737 000000 002570 MOV #0,GBCNT+2 ; GROUP BYTE COUNT
19 041702 010137 002572 MOV R1,BLKCNT ; USE REMAINDER FOR BLOCK COUNT
20 041706 ; 620$:
21 041706 062737 001000 002566 ADD #BLKSIZ,GBCNT ; INCREMENT BYTE COUNT (LOW)
22 041714 005537 002570 ADC GBCNT+2 ; ADD CARRY
23 041720 005301 DEC R1 ; DECREMENT BLOCKS LEFT
24 041722 001371 BNE 620$ ; CONTINUE IF NOT ZERO
25 041724 000240 NOP ;
26 041726 000411 BR 640$ ;
27 041730 ; 630$:
28 041730 012737 000000 002566 MOV #0,GBCNT ; ELSE, RESET
29 041736 012737 000001 002570 MOV #1,GBCNT+2 ; BYTE COUNT
30 041744 012737 000200 002572 MOV #GRPSIZ,BLKCNT ; AND BLOCK COUNT
31 041752 ; 640$:
32 041752 062737 000001 002574 ADD #1,ACCESS ; BUMP ACCESS COUNT
33 041760 032737 000777 002574 BIT #777,ACCESS ; IS IT MOD 512.?
34 041766 001013 BNE 650$ ; BRANCH IF NO
35 ; ELSE, PRINT CURRENT LBN
36 041770 013746 002564 MOV GLBN+2,-(SP) ; PUSH GLBN+2 ON STACK
37 041774 013746 002562 MOV GLBN,-(SP) ; PUSH GLBN ON STACK
38 042000 012746 010166 MOV #INF405,-(SP) ; PUSH #INF405 ON STACK
39 042004 004137 033254 JSR R1,LPNTB ; CALL LPNTB PRINT ROUTINE
40 042010 000006 .WORD ARG.CT ; ARGUMENT COUNT * 2
41 042012 004737 020650 JSR PC,RTIME ; AND RUNTIME
42 042016 ; 650$:
43 042016 000137 041112 JMP 310$ ; ACCESS NEXT GROUP

```

CZLDL40 BBR Replacement Utility MACRO Y05.03c Friday 19-Apr 85 10:42 Page 143
 TEST2: AUTOMATIC BAD BLOCK REPLACEMENT

```

1
2 042022      800$:
3 042022 013746 002564      MOV      GLBN+2,-(SP)      ;PUSH GLBN+2 ON STACK
  042026 013746 002562      MOV      GLBN,-(SP)       ;PUSH GLBN ON STACK
  042032 012746 010166      MOV      @INF405,-(SP)   ;PUSH @INF405 ON STACK
  042036 004137 033254      JSR      R1,LPNTB        ;CALL LPNTB PRINT ROUTINE
  042042 000006      .WORD   ARG.CT          ;ARGUMENT COUNT * 2
4 042044 004737 020650      JSR      PC,RTIME        ; DISPLAY RUNTIME
5
6 042050 104424      TRAP     C$DRPT
7 042052 005737 002650      TST      DEBUG          ; TEST FOR DEBUG
8 042056 001402      BEQ      810$           ; BRANCH IF NO
9 042060 000137 041064      JMP      305$           ; ELSE LOOP
10 042064      810$:
11 042064 005737 002156      TST      DSPRCT         ; TEST FOR DISPLAY
12 042070 001403      BEQ      820$           ; BRANCH IF NO
13 042072 004737 032364      JSR      PC,DSPDSC      ; ELSE, DISPLAY DESCRIPTORS
14 042076 042100      820$:
15 042100      JMP      100$           ; CONTINUE
16 042100 000137 040710
17
18      ; FATAL EXIT; DROP UUT
19 042104      1000$:
20 042104 012737 177777 002442      MOV      @-1,STATUS     ; SET FATAL STATUS
21 042112 013702 002300      MOV      TBLNDX,R2      ; LOAD UNIT INDEX
22 042116 012762 000000 002230      MOV      @0,CTRLST(R2) ; ZERO CONTROLLER TABLE ADDRESS
23 042124 013700 002074      MOV      L$LUN,R0
  042130 104451      TRAP     C$DODU
24 042132 000137 040710      JMP      100$           ; AND CONTINUE
25
26      ; TEST EXIT
27 042136      1200$:
28 042136 104432      TRAP     C$EXIT
  042140 000002      .WORD   L10027-.
29
30      .EVEN
31
32 042142      L10027:
  042142 104401      TRAP     C$ETST
33
35

```

```

1      .SBTTL  HARDWARE PARAMETER CODING SECTION
3      :
4      : THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
5      : THAT ARE USED BY THE SUPERVISOR TO BUILD P TABLES.  THE
6      : MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7      : INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
8      : MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
9      : WITH THE OPERATOR.
10     :
11     :
12     042144 000027      .WORD  L10030-L$HARD/2
13     042146
14     000000      L$HARD::
15     000002      H.UBA   = 0      ;UNIBUS ADDRESS
16     000004      H.VEC   = 2      ;XDA VECTOR
17     000006      H.BRL   = 4      ;BR LEVEL
18     000010      H.BST   = 6      ;BURST RATE
19     000010      H.DRV   = 10     ;DRIVE NUMBER
20
21     042146 000031      .WORD  T$CODE
22     042150 042224      .WORD  MSGUBA
23     042152 160000      .WORD  T$LOLIM
24     042154 177774      .WORD  T$HILIM
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

;PRINT 'UNIBUS ADDRESS OF XDA?'

;PRINT 'VECTOR?'

;PRINT BR LEVEL?'

;PRINT 'UNIBUS BURST RATE?'

;PRINT 'DRIVE #'

```

30     042224      L10030:
31
35     042224      125      116      111  MSGUBA: .ASCIZ  \UNIBUS ADDRESS OF XDA\
36     042252      126      105      103  MSGVEC: .ASCIZ  \VECTOR\
37     042261      102      122      040  MSGBRL: .ASCIZ  \BR LEVEL\
38     042272      125      116      111  MSGBST: .ASCIZ  \UNIBUS BURST RATE\
39     042314      104      122      111  MSGLDR: .ASCIZ  \DRIVE #\
40     .EVEN
41
42
43
44

```



```

1          .SBTTL  SOFTWARE PARAMETER CODING SECTION
2
3          ;**
4          ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
5          ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
6          ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7          ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
8          ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
9          ; WITH THE OPERATOR.
10         ;--
11
12 042324 000000          .WORD L10031-L$SOFT/2
13 042326          L$SOFT::
14
15          .EVEN
16          L10031:
17
18          $PATCH:: .BLKW 50.          ;PATCH AREA (50. WORDS)
19
20 DBUFF:: .BLKB 512.          ;BYTES          ;DATA BUFFER
21 TBUFF:: .BLKB 512.          ;BYTES          ;TEMPORARY BUFFER FOR RCT BLOCK 0
22 IBUFF:: .BLKB 512.          ;BYTES          ;IMAGE BUFFER FOR LBN BEING REPLACED
23 BUFF1:: .BLKB 512.          ;BYTES          ;BUFFER 1
24 BUFF2:: .BLKB 512.          ;BYTES          ;BUFFER 2
25
26          .EVEN
27          .WORD T$FREE
28          .WORD T$SIZE
29          L$LAST::
30
31
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
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1
3 047476 000000
  047500 000005
  047502
4 047502 172150
5 047504 000154
6 047506 000005
7 047510 000077
8 047512 000000
9 047514
11 000001
    
```

```

          .WORD 0
          .WORD L10034-./2-1
L10032:   .WORD 172150
          .WORD 154
          .WORD 5.
          .WORD 63.
          .WORD 0.
L10034:
          .END
    
```

```

; UNIBUS ADDRESS
; VECTOR ADDRESS
; BR LEVEL
; UNIBUS BURST RATE
; DRIVE NUMBER
    
```

Symbol table

ACCESS	002574	G	B. ESTA=	000100	COPY	034276	C#PNTF=	000017	DU. QUE=	010000			
ADR	= 000020	G	B. EVNT=	000001	CR	= 000015	G	C#PNTS=	000016	DU. SPC=	060000		
ALOCM	040054		B. FLAG=	000040	CRLF	014231		C#PNTY=	000015	DU. TER=	040000		
ARG. CT=	000006		B. HADD=	000002	CTLRLS	002230	G	C#PUTB=	000072	D. BB	= 000002		
ASSEMB=	000010		B. MOD =	000004	CTLRTB	002304	G	C#PUTW=	000073	D. BB01=	000004		
AUTORC	002152	G	B. MSCP=	000140	CT. AVL=	100000		C#QIO =	000377	D. BB02=	000010		
BACKMS	022464		B. PHAS=	000004	CT. BRL=	007000		C#RDBU=	000007	D. BB03=	000014		
BACKUP	002652	G	B. RCT =	000003	CT. CMD=	000004		C#REFG=	000047	D. BB04=	000020		
BADRBN=	000024	G	B. SADD=	000001	CT. MSG=	000010		C#REL =	000077	D. BB05=	000024		
BASEMS	002552	G	B. SAEY=	000004	CT. REQ=	000020		C#RESE=	000033	D. BB06=	000030		
BBRCNT	002540	G	B. SAVA=	000002	CT. UNT=	000077		C#REVI=	000003	D. BB07=	000034		
BCNT	002334	G	CACHES=	000620	CT. U50=	000040		C#RFLA=	000021	D. BB08=	000040		
BELL	= 000007	G	CACHID=	000030	CT. VEC=	000777		C#RPT =	000025	D. BB09=	000044		
BITTBL	020612		CACHLS	002254	G	C#AU =	000052	C#SEFG=	000046	D. BB10=	000050		
BIT0	= 000001	G	CACHTB	002310	G	C#AUTO=	000061	C#SPRI=	000041	D. BB11=	000054		
BIT00	= 000001	G	CACHTS	017562		C#BRK =	000022	C#SVEC=	000037	D. BB12=	000060		
BIT01	= 000002	G	CALBLK	031332		C#BSEG=	000004	C#TOME=	000076	D. BB13=	000064		
BIT02	= 000004	G	CALRBN	031256		C#BSUB=	000002	C. BST =	000004	D. BB14=	000070		
BIT03	= 000010	G	CALR1	016042		C#CLCK=	000062	C. DRVN=	000006	D. BB15=	000074		
BIT04	= 000020	G	CALR2	016122		C#CLEA=	000012	C. DRVT=	000020	D. BB16=	000100		
BIT05	= 000040	G	CALR3	016124		C#CLOS=	000035	C. FLG =	000014	D. BCYL=	000146		
BIT06	= 000100	G	CALR4	016126		C#CLP1=	000006	C. HCOM=	000016	D. BE	= 000040		
BIT07	= 000200	G	CALR5	016130		C#CPBF=	000074	C. JAD =	000012	D. BEC	000104		
BIT08	= 000400	G	CALR6	016132		C#CPME=	000075	C. JSR =	000010	D. BGN1=	000106		
BIT09	= 001000	G	CF. ATN=	000200		C#CVEC=	000036	C. REF =	000026	D. BGN2=	000116		
BIT1	= 000002	G	CF. MSC=	000100		C#DCLN=	000044	C. SIZE=	000030	D. BGN3=	000126		
BIT10	= 002000	G	CF. OTH=	000040		C#DODU=	000051	C. TO =	000022	D. BGN4=	000136		
BIT11	= 004000	G	CF. SHD=	000002		C#DRPT=	000024	C. TOH =	000024	D. CYL =	000400		
BIT12	= 010000	G	CF. THS=	000020		C#DU =	000053	C. UADR=	000000	D. DC	= 000002		
BIT13	= 020000	G	CF. 576=	000001		C#EDIT=	000003	C. VEC =	000002	D. DCA =	000001		
BIT14	= 040000	G	CHRMSG=	023110		C#ERDF=	000055	DBM01	035470	D. DCY =	020000		
BIT15	= 100000	G	CHRRPL=	023104		C#ERHR=	000056	DBM02	035556	D. ECC =	010000		
BIT2	= 000004	G	CLKFLG	002206	G	C#ERRO=	000060	DBM03	035601	D. ECTL=	000152		
BIT3	= 000010	G	CLOG	034416		C#ERSF=	000054	DBM04	035667	D. END1=	000112		
BIT4	= 000020	G	CLRBUF	034240		C#ERSO=	000057	DBUFF	042472	G	D. END2=	000122	
BIT5	= 000040	G	CMDIN	002712	G	C#ESCA=	000010	DCODE	002404	G	D. END3=	000132	
BIT6	= 000100	G	CMDMOD	002346	G	C#ESEG=	000005	DEBUG	002650	G	D. END4=	000142	
BIT7	= 000200	G	CMDPCK	002326	G	C#ESUB=	000003	DELTA	002510	G	D. IW =	040000	
BIT8	= 000400	G	CMDREF	002342	G	C#ETST=	000001	DEVNAM	014225		D. PAT =	000000	
BIT9	= 001000	G	CMPUUT	040216		C#EXIT=	000032	DFPTBL	002132	G	D. RET =	001000	
BLDCMD	030632		CNTTBL	020636		C#FREQ=	000101	DIAG	= 177777		D. RO =	004000	
BLD28	034654		CNV28	034756		C#FRME=	000100	DIAGMC=	000000		D. SEQ =	000100	
BLKCNT	002572	G	CON. A	033444		C#GETB=	000026	DIVIDE	034436		D. SERN=	000156	
BLKSTZ=	001000	G	CON. A1	033452		C#GETW=	000027	DIV10	034474		D. SIZE=	000164	
BOE	= 000400	G	CON. A2	033466		C#GMAN=	000043	DRIVTB	002306	G	D. TR =	000020	
BRF	= 020000	G	CON. D	033470		C#GPHR=	000042	DRPUUT	040254		D. WC =	000010	
BUFF1	045472	G	CON. H	033502		C#GPRI=	000040	DSPDSC	032364		D. WCA =	000004	
BUFF2	046472	G	CON. N	033526		C#INIT=	000011	DSPHEL	032150		D. WO =	002000	
BUSFLA	002614	G	CON. N1	033532		C#INLP=	000020	DSPRCT	002156	G	ECCCNT	002536	G
B. BLOC=	000001		CON. O	033514		C#MANI=	000050	DSPRPL	002154	G	ECODE	002352	G
B. CMDR=	000001		CON. QU	033424		C#MAP =	000102	DSTCNT	002534	G	EF. BBR=	000200	
B. CODE=	000002		CON. QX	033442		C#MEM =	000031	DT. AVL=	100000		EF. BBU=	000100	
B. COPY=	000002		CON. R	033550		C#MMU =	000103	DT. UNT=	000077		EF. CON=	000036	G
B. DCD=	000004		CON. R1	033572		C#MSG =	000023	DUP	= 001000		EF. LOG=	000040	
B. DERR=	000010		CON. S	033610		C#OPNR=	000034	DU. DFL=	020000		EF. NEW=	000035	G
B. ECCD=	000020		CON. S1	033614		C#OPNW=	000104	DU. FTL=	050000		EF. PWR=	000034	G
B. EMRE=	000010		COPIES	002423	G	C#PNTB=	000014	DU. INF=	030000		EF. RES=	000037	G

EF.SEX=	000020	ERRLOG	016634	F\$MOD =	000000	HC.MPK=	000020	INF602	010320
EF.STA=	000040 G	ERRLO	016734	F\$MSG =	000011	HC.MSG=	000004	INF603	010405
EMPTY	002512 G	ERRL1	016772	F\$PROT=	000021	HC.PSZ=	000060	INIPAR	023160
EMSTAT	002354 G	ERRL2	017066	F\$PWR =	000017	HC.RSZ=	000004	INISWP	022206
ENBERR	002162 G	ERRL3	017224	F\$RPT =	000012	HC.SIZ=	000164	INITAC	030140
ENBRPL	002160 G	ERRL4	017462	F\$SEG =	000003	HDR001	010472	INITBL	037422
ENBWE	002164 G	ERRME1	014427	F\$SOFT=	000005	HDR101	010667	INITRW	030074
ENDREF	002350 G	ERRMSG	002172 G	F\$SRV =	000010	HDR201	011314	INTCPK	030626
ENTRYS=	000200 G	ERRNBR	002170 G	F\$SUB =	000002	HDR301	012043	INTRCV	002332 G
ERFLAG	002364 G	ERRREF	002360 G	F\$SW =	000014	HDR401	012663	INTSRV	016216 G
ERLAST	016630	ERRSTE	002550 G	F\$TEST=	000001	HDR501	013107	INVCHR	014340
ERLDS	002406 G	ERRTYP	002166 G	F.CRLF	014215	HDR601	013173	IPVAL	002322 G
ERLEXI	017514	ERR.SZ=	000010	F.TEXT	014220	HDR701	013731	ISR	= 000100 G
ERLFMT	002372 G	ERR.TB	016134	GBCNT	002566 G	HELP	= 000000	ISSUE	030204
ERLLBN	002374 G	ERR001	015220 G	GETCDN	033730	HLPFIL	014672	IXE	= 004000 G
ERLMSG	004020 G	ERR101	015266 G	GETCHR	023070	HOE	= 100000 G	I\$AU	= 000041
ERLPSI=	000005	ERR201	015336 G	GETCNT	033664	H.BRL	= 000004	I\$AUTO=	000041
ERLPTB	016616	ERR300	015406 G	GETCNX	033670	H.BST	= 000006	I\$CLN =	000041
ERLVAL	004026 G	ERR301	015504 G	GETCXX	033736	H.DRV =	000010	I\$DU =	000041
ERM001	010527	ERR302	015570 G	GETLBN	023410	H.UBA =	000000	I\$HRD =	000041
ERM101	010727	ERR401	015650 G	GETLOG	023126	H.VEC =	000002	I\$INIT=	000041
ERM102	011061	ERR601	015720 G	GETVSN	032022	IBE	= 010000 G	I\$MOD =	000041
ERM103	011145	ERR701	015770 G	GLBN	002562 G	IBUFF	044472 G	I\$MSG =	000041
ERM104	011240	EVENT	002362 G	GRPSIZ=	000200 G	IDU	= 000040 G	I\$PROT=	000040
ERM201	011346	EVL	= 000004 G	GTDRVT	032136	IER	= 020000 G	I\$PTAB=	000041
ERM202	011453	EXIT10	025600	G\$CNTD=	000200	INCARN=	000040 G	I\$PWR =	000041
ERM203	011711	EXIT11	026122	G\$DELM=	000372	INCRLB	002554 G	I\$RPT =	000041
ERM204	011743	EXIT12	026520	G\$DISP=	000003	INF100	004362	I\$SECT=	000000
ERM300	012076	EXIT13	026612	G\$EXCP=	000400	INF101	004546	I\$SEG =	000041
ERM301	012137	EXIT15	026750	G\$HILI=	000002	INF102	004603	I\$SETU=	000041
ERM302	012202	EXIT16	027122	G\$LOLI=	000001	INF103	004731	I\$SFT =	000041
ERM303	012246	EXIT17	027214	G\$NO	= 000000	INF110	005016	I\$SRV =	000041
ERM304	012313	EXIT3	024770	G\$OFFS=	000400	INF111	005165	I\$SUB =	000041
ERM305	012361	EXIT4	025126	G\$OFFSI=	000376	INF200	005271	I\$TST =	000041
ERM306	012526	EXIT5	025174	G\$PRMA=	000001	INF201	005343	J\$JMP	= 000167
ERM307	012617	EXIT6	025340	G\$PRMD=	000002	INF202	005725	KTMEM	00222E
ERM401	012727	EXIT9	025450	G\$PRML=	000000	INF210	006170	KW.BRL	002212
ERM402	013000	EXTMSK	002370 G	G\$RADA=	000140	INF211	006244	KW.CSR	002210
ERM403	013050	E\$END =	002100	G\$RADB=	000000	INF212	006332	KW.EL	002220
ERM501	013125	E\$LOAD=	000035	G\$RADD=	000040	INF213	006420	KW.HZ	002216
ERM601	013216	FEF	= 000200 G	G\$RADL=	000120	INF220	006452	KW.OUT=	000105 G
ERM602	013242	FERCNT	002530 G	G\$RADO=	000020	INF221	006523	KW.VEC	002214
ERM603	013257	FFREE	002176 G	G\$XFER=	000004	INF301	007002	KW11I	016174 G
ERM604	013334	FLAGS	002356 G	G\$YES =	000010	INF302	007027	LBN	002446 G
ERM605	013402	FMERR	040100	HADDR	002400 G	INF310	007061	LBNDX	002522 G
ERM606	013472	FMT.AS	014223	HCECNT	002532 G	INF311	007135	LBNLO =	000014 G
ERM607	013621	F\$SIZE	002200 G	HCOM	002324 G	INF320	007164	LBNSTA	002516 G
ERM700	013763	F\$AU =	000015	HCOMM	040004	INF321	007242	LF	= 000012 G
ERM701	014024	F\$AUTO=	000020	HC.CCT=	000012	INF322	007630	LINBUF	002716
ERM702	014062	F\$BGN =	000040	HC.CEV=	000100	INF323	010043	LMT28	014277
ERM703	014130	F\$CLEA=	000007	HC.CMD=	000010	INF324	010065	LOADFL	002202 G
ERM704	014157	F\$DU =	000016	HC.CPK=	000104	INF401	010114	LOE	= 040000 G
ERRBLK	002174 G	F\$END =	000041	HC.ESZ=	000004	INF405	010166	LOG	= 000001
ERRC	033632	F\$HARD=	000004	HC.INT=	000000	INF406	010206	LOGMSG=	023146
ERRCNT	002475 G	F\$HW =	000013	HC.ISZ=	000004	INF407	010224	LOGRPL=	023142
ERRD	033644	F\$INIT=	000006	HC.MCT=	000006	INF408	010243	LOT	= 000010 G
ERRFMT	002366 G	F\$JMP =	00000C	HC.MEV=	000014	INF601	010264	LPNT	033302

Symbol table

LPNTB	033254	L\$UNIT	002012	G	MD.SCL=	002000	OP.ESP=	000002	P.CPSP=	000042
LPNTF	033244	L.BADR=	000030	G	MD.SEC=	000100	OP.FLU=	000023	P.CRF =	000000
LPNTS	033274	L.CRF =	000000	G	MD.SEQ=	000020	OP.GCS=	000002	P.CTMO=	000020
LPNTX	033264	L.DS =	000060	G	MD.SER=	000400	OP.GSS=	000001	P.CYLS=	000050
L\$ACP	002110	L.EVNT=	000012	G	MD.SPD=	000001	OP.GUS=	000003	P.DEXT=	000014
L\$APT	002036	L.FLGS=	000011	G	MD.SSH=	000200	OP.MRD=	000030	P.DFLG=	000017
L\$AU	040306	L.FMT =	000010	G	MD.SWP=	000004	OP.MWR=	000031	P.DMDT=	000024
L\$AUT	002070	L.MDCD=	000050	G	MD.VOL=	000002	OP.ONL=	000011	P.DPRG=	000020
L\$AUTO	040122	L.SDI =	000054	G	MD.WBN=	000100	OP.RD =	000041	P.DTMO=	000024
L\$CCP	002106	L10000	002144		MD.WBV=	000400	OP.RLC=	000103	P.ELGF=	000034
L\$CLEA	040124	L10001	002166		MINLBN	002600	OP.RPL=	000024	P.FBBK=	000034
L\$CO	002032	L10002	015264		MODEL	003244	OP.RSD=	000005	P.FLGS=	000011
L\$DEPO	002011	L10003	015334		MSCP =	000000	OP.SCC=	000004	P.GRPS=	000046
L\$DESC	004324	L10004	015404		MSCPMS	003330	OP.SEX=	000007	P.HSTI=	000020
L\$DESP	002076	L10005	015502		MSCPND	004226	OP.SHC=	000102	P.HTMO=	000020
L\$DEVP	002060	L10006	015566		MSCPVB	004230	OP.SSD=	000004	P.LBN =	000034
L\$DISP	002124	L10007	015646		MSGADR=	023456	OP.SUC=	000012	P.MEDI=	000034
L\$DLY	002116	L10010	015716		MSGBRL	042261	OP.WR =	000042	P.MLUN=	000014
L\$DTP	002040	L10011	015766		MSGBST	042272	OSTRE	033422	P.MOD =	000012
L\$DTYP	002034	L10012	016040		MSGGLDR	042314	OSTRNG	033352	P.MODE=	000032
L\$DU	040146	L10013	016162		MSGPCK	002330	O\$APIS=	000000	P.OPCD=	000010
L\$DUT	002072	L10014	016172		MSGUBA	042224	O\$AU =	000000	P.OTRF=	000014
L\$DVTY	004300	L10015	016214		MSGVEC	042252	O\$BGNR=	000001	P.OVRL=	000034
L\$EF	002052	L10016	016222		MULT	034542	O\$BGNS=	000000	P.RBN =	000014
L\$ENVI	002044	L10017	035716		MULTRD	027224	O\$DU =	000001	P.RBNS=	000056
L\$ERRT	002166	L10021	040120		MULTWR	027514	O\$ERRT=	000001	P.RCTC=	000057
L\$ETP	002102	L10022	040122		M.RA60=	000004	O\$GNSW=	000001	P.RCTS=	000054
L\$EXP1	002046	L10023	040144		M.RA80=	000001	O\$POIN=	000001	P.RGID=	000034
L\$EXP4	002064	L10024	040304		M.RA81=	000005	O\$SETU=	000001	P.RGOF=	000040
L\$EXP5	002066	L10025	040312		M.RA82=	000013	PB	033146	P.SHST=	000042
L\$HARD	042146	L10026	040622		M.UNKN=	000000	PF	033122	P.SHUN=	000040
L\$HIME	002120	L10027	042142		NCONF	033406	PHASE	002560	P.STS =	000012
L\$HPCP	002016	L10030	042224		NCONS	033362	PNT =	001000	P.TIME=	000024
L\$HPTP	002022	L10031	042326		NDX.AC=	000002	PNTNUM	033742	P.TRKS=	000044
L\$HW	002132	L10032	047502		NDX.RD=	000000	PNTNUS	033750	P.UADR=	000020
L\$ICP	002104	L10034	047514		NDX.RP=	000003	PRI =	002000	P.UNFL=	000016
L\$INIT	035726	MANDIS	014473		NDX.WR=	000001	PRIMARY	002434	P.UNIT=	000004
L\$LADP	002026	MANRPL	002150	G	NXMAD	002224	PRINTC	033066	P.UNSZ=	000044
L\$LAST	047476	MATBLK	002500	G	NXMI	016154	PRI00 =	000000	P.UNTI=	000024
L\$LOAD	002100	MATFLG	002514	G	NXTSTE	002546	PRI01 =	000040	P.USEF=	000022
L\$LUN	002074	MATOFF	002504	G	OLIMIT	014367	PRI02 =	000100	P.VRSN=	000014
L\$MREV	002050	MATRBN	002462	G	ONEFIL=	000001	PRI03 =	000140	P.VSER=	000050
L\$NAME	002000	MAXLBN	002604	G	ONLINE	002612	PRI04 =	000200	P1 =	100000
L\$PRIO	002042	MDLNDX	003246		OPCODE	002344	PRI05 =	000240	P2 =	040000
L\$PROT	035720	MDLSIZ=	000012		OPHELP	002146	PRI06 =	000300	RA60	003311
L\$PRT	002112	MDLTBL	003250		OPTIN	002714	PRI07 =	000340	RA80	003304
L\$REPP	002062	MDLTEX	003262		OP.ABO=	000001	PS	033216	RA81	003316
L\$REV	002010	MD.CMP=	040000		OP.ACC=	000020	PTABLE	002302	RA82	003323
L\$RPT	035132	MD.CWB=	000010		OP.AVA=	000100	PTYPE	033350	RBN	002452
L\$SOFT	042326	MD.ERR=	010000		OP.AVL=	000010	PX	033172	RBNLO =	000020
L\$SPC	002056	MD.EXP=	100000		OP.CCD=	000021	P.BCNT=	000014	RBNS	002422
L\$SPCP	002020	MD.FEU=	000001		OP.CMP=	000040	P.BUFF=	000020	RCTBLK	002476
L\$SPTP	002024	MD.IMF=	000002		OP.DUP=	000101	P.CLAS=	000033	RCTCOP	002424
L\$STA	002030	MD.NXU=	000001		OP.ELP=	000003	P.CMST=	000020	RCTERR	016370
L\$SW	002146	MD.PRI=	000001		OP.END=	000200	P.CNCL=	000022	RCTLBN	002456
L\$TEST	002114	MD.RIP=	000001		OP.ERS=	000022	P.CNTF=	000016	RCTMAS	002426
L\$TIML	002014	MD.SCH=	004000				P.CNTI=	000024	RCTMSG	003660

Symbol table

RCTOFF	002502	G	SA.STP=	100000	SVCGBL=	000000	T\$FLAG=	000040	UNKNOW	003274	
RCTSIZ	002416	G	SA.S1 =	004000	SVCINS=	000000	T\$FREE=	047514	UNSIZ	002430	G
RCTTBL	002312	G	SA.S2 =	010000	SVCSUB=	000000	T\$GMAN=	000000	UNUSE	002440	G
RCTVAL	003670	G	SA.S3 =	020000	SVCTAG=	000000	T\$HILI=	000377	VBUFF1	002473	G
RDCNT	002556	G	SA.S4 =	040000	SVCTST=	000000	T\$LAST=	000001	VBUFF2	002474	G
RDSTAT	002524	G	SA.TST=	100000	SWMSG0	022525	T\$LOLI=	000000	VIBUFF	002472	G
RDDVALD	002470	G	SA.VCE=	000177	SWMSG1	022553	T\$LSYM=	010000	VOLSN =	000000	G
READLB	023604		SA.VEC=	000177	SWMSG2	022613	T\$LTNO=	000002	VSNBUF	002620	G
RECOVR	002466	G	SA.WRP=	040000	SWMSG3	022644	T\$NEST=	177777	VSNPTR	002616	G
REPEAT	002610	G	SB.DST=	000150	SWMSG4	022707	T\$NSO =	000000	VWP =	000002	G
KEPLAC	024156		SB.ECC=	000350	SWMSG5	022753	T\$NS1 =	000005	W =	000000	
REPLY	002654	G	SB.HCE=	000110	SWMSG6	022777	T\$PCNT=	000000	WAIT	031072	
RESCAN	002513	G	SCNDRY	002436	SWMSG7	023024	T\$PTAB=	010033	WBC =	000001	G
RESET	016224		SEARCH	031400	S\$LSYM=	010000	T\$PTHV=	000001	WFECNT	002542	G
RFLAGS=	000010	G	SETTO	034334	TBLNDX	002300	T\$PTNU=	000001	WRSTAT	002526	G
RG.FLG=	040000		SFLAG	002515	TBUFF	043472	T\$SAVL=	177777	XACCES	004257	
RG.OWN=	100000		SFPTBL	002146	TF.BAD=	100000	T\$SEGL=	177777	XBLOCK	003700	
RNTIM	014233		SNDCMD	030744	TF.BBR=	000200	T\$SIZE=	000007	XCMDF	003370	
RNTIME	020650		SND.S1	037424	TF.BBU=	000100	T\$SUBN=	000000	XCODE	003421	
RNTIMX	021036		SND.S2	037430	TF.DST=	000004	T\$TAGL=	177777	XCOPY	003732	
RNTIM1	014266		SND.S3	037434	TF.ECC=	000010	T\$TAGN=	010035	XDAINT	036670	
RNTIM2	014274		STATBL	020624	TF.FER=	000001	T\$TEMP=	000000	XDAIST	037200	
RPLSTA	002520	G	STATUS	002442	TF.HCE=	000002	T\$TEST=	000002	XDARSD	040002	
RSP.CK	037542		STEP	002544	TF.LOG=	000040	T\$TSTM=	177777	XDARSP	037606	
RSP.S1	037440		STEPER	024226	TF.SEX=	000020	T\$TSTS=	000001	XDASRV	016164	G
RSP.S2	037452		STEPFT	024250	TF.WFE=	000400	T\$AU =	010025	XDCODE	004112	
RSP.S3	037472		STEPSI=	000044	TIMDAT=	000044	T\$AUT=	010022	XDHEX	004217	
RSP.S4	037510		STEPTB	024112	TINDEX=	000012	T\$CLE=	010023	XDINFO	004152	
RTFLAG	002204	G	STEP10	025460	TMPSTA	002444	T\$DAT=	010034	XECODE	003537	
SADDR	002314	G	STEP11	025610	TRKSIZ	002420	T\$DU =	010024	XENDRF	003506	
SAEXP	002320	G	STEP12	026132	TTYIN	002666	T\$HAR=	010030	XESTAT	003625	
SARMSG	003116	G	STEP13	026530	TTYOUT	033242	T\$HW =	010000	XEVENT	004034	
SARVAL	003126	G	STEP15	026640	TXT000	014705	T\$INI=	010021	XEVNT	004035	
SAVAL	002316	G	STEP16	026760	TXT001	014762	T\$MSG=	010012	XFLAGS	003572	
SA.BST=	000374		STEP17	027132	TXT002	014767	T\$PC =	000001	XHADDR	004063	
SA.CMD=	034000		STEP3	024260	TXT005	014774	T\$PRO=	010020	XLBN	014575	
SA.CME=	000070		STEP4	024772	TXT006	015031	T\$PTA=	010033	XMOD	003454	
SA.CM1=	004000		STEP5	025136	TX.BBR	015150	T\$RPT=	010017	XOFF =	000023	G
SA.CNT=	000360		STEP6	025204	TX.BBU	015155	T\$SOF=	010031	XON =	000021	G
SA.CTP=	003400		STEP9	025350	TX.BLK	015174	T\$SRV=	010016	XPHASE	003764	
SA.DI =	000400		STROFF	002506	TX.DST	015213	T\$SW =	010001	XRBN	014615	
SA.ERC=	003777		ST.ABO=	000002	TX.ECH	015143	T\$TES=	010027	XRCT	014537	
SA.ERR=	100000		ST.AOL=	000400	TX.FER	015136	T1	040314	XREAD	004242	G
SA.GO =	000001		ST.AVL=	000004	TX.HCE	015206	T2	040624	XREPLA	004267	G
SA.INE=	000200		ST.CMD=	000001	TX.LB1	015060	UADR	002340	XSADDR	003165	G
SA.INT=	000200		ST.CMP=	000007	TX.LB2	015100	UAM =	000200	XSAEXP	003214	G
SA.LFC=	000002		ST.CNT=	000012	TX.LOG	015167	UF.CMR=	000001	XSAVAL	003136	
SA.MCV=	000017		ST.DAT=	000010	TX.PRI	015117	UF.CMW=	000002	XSTAT	003626	
SA.MP =	000100		ST.DIA=	000037	TX.SEC	015124	UF.INA=	040000	XSTEP	014635	
SA.MSE=	000007		ST.DRV=	000013	TX.SEX	015162	UF.RPL=	100000	XWRITE	004250	
SA.MSG=	003400		ST.HST=	000011	TX.UNS	015131	UF.SCH=	004000	X\$ALWA=	000000	
SA.MS1=	000400		ST.MFE=	000005	TX.WFE	015201	UF.SCL=	002000	X\$FALS=	000040	
SA.NV =	002000		ST.MSK=	000037	TYPSTA	020002	UF.WBN=	000100	X\$OFFS=	000400	
SA.NVE=	000400		ST.OFL=	000003	T\$ARGC=	000002	UF.WPH=	020000	X\$TRUE=	000020	
SA.PRG=	000001		ST.SUB=	000040	T\$CODE=	004052	UF.WPS=	001000	\$MUL =	000037	
SA.QB =	001000		ST.SUC=	000000	T\$ERRN=	000457	JF.576=	000004	\$PATCH	042326	G
SA.STE=	000200		ST.WPR=	000006	T\$EXCP=	000000	UNITIN	021050			

CZUDLA0 BBR Replacement Utility MACRO Y05.03c Friday 19 Apr 85 10:42 Page 146-5
Symbol table

. ABS. 047514 000 (RW,I,GBL,ABS,OVR)
Errors detected: 0

*** Assembler statistics

Work file reads: 986
Work file writes: 860
Size of work file: 38688 Words (152 Pages)
Size of core pool: 14080 Words (55 Pages)
Operating system: RT-11 (Under RTEM-11)

Eapsed time: 00:05:30.00
ZUDLA0.BIN/DS:GBL/EN:ABS:AMA,ZUDLA0/CR=SVC35R/ML,ZUDLA0.MAC

ERR401	39-7*	88-88	88-96	88-111														
ERR601	38-28*	52-42	70-32	76-38	84-68	85-81												
ERR701	39-7*	47-46	48-7	48-19	50-19	51-7												
ERRBLK	19-5*																	
ERRC	101-39	101-46	105-35*															
ERRCNT	21-28*	84-26*	84-46*	85-25*	85-45*													
ERRD	101 48	105-51*																
ERRFMT	20-30*																	
ERRLO	47-21	48-3*																
ERRL1	47-22	48-11*																
ERRL2	47-23	49-3*																
ERRL3	47-24	50-3*																
ERRL4	47-25	51-3*																
ERRLOG	47-31*	88-49																
ERRME1	34-18*	101-44	103-26															
ERRMSG	19-5*																	
ERRNBR	19-5*																	
ERRREF	20-27*																	
ERRSTE	21-80*																	
ERRTYP	19-5*																	
EVENT	20-28*	27-20	39-12	47-36*	49-18													
EVL	5-9*																	
EXIT10	77-49*																	
EXIT11	78-63	78-93*																
EXIT12	79-64	79-82	79-94*															
EXIT13	80-39*																	
EXIT15	81-35	81-41*																
EXIT16	82-38	82-43	82-50*															
EXIT17	83-36*																	
EXIT3	70-16	72-16	72-19*															
EXIT4	73-30	73-36	73-42*															
EXIT5	74-28*																	
EXIT6	75-49*																	
EXIT9	76-34	76-41*																
EXTMSK	20-31*	40-14	40-19*	45-33*	47-44*	48-5*	48-17*	50-17*	51-5*	52-41*	58-12*	58-22*	58-41*	59-12*				
	70-30*	73-39*	76-37*	79-41*	79-75*	79-85*	82-46*	84-40*	84-67*	85-39*	85-58*	85 80*	88-79*	88-87*				
	88-95*	88-103*	88-110*	128-69*	129-35*	129-118*	130-43*	130-53*	139-39*									
F\$AU	1-I02*	136-9	136-15															
F\$AUTO	1-I02*	133-10	133-12															
F\$BGN	1-I02*	1-I17	4-22	5-3	36-7	36-23	37-7	37-28	37-53	37-76	38-7	38-28	39-7	42-11				
	43-20	44-6	44-11	67-102	117-56	117-62	119-9	120-26	127-34	133-10	134-8	134-13	135-8	136-9				
	136-16	136-17	137-3	137-7	137-11	137-72	137-78	137-82	138-3	138-7	143-28	143-32	143-34	144-2				
	144-12	145-12	145-26	146-2	146-3	146-3	146-9	146-10										
F\$CLEA	1-I02*	134-8	134-17															
F\$DU	1-I02*	135-8	135-26															
F\$END	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1-I02	1 I02	1-I02				
	1-I02	1-I02	1-I02*	1-I17	4-22	5-3	36-16	36-35	36-37	37-19	37-21	37-44	37-46	37-67				
	37-69	37-88	37-90	38-19	38-21	38-40	38-42	39-16	39-18	42-13	43-23	44-10	44-13	67-102				
	117-56	117-79	118-40	118-52	127-34	132-39	133-12	134-13	134-17	135-18	135-26	136-11	136-15	136-16				
	136-17	137-3	137-3	137-3	137-7	137-11	137-72	137-78	137-82	137-82	138-3	138-3	138-3	138-7				
	143 28	143-32	143-32	143-34	144-2	144-30	145-14	145-26	146-2	146-3	146-9	146-10						
F\$HARD	1-I02*	144-12	144-30															
F\$HM	1-I02*	3-10	3-17															
F\$INIT	1-I02*	120-26	132-39															
F\$JMP	1-I02*	36-35	36-35	37-19	37-19	37-44	37-44	37-67	37-67	37-88	37-88	38-19	38-19	38-40				
	38-40	39-16	39-16	117-79	117-79	118-40	118-40	127-34	134-13	135-18	135-18	136-11	136-11	137-7				

CZUOLAO BBR Replacement Utility MACRO Y05.03c Friday 19-Apr-85 10:42 Page S-24
 Cross reference table (CREF V05.03)

	144-17	144-17	144-17*	144-18	144-18	144-18*								
TMPSTA	21-17*	84-27*	84-45*	84-60	85-26*	85-44*	85-73							
TRKSIZ	21-5*	58-17*	71-15	94-34										
TTYIN	22-8*	66-25*	66-33	66-34	66-52*	66-56	116-23	117-20						
TTYOUT	99-8*	99-18	99-20	99-22	99-24	99-27*								
TX.BBR	34-44*	54-18												
TX.BBU	34-45*													
TX.BLK	34-48*	53-33	53-46											
TX.DST	34-51*	55-40												
TX.ECH	34-43*	55-41												
TX.FER	34-42*	55-38												
TX.HCE	34-50*	55-39												
TX.LB1	34-36*	138-44												
TX.LB2	34-37*	138-50												
TX.LOG	34-47*													
TX.PRI	34-39*	55-13												
TX.SEC	34-40*	55-23												
TX.SEX	34-46*													
TX.UNS	34-41*	53-35												
TX.WFE	34-49*	54-8												
TXT000	34-30*	137-34												
TXT001	34-31*													
TXT002	34-32*													
TXT005	34-33*													
TXT006	34-34*	61-20												
TYPSTA	53-22*	80-43	137-48	140-18	141-31									
UADR	20-10*	59-49*	60-6*	69-28*	69-36	72-3*	73-25*	75-25*	75-34	76-27*	77-35	78-37	78-73	79-45*
	80-26*	81-28*	81-36*	82-26*	83-24*	86-26	94-53	95-23	98-27*	98-44	139-11*			
UAM	5-9*													
UF.576	10-56*													
UF.CMR	10-47*													
UF.CMW	10-48*													
UF.INA	10-50*													
UF.RPL	10-49*													
UF.SCH	10-51*													
UF.SCL	10-52*													
UF.WBN	10-53*													
UF.WPH	10-54*													
UF.WPS	10-55*													
UNITIN	57-31*	137-18	138-26											
UNKNOW	24-22	24-29*												
UNSIZ	21-10*	59-17*	59-18*	59-62	59-62	65-22	65-23	66-38	66-39	86-29	86-30	138-48	138-49	
UNUSE	21-14*	53-31*	65-33*	98-26*	98-55	98-57	98-80*	117-73						
VBUFF1	21-26*	78-27*	78-47*	81-26										
VBUFF2	21-27*	78-28*	78-83*	81-34										
VIBUFF	21-25*	70-19*	70-24*	72-10*	73-24*	73-32*								
VOLSN	18-3*													
VSNBUF	21-97*	95-28												
VSNPTR	21-96*	59-62	95-46*											
VWP	18-16*													
W	97-28*													
WAIT	88-31	91-19*												
WBC	18-15*													
WFECNT	21-75*	54-9*	65-40*	117-68										
WRSTAT	21-68*													
X\$ALWA	1 I02*													

X\$FALS	1-I02#						
X\$OFFS	1-I02#						
X\$TRUE	1-I02#						
XACCES	28-13	28-20#					
XBLOCK	26-14	26-28#					
XCMDRF	25-18	25-42#					
XCODE	25-19	25-43#					
XCOPY	26-15	26-29#					
XDAINT	57-47	128-27#					
XDAIST	128-58	129-14#					
XDARSD	129-42*	129-52*	129-53	130-20	130-21*	130-30	130-66#
XDARSP	128-77	129-48	130-19#				
XDASRV	43-20#	125-28					
XDCODE	27-30#	50-32					
XDHEX	27-32#	50-39					
XDINFO	27-31#	50-34					
XECODE	25-22	25-46#					
XENDRF	25-21	25-45#					
XESTAT	25-24	25-48#	36-13				
XEVENT	27-13	27-27#	39-12				
XEVNT	27-28#	46-45	49-18				
XFLAGS	25-23	25-47#					
XHADDR	27-14	27-29#	48-20				
XLBN	34-21#	50-21					
XMOD	25-20	25-44#					
XOFF	29-14						
XON	29-13						
XPHASE	26-16	26-30#					
XRBN	34-22#						
XRCT	34-20#						
XREAD	28-11	28-18#					
XREPLA	28-14	28-21#					
XSADDR	23-12	23-29#					
XSAEXP	23-14	23-30#					
XSAVAL	23-13	23-28#					
XSTAT	25-49#						
XSTEP	34-23#						
XWRITE	28-12	28-19#					

