

[illegible][illegible]

DI
V

DI
V

(2)	81	DECLARATIONS
(3)	130	MACRO DEFINITIONS
(4)	199	IRP - CDRP Consistency Check
(5)	241	Static Storage
(5)	242	- HIRT - Host Initiated Replacement Table
(5)	332	- HIR Error Processing Information
(6)	418	DUSINIT HIRT - Initialize Host Initiated Replacement Table
(7)	535	ALLOC POOL
(8)	577	DUSLOCK HIRT - Gain exclusive access to HIRT
(9)	654	GRANT HIRT - Complete granting access to the HIRT
(10)	718	DUSUNLOCK HIRT - Release HIRT access
(11)	814	DUSTEST HIRT RWAITCNT - Accumulate RWAITCNT for HIRT
(12)	856	DUSCANCELL FROM HIRT - Cancel requests from the HIRT
(13)	958	DUSDISCONNECT HIRT - Do HIRT cleanup for a disconnect
(14)	1013	DUSRSTRTO HIRT_CDRP - Do connection failed cleanup of HIRT CDRP
(15)	1076	DUSREPLACE LBN - Replace a failing block
(16)	1834	DUSONLINE_COMPLETE - Perform HIRT operations after ONLINE
(17)	2001	WRITE RCT_BLOCK - Write an RCT sector
(18)	2116	READ RCT_BLOCK - Read an RCT sector
(19)	2208	BUILD RCT_PACKET - Recycle an MSCP end message
(19)	2209	FILL RCT_PACKET - Prepare an MSCP packet for an RCT transfer
(20)	2288	MAP PAGE - Map a page for a transfer
(21)	2318	SEARCH RCT - Locate an available RBN
(22)	2436	TEST_RCT_ENTRY - Test for allocated RBN
(23)	2491	HASH_LBN - Hash an LBN into a RCT block and an offset
(23)	2529	DUSHIR_ERROR - Process error encountered during HIRT processing

```

0000 1      .TITLE  DUHIRT HOST INITIATED REPLACEMENT FOR THE DISK CLASS DRIVER
0000 2      .IDENT  'V04-000'
0000 3
0000 4
0000 5      *****
0000 6      *
0000 7      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      *  ALL RIGHTS RESERVED.
0000 10     *
0000 11     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     *  TRANSFERRED.
0000 17     *
0000 18     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     *  CORPORATION.
0000 21     *
0000 22     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     *
0000 25     *
0000 26     *****
0000 27
0000 28     **
0000 29
0000 30     FACILITY:
0000 31
0000 32         MSCP Disk Class Driver
0000 33
0000 34     ABSTRACT:
0000 35
0000 36         Buddy!  You're in a world of HIRT (Host Initiated Replacement Table).
0000 37
0000 38         This module contains all the routines and data structure definitions
0000 39         needed by the disk class driver to perform host initiated replacement
0000 40         of questionable blocks on disks conforming to the DSA specification.
0000 41
0000 42     ENVIRONMENT:
0000 43
0000 44         This module is linked into DUDRIVER, the VMS disk class driver.
0000 45
0000 46     --
0000 47
0000 48     AUTHOR:  Ralph O. Weber (ghost writer for Robert L. Rappaport)
0000 49
0000 50     CREATION DATE: 21-JAN-1984
0000 51
0000 52     MODIFIED BY:
0000 53
0000 54         V03-004 ROW0398      Ralph O. Weber      21-JUL-1984
0000 55         Setup use of class driver write-lock bit in UCBSW_DEVSTS.
0000 56         Also eliminate alteration and use of DEV$V_SWL bit in
0000 57         UCBSL_DEVCHAR.  That bit is controlled by the file system.

```

```

0000 58 :
0000 59 :
0000 60 : V03-003 ROW0346 Ralph O. Weber 11-APR-1984
0000 61 : > Add worst failure status reporting to insure that I/O
0000 62 : requests producing failed replacement requests get failure
0000 63 : status codes.
0000 64 : > Add several more error logging points.
0000 65 : > Supress error recovery and error correction when testing
0000 66 : the possibly bad block in step 7.
0000 67 : > Use cheap class driver macros for common functions like
0000 68 : testing MSCP success/failure and initializing a MSCP command
0000 69 : packet.
0000 70 :
0000 71 : V03-002 ROW0332 Ralph O. Weber 2-APR-1984
0000 72 : Cause message to appear on the system console whenever an
0000 73 : error occurs during RDT processing.
0000 74 :
0000 75 : V03-001 ROW0331 Ralph O. Weber 29-MAR-1984
0000 76 : Change DUSCANCELFROM_HIRT to count wait count adjustment in
0000 77 : CDRPSW_DUTUCNTR. Also add comments from old DUDRIVER to its
0000 78 : module header.
0000 79 :--

```

```

0000 81          .SBTTL  DECLARATIONS
0000 82  :
0000 83  : INCLUDE FILES:
0000 84  :
0000 85          $CDDDBDEF          ;Define CDDB offsets
0000 86          $CDRPDEF          ;Define CDRP offsets
0000 87          $CDTDEF           ;Define CDT offsets
0000 88          $CRBDEF           ;Define CRB offsets
0000 89          $DDBDEF           ;Define DDB offsets
0000 90          $DEVDEF           ;Define DEVICE CHARACTERISTICS bits
0000 91          $DYNDEF           ;Define DYN symbols
0000 92          $EMBLTDEF         ;Define EMB Log Message Types
0000 93          $FKBDEF           ;Define FKB offsets
0000 94          $IODEF            ;Define I/O FUNCTION codes
0000 95          $IPLDEF           ;Define IPL levels
0000 96          $IRPDEF           ;Define IRP offsets
0000 97          $MSCPDEF          ;Define MSCP packet offsets
0000 98          $MSLGDEF          ;Define MSCP Error Log offsets
0000 99          $PBDEF            ;Define Path Block offsets
0000 100         $PCBDEF           ;Define PCB offsets
0000 101         $PDTDEF           ;Define PDT offsets
0000 102         $PRDEF            ;Define Processor Registers
0000 103         $RCTDEF           ;Define RCT offsets
0000 104         $RDDEF            ;Define RDTE offsets
0000 105         $SBDEF            ;Define System Block Offsets
0000 106         $SCSCMGDEF        ;Define SCS Connect Message offsets
0000 107         $SSDEF            ;Define System Status values
0000 108         $UCBDEF           ;Define UCB offsets
0000 109         $VADEF            ;Define Virtual Address offsets
0000 110         $VECDEF           ;Define INTERRUPT DISPATCH VECTOR offsets
0000 111
0000 112
0000 113         $DUTUDEF          ;Define common class driver CDDB
0000 114                                     ; extensions and other common symbols
0000 115
0000 116
0000 117  : CONSTANTS
0000 118
B6DBC6D 0000 119 TEST_PATTERN=^xB6DBC6D          ; Pattern to write on bad blocks.
0000 120
0000 121  :
0000 122  : MODULE PSECT
0000 123  :
00000000 0000 124          .PSECT $$$115_DRIVER LONG
0000 125  :
0000 126  : SET DEFAULT DISPLACEMENT
0000 127  :
0000 128          .DEFAULT DISPLACEMENT WORD

```



```

0000 130      .SBTTL  MACRO DEFINITIONS
0000 131
0000 132      .MACRO  HIRT SUBSAVE          ; Save return on HIRT substack.
0000 133      POPL   @HIRT$L STKPTR        ; Pop return from stack onto substack.
0000 134      ADDL   #4,HIRT$L STKPTR      ; Bump substack pointer.
0000 135      .ENDM   HIRT_SUBSAVE
0000 136
0000 137      .MACRO  HIRT SUBUNSAVE        ; Pop top of SUBSTACK and push onto stack.
0000 138      SUBL   #4,HIRT$L STKPTR      ; Pop substack.
0000 139      PUSHL  @HIRT$L STKPTR        ; Put top of substack onto top of stack.
0000 140      .ENDM   HIRT_SUBUNSAVE
0000 141
0000 142      .MACRO  HIRT SUBRETURN        ; HIRT_SUBUNSAVE and return to caller.
0000 143      HIRT_SUBUNSAVE
0000 144      RSB
0000 145      .ENDM   HIRT_SUBRETURN
0000 146
0000 147      ;
0000 148      ; Expanded opcode macros - Branch word conditional psuedo opcodes.
0000 149      ;
0000 150
0000 151      ;
0000 152      ; BWNEQ - Branch (word offset) not equal
0000 153      ;
0000 154
0000 155      .MACRO  BWNEQ  DEST,?L1
0000 156      .SHOW
0000 157      BEQL   L1                    ; Branch around if NOT NEQ.
0000 158      BRW    DEST                  ; Branch to destination if NEQ.
0000 159      L1:                          ; Around.
0000 160      .NOSHOW
0000 161      .ENDM   BWNEQ
0000 162
0000 163      ;
0000 164      ; BWEQL - Branch (word offset) equal
0000 165      ;
0000 166
0000 167      .MACRO  BWEQL  DEST,?L1
0000 168      .SHOW
0000 169      BNEQ   L1                    ; Branch around if NOT EQL.
0000 170      BRW    DEST                  ; Branch to destination if EQL.
0000 171      L1:                          ; Around.
0000 172      .NOSHOW
0000 173      .ENDM   BWEQL
0000 174
0000 175      ;
0000 176      ; BWBS - Branch (word offset) bit set.
0000 177      ;
0000 178
0000 179      .MACRO  BWBS    BIT,FIELD,DEST,?L1
0000 180      .SHOW
0000 181      BBC     BIT,FIELD,L1          ; Branch around if bit NOT set.
0000 182      BRW    DEST                  ; Branch to destination if bit set.
0000 183      L1:                          ; Around.
0000 184      .NOSHOW
0000 185      .ENDM   BWBS
0000 186

```

```

0000 187 ;
0000 188 ; BWBC - Branch (word offset) bit clear.
0000 189 ;
0000 190
0000 191 .MACRO BWBC BIT,FIELD,DEST,?L1
0000 192 .SHOW
0000 193 BBS BIT,FIELD,L1 ; Branch around if bit NOT clear.
0000 194 BRW DEST ; Branch to destination if bit clear.
0000 195 L1: ; Around.
0000 196 .NOSHOW
0000 197 .ENDM BWBC

```



```
0000 199 .SBTTL IRP - CDRP Consistency Check
0000 200
0000 201 ; The following set of ASSUME statements will all be true as long as
0000 202 ; the IRP and CDRP definitions remain consistent.
0000 203
0000 204 ASSUME CDRPSL_IOQFL-CDRPSL_IOQFL EQ IRPSL_IOQFL
0000 205 ASSUME CDRPSL_IOQBL-CDRPSL_IOQFL EQ IRPSL_IOQBL
0000 206 ASSUME CDRPSW_IRP_SIZE-CDRPSL_IOQFL EQ IRPSW_SIZE
0000 207 ASSUME CDRPSB_IRP_TYPE-CDRPSL_IOQFL EQ IRPSB_TYPE
0000 208 ASSUME CDRPSB_RMOD-CDRPSL_IOQFL EQ IRPSB_RMOD
0000 209 ASSUME CDRPSL_PID-CDRPSL_IOQFL EQ IRPSL_PID
0000 210 ASSUME CDRPSL_AST-CDRPSL_IOQFL EQ IRPSL_AST
0000 211 ASSUME CDRPSL_ASTPRM-CDRPSL_IOQFL EQ IRPSL_ASTPRM
0000 212 ASSUME CDRPSL_WIND-CDRPSL_IOQFL EQ IRPSL_WIND
0000 213 ASSUME CDRPSL_UCB-CDRPSL_IOQFL EQ IRPSL_UCB
0000 214 ASSUME CDRPSW_FUNC-CDRPSL_IOQFL EQ IRPSW_FUNC
0000 215 ASSUME CDRPSB_EFN-CDRPSL_IOQFL EQ IRPSB_EFN
0000 216 ASSUME CDRPSB_PRI-CDRPSL_IOQFL EQ IRPSB_PRI
0000 217 ASSUME CDRPSL_IOSB-CDRPSL_IOQFL EQ IRPSL_IOSB
0000 218 ASSUME CDRPSW_CHAN-CDRPSL_IOQFL EQ IRPSW_CHAN
0000 219 ASSUME CDRPSW_STS-CDRPSL_IOQFL EQ IRPSW_STS
0000 220 ASSUME CDRPSL_SVAPTE-CDRPSL_IOQFL EQ IRPSL_SVAPTE
0000 221 ASSUME CDRPSW_BOFF-CDRPSL_IOQFL EQ IRPSW_BOFF
0000 222 ASSUME CDRPSL_BCNT-CDRPSL_IOQFL EQ IRPSL_BCNT
0000 223 ASSUME CDRPSW_BCNT-CDRPSL_IOQFL EQ IRPSW_BCNT
0000 224 ASSUME CDRPSL_IOST1-CDRPSL_IOQFL EQ IRPSL_IOST1
0000 225 ASSUME CDRPSL_MEDIA-CDRPSL_IOQFL EQ IRPSL_MEDIA
0000 226 ASSUME CDRPSL_IOST2-CDRPSL_IOQFL EQ IRPSL_IOST2
0000 227 ASSUME CDRPSL_TT_TERM-CDRPSL_IOQFL EQ IRPSL_TT_TERM
0000 228 ASSUME CDRPSB_CARCON-CDRPSL_IOQFL EQ IRPSB_CARCON
0000 229 ASSUME CDRPSQ_NT_PRVMSK-CDRPSL_IOQFL EQ IRPSQ_NT_PRVMSK
0000 230 ASSUME CDRPSL_ABCNT-CDRPSL_IOQFL EQ IRPSL_ABCNT
0000 231 ASSUME CDRPSW_ABCNT-CDRPSL_IOQFL EQ IRPSW_ABCNT
0000 232 ASSUME CDRPSL_OBCNT-CDRPSL_IOQFL EQ IRPSL_OBCNT
0000 233 ASSUME CDRPSW_OBCNT-CDRPSL_IOQFL EQ IRPSW_OBCNT
0000 234 ASSUME CDRPSL_SEGVBN-CDRPSL_IOQFL EQ IRPSL_SEGVBN
0000 235 ASSUME CDRPSL_JNL_SEQNO-CDRPSL_IOQFL EQ IRPSL_JNL_SEQNO
0000 236 ASSUME CDRPSL_DIAGBUF-CDRPSL_IOQFL EQ IRPSL_DIAGBUF
0000 237 ASSUME CDRPSL_SEQNUM-CDRPSL_IOQFL EQ IRPSL_SEQNUM
0000 238 ASSUME CDRPSL_EXTEND-CDRPSL_IOQFL EQ IRPSL_EXTEND
0000 239 ASSUME CDRPSL_ARB-CDRPSL_IOQFL EQ IRPSL_ARB
```

```
0000 241 .SBTTL Static Storage
0000 242 .SBTTL - HIRT - Host Initiated Replacement Table
0000 243
0000 244 :+
0000 245 : The following table is allocated within the Disk Class Driver. There is
0000 246 : only one such table per system. The HIRT is used to control resources
0000 247 : needed by the Host Initiated Replacement of disk blocks algorithms. In
0000 248 : order to limit the resources dedicated to this activity, only one such
0000 249 : replacement is allowed to proceed at any given instant of time. Replacement
0000 250 : requests which cannot be immediately satisfied are queued.
0000 251 :-
0000 252
0000 253 .SAVE
0000 254 .PSECT $$$300_HIRT LONG
0000 255
00000000 0000 256 HIRTSL_RPLQFL: .LONG 0 : Request Queue FLINK.
00000000 0004 257 HIRTSL_RPLQTP: .LONG 0 : Request Queue Tail Pointer.
0000 0008 258 HIRT$W_IOST: .WORD 0 : Static storage for routines.
0000 000A 259 HIRT$W_IOWORST: .WORD 0 : Worst I/O status encountered.
0000 000C 260 .WORD 0
0000 000E 261 HIRT$W_STS: .WORD 0 : HIRT status word.
0010 262
0010 263 $VIELD HIRT,0,<-
0010 264 <ACTIVE,,M>,- : Set means HIRT has been initialized.
0010 265 <BUSY,,M>,- : Set means HIRT being used currently.
0010 266 <FE,,M>,- : Set means force error on original data
0010 267 <MATCH,,M>,- : SEARCH RCT bit - set => LBN matched
0010 268 <EMPTYPE,,M>,- : SEARCH RCT BIT - set => not primary
0010 269 <RESCAN,,M>,- : SEARCH RCT BIT - set => reached NULLS
0010 270 <RCTFULL,,M>,- : SEARCH RCT BIT - set => no more RBNs avail
0010 271 <ERLOGIP,,M>,- : Error Log message has been generated
0010 272 <RCTFE,,M>,- : Write RCT block with Forced Error
0010 273 >
0010 274
00000000 0010 275 HIRTSL_LOOPCNT: .LONG 0 : Loop count used in READ_RCT_BLOCK and
0010 276 : WRITE_RCT_BLOCK.
00000000 0014 277 HIRTSL_OWNUCB: .LONG 0 : If HIRT busy, owner UCB address.
00000000 0018 278 HIRTSL_LBN: .LONG 0 : LBN being replaced for UCB.
00000000 001C 279 HIRTSL_SAVDCDRP: .LONG 0 : CDRP address of I/O request of owner.
00000000 0020 280 HIRTSL_CDRP: .LONG 0 : Address of permanent CDRP for replacement.
0024 281
00000000 0024 282 HIRTSL_PAGE0PTR: .LONG 0 : System Virtual Address of scratch page
0028 283 : needed by Replacement algorithm.
00000000 0028 284 HIRTSL_PAGE1PTR: .LONG 0 : System Virtual Address of scratch page
002C 285 : needed by Replacement algorithm.
00000000 002C 286 HIRTSL_PAGE2PTR: .LONG 0 : System Virtual Address of scratch page
0030 287 : needed by Replacement algorithm.
00000000 0030 288 HIRTSL_PAGE3PTR: .LONG 0 : System Virtual Address of scratch page
0034 289 : needed by Replacement algorithm.
0034 290
00000000 0034 291 HIRTSL_SVAPTE0: .LONG 0 : SVAPTE of page 0.
00000000 0038 292 HIRTSL_SVAPTE1: .LONG 0 : SVAPTE of page 1.
00000000 003C 293 HIRTSL_SVAPTE2: .LONG 0 : SVAPTE of page 2.
00000000 0040 294 HIRTSL_SVAPTE3: .LONG 0 : SVAPTE of page 3.
0044 295
0000 0044 296 HIRT$W_BOFF0: .WORD 0 : BOFF of page 0.
0000 0046 297 HIRT$W_BOFF1: .WORD 0 : BOFF of page 1.
```

0000	0048	298	HIRT\$W_BOFF2:	.WORD	0		; BOFF of page 2.
0000	004A	299	HIRT\$W_BOFF3:	.WORD	0		; BOFF of page 3.
	004C	300					
	004C	301	; Array of words that give the relative RCT sector number contained in a page.				
	004C	302					
0000	004C	303	HIRT\$W_PG0CNTNT:	.WORD	0		; Page 0 contents.
0000	004E	304	HIRT\$W_PG1CNTNT:	.WORD	0		; Page 1 contents.
0000	0050	305	HIRT\$W_PG2CNTNT:	.WORD	0		; Page 2 contents.
0000	0052	306	HIRT\$W_PG3CNTNT:	.WORD	0		; Page 3 contents.
	0054	307					
	0054	308	; Static storage needed by several routines that read and write RCT blocks.				
	0054	309					
0000	0054	310	HIRT\$W_SECTORNO:	.WORD	0		; Sector number.
0000	0056	311	HIRT\$W_PAGENO:	.WORD	0		; Page number.
	0058	312					
	0058	313	; Static storage needed by SEARCH_RCT subroutine.				
	0058	314					
00000000	0058	315	HIRT\$L_RBN:	.LONG	0		; RBN returned to caller.
00000000	005C	316	HIRT\$L_MATCHRBN:	.LONG	0		; Previous RBN that failed.
00000000	0060	317	HIRT\$L_BADRBN:	.LONG	0		; Bad RBN descriptor contents,
	0064	318					; used in STEP15 error recovery.
00000000	0064	319	HIRT\$L_STARTBLK:	.LONG	0		; Sector number of Primary RBN.
00000000	0068	320	HIRT\$L_RCTBLOCK:	.LONG	0		; Current RCT sector number.
00000000	006C	321	HIRT\$L_OFFSET:	.LONG	0		; Offset into current RCT sector.
	0070	322					
	0070	323	; HIRT SUBSTACK - used by single threaded replacement algorithm as a return				
	0070	324	; point stack.				
	0070	325					
00000000	0070	326	HIRT\$L_STKPTR:	.LONG	0		; Pointer to top of SUBSTACK.
00000000	0074	327	HIRT\$L_SUBSTACK:	.LONG	0,0,0,0,0		; SUBSTACK itself.
00000000	0084						
00000005	0088	328	HIRT\$K_SUBSTKLN=<.-HIRT\$L_SUBSTACK>/4				; Total length of SUBSTACK in longwords.
	0088	329					
00000000		330	.RESTORE				

```
0000 332 .SATTIL - HIR Error Processing Information
0000 333
0000 334 ; Constants used in forming HIR error messages
0000 335
0000 336 $VIELD HIRER, 0, <-
0000 337 <STEP, 8, M>, - ; Step number
0000 338 <TYPE, 4, M>, - ; Error type
0000 339 <ONLINE, M> - ; Online (not HIR)
0000 340 >
00000000 0000 341 HIRERSM_REPLACE = 0
0000 342
0000 343 ; Error type codes:
00000001 0000 344 HIRERSK_READ = 1 ; READ
00000002 0000 345 HIRERSK_WRITE = 2 ; WRITE
00000003 0000 346 HIRERSK_RCTFULL = 3 ; RCT FULL
00000004 0000 347 HIRERSK_REPFail = 4 ; REPLACE FAILURE
0000 348
0000 349 .SAVE
00000000 0000 350 .PSECT $$$301_HIR_ERRORS LONG
0000 351
0000 352 HIR_ERR_TYPES:
0000 353
0000 354 .BYTE 0
44 41 45 52 00' 0001 355 .ASCIC /READ/
04 0001
45 54 49 52 57 00' 0006 356 .ASCIC /WRITE/
05 0006
4C 4C 55 46 20 54 43 52 00' 000C 357 .ASCIC /RCT FULL/
08 000C
49 41 46 20 45 43 41 4C 50 45 52 00' 0015 358 .ASCIC /REPLACE FAILURE/
45 52 55 4C 0021
0F 0015
0025 359
0025 360 HIR_ERR_REPLACE:
0025 361
45 43 41 4C 50 45 52 00' 0025 362 .ASCIC /REPLACE/
07 0025
002D 363
002D 364 HIR_ERR_ONLINE:
002D 365
45 4E 49 4C 4E 4F 00' 002D 366 .ASCIC /ONLINE/
06 002D
0034 367
0034 368 HIR_ERR_SEG1:
0034 369
65 72 65 74 6E 75 6F 63 6E 65 20 00' 0034 370 .ASCIC / encountered a /
20 61 20 64 0040
0F 0034
0044 371
0044 372 HIR_ERR_SEG2:
0044 373
20 6E 69 20 72 6F 72 72 65 20 00' 0044 374 .ASCIC / error in /
0A 0044
004F 375
004F 376 HIR_ERR_SEG3:
004F 377
20 70 65 74 73 20 00' 004F 378 .ASCIC / step /
```

```
06 004F
    0056 379
    0056 380 ;
    0056 381 ; Compute maximum HIR message size
    0056 382 ;
    0056 383
00000012 0056 384 DEVNAMSIZ = 18 ; size of a device name
00000007 0056 385 TYPsiz = 7 ; largest error type character count
00000007 0056 386 FUNCSIZ = 7 ; largest REPLACE/ONLINE character count
00000002 0056 387 STEPSIZ = 2 ; max characters in step number
00000023 0056 388 FIXEDSIZ = <. - HIR_ERR_SEG1> + 1 ; size of fixed text
    0056 389
00000012 0056 390 HIRERSK_DEVNAMSIZ = DEVNAMSIZ
00000045 0056 391 SIZE = FIXEDSIZ + DEVNAMSIZ + TYPsiz + FUNCSIZ + STEPSIZ
00000048 0056 392 HIRERSK_MSGSIZE = <SIZE + 3> & ^c3
    0056 393
00000000 0056 394 .RESTORE
    0000 395
    0000 396 ;++
    0000 397 ;
    0000 398 ; HIR_ERROR
    0000 399 ;
    0000 400 ; This macro calls the HIR error reporting routine.
    0000 401 ;
    0000 402 ; Parameters:
    0000 403 ;
    0000 404 ; STEP step number in which the error occurred
    0000 405 ; TYPE error type (one of READ / WRITE / RCTFULL)
    0000 406 ; FUNC function incurring error (one of REPLACE / ONLINE;
    0000 407 ; default = REPLACE)
    0000 408 ;--
    0000 409
    0000 410 .MACRO HIR_ERROR step, type, func=REPLACE
    0000 411 ASSUME HIRERSV ONLINE LE 15
    0000 412 MOVZWL #<HIRERSM 'func' -
    0000 413 +<HIRERSK 'type' @ HIRERSV_TYPE > -
    0000 414 + 'step'>, R0
    0000 415 BSBW DUSHIR_ERROR
    0000 416 .ENDM HIR_ERROR
```

```
0000 418 .SBTTL DUSINIT_HIRT - Initialize Host Initiated Replacement Table
0000 419 :++
0000 420 :
0000 421 : DUSINIT_HIRT - Initialize Host Initiated Replacement Table
0000 422 :
0000 423 : Functional Description:
0000 424 :
0000 425 : This routine initializes the HIRT, if it has not already been
0000 426 : initialized. There is one HIRT per system and it resides in the
0000 427 : disk class driver. It is initialized the first time an intelligent
0000 428 : controller that requires Host Initiated Bad Block Replacement is
0000 429 : brought online.
0000 430 :
0000 431 : HIRT initialization includes setting up its FLINK and BLINK, allocating
0000 432 : a permanent CDRP for it, allocating an RSPID for it, allocating an
0000 433 : MSCP buffer (without Send Credit on any connection) and allocating
0000 434 : four pages of memory that are needed by the replacement algorithm.
0000 435 :
0000 436 : Inputs:
0000 437 :
0000 438 : R3 CDDb address
0000 439 : R4 PDT address
0000 440 : R5 Connection permanent CDRP address
0000 441 :
0000 442 : Outputs:
0000 443 :
0000 444 : Registers R0-R2 are modified.
0000 445 : Registers R3-R5 are preserved.
0000 446 :
0000 447 : Implicit Outputs:
0000 448 :
0000 449 : The HIRT is initialized as described above.
0000 450 :--
0000 451 :
0000 452 DUSINIT_HIRT::
0000 453 :
0000 454 POPL CDDb$S_SAVED_PC(R3) ; Save caller's PC in CDDb.
0000 455 BBBS s^#HIRT$V_ACTIVE,- ; Now see if HIRT already init'ed.
0000 456 HIRT$W_STS,10$
0000 457 BRW END_INIT_HIRT ; Branch around if already initialized.
0000 458 :
0000 459 10$: CLRL HIRT$L_RPLQFL ; Singly linked list with second
0000 460 MOVAB HIRT$L_RPLQFL,- ; longword pointing to tail of list.
0000 461 HIRT$L_RPLQTP
0000 462 :
0000 463 BISW s^#HIRT$M_BUSY,- ; Prevent use of HIRT until fully
0000 464 HIRT$W_STS ; init'ed.
0000 465 :
0000 466 : Allocate the CDRP to be used and re-used during the I/O operations
0000 467 : associated with dynamic Host Initiated Replacement of bad blocks.
0000 468 :
0000 469 :
0000 470 20$: MOVZWL #IRPSK_LENGTH,R1 ; R1 contains amount of space to alloc.
0000 471 BSBW ALLOC_POOL ; Allocate space. Returns R2=>space.
0000 472 :
0000 473 MOVAB #DYN$C_IRP,- ; Make first part of CDRP look like an
0000 474 IRP$B_TYPE(R2) ; IRP.
```

44 A3 8ED0 0000 454
00 00 E3 0004 455
03 000E'CF 0006 456
00A4 31 000A 457
0000'CF D4 000D 459
0000'CF 9E 0011 460
0004'CF 0015 461
000E'CF 02 A8 0018 462
0018 463
001D 464
001D 465
001D 466
001D 467
001D 468
001D 469
51 00C4 8F 3C 001D 470
008F 30 0022 471
0A 90 0025 472
0A A2 0025 473
0027 474


```
08 A2 51 B0 0029 475      MOVW    R1,IRPSW_SIZE(R2)      ; Save type and size inside "IRP".
55 60 A2 9E 002D 476      MOVAB    -CDRPSL_IDQFL(R2),R5      ; R5 => CDRP portion of packet.
0020'CF 55 D0 0031 477      MOVL     R5,HIRTSL_CDRP      ; Save address of replacement CDRP.
      FFA0 8F B0 0036 478      MOVW    #CDRPSL_IDQFL,-      ; Size field in CDRP portion is negative
      08 A5 90 003A 479      MOVW    CDRPSW_CDRPSIZE(R5)      ; offset of IRP from CDRP portion.
      39 90 003C 480      MOVW    #DYN$C_CDRP,-      ; Mark type of CDRP portion.
      0A A5 90 003E 481      MOVW    CDRPSB_CD_TYPE(R5)      ;
      24 A5 D4 0040 482      CLRL     CDRPSL_CDT(R5)      ; So far we have no connection for CDRP.
      28 A5 D4 0043 483      CLRL     CDRPSL_RWCPT(R5)      ; This CDRP will not use RWAITCNT.
      2C A5 D4 0046 484      CLRL     CDRPSL_LBUFH_AD(R5)      ; Signal that no mapping resources allocated
      0049 485
      20 A5 D4 0049 486      CLRL     CDRPSL_RSPID(R5)      ; Clear RSPID to show none yet allocated.
      1C A5 D4 004C 487      CLRL     CDRPSL_MSG_BUF(R5)      ; Likewise show no MSCP buffer.
40 A5 10 9A 004F 488      MOVZBL    #CDRPSM_HIRT,-      ; Set HIRT permanent CDRP flag.
      0053 489      CDRPSL_DUTUFLAGS(R5)
      0053 490
      0053 491
      0053 492      ; Allocate pages from pool to serve as buffers when reading RCT sectors
      0053 493      ; during replacement of bad blocks on a disk.
      0053 494
      0053 495
51 020C 8F 3C 0053 496 50$: MOVZWL    #512+12,R1      ; R1 contains amount of space for a
      0058 497      ; page and a VMS structure header.
      0058 498
55 00D0 C3 9E 0058 499      MOVAB    CDDBSA_PRCMDRP(R3), R5      ; ALLOC_POOL needs R5 => Permanent CDRP.
      0054 30 005D 500      BSBW     ALLOC_POOL      ; Allocate space. Returns R2=>space.
55 0020'CF D0 0060 501      MOVL     HIRTSL_CDRP,R5      ; Restore R5 => Hirt CDRP.
      0065 502
      0065 503      ASSUME    CDDBSB_SUBTYPE EQ CDDBSB_TYPE+1
      B0 0065 504      MOVW    #DYN$C_CLASSDRV-      ; Place type and subtype descriptors
      0066 505      !<DYN$C_CD BBRPGa8>,-      ; into header using convenient (CDDB)
      0066 506      CDDBSB_TYPE(R2)      ; offset definition.
      08 A2 0264 8F B0 006B 507      MOVW    R1,CDDBSW_SIZE(R2)      ; Also place size into header.
      52 0C A2 9E 006F 508      MOVAB    12(R2),R2      ; R2 => beyond VMS structure header.
      51 D4 0073 509      CLRL     R1      ; Clear loop index register.
      0075 510
      0024'CF41 D5 0075 511 80$: TSTL     HIRTSL_PAGEOPTR[R1]      ; Test where to put address of allocated pag
      04 13 007A 512      BEQL     90$      ; EQL implies we have found a depository.
      51 D6 007C 513      INCL     R1      ; Else bump index register
      F5 11 007E 514      BRB      80$      ; and go back and try again.
      0080 515
      0024'CF41 52 D0 0080 516 90$: MOVL     R2,HIRTSL_PAGEOPTR[R1]      ; Else save Page address.
0044'CF41 52 FE00 8F AB 0086 517      BICW3    #^XFE00,R2,-      ; Calculate BOFF of page just allocated
      008F 518      HIRTSL_BOFF0[R1]      ; and save it in the Indexed slot.
      09 EF 008F 519      EXTZV    S^#VASS_VPN,-      ; Now calculate SVAPTE of allocated
      52 52 15 0091 520      S^#VASS_VPN,R2,R2      ; page. First get VPN.
50 00000000'GF D0 0094 521      MOVL     G^MMGSGC_SPTBASE,R0      ; Then R0 => base of system page table.
      0098 522
      0034'CF41 6042 DE 009B 523      MOVAL    (R0)[R2],HIRTSL_SVAPTE0[R1]
      00A2 524      ; Move SVAPTE into proper slot.
      03 51 D1 00A2 525      CMPL     R1,#3      ; See if we are done allocating.
      AC 19 00A5 526      BLSS     50$      ; LSS implies NO, so we go to try again.
      00A7 527
      000E'CF 02 AA 00A7 528      BICW     s^#HIRTSM_BUSY,-      ; Allow use of HIRT now that it has
      00AC 529      HIRTSM_STS      ; been initialized.
55 00D0 C3 9E 00AC 530      MOVAB    CDDBSA_PRCMDRP(R3), R5      ; Get controller permanent CDRP in R5.
      00B1 531
```

DUHIRT
V04-000

H 13
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
DUS\$INIT_HIRT - Initialize Host Initiated 5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 13
(6)

DU
VC

44 B3 17 00B1 532 END_INIT_HIRT:
00B1 533 JMP @CDDBSL_SAVED_PC(R3) ; Return to caller.


```
00B4 535      .SBTTL  ALLOC_POOL
00B4 536
00B4 537      ;+
00B4 538      ; This subroutine allocates and zeroes nonpaged pool.
00B4 539
00B4 540      ; Inputs:
00B4 541
00B4 542      ;         R1          -# bytes of pool to allocate
00B4 543      ;         R5          -Addr of CDRP
00B4 544
00B4 545      ; Outputs:
00B4 546
00B4 547      ;         R0          -0/1 for fail/success
00B4 548      ;         R1          -# bytes actually allocated
00B4 549      ;         R2          -Addr of buffer allocated
00B4 550      ; -
00B4 551
00B4 552      ALLOC_POOL:
00B4 553      ; Allocate and zero pool
00B4 554      PUSHL  R3
00B4 555      JSB   G^EX$ALONONPAGED
00B4 556      BLBC  R0,10$
00B4 557      PUSHR  #^M<R0,R1,R2,R4,R5>
00B4 558      MOVCS  #0,(SP),#0,R1,(R2)
00B4 559
00B4 560      POPR   #^M<R0,R1,R2,R4,R5>
00B4 561      POPL   R3
00B4 562      RSB
00B4 563
00B4 564      10$:
00B4 565      ; Allocation failure.
00B4 566      ; Prepare to wait awhile before trying again.
00B4 567
00B4 568      POPL   CDRP$$_FR3(R5)
00B4 569      MOVL   R4, CDRP$$_FR4(R5)
00B4 570      POPL   CDRP$$_SAVD_RTN(R5)
00B4 571      MOVL   R1, CDRP$$_RSPID(R5)
00B4 572      FORK_WAIT
00B4 573      MOVL   CDRP$$_RSPID(R5), R1
00B4 574      CLRL   CDRP$$_RSPID(R5)
00B4 575      PUSHL  CDRP$$_SAVD_RTN(R5)
00B4 576      BRB   ALLOC_POOL
00B4 577      ; Save R3 in R5=>UCB or CDRP.
00B4 578      ; Likewise R4
00B4 579      ; and caller's return address.
00B4 580      ; Save allocation size.
00B4 581      ; Wait awhile.
00B4 582      ; Restore size of block to allocate.
00B4 583      ; Restore CDRP field.
00B4 584      ; Restore caller's return address.
00B4 585      ; Go try again.
```

62 51 00 6E 00 2C 00C1 558 MOVCS #0,(SP),#0,R1,(R2)

37 BA 00C7 560 POPR #^M<R0,R1,R2,R4,R5>

53 8ED0 00C9 561 POPL R3

05 00CC 562 RSB

00CD 563

10 A5 8ED0 00CD 567 POPL CDRP\$\$_FR3(R5)

14 A5 54 D0 00D1 568 MOVL R4, CDRP\$\$_FR4(R5)

18 A5 8ED0 00D5 569 POPL CDRP\$\$_SAVD_RTN(R5)

20 A5 51 D0 00D9 570 MOVL R1, CDRP\$\$_RSPID(R5)

51 20 A5 D0 00DD 571 FORK_WAIT

20 A5 D4 00E3 572 MOVL CDRP\$\$_RSPID(R5), R1

18 A5 DD 00E7 573 CLRL CDRP\$\$_RSPID(R5)

C5 11 00EA 574 PUSHL CDRP\$\$_SAVD_RTN(R5)

00ED 575 BRB ALLOC_POOL

```
00EF 577 .SBTTL DUSLOCK_HIRT - Gain exclusive access to HIRT
00EF 578 :++
00EF 579 :
00EF 580 : DUSLOCK_HIRT - Gain exclusive access to HIRT
00EF 581 :
00EF 582 : Functional Description:
00EF 583 :
00EF 584 : Gain exclusive access to the H(ost) I(nitiated) bad block
00EF 585 : R(eplacement) T(able). Also, lockout new activity on the unit by
00EF 586 : bumping UCBSW_RWAITCNT.
00EF 587 :
00EF 588 : Inputs:
00EF 589 :
00EF 590 : R3 UCB address
00EF 591 : R5 a fork block
00EF 592 :
00EF 593 : Implicit Inputs:
00EF 594 :
00EF 595 : (SP) caller's return address
00EF 596 : 4(SP) caller's caller's return address
00EF 597 :
00EF 598 : Outputs:
00EF 599 :
00EF 600 : R0 - R2 are modified.
00EF 601 : All other registers are preserved.
00EF 602 :
00EF 603 : Implicit Outputs:
00EF 604 :
00EF 605 : HIRT is owned by caller.
00EF 606 :--
00EF 607 :
00EF 608 DUSLOCK_HIRT::
00EF 609 :
000E'CF 00 E0 00EF 610 BBS s^#HIRTSV_ACTIVE, - ; Make sure HIRT is active.
00EF 611 HIRTSW_STS, 5$
00F4 611 :
00F5 612 BUG_CHECK DISKCLASS,FATAL
00F9 613 :
00F9 614 5$: INCW UCBSW_RWAITCNT(R3) ; Prevent new CDRP's from being begun
00FC 615 : ; on this UCB.
000E'CF 01 E3 00FC 616 BBBS s^#HIRTSV_BUSY, - ; Allocate the Host Initiated Replac
00FC 617 HIRTSW_STS, 10$ ; ment table (HIRT).
0101 617 :
0102 618 :
0102 619 MOVQ R3, FKBSL_FR3(R5) ; If here, already allocated, save
0106 620 POPL FKBSL_FPC(R5) ; thread context in fork block.
010A 621 :
010A 622 : Thread R5 (a fork block) onto the tail of the singly threaded list
010A 623 : of fork blocks awaiting use of the HIRT. The listhead is a quadword
010A 624 : whose first longword points to the first fork block on the list and
010A 625 : whose second longword points to the last forkblock on the list. An
010A 626 : empty list is characterized by having the first longword contain a
010A 627 : zero with the second longword pointing to the first longword. Each
010A 628 : fork block on the list, has the first longword of its link quadword
010A 629 : pointing to the next fork block on the list, with the last fork
010A 630 : block containing a zero in this longword. The second longword of
010A 631 : each fork block's link quadword contains the address of the CDDB of
010A 632 : the intelligent controller associated with the device unit
010A 633 : attempting to gain exclusive use of the HIRT.
```

```

010A 634
010A 635
010A 636
010A 637
65 D4 010A 638 CLRL FKBSL_FQFL(R5) ; Prepare this fork block to be at
04 A5 00BC C3 D0 010C 639 ; tail of the list.
0004'DF 65 9E 0112 640 MOVL UCBSL_CDDDB(R3), - ; Second longword of link quadword
0004'CF 65 9E 0112 641 FKBSL_FQBL(R5) ; points to CDDDB.
0117 642 MOVAB FKBSL_FQFL(R5), - ; Move address of this fork block into
0117 643 @HIRTSL_RPLQTP ; forward ptr of previous tail.
011C 644 MOVAB FKBSL_FQFL(R5), - ; Also move address of this fork block
05 011C 645 HIRTSL_RPLQTP ; to list tail pointer.
011D 646 RSB ; Terminate this execution thread by
011D 647 ; returning to caller's caller.
011D 648
011D 649 10$: ; The HIRT is owned.
01 10 011D 650 BSBB GRANT_HIRT ; Call to initialize various structures
011F 651 ; with data of the new HIRT owner.
05 011F 652 RSB ; And return to caller who now owns HIRT.

```

```

0120 654 .SBTTL GRANT_HIRT - Complete granting access to the HIRT
0120 655 :++
0120 656 :
0120 657 : GRANT_HIRT - Complete granting access to the HIRT
0120 658 :
0120 659 : Functional Description:
0120 660 :
0120 661 : This routine is called from DUSLOCK_HIRT and DUSUNLOCK_HIRT, upon
0120 662 : granting ownership of the HIRT to a thread. GRANT_HIRT initializes
0120 663 : various data fields reflecting this ownership and facilitating the
0120 664 : thread's use of the HIRT CDRP.
0120 665 :
0120 666 : Note:
0120 667 : Since both subroutines that require ownership of the HIRT, REPLACE_LBN
0120 668 : and ONLINE_COMPLETE, make use of the user's original RSPID so as to be
0120 669 : able to co-relate all Error Log messages generated by a user I/O
0120 670 : request, GRANT_HIRT passes the RSPID from the user CDRP to the HIRT
0120 671 : permanent CDRP.
0120 672 :
0120 673 : Inputs:
0120 674 :
0120 675 : R3      UCB address
0120 676 : R5      User CDRP address
0120 677 :
0120 678 : Outputs:
0120 679 :
0120 680 : Various HIRT and CDRP fields updated.
0120 681 :--
0120 682 :
0120 683 GRANT_HIRT:
0120 684 :
0120 685 : PUSHL CDRP$R_RSPID(R5)      ; Pass current RSPID to HIRT CDRP.
0120 686 : CLRL  CDRP$R_RSPID(R5)      ; Prevent spurious deallocates.
0120 687 : MOVL  R5, HIRT$R_SAVDCDRP    ; Save given R5.
0120 688 : MOVAB HIRT$R_SUBSTACK, -     ; Initialize SUBSTACK in HIRT.
0120 689 : HIRT$R_STKPTR
0120 690 : MOVL  R3, HIRT$R_OWNUCB      ; Indicate who owns HIRT.
0120 691 :
0120 692 : MOVL  (SP), R5              ; Get RSPID.
0120 693 : MOVQ  R0, -(SP)            ; Save registers.
0120 694 : FIND_RSPID_RDTE            ; Lookup RDT for RSPID.
0120 695 : BLBS  R0, 10$              ; Branch if lookup successful.
0120 696 : BUG CHECK DISKCLASS,FATAL ; Else, major inconsistency.
0120 697 : 10$: MOVE  HIRT$R_CDRP, -     ; For now pass ownership of RDTE to HIRT
0120 698 : RD$R_CDRP(R5)              ; permanent CDRP.
0120 699 : MOVQ  (SP)+, R0            ; Restore saved registers.
0120 700 : MOVL  HIRT$R_CDRP, R5      ; R5 => permanent replacement CDRP.
0120 701 : POPL  CDRP$R_RSPID(R5)    ; Pickup RSPID to use thruout replacement.
0120 702 : CLRL  CDRP$R_LBUFH_AD(R5) ; Indicate no resources yet allocated
0120 703 : CLRL  CDRP$R_MSG_BUF(R5)  ; except RSPID.
0120 704 : MOVL  R3, CDRP$R_UCB(R5)   ; Make HIRT permanent CDRP => this UCB.
0120 705 :                                ; This allow UNIBUS mapping to work.
0120 706 : MOVL  UCB$R_CDT(R3), -     ; Place CDT pointer into CDRP for handy
0120 707 : CDRP$R_CDT(R5)            ; reference by SCS routines. Note this
0120 708 :                                ; must be done each time the HIRT is
0120 709 :                                ; locked since we may be using a different
0120 710 :                                ; port (and therefore CONNECTION) each

```

DUHIRT
V04-000

M 13
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
GRANT_HIRT - Complete granting access to 5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 18
(9)

56 A3	9E	016B	711			
28 A5		016B	712	MOVAB	UCBSW_RWAITCNT(R3),-	; time.
0080 8F	AA	016E	713		CDRPSR_RWCPTTR(R5)	; Point CDRP field to UCB field.
000E CF		0170	714	BICW	#HIRTSM_ERLOGIP,-	; Initialize bit.
	05	0174	715		HIRTSM_STS	
		0177	716	RSB		; Return to caller.

```

0178 718 .SBTTL DU$UNLOCK_HIRT - Release HIRT access
0178 719 :++
0178 720 :
0178 721 : DU$UNLOCK_HIRT - Release HIRT access
0178 722 :
0178 723 : Functional Description:
0178 724 :
0178 725 :     Caller wishes to relinquish exclusive control of the HIRT.
0178 726 :     It becomes the current owner's obligation to restart the first
0178 727 :     thread (if any are there) that may be waiting on the HIRT wait
0178 728 :     list.
0178 729 :
0178 730 : Note:
0178 731 :
0178 732 :     DU$UNLOCK_HIRT passes back the user's RSPID from the HIRT permanent
0178 733 :     CDRP to the user's CDRP.
0178 734 :
0178 735 : Inputs:
0178 736 :
0178 737 :     R3      UCB of HIRT owner
0178 738 :
0178 739 : Implicit Inputs:
0178 740 :
0178 741 :     HIRT owned by caller
0178 742 :
0178 743 : Outputs:
0178 744 :
0178 745 :     R5      Original CDRP address
0178 746 :     All other registers are preserved.
0178 747 :
0178 748 : Implicit Outputs:
0178 749 :
0178 750 :     HIRT ownership relinquished. If any threads are on the HIRT wait list,
0178 751 :     first of these is granted HIRT ownership and is started up.
0178 752 : --
0178 753 :
0178 754 DU$UNLOCK_HIRT::
0178 755 :
0178 756 :     DECW    UCBSW_RWAITCNT(R3)      ; Decrement to again allow normal I/O.
0178 757 :     MOVQ    R0, -7(SP)                ; Save some registers.
0178 758 :     PUSH    #^M<R2,R3,R4>            ; Save more registers.
0178 759 :     MOVL    R3, R5                    ; Setup UCB for UNSTALLUCB.
0178 760 :     JSB     G^SCS$UNSTALLUCB         ; Call to start up IRP's on UCB$L_10QFL.
0178 761 :     POPR    #^M<R2,R3,R4>            ; Restore registers.
0178 762 :
0178 763 :     MOVL    HIRT$L_CDRP, R5           ; R5 => HIRT CDRP.
0178 764 :     PUSH    CDRP$L_RSPID(R5)         ; Save current RSPID so as to restore to
0178 765 :                                     ; user CDRP.
0178 766 :     BEQL    15$                       ; EQL implies RSPID has been deallocated
0178 767 :                                     ; due to re-CONNECT. Branch around.
0178 768 :     CLRL    CDRP$L_RSPID(R5)         ; Prevent spurious deallocates.
0178 769 :     MOVL    (SP), R5                 ; Get RSPID.
0178 770 :     FIND_RSPID_RDTE                  ; Lookup RDT entry for RSPID.
0178 771 :     BLBS    R0, 5$                   ; Branch if lookup successful.
0178 772 :     BUG CHECK DISKCLASS, FATAL       ; Else, major inconsistency.
0178 773 :     MOVE    HIRT$L_SAVDCDRP, R0      ; Get saved CDRP address.
0178 774 :     BNEQ    10$                       ; Branch if there still is a saved CDRP.

```



```

0205 814 .SBTTL DUSTEST_HIRT_RWAITCNT - Accumulate RWAITCNT for HIRT
0205 815 :++
0205 816 :
0205 817 : DUSTEST_HIRT_RWAITCNT - Accumulate RWAITCNT for HIRT
0205 818 :
0205 819 : Functional Description:
0205 820 :
0205 821 : This routine accumulates an RWAITCNT value for the input UCB based
0205 822 : upon the amount RWAITCNT has been increment for HIRT usage.
0205 823 :
0205 824 : Inputs:
0205 825 :
0205 826 : R0 RWAITCNT accumulator
0205 827 : R5 UCB address
0205 828 :
0205 829 : Outputs:
0205 830 :
0205 831 : R0 RWAITCNT accumulator (with additions for HIRT usage)
0205 832 : R1 destroyed
0205 833 :
0205 834 : All other registers preserved.
0205 835 :--
0205 836 :
0205 837 DUSTEST_HIRT_RWAITCNT::
0205 838 :
1D 000E'CF 01 E1 0205 839 BBC s^#HIRT$V_BUSY, HIRT$W_STS, 90$ ; Branch if HIRT not busy.
0208 840 :
0014'CF 55 D1 0208 841 CMPL R5, HIRT$L_OWNUCB ; Is the UCB the HIRT owner?
02 12 0210 842 BNEQ 10$ ; Branch if not HIRT owner.
50 D6 0212 843 INCL R0 ; Else, increment RWAITCNT.
0214 844 :
0214 845 10$: ASSUME FKB$FQFL EQ 0
51 0000'CF 9E 0214 846 MOVAB HIRT$C_RPLQFL, R1 ; Init. 'previous' wait CDRP.
51 61 D0 0219 847 11$: MOVL FKB$FQFL(R1), R1 ; Link to next waiting CDRP.
0A 13 021C 848 BEQL 90$ ; Branch if no more waiters.
BC A1 55 D1 021E 849 CMPL R5, CDRP$L_UCB(R1) ; Is this waiter for this UCB?
F5 12 0222 850 BNEQ 11$ ; Branch if not right UCB.
50 D6 0224 851 INCL R0 ; Else, increment RWAITCNT.
F1 11 0226 852 BRB 11$ ; Loop, till no more waiters.
0228 853 :
05 0228 854 90$: RSB ; All done; exit.

```



```
0229 856 .SBTTL DUSCANCEL_FROM_HIRT - Cancel requests from the HIRT
0229 857 :++
0229 858 :
0229 859 : DUSCANCEL_FROM_HIRT - Cancel requests from the HIRT
0229 860 :
0229 861 : Functional Description:
0229 862 :
0229 863 : This routine is called to locate and cancel any I/O requests current
0229 864 : or pending for host initiated replacement. The queue of pending
0229 865 : requests is scanned. The then current HIRT owner is tested.
0229 866 :
0229 867 : The HIRT wait queue is scanned and all CDRPs that meet the cancel
0229 868 : criteria are removed from the HIRT wait queue and queued for I/O post
0229 869 : processing. The current owner of the HIRT (if any) is similarly
0229 870 : tested against the cancel criteria and if needed it too is queued for
0229 871 : I/O post processing. The HIRT is left 'ownerless' in the sense that
0229 872 : HIRT$S_SAVDCDRP is left zero. This allows the current HIRT I/O to
0229 873 : continue until it completes on its own. Then, when the HIRT is
0229 874 : UNLOCKED, the 'ownerless' state is noticed and the HIRT thread for the
0229 875 : former owner is evaporated.
0229 876 :
0229 877 : Inputs:
0229 878 :
0229 879 : R3 UCB address
0229 880 : R5 Cancel CDRP address
0229 881 :
0229 882 : Implicit Inputs:
0229 883 :
0229 884 : CDRP$W_DUTUCNTR(R5) count of number of times to increment RWAITCNT
0229 885 : after cancel is completed.
0229 886 :
0229 887 : Outputs:
0229 888 :
0229 889 : R0 through R2 are destroyed
0229 890 : All other registers are preserved.
0229 891 :
0229 892 : Implicit Outputs:
0229 893 :
0229 894 : CDRP$W_DUTUCNTR(R5) count of number of times to increment RWAITCNT
0229 895 : after cancel is completed.
0229 896 :--
0229 897 :
0229 898 DUSCANCEL_FROM_HIRT::
0229 899 :
000E'CF 01 E1 0229 900 BBC s^#HIRT$V BUSY, - ; Is the HIRT busy? If not, there is
79 0229 901 HIRT$W_STS, 900$ ; nothing to do: so branch to exit.
0229 902 :
0229 903 ; Scan the HIRT pending requests queue
0229 904 :
0229 905 ASSUME FKB$S_FQFL EQ 0
51 0000'CF 9E 0229 906 MOVAB HIRT$C_RPLQFL, R1 ; Get 'previous' CDRP on wait list.
0234 907 :
52 61 D0 0234 908 10$: MOVL FKB$S_FQFL(R1), R2 ; Get next CDRP.
3C 13 0237 909 BEQL 100$ ; Branch if no more CDRPs on wait list.
04 A2 00BC C3 D1 0239 910 CML UCB$S_CDDDB(R3), - ; Is CDRP for this CDDDB?
023F 911 FKB$S_FQBL(R2)
06 12 023F 912 BNEQ 40$ ; Branch if not the right CDDDB.
```

```

51 52 D0 0241 913 IFCANCEL cdrp=(R2), then=70$ ; Branch if CDRP should be canceled.
    E8 11 0247 914 40$: MOVL R2, R1 ; Current becomes previous.
    024A 915 BRB 10$ ; Loop through all waiting CDRPs.
    024C 916
61 62 D0 024C 917 70$: MOVL FKB$FQFL(R2), - ; Unlink cancelable CDRP.
    024F 918 FKB$FQFL(R1)
    05 12 024F 919 BNEQ 75$ ; If cancelable CDRP was last,
0004'CF 51 D0 0251 920 MOVL R1, HIRT$L_RPLQTP ; adjust queue tail pointer.
50 52 D0 0256 921 75$: MOVL R2, R0 ; Setup CDRP to cancel.
44 A5 B6 0259 922 INCW CDRP$W_DUTUCNTR(R5) ; Account for RWAITCNT increment during
    025C 923 ; attempt to lock the HIRT.
    3F BB 025C 924 PUSHF #^M<R0,R1,R2,R3,R4,R5> ; Save registers.
55 50 D0 025E 925 MOVL R0, R5 ; Setup for message deallocate.
    0261 926 DEALLOC_MSG_BUF ; Deallocate End Message that told of
    0264 927 ; block to be replaced.
    3F BA 0264 928 POPF #^M<R0,R1,R2,R3,R4,R5> ; Restore registers.
    0266 929 POST_CDRP status=SS$_CANCEL ; Insert IRP/CDRP in IOPST queue.
    BF 11 0273 930 BRB 10$ ; Branch back to scan entire list.
    0275 931
    0275 932 100$: ; Is the HIRT owner a cancelable CDRP? If so retrieve this HIRT owner
    0275 933 ; CDRP, clear HIRT$L_SAVDCDRP, and POST CDRP the retrieved CDRP. Note
    0275 934 ; this works in conjunction with DUS$UNLOC^ HIRT and DUS$STRTQ_HIRT_CDRP
    0275 935 ; which must be prepared to find HIRT$L_SAVDCDRP = 0.
    0275 936
BC A5 0014'CF D1 0275 937 CMPL HIRT$L_OWNUCB, - ; Check for correct HIRT owner UCB.
    027B 938 CDRP$L_UCB(R5)
    2B 12 027B 939 BNEQ 900$ ; Branch in wrong HIRT owner.
52 001C'CF D0 027D 940 MOVL HIRT$L_SAVDCDRP, R2 ; Get CDRP owner of HIRT.
    24 13 0282 941 BEQL 900$ ; Branch if owner already canceled,
    0284 942 ; replacement running to completion.
    0284 943 IFNOCANCEL cdrp=(R2), then=900$ ; Branch if owner shouldn't be canceled.
001C'CF D4 028A 944 CLRL HIRT$L_SAVDCDRP ; Else, indicate HIRT owner canceled.
50 52 D0 028E 945 MOVL R2, R0 ; Setup CDRP to cancel.
    0291 946 ; Following instruction deleted due to its causing RWAITCNT to be
    0291 947 ; decremented twice; once here and once Replacement runs to completion.
    0291 948 ; INCW CDRP$W_DUTUCNTR(R5) ; Account for owning the HIRT.
    3F BB 0291 949 PUSHF #^M<R0,R1,R2,R3,R4,R5> ; Save registers.
55 50 D0 0293 950 MOVL R0, R5 ; Setup for message deallocate.
    0296 951 DEALLOC_MSG_BUF ; Deallocate End Message that told of
    0299 952 ; block to be replaced.
    3F BA 0299 953 POPF #^M<R0,R1,R2,R3,R4,R5> ; Restore registers.
    029B 954 POST_CDRP status=SS$_CANCEL ; Insert IRP/CDRP in IOPST queue.
    02A8 955
    05 02A8 956 900$: RSB ; Return to caller.
```

```
02A9 958 .SBTTL DUS$DISCONNECT_HIRT - Do HIRT cleanup for a disconnect
02A9 959 :++
02A9 960 :
02A9 961 : DUS$DISCONNECT_HIRT - Do HIRT cleanup for a disconnect
02A9 962 :
02A9 963 : Functional Description:
02A9 964 :
02A9 965 : Scan the HIRT wait queue for CDRPs belonging to this CDDB. Remove
02A9 966 : them and place on the restart queue. This must be done before the RDT
02A9 967 : resource wait is scanned. It is essential to deallocate SCS resources
02A9 968 : held by CDRPs on the HIRT wait queue before scanning any of the SCS
02A9 969 : resource wait queues.
02A9 970 :
02A9 971 : Inputs:
02A9 972 :
02A9 973 : R3 CDDB address
02A9 974 : R4 PDT address
02A9 975 : R5 Permanent CDRP address
02A9 976 :
02A9 977 : Outputs:
02A9 978 :
02A9 979 : R0 through R2 are destroyed.
02A9 980 : All other registers are preserved.
02A9 981 :--
02A9 982 :
02A9 983 DUS$DISCONNECT_HIRT::
02A9 984 :
000E'CF 00 E1 02A9 985 BBC s^#HIRTSV_ACTIVE, - ; See if HIRT has been activated.
30 02AE 986 HIRTSW_STS, 99$ ; If HIRT not active, branch around.
55 DD 02AF 987 PUSHL R5 ; Save a register.
02B1 988 :
50 0000'CF 9E 02B1 989 30$: ASSUME FKB$FQFL EQ 0
02B1 990 MOVAB HIRTSF_RPLQFL, R0 ; Get 'previous' CDRP on wait list.
02B6 991 :
55 60 D0 02B6 992 40$: MOVL FKB$FQFL(R0), R5 ; Get next CDRP on wait list.
21 13 02B9 993 BEQL 90$ ; Branch if no more waiting CDRPs.
04 A5 53 D1 02BB 994 CMPL R3, FKB$FQBL(R5) ; See if waiter has right CDDB.
16 12 02BF 995 BNEQ 60$ ; Branch if wrong CDDB.
60 65 D0 02C1 996 MOVL FKB$FQFL(R5), - ; Let previous point to next.
02C4 997 FKB$FQFL(R0)
02C4 998 BNEQ 50$ ; Branch if current CDRP is not last.
0004'CF 05 12 02C4 998 BNEQ 50$ ; Branch if current CDRP is not last.
50 50 D0 02C6 999 MOVL R0, HIRTSF_RPLQTP ; Else, previous is new end.
BC A5 D0 02CB 1000 50$: MOVL CDRPSF_UCB(R5), R0 ; Get UCB of interest.
56 A0 B7 02CF 1001 DECW UCBSF_RWAITCNT(R0) ; Decrement count incremented during
02D2 1002 ; attempt to allocate HIRT.
FD2B' 30 02D2 1003 BSBW DUTUS$INSERT_RESTARTQ ; Insert this CDRP in restart queue.
DA 11 02D5 1004 BRB 30$ ; Branch back to re-scan entire
02D7 1005 ; HIRT wait queue.
02D7 1006 60$: ; Setup to move to next waiting CDRP.
50 55 D0 02D7 1007 MOVL R5, R0 ; Current becomes previous.
DA 11 02DA 1008 BRB 40$ ; Loop back.
02DC 1009 :
55 8ED0 02DC 1010 90$: POPL R5 ; Restore saved register.
05 02DF 1011 99$: RSB ; Return to caller.
```

```
02E0 1013 .SBTTL DUSRSTRTO_HIRT_CDRP - Do connection failed cleanup of HIRT CDRP
02E0 1014 :++
02E0 1015 :
02E0 1016 : DUSRSTRTO_HIRT_CDRP - Do connection failed cleanup of HIRT CDRP
02E0 1017 :
02E0 1018 : Functional Description:
02E0 1019 :
02E0 1020 : This routine is called by DUTUSINSERT_RESTARTQ when it is discovered
02E0 1021 : that the CDRP destined for the restart queue is the HIRT permanent
02E0 1022 : CDRP. This action is taken instead of placing the HIRT permanent CDRP
02E0 1023 : on the restart queue.
02E0 1024 :
02E0 1025 : The CDRP owning the HIRT is located and processed with a recursive
02E0 1026 : call to DUTUSINSERT_RESTARTQ. Any mapping resources owned by the HIRT
02E0 1027 : permanent CDRP are copied to one of the CDDB permanent CDRPs. This
02E0 1028 : allows the resources to be deallocated sometime after the connection
02E0 1029 : is DISCONNECTED. This prevents "insane" servers for incorrectly
02E0 1030 : overwriting memory due to reallocation of mapping resources. Finally,
02E0 1031 : the HIRT is unlocked, thus making it available for some other
02E0 1032 : replacement operation.
02E0 1033 :
02E0 1034 : Inputs:
02E0 1035 :
02E0 1036 : R3 CDDB address
02E0 1037 : R4 PDT address
02E0 1038 : R5 HIRT permanent CDRP address
02E0 1039 :
02E0 1040 : Outputs:
02E0 1041 :
02E0 1042 : R0 is destroyed.
02E0 1043 : All other registers are preserved.
02E0 1044 :--
02E0 1045 :
02E0 1046 DUSRSTRTO_HIRT_CDRP::
02E0 1047 :
02E0 1048 PUSH R5 ; Save permanent replacement CDRP addr.
02E2 1049 MOVL HIRT$L_SAVDCDRP, R5 ; Get HIRT owner CDRP address.
02E7 1050 BEQL 10$ ; Branch if HIRT owner was canceled.
02E9 1051 BBS #CDRPSV_PERM, - ; Branch if HIRT owner was a CDDB
02EE 1052 CDRPS$L_DUTUFLAGS(R5), - ; permanent CDRP.
02EE 1053 10$
02EE 1054 BSBW DUTUSINSERT_RESTARTQ ; Insert HIRT owner on restart queue.
02F1 1055 10$: POPL R5 ; Restore HIRT permanent CDRP addr.
02F4 1056 :
02F4 1057 TSTL CDRPS$L_LBUFH_AD(R5) ; Were mapping resources allocated?
02F7 1058 BEQL 20$ ; Branch if no mapping res. allocated.
02F9 1059 CLRL CDRPS$L_LBUFH_AD(R5) ; Prevent duplicate deallocations.
02FC 1060 MOVAB CDDBSA_PRCMDRP(R3), R0 ; Get CDDB permanent CDRP address.
0301 1061 MOVAB CDRPS$L_LBUFHNDL(R0), - ; Put address of Local Buffer Handle
0306 1062 CDRPS$L_LBUFH_AD(R0) ; field into field that points to it.
0306 1063 ASSUME CDRPS$L_UBARSRC EQ CDRPS$L_LBUFHNDL+12
0306 1064 MOVQ CDRPS$L_LBUFHNDL(R5), - ; Copy contents of buffer handle to
0308 1065 CDRPS$L_LBUFHNDL(R0) ; CDDB permanent CDRP. Also copy
0308 1066 MOVQ CDRPS$L_LBUFHNDL+8(R5), - ; CDRPS$L_UBARSRC in case this is
0310 1067 CDRPS$L_LBUFHNDL+8(R0) ; a UNIBOS controller.
0310 1068 :
0310 1069 20$: PUSH R3 ; Save CDDB address.
```

DUHIRT
V04-000

H 14
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
DUSRSTRTQ_HIRT_CDRP - Do connection fail 5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 26
(14)

DU
VO

53	BC	A5	D0	0312	1070	MOVL	CDRPSL UCB(R5), R3	; Setup UCB for unlocking HIRT.
		FE5F	30	0316	1071	BSBW	DUSUNLOCK_HIRT	; Release HIRT.
		53	8ED0	0319	1072	POPL	R3	; Restore CDOB address.
				031C	1073			
			05	031C	1074	RSB		; Return

```
031D 1076 .SBTTL DUSREPLACE_LBN - Replace a failing block
031D 1077 :++
031D 1078 :
031D 1079 : DUSREPLACE_LBN - Replace a failing block
031D 1080 :
031D 1081 : Functional Description:
031D 1082 :
031D 1083 : Perform dynamic bad block replacement. At the time of invocation,
031D 1084 : the HIRT is already owned by the caller.
031D 1085 :
031D 1086 : Also entry to this routine made be made by branching (from subroutine
031D 1087 : ONLINE_COMPLETE) to labels STEP7 and STEP11.
031D 1088 :
031D 1089 : Inputs:
031D 1090 :
031D 1091 : R3 UCB address
031D 1092 : R5 HIRT permanent CDRP address
031D 1093 :
031D 1094 : Implicit Inputs:
031D 1095 :
031D 1096 : CDRP$L_RSPID(R5) user RSPID
031D 1097 : HIRT$L_SAVDCDRP original user CDRP address
031D 1098 :
031D 1099 : HIRT owned by caller which implies HIRT SUBSTACK is operative
031D 1100 :
031D 1101 : Outputs:
031D 1102 :
031D 1103 : R0 Replacement status
031D 1104 : R1 Setting for CDRP$V_ERLIP
031D 1105 : R3 UCB address (unchanged)
031D 1106 :
031D 1107 : R2, R4, R5 destroyed.
031D 1108 : All other registers preserved.
031D 1109 :--
031D 1110 :
031D 1111 :
031D 1112 : DUSREPLACE_LBN::
031D 1113 :
031D 1114 : HIRT_SUBSAVE ; Save callers return point on SUBSTACK.
50 001C'CF DO 0327 1115 : MOVL HIRT$L_SAVDCDRP,R0 ; R0 => original CDRP.
50 1C A0 DO 032C 1116 : MOVL CDRP$L_MSG_BUF(R0),R0 ; R0 => END PACKET.
1C A0 DO 0330 1117 : MOVL MSCP$L_FRST_BAD(R0),- ; Indicate which LBN we are
0018'CF 0333 1118 : HIRT$L_LBN ; fixing on this unit.
000A'CF 01 BO 0336 1119 : MOVW #SS$ NORMAL, - ; Initialize worst case I/O status.
0338 1120 : HIRT$L_IOWORST
0338 1121 :
0338 1122 : ; Invalidate contents of incore scratch pages.
0338 1123 :
0338 1124 : ASSUME HIRT$L_PG0CNTNT+2 EQ HIRT$L_PG1CNTNT
004C'CF 01 CE 0338 1125 : MNEGL #1,HIRT$L_PG0CNTNT ; Invalidate pages 0 and 1.
0340 1126 :
0340 1127 : ASSUME HIRT$L_PG2CNTNT+2 EQ HIRT$L_PG3CNTNT
0050'CF 01 CE 0340 1128 : MNEGL #1,HIRT$L_PG2CNTNT ; Invalidate pages 2 and 3.
0345 1129 :
0345 1130 :
0345 1131 : ALLOC_MSG_BUF ; Allocate a send credit.
03 50 E8 0348 1132 : BLBS R0,10$ ; Branch around if successful alloc.
```



```
04BD 31 034B 1133 BRW REPLACE_CONNECT_FAILURE ; If we had an allocation failure branch,
034E 1134 10$:
034E 1135
034E 1136 ; STEP 4 of replacement algorithm.
034E 1137 ; We clear (zero) a sector sized buffer, then read the current
034E 1138 ; contents of the bad block into the buffer. The buffer is cleared first
034E 1139 ; for the rare case when no data can be transferred (such as a valid sync
034E 1140 ; pattern is not detected). The read is performed with error recovery and
034E 1141 ; error correction enabled. In addition to saving the data, we
034E 1142 ; remember whether or not the read succeeded. The saved data is
034E 1143 ; considered valid if the read succeeded, or invalid if it did not.
034E 1144
034E 1145
034E 1146 ; First we zero out the buffer to receive the contents of the failing block.
034E 1147
034E 1148 PUSHR #^M<R2,R3,R4,R5> ; Save registers.
0350 1149 MOVCS #0, #0, #512, - ; Clear page 1 prior to read.
0358 1150 @HIRT$L_PAGE1PTR
035B 1151 POPR #^M<R2,R3,R4,R5> ; Restore registers.
035D 1152
035D 1153 ; Step 4 continued. Prepare the CDRP with SVAPTE, BOFF, and BCNT of page 1
035D 1154 ; (receiving field for upcoming read) so as to facilitate mapping of
035D 1155 ; this region.
035D 1156
035D 1157 MOVL #1,R0 ; Pass page to map to subroutine.
0360 1158 BSBW MAP_PAGE ; Map page 1.
0363 1159
0363 1160 ; Step 4 continued. Prepare the MSCP packet to read failing block into page 1.
0363 1161
0363 1162 BSBW FILL_RCT_PACKET ; Subroutine that fills most fields in
0366 1163 ; MSCP packet. Returns R2=>MSCP packet.
0366 1164 MOVB #MSCP$K_OP_READ, - ; Copy the READ opcode, field not filled
0368 1165 MSCP$B_OPCODE(R2) ; by above subroutine.
036A 1166 MOVL HIRT$L_LBN, - ; And also the LBN of the bad block.
036E 1167 MSCP$L_LBN(R2)
0370 1168
0370 1169 SEND_MSCP_MSG ; Send message to the MSCP server.
0373 1170
0373 1171 BBC #MSCP$V_EF_ERLOG, - ; Test for error log message generated
0375 1172 MSCP$B_FLAGS(R2), 15$ ; and branch around if not.
0378 1173 BISW #HIRT$M_ERLOGIP, - ; Else remember that error log messages
037C 1174 HIRT$W_STS ; Have been generated.
037F 1175 15$:
037F 1176
037F 1177 UNMAP ; Release mapping resources.
0382 1178 CLRL CDRP$L_LBUFH_AD(R5) ; Show no mapping resources allocated.
0385 1179 MOVL CDRP$L_MSG_BOFF(R5), R2 ; Refresh R2 after unmap.
0389 1180
0389 1181 ; Remember status of read of failing block so as to be able to write it later
0389 1182 ; with or without the forced error flag.
0389 1183
0389 1184 BICW s^#HIRT$M_FE, HIRT$W_STS ; Initialize bit.
038E 1185 IF MSCP SUCCESS, Then=20$ ; Branch if READ succeeded.
0394 1186 BISW s^#HIRT$M_FE, HIRT$W_STS ; Set bit if read failed.
0399 1187 20$:
0399 1188
0399 1189 ; Step 5.
```

0200 8F 00 FE AF 3C BB 034E 1148
0028'DF 2C 0350 1149
3C BA 035B 1151
50 01 D0 035D 1157
0748 30 0360 1158
0716 30 0363 1159
21 90 0366 1163
08 A2 0368 1165
0018'CF D0 036A 1166
1C A2 036E 1167
05 E1 0373 1170
07 09 A2 0375 1172
0080 8F A8 0378 1173
000E'CF 037C 1174
2C A5 D4 0382 1178
52 1C A5 D0 0385 1179
000E'CF 04 AA 0389 1184
000E'CF 04 A8 038E 1185
0394 1186
0399 1187
0399 1188
0399 1189

```
0399 1190 : Record the data obtained when the bad block was read during step 4 in
0399 1191 : sector 1 of each RCT copy. If the data cannot be successfully recorded
0399 1192 : in the RCT, report the error to the error log and go to step 18. Note
0399 1193 : that the Multi-Write algorithm used to record the data in sector 1 uses
0399 1194 : write-compare operations to guarantee that the data is successfully
0399 1195 : recorded.
0399 1196 :
0399 1197 :
0399 1198 STEP5:
0399 1199 :
0399 1200 : Write contents of page 1 to sector 1 of each RCT copy.
0399 1201 :
0399 1202      MOVL    #1,R0                ; Pass sector and page number to routine,
004E'CF 50 01 D0 039C 1203      MOVW    R0,HIRT$W_PG1CNTNT        ; Indicate that page 1 contains RCT sector
                                ; #1.
                                0565 30 03A1 1204      BSBW    WRITE_RCT_BLOCK        ; Call internal subroutine to write.
                                OB 50 E8 03A4 1205      BLBS    R0,STEP6            ; LBS implies successful write to at
                                03A7 1206      HIR_ERROR -                ; Signal HIR error.
                                03A7 1207      step=5, type=WRITE
                                0434 31 03AF 1208      BRW     STEP18                ; And branch to step 18.
                                03B2 1209
                                03B2 1210 : Step 6. Record bad block's LBN, whether or not the saved data is valid and
                                03B2 1211 : the fact that we are now in phase 1 of replacement in sector 0 of each
                                03B2 1212 : RCT copy. This means that we read sector 0 modify it and
                                03B2 1213 : then rewrite the updated sector to each RCT copy. If we cannot
                                03B2 1214 : read any sector 0 successfully, we go to step 18. If we cannot
                                03B2 1215 : successfully write at least one sector 0, we go to step 17.
                                03B2 1216 :
                                03B2 1217 :
                                03B2 1218 STEP6:
                                50 7C 03B2 1219      CLRQ    R0                ; Prepare to read sector #0 into page #0.
                                03B4 1220      ; We pass the page number in R0 and
                                03B4 1221      ; the sector number in R1.
                                03B4 1222 :
                                060C 30 03B4 1223      BSBW    READ_RCT_BLOCK        ; Call to read RCT block.
                                OB 50 E8 03B7 1224      BLBS    R0,10$            ; LBS implies successful read.
                                03BA 1225      HIR_ERROR -                ; Signal HIR error.
                                03BA 1226      step=6, type=READ
                                0421 31 03C2 1227      BRW     STEP18                ; On failure goto step 18.
                                03C5 1228 10$:
                                50 0024'CF D0 03C5 1229      MOVL    HIRT$W_PAGE0PTR,R0        ; R0 => sector 0 in memory.
                                0018'CF D0 03CA 1230      MOVL    HIRT$W_LBN,-        ; Copy bad block's LBN to RCT sector
                                OC A0 03CE 1231      RCT$W_LBN(R0)        ; 0 copy in memory.
                                03D0 1232 :
                                08 A0 8000 8F A8 03D0 1233      BISW    #RCT$W_RP1,RCT$W_FLAGS(R0); Set bit to signal phase 1.
                                AA 03D6 1234      BICW    #RCT$W_RP2,-        ; Clear bit to signal not phase 2
                                03D7 1235      !RCT$W_BR-        ; clear bad RBN flag,
                                03D7 1236      !RCT$W_FE,-        ; and also clear force error before
                                08 A0 6080 8F 03D7 1237      RCT$W_FLAGS(R0)        ; testing for valid data.
                                03DC 1238 :
                                000E'CF 02 E1 03DC 1239      BBC     s^#HIRT$W_FE,-        ; See if original data is valid.
                                06 03E1 1240      HIRT$W_STS, 20$
                                08 A0 0080 8F A8 03E2 1241      BISW    #RCT$W_FE,-        ; Set force error if appropriate.
                                03E8 1242      RCT$W_FLAGS(R0)
                                03E8 1243 20$:
                                50 D4 03E8 1244      CLRL    R0                ; Rewrite page 0.
                                051C 30 03EA 1245      BSBW    WRITE_RCT_BLOCK        ; Go to rewrite sector 0.
                                OB 50 E8 03ED 1246      BLBS    R0,STEP7            ; LBS implies successful rewrite.
```



```

03CA 31 03F0 1247 HIR_ERROR - ; Signal HIR error.
03F0 1248 step=6, type=WRITE
03F8 1249 BRW STEP17 ; And go to step 17.
03FB 1250 ; STEP7.
03FB 1251 ; Write and read test patterns on the suspected bad block to determine
03FB 1252 ; whether or not it is in fact a bad block.
03FB 1253 ; Go to step 9 if the test patterns fail.
03FB 1254 ; indicating that the block is indeed bad. Continue with step 8 if the
03FB 1255 ; test patterns succeed, indicating that the block may be good. The test
03FB 1256 ; patterns fail if either the block is again reported as a bad block or if
03FB 1257 ; the test patterns cannot be written and read back correctly.
03FB 1258 ;
03FB 1259 ;
03FB 1260 STEP7:
51 52 002C'CF D0 03FB 1261 MOVL HIR$SL_PAGE2PTR,R2 ; R2 => target page.
B6DBCB6D 8F D0 0400 1262 MOVL #TEST_PATTERN,R1 ; Get test pattern to write to bad block.
50 80 8F 9A 0407 1263 MOVZBL #512/4,R0 ; Loop counter set to # longwords in block.
82 51 D0 040B 1264 10$: MOVL R1,(R2)+ ; Copy test pattern to page 2.
FA 50 F5 040E 1265 SOBGTR R0,10$ ; Loop thru page.
50 02 D0 0411 1266 MOVL #2,R0 ; Pass page to map to subroutine.
0694 30 0414 1267 BSBW MAP_PAGE ; Call to map page 2.
0417 1270 ; Call subroutine that recycles current END PACKET, recycles current RSPID and
0417 1271 ; fills in most relevant data in the MSCP packet.
0417 1272 ;
063B 30 0417 1273 BSBW BUILD_RCT_PACKET ; Build a packet to transfer mapped
041A 1274 ; page to random LBN.
041A 1275
041A 1276 ASSUME MSCPSW_MODIFIER EQ MSCPSB_OPCODE+2
08 A2 43000022 8F D0 041A 1277 MOVL #MSCPSB_OP_WRITE- ; Fill in field not prepared by
0422 1278 !<<MSCPSM_MD_COMP - ; BUILD_RCT_PACKET.
0422 1279 !MSCPSM_MD_SECOR -
0422 1280 !MSCPSM_MD_SEREC> @ 16>, -
0422 1281 MSCPSB_OPCODE(R2)
0422 1282
0422 1283 MOVL HIR$SL_LBN,- ; Fill in field filled in incorrectly
0422 1284 MSCPSL_LBN(R2) ; by BUILD_RCT_PACKET.
0426 1285 SEND_MSCP_MSG ; Send message to the MSCP server.
0428 1286
042B 1287 BBC #MSCPSV_EF_ERLOG,- ; Test for error log message generated
07 09 A2 042D 1288 MSCPSB_FLAGS(R2),15$ ; and branch around if not.
0080 8F A8 0430 1289 BISW #HIR$M_ERLOGIP,- ; Else remember that error log messages
000E'CF 0434 1290 HIR$W_STS ; Have been generated.
0437 1291 15$: UNMAP ; If write no good, give up resources.
0437 1292 CLKL CDRPSL_LBUFH_AD(R5) ; And show that deallocation was done.
52 2C A5 D4 043A 1293 MOVL CDRPSL_MSG_BUF(R5),R2 ; Refresh R2 => END PACKET after unmap.
1C A5 D0 043D 1294 IF_MSCP SUCCESS, then=30$ ; Branch if WRITE successful.
0441 1295
0447 1296 20$: BRW STEP9 ; Proceed to next step of replacement.
0447 1297
00B4 31 0447 1298 30$: BBS #MSCPSV_EF_BBLKR,- ; If bad block reported again on write,
044A 1299 MSCPSB_FLAGS(R2),20$ ; then branch back to proceed with
044C 1300 ; replacement.
F8 09 A2 E0 044C 1301
044F 1302
044F 1303 40$:

```

```
0200 8F 00 FE AF 3C BB 044F 1304
002C'DF 2C 044F 1305 ; Clear page to receive test pattern written block.
3C BA 044F 1306
0451 1307 PUSHF #^M<R2,R3,R4,R5> ; Save registers.
0459 1308 MOVCS #0, #0, #512, - ; Clear page 2 prior to read.
045C 1309 @HIRTSL, PAGE2PTR
045E 1310 POPR #^M<R2,R3,R4,R5> ; Restore registers.
045E 1311
045E 1312 ; Call subroutine that recycles current END PACKET, recycles current RSPID and
045E 1313 ; fills in most relevant data in the MSCP packet.
045E 1314
045E 1315 MOVL #2, R0 ; Pass page to map to subroutine.
0647 30 0461 1316 BSBW MAP_PAGE ; Call to map page 2.
05EE 30 0464 1317 BSBW BUILD_RCT_PACKET ; Build MSCP packet to transfer page 2
0467 1318
0467 1319 ASSUME MSCPSW MODIFIER EQ MSCPSB_OPCODE+2
08 A2 43000021 8F D0 0467 1320 MOVL #MSCPSB_OP_READ- ; Fill in field not prepared by
046F 1321 !<<MSCPSM_MD_COMP - ; BUILD_RCT_PACKET.
046F 1322 !MSCPSM_MD_SECOR -
046F 1323 !MSCPSM_MD_SEREC> @ 16>, -
046F 1324 MSCPSB_OPCODE(R2)
046F 1325
0018'CF D0 046F 1326 MOVL HIRTSL_LBN, - ; Fill in field filled in incorrectly
1C A2 0473 1327 MSCPSL_LBN(R2) ; by BUILD_RCT_PACKET.
0475 1328 SEND_MSCP_MSG ; Send message to the MSCP server.
0478 1329
07 09 A2 05 E1 0478 1330 BBC #MSCPSV_EF_ERLOG, - ; Test for error log message generated
0080 8F A8 047A 1331 MSCPSB_FLAGS(R2), 45$ ; and branch around if not.
000E'CF 047D 1332 BLSW #HIRTSM_ERLOGIP, - ; Else remember that error log messages
0481 1333 HIRTSM_STS ; Have been generated.
0484 1334 45$:
0484 1335 UNMAP ; Give up MAP resources.
52 2C A5 D4 0487 1336 CLRL CDRPSL_LBUFH_AD(R5) ; And show that deallocation was done.
1C A5 D0 048A 1337 MOVL CDRPSL_MSG_BUF(R5), R2 ; Refresh R2 => END PACKET after unmap.
048E 1338
048E 1339 IF_MSCP SUCCESS, then=60$ ; Branch if WRITE successful.
0494 1340 50$:
0067 31 0494 1341 BRW STEP9 ; On any error, goto step 9.
0497 1342 60$:
07 E0 0497 1343 BBS #MSCPSV_EF_BBLKR, - ; If bad block reported on read,
F8 09 A2 0499 1344 MSCPSB_FLAGS(R2), 50$ ; then branch back to proceed with
049C 1345 ; replacement of same.
51 52 002C'CF D0 049C 1346 MOVL HIRTSL_PAGE2PTR, R2 ; R2 => target page.
B6DBC6D 8F D0 04A1 1347 MOVL #TEST_PATTERN, R1 ; Test pattern to compare to bad block.
50 80 8F 9A 04A8 1348 MOVZBL #512/4, R0 ; Loop counter set to # longwords in block.
04AC 1349 70$:
82 51 D1 04AC 1350 CMPL R1, (R2)+ ; Compare test pattern to page 2.
E3 12 04AF 1351 BNEQ 50$ ; On any discrepancy, branch.
F8 50 F5 04B1 1352 SOBGTR R0, 70$ ; Loop thru page.
04B4 1353
04B4 1354 ; STEP8.
04B4 1355 ; We write the saved data back out to the bad block using a
04B4 1356 ; write-compare operation. The write-compare is performed with the "force
04B4 1357 ; error" modifier if and only if the saved data is invalid. Go to step 13
04B4 1358 ; if the write-compare both succeeds AND the block is no longer reported as
04B4 1359 ; a bad block -- the original problem was a transient. The write-compare
04B4 1360 ; succeeds if no error is detected and the saved data is valid or if only a
```

```
04B4 1361 : forced error is detected and the saved data is invalid.
04B4 1362 :
04B4 1363 :
04B4 1364 STEP8:
04B4 1365 :
04B4 1366 : Try to write original data out to block originally reported as bad since
04B4 1367 : error now appears to have been transient.
04B4 1368 :
50 01 D0 04B4 1369 MOVL #1,R0 ; Data from bad block is in page 1.
05F1 30 04B7 1370 BSBW MAP_PAGE ; Map page 1.
0598 30 04BA 1371 BSBW BUILD_RCT_PACKET ; Recycle etc., and fill in packet.
04BD 1372 :
04BD 1373 ASSUME MSCPSW_MODIFIER EQ MSCPSB_OPCODE+2
D0 04BD 1374 MOVL #MSCPSW_OP_WRITE- ; Fill in field not prepared by
04BE 1375 !<MSCPSW_MD_COMP@16>,- ; BUILD_RCT_PACKET.
04BE 1376 MSCPSB_OPCODE(R2)
04C5 1377 :
04C5 1378 BBC s^#HIRT$V_FE,- ; See if original data is valid.
04CA 1379 HIRT$W_STS,10$
0A A2 1000 8F A8 04CB 1380 BISW #MSCPSW_MD_ERROR,- ; Set force error modifier if
04D1 1381 MSCPSW_MODIFIER(R2) ; original data is invalid.
04D1 1382 10$:
04D1 1383 MOVL HIRT$W_LBN,- ; Fill in field filled in incorrectly
04D5 1384 MSCPSW_LBN(R2) ; by BUILD_RCT_PACKET.
04D7 1385 SEND_MSCP_MSG ; Send message to the MSCP server.
04DA 1386 :
04DA 1387 BBC #MSCPSW_EF_ERLOG,- ; Test for error log message generated
04DC 1388 MSCPSB_FLAGS(R2),15$ ; and branch around if not.
0080 8F A8 04DF 1389 BISW #HIRT$W_ERLOGIP,- ; Else remember that error log messages
000E'CF 04E3 1390 HIRT$W_STS ; Have been generated.
04E6 1391 15$:
04E6 1392 UNMAP ; If write no good, give up resources.
04E9 1393 CLRL CDRPSL_LBUFH_AD(R5) ; And show that deallocation was done.
52 2C A5 D4 04EC 1394 MOVL CDRPSL_MSG_BUF(R5),R2 ; Refresh R2 => END PACKET after unmap.
04F0 1395 :
04F0 1396 IF MSCP FAILURE, then=STEP9 ; Branch if problem not transient.
04F6 1397 BBS #MSCPSW_EF_BBLKR,- ; If bad block reported on write,
04F8 1398 MSCPSB_FLAGS(R2),STEP9 ; then branch ahead to proceed with
04FB 1399 ; replacement of same.
01B2 31 04FB 1400 BRW STEP13 ; Branch if error was transient.
04FE 1401 :
04FE 1402 : STEP9.
04FE 1403 : We scan the RCT and determine what new RBN the bad block
04FE 1404 : should be replaced with, whether or not the the bad block has been
04FE 1405 : previously replaced, and (if it has previously been replaced) the bad
04FE 1406 : block's old RBN. The RCT is NOT updated at this time. If the RCT scan
04FE 1407 : fails, we report the error to the error log and go to step 16.
04FE 1408 :
04FE 1409 :
04FE 1410 :
04FE 1411 STEP9:
05DA 30 04FE 1412 BSBW SEARCH_RCT ; Routine to search the RCT for an RBN.
22 50 E8 0501 1413 BLBS R0,STEP10 ; LBC implies success, so goto step 10.
000E'CF 06 E0 0504 1414 BBS s^#HIRT$V_RCTFULL,- ; Check for RCTFULL error and
0A 0509 1415 HIRT$W_STS,910$ ; branch if that is the problem.
050A 1416 HIR_ERROR - ; Else, signal HIR READ error.
050A 1417 step=9, type=READ
```

```
OF 11 0512 1418 BRB 980$ ; Join common branch to step 16.
000A'CF 216C 8F B0 0514 1419 910$: HIR_ERROR -
0514 1420 ; Signal HIR RCTFULL error.
051C 1421 MOVW #SS$ BADRCT, - ; Also, supply a worst case error
0523 1422 HIRSW_IOWORST ; status.
025D 31 0523 1423 980$: BRW STEP16 ; Go to step 16 after any failure.
0526 1424
0526 1425 ; STEP10.
0526 1426 ; Record the new RBN, whether or not the bad block has been previously
0526 1427 ; replaced, the bad block's old RBN (if it has been previously replaced),
0526 1428 ; and the fact that we are in phase 2 of bad block replacement in sector 0
0526 1429 ; of each RCT copy. The RCT must be updated without reading sector 0,
0526 1430 ; instead using the copy of sector 0 last read from or written to the RCT.
0526 1431 ; If the RCT cannot be updated, report the error to the error log and go to
0526 1432 ; step 16.
0526 1433
0526 1434
0526 1435
0526 1436 STEP10:
50 0024'CF D0 0526 1437 MOVL HIRSL_PAGEPTR,R0 ; R0 => Page 0.
0058'CF D0 052B 1438 MOVL HIRSL_RBN,- ; Update date to sector 0 copy in
10 A0 052F 1439 RCTSL_RBN(R0) ; in memory.
000E'CF 03 E1 0531 1440 BBC s^#HIRSV_MATCH,- ; See if we had a failing RBN, and
0C 0536 1441 HIRSW_STS,10$ ; if NOT, branch around.
2000 8F A8 0537 1442 BISW #RCTSM_BR,- ; Indicate failing RBN in sector 0
08 A0 053B 1443 RCTSW_FLAGS(R0) ; flags word.
005C'CF D0 053D 1444 MOVL HIRSC_MATCHRBN,- ; And also indicate the failing RBN.
14 A0 0541 1445 RCTSL_BAD_RBN(R0)
0543 1446 10$:
8000 8F AA 0543 1447 BICW #RCTSM_RP1,- ; Show that we are leaving phase 1
08 A0 0547 1448 RCTSW_FLAGS(R0) ; of replacement processing.
4000 8F A8 0549 1449 BISW #RCTSM_RP2,- ; And entering phase 2.
08 A0 054D 1450 RCTSW_FLAGS(R0)
054F 1451
50 D4 054F 1452 CLRL R0 ; Rewrite page 0.
03B5 30 0551 1453 BSBW WRITE_RCT_BLOCK ; Go write the sector.
0B 50 E8 0554 1454 BLBS R0,STEP11 ; If success, go to next step.
0557 1455 HIR_ERROR - ; Signal HIR error.
0557 1456 step=10, type=WRITE
0221 31 055F 1457 BRW STEP16 ; Branch on failure.
0562 1458
0562 1459 ; STEP11.
0562 1460 ; We update the RCT to indicate that the bad block has been
0562 1461 ; replaced with the new RBN, and that the old RBN (if any) is unusable. If
0562 1462 ; this requires updating two blocks in the RCT, then both blocks must be
0562 1463 ; read before either is written. If a block cannot be read successfully,
0562 1464 ; report the error to the error log and go to step 16. If a block cannot
0562 1465 ; be written successfully, report the error to the error log and go to step
0562 1466 ; 15.
0562 1467
0562 1468
0562 1469
0562 1470 STEP11:
50 0058'CF 07 00 EF 0562 1471 EXTZV #0,#7,HIRSL_RBN,R0 ; R0 = offset in sector of RBN descriptor.
51 002C'CF D0 0569 1472 MOVL HIRSL_PAGEPTR,R1 ; R1 => sector containing RBN descriptor.
50 6140 DE 056E 1473 MOVAL (R1)[R0],R0 ; R0 => RBN descriptor.
0572 1474
```

```
20000000 8F C9 0572 1475 BISL3 #RCTSM_ALLOCATED,- ; Put LBN being replaced into descriptor.
60 0018'CF 0578 1476 HIRTSL_LBN,(R0) ; and or in ALLOCATED bit.
000E'CF 04 E1 057C 1477 BBC s^#HIRT$V_EMPTYTYPE, - ; Branch if primary RBN allocation.
07 C581 1478 HIRT$W_STS,10$
60 10000000 8F C8 0582 1479 BISL #RCTSM_NONPRIME,(R0) ; Indicate non prime allocation.
0589 1480 10$:
000E'CF 03 E0 0589 1481 BBS s^#HIRT$V_MATCH, - ; Branch if RCT search showed RBN failed.
03 058E 1482 HIRT$W_STS,20$
0063 31 058F 1483 BRW 60$ ; If NOT RBN failure, skip RBN
0592 1484 ; descriptor update.
0592 1485 20$:
51 005C'CF F9 8F 78 0592 1486 ASHL #-7,HIRTSL_MATCHRBN,R1 ; R1 = relative RCT block containing
0599 1487 ; bad RBN descriptor.
51 02 C0 0599 1488 ADDL #2,R1 ; Add in sectors 0 and 1.
0050'CF 51 B1 059C 1489 CMPW R1,HIRT$W_PG2CNTNT ; Page 2 contains RBN descriptor of
05A1 1490 ; allocatable RBN and maybe also
05A1 1491 ; descriptor of bad RBN.
1B 13 05A1 1492 BEQL 40$ ; EQL implies both descriptors in same
05A3 1493 ; block.
50 03 D0 05A3 1494 MOVL #3,R0 ; Indicate that we want to read into
05A6 1495 ; page 3.
041A 30 05A6 1496 BSBW READ_RCT_BLOCK ; Read sector (R1) into page 3.
0B 50 E8 05A9 1497 BLBS R0,30$ ; If success, continue.
05AC 1498 HIR_ERROR - ; Signal HIR error.
05AC 1499 step=11, type=READ
01CC 31 05B4 1500 BRW STEP16 ; Branch on failure.
05B7 1501
05B7 1502 30$:
52 0030'CF D0 05B7 1503 MOVL HIRTSL_PAGE3PTR,R2 ; R2 => page with bad RBN descriptor.
05 11 05BC 1504 BRB 50$ ; Branch around.
05BE 1505 40$:
52 002C'CF D0 05BE 1506 MOVL HIRTSL_PAGE2PTR,R2 ; R2 => page with bad RBN descriptor.
05C3 1507 50$:
50 005C'CF 07 00 EF 05C3 1508 EXTZV #0,#7,HIRTSL_MATCHRBN,R0 ; R0 = offset of bad RBN descriptor.
50 6240 DE 05CA 1509 MOVAL (R2)[R0],R0 ; R0 => bad RBN descriptor.
0060'CF 60 D0 05CE 1510 MOVL (R0),HIRTSL_BADRBND ; Save Bad RBN descriptor in case
05D3 1511 ; we have to restore due to failure.
60 40000000 8F D0 05D3 1512 MOVL #RCTSM_UNUSABLE,(R0) ; Clear LBN and mark unusable bit in
05DA 1513 ; descriptor.
002C'CF 52 D1 05DA 1514 CMPL R2,HIRTSL_PAGE2PTR ; See if both descriptors in same page.
14 13 05DF 1515 BEQL 60$ ; EQL implies yes. Go do only 1 write.
05E1 1516
50 03 D0 05E1 1517 MOVL #3,R0 ; Rewrite page 3 [R0].
0322 30 05E4 1518 BSBW WRITE_RCT_BLOCK ; Go write.
0B 50 E8 05E7 1519 BLBS R0,60$ ; If success, continue.
05EA 1520 HIR_ERROR - ; Signal HIR error.
05EA 1521 step=11, type=WRITE
00FA 31 05F2 1522 BRW STEP15_A ; Branch on failure.
05F5 1523
05F5 1524 60$:
50 02 D0 05F5 1525 MOVL #2,R0 ; Prepare to write page 2.
030E 30 05F8 1526 BSBW WRITE_RCT_BLOCK ; Go write.
0B 50 E8 05FB 1527 BLBS R0,STEP12 ; If success, goto next step.
05FE 1528 HIR_ERROR - ; Signal HIR error.
05FE 1529 step=11, type=WRITE
010E 31 0606 1530 BRW STEP15_B ; Branch on failure.
0609 1531
```

```
0609 1532 : STEP12.
0609 1533 : We use the REPLACE command to revector the bad block to the
0609 1534 : chosen replacement block, then use the standard WRITE command (addressed
0609 1535 : to the bad block's LBN) with the "compare" modifier asserted to store the
0609 1536 : saved data in the replacement block. The write-compare is performed with
0609 1537 : the "force error" modifier if and only if the saved data is invalid.
0609 1538 : Note that the REPLACE command implicitly verifies that a head or servo
0609 1539 : track failure has not occurred, causing a large number of improper
0609 1540 : replacements. If the REPLACE command fails, go to step 15. If the WRITE
0609 1541 : command fails, go to step 9 to re-scan the RCT for another RBN. Note
0609 1542 : that the current new RBN will become the old RBN for this next pass.
0609 1543 : Either failure will have already been reported to the error log. The
0609 1544 : WRITE command succeeds if no error is detected and the saved data is
0609 1545 : valid or if only a forced error is detected and the saved data is
0609 1546 : invalid.
0609 1547 :
0609 1548 :
0609 1549 STEP12:
0609 1550 RECYCH_MSG_BUF ; Recycle END PACKET into MSCP buffer.
03 50 E8 060C 1551 BLBS RO,5$ ; LBS means allocation success.
01F9 3i 060F 1552 BRW REPLACE_CONNECT_FAILURE ; Allocation failure means CONNECTION
0612 1553 ; failure.
0612 1554 5$:
0612 1555 INIT_MSCP_MSG ucb=(R3) ; Initialize MSCP packet for REPLACE.
0615 1556
0615 1557 ASSUME MSCPSW_MODIFIER EQ MSCPSB_OPCODE+2
DO 0615 1558 MOVL #MSCPSR_OP_REPLACE ; Fill in field not prepared by
0616 1559 !<MSCPSM_MD_EXPRS@16>,- ; BUILD_RCT_PACKET.
0616 1560 MSCPSB_OPCODE(R2)
08 A2 80000014 8F 061D 1561
000E'CF 04 E0 061D 1562 BBS s^#HIRTSV_EMPTYTYPE, - ; See if primary or secondary RBN,
04 ; branch if secondary.
01 AB 0622 1563 BBSW #MSCPSM_MD_PRIMR,- ; Set primary modifier if
0A A2 0623 1564 MSCPSW_MODIFIER(R2) ; called for.
0625 1565 10$:
UC A2 0058'CF DO 0627 1567 MOVL HIRTSI_RBN,MSCPSI_RBN(R2); Fill in special REPLACE field.
0018'CF DO 062D 1568 MOVL HIRTSI_LBN,- ; Fill in field filled in incorrectly
1C A2 0631 1569 MSCPSI_LBN(R2) ; by BUILD_RCT_PACKET.
0633 1570 SEND_MSCP_MSG ; Send message to the MSCP server.
0636 1571
07 09 A2 E1 0636 1572 BBC #MSCPSV_EF_ERLOG,- ; Test for error log message generated
0080 8F AB 0638 1573 MSCPSB_FLAGS(R2),15$ ; and branch around if not.
000E'CF 063B 1574 BBSW #HIRTSV_ERLOGIP,- ; Else remember that error log messages
063F 1575 HIRTSV_STS ; Have been generated.
0642 1576 15$:
0642 1577 IF MSCP SUCCESS, then=20$ ; Branch if REPLACE was successful.
0648 1578 HIR_ERROR - ; Signal HIR error.
0648 1579 step=12, type=REPFAIL
00C4 31 0650 1580 BRW STEP15_B
0653 1581 20$:
50 01 DO 0653 1582 MOVL #1,R0 ; Data from bad block is in page 1.
0452 30 0656 1583 BSBW MAP_PAGE ; Map page 1.
03F9 30 0659 1584 BSBW BUID_RCT_PACKET ; Recycle etc., and fill in packet.
065C 1585
065C 1586 ASSUME MSCPSW_MODIFIER EQ MSCPSB_OPCODE+2
DO 065C 1587 MOVL #MSCPSR_OP_WRITE- ; Fill in field not prepared by
065D 1588 !<MSCPSM_MD_COMP@16>,- ; BUILD_RCT_PACKET.
```



```
08 A2 40000022 8F      065D 1589      MSCPSB_OPCODE(R2)
                        0664 1590
                        000E'CF 02 E1 0664 1591      BBC      s^#HIRT$V FE, -      ; See if original data is valid.
                        06      0669 1592      HIRT$W STS, 30$
0A A2 1000 8F      A8 066A 1593      BISW      #MSCPSM MD_ERROR, -      ; Set force error modifier if
                        0670 1594      MSCPSW_MODIFIER(R2)      ; original data is invalid.
                        0670 1595 30$:      MOVL      HIRT$L_LBN, -      ; Fill in field filled in incorrectly
                        0670 1596      MSCPSL_LBN(R2)      ; by BUILD_RCT_PACKET.
                        0674 1597      SEND_MSCP_MSG      ; Send message to the MSCP server.
                        0676 1598
                        0679 1599
                        07 09 05 E1 0679 1600      BBC      #MSCPSV EF_ERLOG, -      ; Test for error log message generated
                        0080 A2 067B 1601      MSCPSB_FLAGS(R2), 35$      ; and branch around if not.
                        0080 8F A8 067E 1602      BISW      #HIRT$M_ERLOGIP, -      ; Else remember that error log messages
                        000E'CF      0682 1603      HIRT$W_STS      ; Have been generated.
                        0685 1604 35$:      UNMAP      ; If write no good, give up resources.
                        0685 1605      CLRL      CDRPSL_LBUFH_AD(R5)      ; And show that deallocation was done.
                        52 2C A5 D4 0688 1606      MOVL      CDRPSL_MSG_BUF(R5), R2      ; Refresh R2 => END PACKET after unmap.
                        1C A5 D0 068B 1607
                        068F 1608
                        068F 1609      IF_MSCP SUCCESS, then=STEP13      ; If WRITE successful go to step 13.
                        0695 1610
                        0695 1611      ; ASSUME that the force data subcode is zero
                        0A 08 B1 0695 1612      CMPW      #MSCPSK ST_DATA, -      ; See if data error with force error
                        0A A2 0697 1613      MSCPSW_STATUS(R2)      ; subcode.
                        0A 12 0699 1614      BNEQ      50$      ; If T, then branch to take action.
                        069B 1615
                        000E'CF 02 E0 069B 1616      BBS      s^#HIRT$V FE, -      ; To STEP13 if force error expected.
                        0F      06A0 1617      HIRT$W_STS, STEP13
                        06A1 1618      BUG_CHECK      DISKCLASS, FATAL      ; Shouldn't happen.
                        06A5 1619 50$:      HIR_ERROR -      ; Signal HIR error.
                        06A5 1620      step=12, type=WRITE
                        FE4E 31 06AD 1622      BRW      STEP9      ; Following algorithm, goto step 9.
                        0680 1623
                        0680 1624      ; STEP13.
                        0680 1625      ; We update sector 0 of the RCT copies to indicate that we are no
                        0680 1626      ; longer in the middle of replacing a bad block. The RCT must be updated
                        0680 1627      ; without reading sector 0, instead using the copy of sector 0 last read
                        0680 1628      ; from or written to the RCT. If the RCT cannot be updated, report the
                        0680 1629      ; error to the error log and go to step 17.
                        0680 1630
                        0680 1631
                        0680 1632      STEP13:
                        50 0024'CF D0 0680 1633      MOVL      HIRT$L_PAGEPTR, R0      ; R0 => page 0, which contains sector 0.
                        AA      0685 1634      BICW      #RCTSM_RP1-      ; Reset flags in sector zero. We are
                        0686 1635      !RCTSM_RP2-      ; NOT in phase 1 nor in phase 2.
                        0686 1636      !RCTSM_BR-      ; Also we clear Bad RBN flag and
                        0686 1637      !RCTSM_FE, -      ; force error flag as well.
                        08 A0 E080 8F 0686 1638      RCTSW_FLAGS(R0)
                        068B 1639
                        068B 1640      ; ASSUME RCT$L_RBN EQ RCT$L_LBN+4
                        0C A0 7C 068B 1641      CLRQ      RCT$L_LBN(R0)      ; Zero out RBN and LBN.
                        14 A0 D4 068E 1642      CLRL      RCT$L_BAD_RBN(R0)      ; Also clear BADRBN field.
                        06C1 1643
                        50 D4 06C1 1644      CLRL      R0      ; Rewrite page 0.
                        0243 30 06C3 1645      BSBW      WRITE_RCT_BLOCK      ; Go write page into sector.
```

```
OB 50 E8 06C6 1646 BLBS R0,STEP14 ; LBS is success.
          06C9 1647 HIR_ERROR - ; Signal HIR error.
          06C9 1648 ;
00F1 31 06D1 1649 BRW STEP13, type=WRITE ; After failure goto step 17.
          06D4 1650 ;
          06D4 1651 ; STEP14.
          06D4 1652 ; We set the success return code and return to our internal caller.
          06D4 1653 ;
          06D4 1654 ;
          06D4 1655 STEP14:
          06D4 1656 DEALLOC_MSG_BUF
          06D7 1657 CLRL R1 ; Prepare to return ERLIP bit if set.
          06D9 1658 BBCC s^#HIRT$V_ERLOGIP,- ; Branch around if clear and clear if
          06DB 1659 HIRT$W_ST5,10$ ; already set.
          06DF 1660 BISW #CDRPSM_ERLIP,R1 ; Set bit in R1 so as to return to caller.
          06E2 1661 10$: MOVZWL S^#SSS_NORMAL,R0 ; Prepare to return status to caller.
          06E5 1662 HIRT_SUBRETURN ; Return to caller.
          06EF 1663 ;
          06EF 1664 ; STEP15 - If here we failed to update the RCT (STEP11) or we failed in the
          06EF 1665 REPLACE (STEP12).
          06EF 1666 ; We restore the RCT to indicate that the new RBN is unallocated
          06EF 1667 ; and usable and that the bad block is either not replaced or revector to
          06EF 1668 ; the old RBN, whichever was it's original status. The RCT must be updated
          06EF 1669 ; without reading any blocks from it, instead using the copies of the
          06EF 1670 ; relevent blocks that were read from the RCT in step 11. Any errors are
          06EF 1671 ; reported to the error log but otherwise ignored.
          06EF 1672 ;
          06EF 1673 ;
          06EF 1674 ;
          06EF 1675 STEP15_A:
          06EF 1676 ; Here we failed to write the RCT block containing the Bad RBN only when
          06EF 1677 ; this descriptor resided in a different RCT block than that of the
          06EF 1678 ; selected RBN. Because of the way that STEP11 works the copy
          06EF 1679 ; of the RCT Block is in page 3.
          06EF 1680 ;
          06EF 1681 ;
          50 005C'CF 52 0030'CF D0 06EF 1682 MOVL HIRT$L_PAGE3PTR,R2 ; R2 => copy of page 3 in memory.
          07 00 EF 06F4 1683 EXTZV #0,#7,HIRT$L_MATCHRBN,R0 ; R0 = offset (longword) of Bad RBN
          06FB 1684 ; descriptor in page 3.
          50 6240 DE 06FB 1685 MOVAL (R2)[R0],R0 ; R0 => Bad RBN descriptor slot.
          60 0060'CF D0 06FF 1686 MOVL HIRT$L_BADRBN,(R0) ; Restore Bad RBN descriptor.
          50 03 D0 0704 1687 MOVL #3,R0 ; Prepare to try to rewrite page 3.
          01FF 30 0707 1688 BSBW WRITE_RCT_BLOCK ; Try to rewrite.
          08 50 E8 070A 1689 BLBS R0, 90$ ; Branch if WRITE succeeded.
          070D 1690 HIR_ERROR - ; Else, signal HIR error.
          070D 1691 ;
          6C 11 0715 1692 90$: BRB STEP15, type=WRITE ; Always go to step 16.
          0717 1693 ;
          0717 1694 STEP15_B:
          0717 1695 ;
          0717 1696 ; If here we failed in the REPLACE operation or in updating the RCT block
          0717 1697 ; containing the selected RBN descriptor. So we try to restore the
          0717 1698 ; RCT sector(s) that contained the RBN descriptor(s). If there was
          0717 1699 ; no Bad RBN, then we simply want to restore the contents of page 2
          0717 1700 ; to the sector indicated by HIRT$W_PG2CNTNT, after clearing the
          0717 1701 ; RBN descriptor slot associated with the selected RBN.
          0717 1702 ;
```



```
000E'CF 03 E1 0717 1703 BBC s^#HIRT$V MATCH, - ; If NO Bad RBN, branch ahead to
43 071C 1704 HIRT$W_ST5,20$ ; restore only one descriptor.
071D 1705
071D 1706 ; If there was a Bad RBN, and its descriptor happened to reside in the
071D 1707 ; same RCT sector as the descriptor of the selected RBN, then we first
071D 1708 ; first restore the old contents of the Bad RBN descriptor then
071D 1709 ; if this descriptor resided in a different sector than the selected
071D 1710 ; RBN's descriptor, we rewrite this other sector first. Note the
071D 1711 ; other sector is contained in page 3 while the selected
071D 1712 ; RBN's descriptor is always in page 2.
071D 1713
071D 1714
51 52 002C'CF DO 071D 1714 MOVL HIRT$L_PAGE2PTR,R2 ; R2 => copy of page 2.
005C'CF F9 8F 78 0722 1715 ASHL #-7,HIRT$L_MATCHRBN,R1 ; Calculate RCT sector # for Bad RBN.
51 02 CO 0729 1716 ADDL #2,R1 ; Add in RCT sectors 0 and 1.
0050'CF 51 B1 072C 1717 CMPW R1,HIRT$W_PG2CNTNT ; See if in same sector.
05 13 0731 1718 BEQL 10$ ; EQL implies yes.
0733 1719
52 0030'CF DO 0733 1720 MOVL HIRT$L_PAGE3PTR,R2 ; R2 => copy of page 3.
50 005C'CF 07 00 EF 0738 1721 10$: EXTZV #0,#7,HIRT$L_MATCHRBN,R0 ; R0 = offset (longword) of Bad RBN
0738 1722 ; descriptor in page 2 or 3.
073F 1723 ; R0 => Bad RBN descriptor slot.
50 6240 DE 073F 1724 MOVAL (R2)[R0],R0 ; Restore Bad RBN descriptor.
60 0060'CF DO 0743 1725 MOVL HIRT$L_BADRBN,(R0) ; See if we have to do both pages.
002C'CF 52 D1 0748 1726 CMPL R2,HIRT$L_PAGE2PTR ; EQL implies NO, only page 2.
11 13 074D 1727 BEQL 20$ ; Prepare to try to rewrite page 3.
50 03 DO 074F 1728 MOVL #3,R0 ; Try to rewrite.
01B4 30 0752 1729 BSBW WRITE RCT_BLOCK ; Branch if WRITE succeeded.
08 50 E8 0755 1730 B'BS R0, 20$ ; Else, signal HIR error.
0758 1731 HIR_ERROR -
0758 1732 step=15, type=WRITE
0760 1733 20$:
0760 1734 ; Here we clear the selected RBN's descriptor and rewrite the sector from
0760 1735 ; page 2.
0760 1736
0760 1737
50 52 002C'CF DO 0760 1738 MOVL HIRT$L_PAGE2PTR,R2 ; R2 => page 2 in memory.
0058'CF 07 00 EF 0765 1739 EXTZV #0,#7,HIRT$L_RBN,R0 ; R0 = offset (longword) of selected RBN
076C 1740 ; descriptor in page 2.
50 6240 DE 076C 1741 MOVAL (R2)[R0],R0 ; R0 => RBN descriptor slot.
60 D4 0770 1742 CLRL (R0) ; Restore to available RBN descriptor.
0772 1743
50 02 DO 0772 1744 MOVL #2,R0 ; Prepare to try to rewrite page 2.
0191 30 0775 1745 BSBW WRITE RCT_BLOCK ; Try to rewrite.
08 50 E8 0778 1746 BLBS R0, STEP16 ; Branch if WRITE succeeded.
0778 1747 HIR_ERROR - ; Else, signal HIR error.
0778 1748 step=15, type=WRITE
0783 1749 ; ----- BRB STEP16 ; Always continue with step 16.
0783 1750
0783 1751 ; STEP16.
0783 1752 ; We use the standard WRITE command (addressed to the bad block's
0783 1753 ; LBN) to restore the saved data. The write is performed with the "force
0783 1754 ; error" modifier if and only if the saved data is invalid. Any errors are
0783 1755 ; reported to the error log but otherwise ignored.
0783 1756
0783 1757
50 01 DO 0783 1758 STEP16: MOVL #1,R0 ; Prepare to try to write original data
0783 1759
```

```
0322 30 0786 1760 ; to bad block that we could not replace.
02C9 30 0786 1761 ; Map page 1.
0789 1762 ; Build MSCP packet.
078C 1763
078C 1764
22 DO 078C 1765 ASSUME MSCPSW_MODIFIER EQ MSCPSB_OPCODE+2
08 A2 078C 1766 MOVL #MSCPSW_OP_WRITE, - ; Put in write opcode.
000E'CF 02 E1 078E 1766 MSCPSB_OPCODE(R2)
06 0790 1767 BBC s^#HIRT$V_FE, - ; Branch around if NO forced error.
1000 8F AB 0795 1768 HIRT$W_STS,10$
0A A2 0796 1769 BISW #MSCPSW_MD_ERROR, - ; Set forced error bit modifier on.
079A 1770 MSCPSW_MODIFIER(R2)
0018'CF 079C 1771 10$: MOVL HIRT$L_LBN, - ; Indicate LBN to write.
1C A2 DO 079C 1772 MSCPSL_LBN(R2)
07A0 1773 SEND MSCP MSG ; Send message to the MSCP server.
07A2 1774 IF MSCP SUCCESS, then=12$ ; Branch if WRITE succeeded.
07A5 1775 HIR_ERROR - ; Else, signal HIR error.
07AB 1776
07AB 1777 step=16, type=WRITE
07 09 A2 05 E1 07B3 1778 12$: BBC #MSCPSV_EF_ERLOG, - ; Test for error log message generated
07B8 1779 MSCPSB_FLAGS(R2),15$ ; and branch around if not.
0080 8F AB 07B8 1780 BISW #HIRT$M_ERLOGIP, - ; Else remember that error log messages
000E'CF 07BC 1781 HIRT$W_STS ; Have been generated.
07BF 1782 15$:
07BF 1783 UNMAP
2C A5 D4 07C2 1784 CLRL CDRPSL_LBUFH_AD(R5) ; Indicate no mapping resources.
07C5 1785
07C5 1786 ; STEP17.
07C5 1787 ; We update sector 0 of the RCT copies to indicate that it is no
07C5 1788 ; longer in the middle of replacing a bad block. The RCT must be updated
07C5 1789 ; without reading sector 0, instead using the copy of sector 0 last read
07C5 1790 ; from or written to the RCT. Any errors are reported to the error log but
07C5 1791 ; otherwise ignored.
07C5 1792
07C5 1793
07C5 1794
07C5 1795 STEP17:
50 0024'CF DO 07C5 1796 MOVL HIRT$L_PAGEOPTR,R0 ; R0 => page 0, which contains sector 0.
AA 07CA 1797 BICW #RCTSM_RP1- ; Reset flags in sector zero. We are
07CB 1798 !RCTSM_RP2- ; NOT in phase 1 nor in phase 2.
07CB 1799 !RCTSM_BR- ; Also we clear Bad RBN flag and
07CB 1800 !RCTSM_FE, - ; force error flag as well.
07CB 1801 RCTSW_FLAGS(R0)
08 A0 E080 8F 07D0 1802
07D0 1803 ;
07D0 1804 ASSUME RCT$L_RBN EQ RCT$L_LBN+4
14 A0 7C 07D0 1804 CLRL RCT$L_LBN(R0) ; Zero out RBN and LBN.
D4 07D3 1805 CLRL RCT$L_BAD_RBN(R0) ; Also clear BADRBN field.
07D6 1806
07D6 1807 CLRL R0 ; Rewrite page 0.
012E 30 07D8 1808 BSBW WRITE RCT_BLOCK ; Go write page into sector.
08 50 E8 07DB 1809 BLBS R0, STEP18 ; Branch if WRITE successful.
07DE 1810 HIR_ERROR - ; Else, signal HIR error.
07DE 1811 step=17, type=WRITE
07E6 1812
07E6 1813 ; STEP18.
07E6 1814 ; We set the failure return code and return to our internal caller.
07E6 1815
07E6 1816
```

```

07E6 1817 STEP18:
07E6 1818 DEALLOC_MSG_BUF
07E9 1819 CLRL R1 ; Prepare to return ERLOGIP bit if set.
07E8 1820 BBCC s^#HIRTSV_ERLOGIP, - ; Branch around if clear and clear if
07F0 1821 HIRTSV_STS,10$ ; already set.
51 04 A8 07F1 1822 BISW #CDRPSM_ERLIP,R1 ; Set bit in R1 so as to return to caller.
07F4 1823 10$:
50 0008'CF 3C 07F4 1824 MOVZWL HIRTSV_IOST,R0 ; Indicate failure to caller.
05 50 E9 07F9 1825 BLBC R0,15$ ; Branch if error already found.
50 000A'CF 3C 07FC 1826 MOVZWL HIRTSV_IOWORST,R0 ; Else, get worst case I/O status.
0801 1827 15$: HIRT_SUBRETURN
080B 1828
080B 1829 REPLACE_CONNECT_FAILURE: ; Come here if CONNECTION failure
080B 1830 ; anywhere in REPLACE logic.
080B 1831
F7F2' 31 080B 1832 BRW DUTUSKILL_THIS_THREAD ; Branch to kill this thread.

```

```
080E 1834 .SBTTL DU$ONLINE_COMPLETE - Perform HIRT operations after ONLINE
080E 1835 :++
080E 1836 :
080E 1837 : DU$ONLINE_COMPLETE - Perform HIRT operations after ONLINE
080E 1838 :
080E 1839 : Functional Description:
080E 1840 :
080E 1841 : Complete bringing a unit ONLINE when it is attached to a controller
080E 1842 : that require HOST INITIATED dynamic bad block replacement. This
080E 1843 : routine reads sector zero of the RCT to see if the disk went offline
080E 1844 : in the middle of bad block replacement. If so the replacement is
080E 1845 : completed.
080E 1846 :
080E 1847 : Inputs:
080E 1848 :
080E 1849 : R3 UCB address for the unit that being brought ONLINE
080E 1850 : R5 HIRT permanent CDRP address
080E 1851 :
080E 1852 : Implicit Inputs:
080E 1853 :
080E 1854 : HIRT$L_SAVDCDRP CDRP that describes the current operation
080E 1855 :
080E 1856 : HIRT is owned by the current thread
080E 1857 : HIRT SUBSTACK is reset
080E 1858 :
080E 1859 : Outputs:
080E 1860 :
080E 1861 : R0 Replacement status
080E 1862 : R1 Setting for CDRP$V_ERLIP
080E 1863 : R3 UCB address (unchanged)
080E 1864 :
080E 1865 : R2, R4, R5 destroyed.
080E 1866 : All other registers preserved.
080E 1867 :--
080E 1868 :
080E 1869 :
080E 1870 DU$ONLINE_COMPLETE::
080E 1871 :
080E 1872 HIRT_SUBSAVE ; Save return point on SUBSTACK.
0818 1873 :
004C'CF 01 CE 0818 1874 ASSUME HIRT$W_PGOCNTNT+2 EQ HIRT$W_PG1CNTNT
0818 1875 MNEGL #1,HIRT$W_PGOCNTNT ; Invalidate contents of pages 0 and 1.
081D 1876 :
0050'CF 01 CE 081D 1877 ASSUME HIRT$W_PG2CNTNT+2 EQ HIRT$W_PG3CNTNT
081D 1878 MNEGL #1,HIRT$W_PG2CNTNT ; Invalidate contents of pages 2 and 3.
0822 1879 :
0822 1880 : Here we want to read sector 0 of the RCT into page 0 so as to be able to
0822 1881 : determine whether or not we went down in the middle of Dynamic Bad
0822 1882 : Block replacement.
0822 1883 : Note we have NOT allocated a Message Buffer. READ_RCT_BLOCK (via a call
0822 1884 : to BUILD_RCT_PACKET) will do it for us.
0822 1885 :
0822 1886 CLRQ R0 ; Indicate read sector 0 (R1) into
0824 1887 : page 0 (R0).
0824 1888 BSBW READ_RCT_BLOCK ; Read indicated sector into page.
0827 1889 BLBS R0,20$ ; LBS means successful read.
082A 1890 HIR_ERROR - ; Signal HIR error.
```

```
082A 1891 step=1, func=ONLINE, -
082A 1892 type=READ
FFB1 31 0832 1893 BRW STEP18 ; Goto deallocate and set RCT_FAILURE
0835 1894 ; status code before returning.
0835 1895 20$:
0835 1896
0835 1897 ; Here we do a write of page 0 to sector 0 (all copies) to insure that we
0835 1898 ; did not crash in the middle of an update of sector 0 and thereby
0835 1899 ; get a set of inconsistent copies.
0835 1900
0835 1901 CLRL R0 ; Rewrite page 0.
00CF 30 0837 1902 BSBW WRITE_RCT_BLOCK ; Write indicated sector from page.
OB 50 E8 083A 1903 BLBS R0,30$ ; LBS means successful write.
083D 1904 HIR_ERROR - ; Signal HIR error.
083D 1905 step=2, func=ONLINE, -
083D 1906 type=WRITE
FF9E 31 0845 1907 BRW STEP18 ; Goto deallocate and set RCT_FAILURE
0848 1908 ; status code before returning.
0848 1909 30$:
50 0024'CF D0 0848 1910 MOVL HIRTSL_PAGEOPTR,R0 ; R0 => sector 0 in memory.
B3 084D 1911 BITW #RCTSM-RP1- ; Test for phase 1 of replacement
084E 1912 !RCTSM-RP2,- ; or phase 2.
084E 1913 RCTSM_FLAGS(R0)
OB A0 C000 8F 12 0853 1914 BNEQ 40$ ; NEQ implies that we were in the
0855 1915 ; middle of replacement.
0855 1916 DEALLOC_MSG_BUF ; Else we deallocate the buffer
0858 1917 CLRL R1 ; Prepare to return ERLOGIP bit if set.
000E'CF 07 E5 085A 1918 BBCC s^#HIRT$V_ERLOGIP, - ; Branch around if clear and clear if
085F 1919 HIRT$V_STS,35$ ; already set.
51 04 A8 0860 1920 BISW #CDRPSM_ERLIP,R1 ; Set bit in R1 so as to return to caller.
0863 1921 35$:
50 01 3C 0863 1922 MOVZWL #SS$ _NORMAL,R0 ; deallocate the RSPID, and we return
0866 1923 ; to caller with a success status.
0866 1924 HIRT_SUBRETURN ; Return.
0870 1925 40$:
OC A0 D0 0870 1926 MOVL RCTSL_LBN(R0),- ; Restore LBN to replace to HIRT.
0018'CF 0873 1927 HIRTSL_LBN
50 01 D0 0876 1928 MOVL #1,R0 ; Read into page 1.
51 01 D0 0879 1929 MOVL #1,R1 ; From RCT sector 1.
0144 30 087C 1930 BSBW READ_RCT_BLOCK ; Read RCT block.
OB 50 E8 087F 1931 BLBS R0,50$ ; LBS means successful read.
0882 1932 HIR_ERROR - ; Signal HIR error.
0882 1933 step=3, func=ONLINE, -
0882 1934 type=READ
FF59 31 088A 1935 BRW STEP18 ; Goto deallocate and set RCT_FAILURE
088D 1936 ; status code before returning.
088D 1937 50$:
50 0024'CF D0 088D 1938 MOVL HIRTSL_PAGEOPTR,R0 ; Again, R0 => sector 0 contents.
0892 1939
AA 0892 1940 BICW #HIRTSM_FE- ; Initialize incore bits.
0893 1941 !HIRTSM_MATCH-
0893 1942 !HIRTSM_EMPTY-
0893 1943 !HIRTSM_RESCAN-
0893 1944 !HIRTSM_RCTFULL,-
000E'CF 007C 8F 0893 1945 HIRTSM_STS
0899 1946
07 E1 0899 1947 BBC #RCTSV_FE,- ; Branch if no forced error.
```

```
05 08 A0 000E'CF 04 AB 089B 1948 RCT$W_FLAGS(R0),60$
089E 1949 BISW s^#HIRT$M_FE,HIRT$W_STS ; Set incore forec error bit.
08A3 1950 60$: BBC #RCT$V_RP1,- ; See if NOT in phase 1.
03 08 A0 0F E1 08A3 1951 RCT$W_FLAGS(R0),70$
FB50 31 08A5 1952 BRW STEP7 ; Branch into step 7 to continue.
08AB 1953 70$: BBS #RCT$V_RP2,- ; Sanity check.
04 08 A0 0E E0 08AB 1954 RCT$W_FLAGS(R0),75$
08AD 1955 BUG_CHECK DISKCLASS,FATAL
08B0 1956 75$:
08B4 1957 MOVL RCT$L_RBN(R0),- ; Remember RBN that we had selected.
0058'CF 10 A0 D0 08B4 1959 HIRT$C_RBN
08B7 1960
08BA 1961 BBC #RCT$V_BR,- ; See if we had had bad RBN.
08 08 A0 0D E1 08BA 1962 RCT$W_FLAGS(R0),80$ ; Clear means no.
000E'CF 08 AB 08BC 1963 s^#HIRT$M_MATCH,- ; We set equivalent bit in core.
08BF 1964
08C4 1965 MOVL RCT$L_BAD_RBN(R0),- ; Copy the old bad RBN.
005C'CF 14 A0 D0 08C4 1966 HIRT$C_MATCHRBN
08C7 1967 80$:
08CA 1968 ; Here figure out whether this is prime RBN.
08CA 1969
08CA 1970
08CA 1971
0309 30 08CA 1972 BSBW HASH_LBN ; Hash HIRT$C_LBN to produce values
08CD 1973 ; for HIRT$C_RCTBLOCK and HIRT$C_OFFSET.
51 0068'CF 02 C3 08CD 1974 SUBL3 #2,HIRT$C_RCTBLOCK,R1 ; Subtract out RCT sectors 0 and 1.
51 51 07 78 08D3 1975 ASHL #7,R1,R1 ; R1 = relative block containing prime
08D7 1976 ; RBN descriptor * 128.
51 006C'CF C0 08D7 1977 ADDL HIRT$C_OFFSET,R1 ; R1 = prime RBN.
0058'CF 51 D1 08DC 1978 CMPL R1,HIRT$C_RBN ; See if this the one.
05 13 08E1 1979 BEQL 90$ ; EQL implies yes.
10 AB 08E3 1980 BISW #HIRT$M_EMPTYTYPE,- ; Set bit meaning NOT prime RBN.
000E'CF 08E5 1981
08E8 1982 90$:
08E8 1983 ; Here read sector containing allocatable RBN into page 2.
08E8 1984
08E8 1985
51 0058'CF F9 8F 78 08E8 1986 ASHL #-7,HIRT$C_RBN,R1 ; R1 = relative RCT block containing
08EF 1987 ; this RBN descriptor.
51 02 C0 08EF 1988 ADDL #2,R1 ; Add in sectors 0 and 1.
50 02 D0 08F2 1989 MOVL #2,R0 ; Read into page 2.
00CB 30 08F5 1990 BSBW READ_RCT_BLOCK ; Go read RCT sector.
08 50 E8 08F8 1991 BLBS R0,100$ ; Branch on success.
08FB 1992
08FB 1993 HIR_ERROR - ; Signal HIR error.
08FB 1994 step=4, func=ONLINE, -
08FB 1995 type=READ
FEE0 31 0903 1996 BRW STEP18 ; Goto deallocate and set RCT_FAILURE
0906 1997 ; status code before returning.
0906 1998 100$:
FC59 31 0906 1999 BRW STEP11
```



```
0909 2001 .SBTTL WRITE_RCT_BLOCK - Write an RCT sector
0909 2002
0909 2003 :+
0909 2004 : WRITE_RCT_BLOCK - internal subroutine to write a block to a particular
0909 2005 : relative sector in each RCT copy.
0909 2006
0909 2007 : INPUTS:
0909 2008 : R0 = page number from which we write the sector
0909 2009 : R3 => UCB
0909 2010 : R5 => CDRP
0909 2011 : HIRTSW_PGOCNTNT[R0] contains relative sector number to write
0909 2012 : CDRP$$_MSG_BUF contains an END PACKET to recycle
0909 2013 : CDRP$$_RSPID contains an RSPID to recycle
0909 2014
0909 2015 : OUTPUTS:
0909 2016 : R0 - LBS indicates we were successful in writing at least one
0909 2017 : RCT.
0909 2018 : R0 - LBC indicates failure in all writes.
0909 2019
0909 2020 : SIDE EFFECTS:
0909 2021 : In this routine we use HIRTSW_IOST as the repository of the
0909 2022 : combined status of the writes that we execute. In other words, if
0909 2023 : on finishing the writes, the low bit of HIRTSW_IOST is set,
0909 2024 : one or more of the writes was successful.
0909 2025
0909 2026 : NOTE:
0909 2027 : Since this subroutine is one of those that calls SCS routines which
0909 2028 : may fork, and since we may not leave anything permanent on the stack,
0909 2029 : the caller's return point is popped off the stack and pushed onto
0909 2030 : the HIRT SUBSTACK via use of the HIRT_SUBSAVE macro. Return to the
0909 2031 : caller is effected by use of the HIRT_SUBRETURN macro.
0909 2032 :
0909 2033 :
0909 2034 : WRITE_RCT_BLOCK:
0909 2035
0909 2036 : HIRT_SUBSAVE
0909 2037 : MOVW R0,HIRTSW_PAGENO ; Save return point in SUBSTACK.
0909 2038 : MOVW HIRTSW_PGOCNTNT[R0],- ; Save input argument as to which page.
0909 2039 : HIRTSW_SECTORNO ; Also save input argument as to which
0909 2040 : MOVW #SS$ BADRCT,HIRTSW_IOST ; sector (page) to write.
0909 2041 : CLRL HIRTS$_LOOPCNT ; Initialize combined status word.
0909 2042 : ; Initialize loop counter. Note loop
0909 2043 : ; counter is longword even though we
0909 2044 : ; only use one byte since we MULL2
0909 2045 : ; with the counter in BUILD_RCT_PACKET.
0909 2046 : BBS #UCBS$_MSCP WRTP, - ; If disk is software write protected
0909 2047 : UCBS$_DEVSTS(R3), 46$ ; branch around and reject.
0909 2048 : 20$:
0909 2049 : BICW #HIRTS$_RCTFE,- ; Initialize flag each time thru loop.
0909 2050 : HIRTS$_SIS
0909 2051 : CMPB HIRTS$_LOOPCNT,- ; See if we are all done with all
0909 2052 : UCBS$_DU_RCTCPYS(R3) ; RCT copies.
0909 2053 : BLSSU 30$ ; LSSU implies NOT done.
0909 2054 : BRB 70$ ; If done, branch around.
0909 2055 : 30$:
0909 2056 : MOVZWL HIRTS$_PAGENO,R0 ; R0 contains which page to map.
0909 2057 : BSBW MAP_PAGE ; Map page selected by R0.
```

0056'CF	50	B0	0913	2037
0054'CF	004C'CF40	B0	0918	2038
0008'CF	216C 8F	B0	0920	2039
	0010'CF	D4	0927	2041
			0928	2042
			0928	2043
5F 68 A3	0D	E0	0928	2044
	0100 8F	AA	0930	2045
	000E'CF		0930	2046
	0010'CF	91	0937	2047
	0000'C3		0938	2048
	02	1F	093E	2049
	6A	11	0940	2050
			0942	2051
50	0056'CF	3C	0942	2052
	0161	30	0947	2053
			094A	2057

```
094A 2058 ; Recycle the current END PACKET, the current RSPID and then prepare
094A 2059 ; the MSCP packet to write page into the next RCT copy at relative
094A 2060 ; sector of this copy. All this is accomplished by BUILD_RCT_PACKET.
094A 2061
0108 30 094A 2062 BSBW BUILD_RCT_PACKET ; Routine fills most MSCP fields.
094D 2063 ; Returns R2 => MSCP packet.
094D 2064 MOVW #MSCPSK_OP_WRITE, - ; Copy the WRITE opcode.
08 A2 90 094F 2065 MSCPSB_OPCODE(R2)
4000 8F 80 0951 2066 MOVW #MSCPSB_MD_COMP, - ; Move in compare modifier to get a
0A A2 0955 2067 MSCPSB_MODIFIER(R2) ; write compare operation.
0957 2068
000E'CF 08 E1 0957 2069 BBC s^#HIRT$V_RCTFE, - ; Bit clear says write WITHOUT force
06 095C 2070 HIRT$W_STS,40$ ; error.
1000 8F A8 095D 2071 BISW #MSCPSB_MD_ERROR, - ; Set force error modifier.
0A A2 0961 2072 MSCPSB_MODIFIER(R2)
0963 2073 40$:
0963 2074 SEND MSCP_MSG ; Send message to the MSCP server.
0966 2075 UNMAP ; Unmap page.
2C A5 D4 0969 2076 CLRL CDRP$L_LBUFH_AD(R5) ; Indicate no mapping resources
096C 2077 ; currently allocated.
52 1C A5 D0 096C 2078 MOVL CDRP$L_MSG_BUF(R5),R2 ; Refresh R2 => End message.
0970 2079
0970 2080 BBC #MSCPSV_EF_ERLOG, - ; Test for error log message generated
07 09 A2 E1 0972 2081 MSCPSB_FLAGS(R2),45$ ; and branch around if not.
0080 8F A8 0975 2082 BISW #HIRT$M_ERLOGIP, - ; Else remember that error log messages
000E'CF 0979 2083 HIRT$W_STS ; Have been generated.
097C 2084 45$:
097C 2085
097C 2086 ; See if write succeeded and if so set HIRT$W_IOST to success; otherwise
097C 2087 ; do nothing. In this way if one or more writes succeed, HIRT$W_IOST
097C 2088 ; will have a success indication.
097C 2089
000E'CF 08 E0 097C 2090 BBS s^#HIRT$V_RCTFE, - ; Branch around status update if
24 0981 2091 HIRT$W_STS,60$ ; we had force error.
0982 2092 IF_MSCP SUCCESS, then=50$, - ; Branch if request was successful,
0982 2093 status=R0 ; leaving MSCP status in R0.
50 06 B1 098A 2094 CMPW #MSCPSK_ST_WRTPR, R0 ; Check for write protected.
09 12 098D 2095 BNEQ 48$ ; If NOT, some other error.
098F 2096 46$:
0008'CF 025C 8F 80 098F 2097 MOVW #SS$_WRTLCK,HIRT$W_IOST; Indicate why we couldn't write.
14 11 0996 2098 BRB 70$ ; And branch around.
0998 2099 48$:
0100 8F A8 0998 2100 BISW #HIRT$M_RCTFE, - ; Set force error flag and
000E'CF 099C 2101 HIRT$W_STS ; branch back to rewrite it.
A1 11 099F 2102 BRB 30$
09A1 2103 50$:
0008'CF 01 B0 09A1 2104 MOVW S^#SS$_NORMAL, - ; If success, remember it in static
09A3 2105 HIRT$W_IOST ; HIRT field.
09A6 2106 60$:
0010'CF D6 09A6 2107 INCL HIRT$L_LOOPCNT ; Increment loop counter.
84 11 09AA 2108 BRB 20$ ; And branch back to do next copy.
09AC 2109
09AC 2110 70$:
50 0008'CF 3C 09AC 2111 MOVZWL HIRT$W_IOST,R0 ; Here after we finish all RCT copies.
05 50 E8 09B1 2112 BLBS R0, 75$ ; Return status to caller.
000A'CF 50 B0 09B4 2113 MOVW R0, HIRT$W_IOWORST ; Branch if successful.
09B9 2114 75$: HIRT_SUBRETURN ; Else, save "worst" error.
; Return to caller.
```



```
09C3 2116 .SBTTL READ_RCT_BLOCK - Read an RCT sector
09C3 2117
09C3 2118 :+
09C3 2119 : READ_RCT_BLOCK - internal routine to read contents of a relative sector of
09C3 2120 : the RCT.
09C3 2121
09C3 2122 : INPUTS:
09C3 2123 : R0 = page number of page into which we read the sector
09C3 2124 : R1 = relative sector number to read
09C3 2125 : R3 => UCB
09C3 2126 : R5 => CDRP
09C3 2127 : CDRP$L_MSG BUF contains an END PACKET to recycle
09C3 2128 : CDRP$L_RSPID contains an RSPID to recycle
09C3 2129
09C3 2130 : OUTPUTS:
09C3 2131 : R0 - LBS indicates we were successful in the block from one of the
09C3 2132 : RCT copies.
09C3 2133 : R0 - LBC indicates failure in reading from all RCT copies.
09C3 2134
09C3 2135 : SIDE EFFECTS:
09C3 2136 : We use HIRT$W_IOST to temporarily save the status to be returned
09C3 2137 : to out caller.
09C3 2138
09C3 2139 : NOTE:
09C3 2140 : Since this subroutine is one of those that calls SCS routines which
09C3 2141 : may fork, and since we may not leave anything permanent on the stack,
09C3 2142 : the caller's return point is popped off the stack and pushed onto
09C3 2143 : the HIRT SUBSTACK via use of the HIRT_SUBSAVE macro. Return to the
09C3 2144 : caller is effected by use of the HIRT_SUBRETURN macro.
09C3 2145 :
09C3 2146 :
09C3 2147 READ_RCT_BLOCK:
09C3 2148
09C3 2149 HIRT_SUBSAVE ; Save return point in SUBSTACK.
09C3 2150 MOVW R0,HIRT$W_PAGENO ; Save input argument as to which page.
09C3 2151 MOVW R1,HIRT$W_SECTORNO ; Also save input argument as to which
09C3 2152 ; sector to write.
09C3 2153 MNEGW #1,HIRT$W_PGOCNTNT[R0] ; Invalidate page(R0) contents.
09C3 2154 CLRL HIRT$L_LOOPCNT ; Initialize loop counter.
09C3 2155 20$:
09C3 2156 CMPB HIRT$L_LOOPCNT,- ; See if we are all done with all
09C3 2157 UCB$B_DU_RCTCPYS(R3) ; RCT copies.
09C3 2158 BLSSU 30$ ; LSSU implies NOT done.
09C3 2159 J9EA
09C3 2160 MOVW #SS$_BADRCT,HIRT$W_IOST ; Pass failure to our caller.
09C3 2161 BRB 50$ ; If done, branch around.
09C3 2162 30$:
09C3 2163 MOVZWL HIRT$W_PAGENO,R0 ; R0 contains which page to map.
09C3 2164 BSBW MAP_PAGE ; Map page selected by R0.
09C3 2165
09C3 2166 ; Recycle the current END PACKET, the current RSPID and then prepare
09C3 2167 ; the MSCP packet to read page from the next RCT copy at relative
09C3 2168 ; sector of this copy. All this is accomplished by BUILD_RCT_PACKET.
09C3 2169
09C3 2170 BSBB BUILD_RCT_PACKET ; Routine fills in most of MSCP packet.
09C3 2171 ; Returns R2 => MSCP packet.
09C3 2172 MOVB #MSCP$K_OP_READ,- ; Copy the READ opcode since this field
```

0056'CF	50	B0	09CD	2150
0054'CF	51	B0	09D2	2151
004C'CF40	01	AE	09D7	2152
0010'CF		D4	09DD	2153
0010'CF	91		09E1	2154
0000'C3			09E5	2155
09	1F		09E8	2156
			09EA	2157
0008'CF	216C	8F	09EA	2158
	48	11	09F1	2159
			09F3	2160
50	0056'CF	3C	09F3	2161
	00B0	30	09F8	2162
			09FB	2163
			09FB	2164
			09FB	2165
			09FB	2166
			09FB	2167
			09FB	2168
			09FB	2169
58	10		09FB	2170
			09FD	2171
21	90		09FD	2172

```
08 A2 09FF 2173          MSCPSB_OPCODE(R2)      ; is not filled in by BUILD_RCT_PACKET.
4000 8F B0 0A01 2174      MOVW      #MSCPSB_MD_COMP,-    ; Move in compare modifier to get a
0A A2    0A05 2175          MSCPSW_MODIFIER(R2)      ; compare operation.
          0A07 2176
          0A07 2177      SEND_MSCP_MSG      ; Send message to the MSCP server.
          0A0A 2178      UNMAP      ; Unmap page.
2C A5 D4 0A0D 2179      CLRL      CDRPSL_LBUFH_AD(R5)    ; Indicate no mapping resources
          0A10 2180          ; currently allocated.
52 1C A5 D0 0A10 2181      MOVL      CDRPSL_MSG_BUF(R5),R2 ; Refresh R2 => End message.
          0A14 2182
          05 E1 0A14 2183      BBC      #MSCPSV_EF_ERLOG,-    ; Test for error log message generated
07 09 A2    0A16 2184          MSCPSB_FLAGS(R2),35$      ; and branch around if not.
0080 8F A8 0A19 2185      BISW      #HIRTSM_ERLOGIP,-      ; Else remember that error log messages
000E'CF    0A1D 2186          HIRT$W_STS      ; Have been generated.
          0A20 2187 35$:
          0A20 2188
          0A20 2189 ; See if read succeeded and if so we now have a valid copy of the sector so we
          0A20 2190 ; simply continue. If we did not succeed we bump the loop counter and
          0A20 2191 ; go back to try and read the sector from the next (if any) RCT copy.
          0A20 2192
          0A20 2193      IF_MSCP_SUCCESS, then=40$      ; Branch if request was successful.
0010'CF D6 0A26 2194      INCL      HIRT$W_LOOPCNT      ; Increment loop counter.
          B5 11 0A2A 2195      BRB      20$      ; And branch back to try next copy.
          0A2C 2196
          0A2C 2197 40$:
50 0056'CF 3C 0A2C 2198      MOVZWL      HIRT$W_PAGENO,R0      ; Here after we finish all RCT copies.
          0054'CF B0 0A31 2199      MOVW      HIRT$W_SECTORNO,-    ; R0 = page number into which we read.
          004C'CF40    0A35 2200          HIRT$W_PGOCNTNT[R0]    ; Update contents of this page by
0008'CF 01 B0 0A39 2201      MOVW      S^#SS$-NORMAL,HIRT$W_IOST; remembering sector therein contained.
          0A3E 2202 50$:
50 0008'CF 3C 0A3E 2203      MOVZWL      HIRT$W_IOST,R0      ; Return status to caller.
          05 50 E8 0A43 2204      BLBS      R0, 55$      ; Branch if successful.
000A'CF 50 B0 0A46 2205      MOVW      R0, HIRT$W_IOWORST    ; Else, save 'worst' error.
          0A4B 2206 55$:      HIRT_SUBRETURN      ; Return to caller.
```

```

0A55 2208      .SBTTL BUILD_RCT_PACKET - Recycle an MSCP end message
0A55 2209      .SBTTL FILL_RCT_PACKET - Prepare an MSCP packet for an RCT transfer
0A55 2210
0A55 2211      ;+
0A55 2212      BUILD_RCT_PACKET - internal subroutine to recycle the current END PACKET
0A55 2213      and then to fall thru to
0A55 2214
0A55 2215      FILL_RCT_PACKET - which prepares an MSCP packet to do an I/O transfer
0A55 2216      to or from the RCT.
0A55 2217
0A55 2218      INPUTS:
0A55 2219      R3 => UCB
0A55 2220      R5 => CDRP
0A55 2221      CDRP$L_RSPID contains a RSPID to re-cycle
0A55 2222      CDRP$L_MSG_BUF address of MSCP buffer to re-cycle or 0 (zero)
0A55 2223      0 (zero) means that we must here allocate an MSCP buffer
0A55 2224      CDRP$L_BUFHNDL contains 96 bit buffer handle
0A55 2225      UCB$L_ABCNT contains which RCT copy we are accessing
0A55 2226      HIRT$Q_SECTORNO contains which relative sector number in the RCT copy
0A55 2227
0A55 2228      OUTPUTS:
0A55 2229      R2 => MSCP PACKET
0A55 2230      Registers R0 and R1 are modified
0A55 2231
0A55 2232      NOTE:
0A55 2233      Since BUILD_RCT_PACKET is one of those that calls SCS routines which
0A55 2234      may fork, and since we may not leave anything permanent on the stack,
0A55 2235      the caller's return point is popped off the stack upon entry to
0A55 2236      this entrypoint and pushed onto the HIRT SUBSTACK via use of the
0A55 2237      HIRT_SUBSAVE macro. Upon return from those SCS routines, the caller's
0A55 2238      return point is restored to the normal stack via use of the
0A55 2239      HIRT_SUBUNSAVE macro. all this is done prior to entrypoint
0A55 2240      FILL_RCT_PACKET so that we may fall into this routine and then use
0A55 2241      its RSB to return to our caller.
0A55 2242
0A55 2243
0A55 2244      BUILD_RCT_PACKET:
0A55 2245
0A55 2246      HIRT_SUBSAVE ; Save return point on HIRT SUBSTACK.
0A55 2247
0A55 2248      ASSUME MSCP$L_CMD_REF EQ 0
0A55 2249      TSTL CDRP$L_MSG_BUF(R5) ; See if we need a Message Buffer.
0A55 2250      BEQL 20$ ; EQL means Buffer needed.
0A55 2251
0A55 2252      RECYCH_MSG_BUF ; Else Recycle END PACKET into MSCP buffer.
0A55 2253      BLBS R0,30$ ; LBS means allocation success.
0A55 2254      10$: BRW REPLACE_CONNECT_FAILURE ; Allocation failure means CONNECTION
0A55 2255      ; failure.
0A55 2256      20$:
0A55 2257      ALLOC_MSG_BUF ; Allocate a Message Buffer.
0A55 2258      BLBC R0,10$ ; LBC means allocation failure.
0A55 2259      30$:
0A55 2260      HIRT_SUBUNSAVE ; Restore caller's return point.
0A55 2261
0A55 2262      FILL_RCT_PACKET: ; Alternate entry that only fills in packet.
0A55 2263
0A55 2264      INIT_MSCP_MSG ucb=(R3) ; Initialize MSCP command packet.

```

```
0C A2 0200 8F 3C 0A7F 2265      MOVZWL #512, - ; Setup transfer byte count.
                                0A85 2266      MSCP$L_BYTE CNT(R2)
                                30 A5 7D 0A85 2267      MOVQ CDRP$LBUFHNDL(R5),- ; Copy 96 bit buffer handle.
                                10 A2 0A88 2268      MSCP$B_BUFFER(R2)
                                38 A5 D0 0A8A 2269      MOVL CDRP$LBUFHNDL+8(R5),- ;
                                18 A2 0A8D 2270      MSCP$B_BUFFER+8(R2) ;
                                0A8F 2271 ;
                                0A8F 2272 ; Calculate LBN of relative sector for this RCT copy.
                                0A8F 2273 ; It is done by multiplying the number of RCT copies already written,
                                0A8F 2274 ; (contained in HIRT$L_LOOPCNT) by the size of an RCT copy (contained in
                                0A8F 2275 ; UCBSW_DU_RCTSIZE), adding in the LBN of the base of the first RCT copy
                                0A8F 2276 ; (UCBSW_DU_USIZE) and then adding in the relative sector number
                                0A8F 2277 ; passed to us when we were called (HIRT$W_SECTORNO).
                                0A8F 2278 ;
                                50 0000'C3 3C 0A8F 2279      MOVZWL UCBSW_DU_RCTSIZE(R3),R0 ; R0 contains size of one RCT copy.
                                50 0010'CF C4 0A94 2280      MULL2 HIRT$L_LOOPCNT,R0 ; R0 contains COPY# * (COPYSIZE).
                                50 0000'C3 C0 0A99 2281      ADDL UCBSW_DU_USIZE(R3),R0 ; R0 contains LBN of base of this copy.
                                51 0054'CF 3C 0A9E 2282      MOVZWL HIRT$W_SECTORNO,R1 ; R1 contains input relative sector #.
                                50 50 51 C0 0AA3 2283      ADDL R1,R0 ; R0 contains LBN.
                                1C A2 50 D0 0AA6 2284      MOVL R0,MSCP$L_LBN(R2) ; Move LBN to MSCP packet.
                                0AAA 2285 ;
                                05 0AAA 2286      RSB ; Return to caller.
```

```
0AAB 2288 .SBTTL MAP_PAGE - Map a page for a transfer
0AAB 2289
0AAB 2290 :+
0AAB 2291 : MAP_PAGE - internal subroutine to map the page selected by R0.
0AAB 2292 :
0AAB 2293 : INPUTS:
0AAB 2294 : R0 contains the number of the page to map.
0AAB 2295 : R5 => CDRP
0AAB 2296 :
0AAB 2297 : OUTPUTS:
0AAB 2298 : CDRP$L_SVAPTE, CDRP$W_BOFF, CDRP$L_BCNT and set to page parameters.
0AAB 2299 :
0AAB 2300 : CDRP$L_LBUFH_AD set to => CDRP$L_LBUFHNDL
0AAB 2301 :
0AAB 2302 : Mapping resources allocated.
0AAB 2303 :
0AAB 2304 MAP_PAGE:
0AAB 2305 HIRT_SUBSAVE ; Save caller's return point on SUBSTACK.
CC A5 0034'CF40 D0 0AB5 2306 MOVL HIRT$L_SVAPTE0[R0],- ; Copy mapping date for relative page
0AAB 2307 CDRP$L_SVAPTE(R5) ; to CDRP.
D0 A5 0044'CF40 B0 0ABC 2308 MOVW HIRT$W_BOFF0[R0],- ; Copy BOFF as well as SVAPTE.
0AAB 2309 CDRP$W_BOFF(R5)
D2 A5 0200 8F 3C 0AC3 2310 MOVZWL #512,CDRP$L_BCNT(R5) ; Finally copy BCNT for page.
0AAB 2311
0AAB 2312 MOVAB CDRP$L_LBUFHNDL(R5),- ; Point CDRP field to local buffer
30 A5 9E 0AC9 2313 CDRP$L_LBUFH_AD(R5) ; handle field.
2C A5 0ACC 2314 MAP_IRP ; Map page.
0AAB 2315
0AAB 2316 HIRT_SUBRETURN ; Return to caller.
```

```

OADB 2318      .SBTTL SEARCH_RCT - Locate an available RBN
OADB 2319
OADB 2320      :+
OADB 2321      SEARCH_RCT - internal subroutine to search the RCT for an available RBN
OADB 2322      to allocate for the current failing LBN. This routine is called from
OADB 2323      STEP9 of the replacement algorithm and is only done here as an
OADB 2324      internal subroutine to simplify the reading of that algorithm.
OADB 2325
OADB 2326      INPUTS:
OADB 2327      R3 => UCB
OADB 2328      R5 => CDRP
OADB 2329      HIRT$L_LBN LBN that is failing
OADB 2330      UCBSW_DU_LBNPTRK number of LBNs on a track of this unit
OADB 2331      UCBSB_DU_RBNPTRK number of RBNs on a track of this unit
OADB 2332
OADB 2333      OUTPUTS:
OADB 2334      R0 = $$$ NORMAL then:
OADB 2335      HIRT$L_RBN - new RBN selected to replace the failing LBN
OADB 2336      and HIRT$V_EMPTYTYPE clear means this is a primary RBN, else
OADB 2337      secondary RBN.
OADB 2338
OADB 2339      If HIRT$V_MATCH set this implies that the LBN which failed
OADB 2340      had previously been replaced by an RBN which in
OADB 2341      turn has failed. This failing RBN is in
OADB 2342      HIRT$L_MATCHRBN.
OADB 2343
OADB 2344      R0 = 0 then we could not find an allocatable RBN and HIRT$L_RBN is
OADB 2345      not valid. The cause of the failure to find an RBN is
OADB 2346      transmitted to the caller by:
OADB 2347
OADB 2348      HIRT$V_RCTFULL set implies that the RCT on the disk
OADB 2349      is full
OADB 2350      HIRT$V_RCTFULL clear implies we had a read error on
OADB 2351      some RCT sector.
OADB 2352
OADB 2353
OADB 2354      SEARCH_RCT:
OADB 2355
OADB 2356      HIRT_SUBSAVE      ; Save return on HIRT substack.
OADB 2357
OADB 2358      BSBW      HASH_LBN      ; Hash LBN value in HIRT$L_LBN returning
OADB 2359      ; HIRT$L_RCTBLOCK and HIRT$L_OFFSET.
OADB 2360
OADB 2361      MOVL      HIRT$L_RCTBLOCK,-      ; And remember the starting sector
OADB 2362      HIRT$L_STARTBLK      ; number in static storage.
OADB 2363
OADB 2364      ; Here we initialize a few bits.
OADB 2365
OADB 2366      BICW      #HIRT$M_MATCH-      ; Initialize the following flags.
OADB 2367      !HIRT$M_EMPTYTYPE-      ; Match set implies valid MATCHRBN,
OADB 2368      !HIRT$M_RESCAN-      ; EMPTYTYPE set implies secondary RBN.
OADB 2369      !HIRT$M_RCTFULL,-      ; Rescan implies reached Nulls,
OADB 2370      HIRT$W_STS      ; and RCTFULL means the RCT is full.
OADB 2371
OADB 2372      ; Here we prepare to read the RCT sector containing the primary RBN descriptor.
OADB 2373
OADB 2374      MOVL      #2,R0      ; Prepare to read into page #2.
```

00EE 30 OADB 2357 HIRT_SUBSAVE ; Save return on HIRT substack.

0068'CF 0064'CF OADB 2358 BSBW HASH_LBN ; Hash LBN value in HIRT\$L_LBN returning

OADB 2359 ; HIRT\$L_RCTBLOCK and HIRT\$L_OFFSET.

OADB 2360

0068'CF 0064'CF OADB 2361 MOVL HIRT\$L_RCTBLOCK,- ; And remember the starting sector

OADB 2362 HIRT\$L_STARTBLK ; number in static storage.

OADB 2363

OADB 2364 ; Here we initialize a few bits.

OADB 2365

AA OADB 2366 BICW #HIRT\$M_MATCH- ; Initialize the following flags.

OADB 2367 !HIRT\$M_EMPTYTYPE- ; Match set implies valid MATCHRBN,

OADB 2368 !HIRT\$M_RESCAN- ; EMPTYTYPE set implies secondary RBN.

OADB 2369 !HIRT\$M_RCTFULL,- ; Rescan implies reached Nulls,

000E'CF 0078 8F OADB 2370 HIRT\$W_STS ; and RCTFULL means the RCT is full.

OADB 2371

OADB 2372 ; Here we prepare to read the RCT sector containing the primary RBN descriptor.

OADB 2373

50 02 00 OADB 2374 MOVL #2,R0 ; Prepare to read into page #2.


```
51 0068'CF D0 0AF9 2375      MOVL  HIRT$R_RCTBLOCK,R1      ; And we read this relative sector #.
      FEC2 30 0AFE 2376      BSBW  READ_RCT_BLOCK      ; Subroutine does read.
      77 50 E9 0B01 2377      BLBC  R0,SEARCH_RTN      ; LBC implies read failure.
      0B04 2378
      0B04 2379 ; Here we scan the RCT sector containing the primary RBN descriptor. The
      0B04 2380 ; method of scanning is to scan outward from the primary RBN descriptor.
      0B04 2381
      0B04 2382 10$:
      52 D4 0B04 2383      CLRL  R2      ; Set up delta.
      0B06 2384 20$:
51 006C'CF 52 C1 0B06 2385      ADDL3 R2,HIRT$R_OFFSET,R1      ; R1 = next entry to test in first RCT
      0B0C 2386      ; sector to scan.
      0F 19 0B0C 2387      BLSS  40$      ; LSS implies invalid offset into page.
      0B0E 2388 30$:
0000007F 8F 51 D1 0B0E 2389      CMPL  R1,#127      ; See if we are within sector page.
      06 14 0B15 2390      BGTR  40$      ; GTR implies no, out of bounds, go
      0B17 2391      ; to increment delta.
      006B 30 0B17 2392      BSBW  TEST_RCT_ENTRY      ; If in bounds, go test RCT entry.
      5E 50 E8 0B1A 2393      BLBS  R0,SEARCH_RTN      ; LBS implies success.
      0B1D 2394 40$:
      52 52 CE 0B1D 2395      MNEGL R2,R2      ; Negate delta.
      E4 19 0B20 2396      BLSS  20$      ; Branch to try again if negative.
      52 D6 0B22 2397      INCL  R2      ; Else increment delta.
00000080 8F 52 D1 0B24 2398      CMPL  R2,#128      ; See if delta too big.
      D9 19 0B2B 2399      BLSS  20$      ; LSS implies not too big.
      0B2D 2400      ; Else we fall thru to try next sector.
      0B2D 2401 NEXT:
      0068'CF D6 0B2D 2402      INCL  HIRT$R_RCTBLOCK      ; Increment RCT sector to scan.
      0B31 2403 10$:
      0068'CF D1 0B31 2404      CMPL  HIRT$R_RCTBLOCK,-      ; See if we are all done with search.
      0064'CF 0B35 2405      HIRT$R_STARTBLK
      38 13 0B38 2406      BEQL  SEARCH_FAIL      ; EQL means that we are finished.
      0B3A 2407
      50 02 D0 0B3A 2408      MOVL  #2,R0      ; Prepare to read into page 2.
51 0068'CF D0 0B3D 2409      MOVL  HIRT$R_RCTBLOCK,R1      ; And to read this sector.
      FE7E 30 0B42 2410      BSBW  READ_RCT_BLOCK      ; Go to read sector into page.
      33 50 E9 0B45 2411      BLBC  R0,SEARCH_RTN      ; LBC implies read failure
      0B48 2412
      0F 002C'DF E1 0B48 2413      BBC  #RCT$V_NULL,-      ; Before linear scan of this sector,
      0068'CF 02 D0 0B4A 2414      @HIRT$C_PAGE2PTR,20$      ; see if we are beyond RCT.
      000E'CF 05 E3 0B4E 2415      MOVL  #2,HIRT$R_RCTBLOCK      ; Here beyond RCT. Wrap to start and
      D8 0B53 2416      BBCS  s^HIRT$V_RESCAN,-      ; go back to search some more after
      0B58 2417      HIRT$R_STS,10$      ; setting bit that says we have wrapped.
      0B59 2418      BUG_CHECK DISKCLASS,FATAL ; Impossible situation.
      0B5D 2419 20$:
      52 D4 0B5D 2420      CLRL  R2      ; Clear loop index register.
      0B5F 2421 30$:
      51 52 D0 0B5F 2422      MOVL  R2,R1      ; Pass RCT entry of interest to routine.
      0020 30 0B62 2423      BSBW  TEST_RCT_ENTRY      ; Call subroutine to test entry.
      13 50 E8 0B65 2424      BLBS  R0,SEARCH_RTN      ; LBS means we have the RBN, go from loop.
EF 52 00000080 8F F2 0B68 2425      AOBLS  #128,R2,30$      ; If we return here, (entry not avail.)
      0B70 2426      ; then loop back after incrementing R2.
      BB 11 0B70 2427      BRB  NEXT      ; If we fall thru, goto NEXT sector.
      0B72 2428
      0B72 2429 SEARCH_FAIL:
      50 D4 0B72 2430      CLRL  R0      ; Indicate failure to caller and
      0040 8F A8 0B74 2431      BLSW  #HIRT$R_RCTFULL,-      ; indicate reason for failure.
```

DUHIRT
V04-000

I 16
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
SEARCH_RCT - Locate an available RBN 5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 53
(21)

000E'CF

0878 2432 HIRTSW_STS
0878 2433 SEARCH_RTN:
0878 2434 HIRT_SUBRETURN

; Return to caller.


```
OB85 2436 .SBTTL TEST_RCT_ENTRY - Test for allocated RBN
OB85 2437
OB85 2438 :+
OB85 2439 : TEST_RCT_ENTRY - internal subroutine called to test an RCT entry to see
OB85 2440 : if it represents an allocatable RBN or if it is already allocated.
OB85 2441 :
OB85 2442 : INPUTS:
OB85 2443 : R1 = index of RCT entry.
OB85 2444 :
OB85 2445 : OUTPUTS:
OB85 2446 : R0 = success code.
OB85 2447 : = $$$ NORMAL then HIRTSL_RBN is set to the RBN associated with the
OB85 2448 : RCT entry defined by HIRTSL_RCTBLOCK and R1 (entry index).
OB85 2449 :
OB85 2450 : = 0 implies that the entry is not allocatable. In addition if
OB85 2451 : the RBN is currently allocated and if it is allocated to this
OB85 2452 : LBN (i.e. to HIRTSL_LBN), then HIRTSLV_MATCH is set in
OB85 2453 : HIRTSLV_STIS and the RBN associated with the current entry is
OB85 2454 : stored in HIRTSL_MATCHRBN.
OB85 2455 :
OB85 2456 :
OB85 2457 :
OB85 2458 TEST_RCT_ENTRY:
OB85 2459 PUSHL R2 ; Save register.
50 0068'CF 52 DD OB85 2459 SUBL3 #2,HIRTSL_RCTBLOCK,R0 ; R0 = found sector without bias of 2.
50 50 07 C3 OB87 2460 ASHL #7,R0,R0 ; Multiply by 128.
50 50 51 C0 OB8D 2461 ADDL R1,R0 ; R0 = RBN associated with this entry.
OB85 2462
OB85 2463
OB85 2464 MOVL HIRTSL_PAGE2PTR,R2 ; R2 => page 2, which contains sector.
52 002C'CF 52 DD OB89 2464 TSTL (R2)[R1] ; Test contents of current entry.
6241 D5 OB99 2465 BNEQ 10$ ; NEQ implies that it is not available.
0A 12 OB9C 2466 MOVL R0,HIRTSL_RBN ; Save RBN of this entry in HIRT.
0058'CF 50 DD OB9E 2467 MOVZWL #$$$_NORMAL,R0 ; Set success code.
50 01 3C OBA3 2468 BRB 40$ ; And branch to return to caller.
2A 11 OBA6 2469
OB85 2470 10$:
52 6241 DE OBA8 2471 MOVAL (R2)[R1],R2 ; R2 => entry of interest.
10 A8 OBAC 2472 BISW #HIRTSL_EMPTYTYPE,- ; Set bit meaning any find will now
000E'CF 0BAE 2473 HIRTSLV_STIS ; have to be a secondary RBN.
1B 62 1D E1 OB81 2474 BBC #RCTSV_ALLOCATED,(R2),30$ ; If clear, then unusable RBN.
OB85 2475
OB85 2476 EXTZV #RCTSV_LBN,- ; If allocated, see if for this LBN.
52 62 1C OB87 2477 #RCTSV_LBN,(R2),R2 ; R2 = LBN for this RBN.
0018'CF 52 D1 OBBA 2478 CMPL R2,HIRTSL_LBN ; See if this LBN.
0F 12 OB8F 2479 BNEQ 30$ ; NEQ means not for this LBN.
000E'CF 03 E3 OBC1 2480 BBBS s^#HIRTSLV_MATCH,- ; Set bit that means we have a match.
04 OBC6 2481 HIRTSLV_STIS,20$
OB85 2482 BUG_CHECK DISKCLASS,FATAL ; Impossible situation.
OB85 2483 20$:
005C'CF 50 DD OBCB 2483 MOV. R0,HIRTSL_MATCHRBN ; Save RBN that matched.
50 D4 OBD0 2485 30$:
OBD0 2486 CLRL R0 ; Failure to find allocatable RBN.
52 8ED0 OBD2 2487 40$:
05 OBD2 2488 POPL R2 ; Restore register.
OBD5 2489 RSB ; Return to caller.
```

```

OBD6 2491 .SBTTL HASH_LBN - Hash an LBN into a RCT block and an offset
OBD6 2492
OBD6 2493 :+
OBD6 2494 : HASH_LBN - internal routine to hash HIRT$L_LBN giving HIRT$L_RCTBLOCK and
OBD6 2495 : HIRT$L_OFFSET.
OBD6 2496
OBD6 2497 : INPUTS:
OBD6 2498 :
OBD6 2499 : R3 => UCB
OBD6 2500 : HIRT$L_LBN
OBD6 2501
OBD6 2502 : OUTPUTS:
OBD6 2503 :
OBD6 2504 : HIRT$L_RCTBLOCK = RCT sector containing prime RBN descriptor for this
OBD6 2505 : LBN
OBD6 2506 : HIRT$L_OFFSET = offset of prime RBN descriptor in sector.
OBD6 2507
OBD6 2508 : SIDE EFFECTS:
OBD6 2509 :
OBD6 2510 : Registers R0 an R1 altered.
OBD6 2511 :
OBD6 2512 :
OBD6 2513 HASH_LBN:
OBD6 2514
OBD6 2515 MOVZWL UCB$W_DU_LBNPTRK(R3),R0 ; R0 contains LBNs per track.
51 50 0000'C3 3C OBD6 2516 DIVL3 R0,HIRT$L_LBN,P1 ; R1 = QUO(LBN/(LBNs per track)).
0018'CF 50 C7 OBD6 2517
OBD6 2518 MOVZBL UCB$B_DU_RBNPTRK(R3),R0 ; R0 = RBNs per track.
50 0000'C3 9A OBE1 2519 MULL R1,R0 ; R0 = (RBNs per)*QUO(LBN/(LBNs per))
50 51 C4 OBE6 2520 CLRL R1 ; Clear high order part of dividend.
51 51 D4 OBE9 2521
OBD6 2522 EDIV #128,R0,- ; Divide result by 128 giving the
50 0000080 8F 7B OBE8 2523 HIRT$L_RCTBLOCK,- ; quotient and the
006C'CF 0068'CF OBF2 2524 HIRT$L_OFFSET ; remainder.
OBD6 2525
OBD6 2526 ADDL #2,HIRT$L_RCTBLOCK ; Add in sector 0 and sector 1.
0068'CF 02 C0 OBF8 2527 RSB ; Return to caller
05 OBF0

```

```

OBFE 2529 .SBTTL DUSHIR_ERROR - Process error encountered during HIRT processing
OBFE 2530 :++
OBFE 2531 :
OBFE 2532 : DUSHIR_ERROR - Process error encountered during HIRT processing
OBFE 2533 :
OBFE 2534 : Functional Description:
OBFE 2535 :
OBFE 2536 : This routine performs any operations necessary to inform the world
OBFE 2537 : that an error was encountered during HIRT processing. It is invoked
OBFE 2538 : via the HIR_ERROR macro.
OBFE 2539 :
OBFE 2540 : Currently, the error processing consists of broadcasting a message to
OBFE 2541 : OPA0. The general text of the error message is:
OBFE 2542 :
OBFE 2543 : %<devnam> encountered a <type> error in <func> step <n>
OBFE 2544 :
OBFE 2545 : Where:
OBFE 2546 :
OBFE 2547 : <devnam> is the device name
OBFE 2548 : <type> is one of READ / WRITE / RCT FULL
OBFE 2549 : <func> is one of REPLACE / ONLINE
OBFE 2550 : <n> is a number giving the failing step in the
OBFE 2551 : replacement algorithm
OBFE 2552 :
OBFE 2553 : Inputs:
OBFE 2554 :
OBFE 2555 : R0 A parameter giving the <type>, <func>, and <n> values above
OBFE 2556 : R3 UCB address
OBFE 2557 :
OBFE 2558 : Outputs:
OBFE 2559 :
OBFE 2560 : R0 is destroyed
OBFE 2561 : All other registers are preserved.
OBFE 2562 :--
OBFE 2563 :
OBFE 2564 : DUSHIR_ERROR:
OBFE 2565 :
OBFE 2566 : PUSH R1, R2, R3, R4, R5, R6, R7 ; Save some registers.
OBFE 2567 : MOVAB -HIRERR$K_MSGSIZE(SP), SP ; Make message space on stack.
OBFE 2568 : MOVL SP, R6 ; Save base of message space.
OBFE 2569 : MOVL R0, R7 ; Copy error parameter.
OBFE 2570 :
OBFE 2571 : ; Form device name.
OBFE 2572 : MOVL R3, R5 ; Move UCB address.
OBFE 2573 : MOVL R6, R1 ; Setup buffer address.
OBFE 2574 : MOVB #^A/%/, (R1)+ ; Insert percent sign.
OBFE 2575 : MOVL #HIRERR$K_DEVNAMSIZ, R0 ; Setup buffer size.
OBFE 2576 : MNEGL #1, R4 ; Setup formation code.
OBFE 2577 : JSB G^IO$CVT_DEVNAM ; Get device name for UCB.
OBFE 2578 : ADDL3 R1, R6, R3 ; Init working buffer pointer.
OBFE 2579 : INCL R3 ; Adjust for percent sign.
OBFE 2580 :
OBFE 2581 : ; Copy first fixed segment.
OBFE 2582 : MOVAB HIR_ERR_SEG1, R1 ; Get string address.
OBFE 2583 : BSBB COPY_ASCII ; Copy string.
OBFE 2584 :
OBFE 2585 : ; Insert proper <type> segment.

```

```
52 57 04 08 EF 0C2E 2586 EXTZV #HIRERSV_TYPE, #HIRERS$ _TYPE, R7, R2 ; Get type number.
51 0000'CF 9E 0C33 2587 MOVAB HIR_ERR_TYPES, R1 ; Get error types strings base.
50 81 9A 0C38 2588 32$: MOVZBL (R1)+, R0 ; Get length of this message.
51 50 C0 0C3B 2589 ADDL R0, R1 ; Point to next message.
F7 52 F5 0C3E 2590 SOBGTR R2, 32$ ; Loop till message located.
53 10 0C41 2591 BSBB COPY_ASCII ; Copy <type> string.
0C43 2592
0C43 2593 ; Copy second fixed segment.
51 0044'CF 9E 0C43 2594 MOVAB HIR_ERR_SEG2, R1 ; Get string address.
4C 10 0C48 2595 BSBB COPY_ASCII ; Copy string.
0C4A 2596
0C4A 2597 ; Insert proper <func> segment.
51 0025'CF 9E 0C4A 2598 MOVAB HIR_ERR_REPLACE, R1 ; Assume REPLACE.
05 57 0C E1 0C4F 2599 BBC #HIRERSV_ONLINE, R7, 45$ ; Branch if not ONLINE.
51 002D'CF 9E 0C53 2600 MOVAB HIR_ERR_ONLINE, R1 ; Else, get ONLINE.
3C 10 0C58 2601 45$: BSBB COPY_ASCII ; Copy <func> string.
0C5A 2602
0C5A 2603 ; Copy third fixed segment.
51 004F'CF 9E 0C5A 2604 MOVAB HIR_ERR_SEG3, R1 ; Get string address.
35 10 0C5F 2605 BSBB COPY_ASCII ; Copy string.
0C61 2606
0C61 2607 ; Convert two digits of <n> and insert them.
0C61 2608 ASSUME HIRERSV_STEP EQ 0
0C61 2609 ASSUME HIRERS$ _STEP EQ 8
50 50 57 9A 0C61 2610 MOVZBL R7, R0 ; Get step number.
50 63 8F 91 0C64 2611 CMPB #99, R0 ; Is number to big?
OF 15 0C68 2612 BLEQ 60$ ; Branch if number to big.
51 51 D4 0C6A 2613 CLRL R1 ; Quadword extend number.
50 50 0A 7B 0C6C 2614 EDIV #10, R0, R0, R1 ; Split digits.
83 50 30 81 0C71 2615 ADDB3 #^A/O/, R0, (R3)+ ; Insert tens digit.
83 51 30 81 0C75 2616 ADDB3 #^A/O/, R1, (R3)+ ; Insert units digit.
0C79 2617
0C79 2618 60$: ; Compute message size and broadcast message to OPA0:.
51 52 56 D0 0C79 2619 MOVL R6, R2 ; Setup base message address.
53 52 C3 0C7C 2620 SUBL3 R2, R3, R1 ; Setup message size.
00000000'GF 9E 0C80 2621 MOVAB G^OPA$UCB0, R5 ; Get OPA0 UCB address.
00000000'GF 16 0C87 2622 JSB G^IOC$BROADCAST ; Broadcast message.
0C8D 2623
5E 4B AE 9E 0C8D 2624 MOVAB HIRERSK_MSGSIZE(SP), SP ; Clear message from stack.
00FE 8F BA 0C91 2625 POPR #^M<R1,R2,R3,R4,R5,R6,R7> ; Restore saved registers.
05 0C95 2626 RSB ; Exit.
0C96 2627
0C96 2628 :++
0C96 2629 :
0C96 2630 : Routine to copy ASCII string to buffer.
0C96 2631 :
0C96 2632 : Inputs:
0C96 2633 :
0C96 2634 : R1 ASCII string address
0C96 2635 : R3 buffer address
0C96 2636 :
0C96 2637 : Outputs:
0C96 2638 :
0C96 2639 : R3 updated buffer address (complements of MOV(C3))
0C96 2640 :
0C96 2641 : R0 through R5 are altered.
0C96 2642 : All other registers are preserved.
```

```

        0C96 2643 :--
        0C96 2644
        0C96 2645 COPY_ASCII:
63      50      81      9A 0C96 2646      MOVZBL (R1)+, R0      ; Get string size.
        61      50      28 0C99 2647      MOVC3  R0, (R1), (R3) ; Copy string.
        05      0C9D 2648      RSB
        0C9E 2649
        0C9E 2650
        0C9E 2651
        0C9E 2652      .END

```

DUHIRT
Symbol table

C 1
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 59
(23)

DU
VO

ALLOC_POOL	000000B4	R	02	CDRPSL TT TERM	=	FFFFFFDC		
ATE_MSCPCODE	00000002			CDRPSL_UBARSRCE	=	0000003C		
ATE_OFFSET	00000000			CDRPSL_UCB	=	FFFFFFBC		
ATE_SSCODE	00000003			CDRPSL_WIND	=	FFFFFFB8		
BIT...	= 0000000D			CDRPSM_ERLIP	=	00000004		
BUG\$ DISKCLASS	*****	X	02	CDRPSM_HIRT	=	00000010		
BUILD_RCT_PACKET	00000A55	R	02	CDRPSQ_NT_PRVMSK	=	FFFFFFE0		
CDDBSA_2PFKB	00000174			CDRPSL_LBUFHNDL	=	00000030		
CDDBSA_DAPCDRP	00000194			CDRPSV_PERM	=	00000003		
CDDBSA_DAPIRP	00000134			CDRPSW_ABCNT	=	FFFFFFE0		
CDDBSA_PRCMDRP	000000D0			CDRPSW_BCNT	=	FFFFFFD2		
CDDBSA_PRMIRP	00000070			CDRPSW_BOFF	=	FFFFFFD0		
CDDSSB_SUBTYPE	= 0000000B			CDRPSW_CDRPSIZE	=	00000008		
CDDSSB_TYPE	= 0000000A			CDRPSW_CHAN	=	FFFFFFC8		
CDDBSK_LENGTH	= 00000070			CDRPSW_DUTUCNTR	=	00000044		
CDDBSL_CANCLOBL	000000B4			CDRPSW_FUNC	=	FFFFFFC0		
CDDBSL_CANCLOFL	000000B0			CDRPSW_IRP_SIZE	=	FFFFFFA8		
CDDBSL_CDT	000000F4			CDRPSW_OBCNT	=	FFFFFFE4		
CDDBSL_DAPCDT	000001B8			CDRPSW_STS	=	FFFFFFCA		
CDDBSL_DAPUCB	00000150			COPY_ASCIC		00000C96	R	02
CDDBSL_PRMUCB	0000008C			DEVNAMSIZ	=	00000012		
CDDBSL_SAVED_PC	= 00000044			DUSCANCEL FROM HIRT		00000229	RG	02
CDDBSW_SIZE	= 00000008			DUSDISCONNECT_HIRT		000002A9	RG	02
CDRPSB_CARCON	= FFFFFFFDC			DUSHIR_ERROR		00000BFE	R	02
CDRPSB_CD_TYPE	= 0000000A			DUSINIT_HIRT		00000000	RG	02
CDRPSB_EFN	= FFFFFFFC2			DUSLOCK_HIRT		000000EF	RG	02
CDRPSB_IRP_TYPE	= FFFFFFFAA			DUSONLINE_COMPLETE		0000080E	RG	02
CDRPSB_PRI	= FFFFFFFC3			DUSREPLACE_LBN		0000031D	RG	02
CDRPSB_RMOD	= FFFFFFFAB			DUSRSTRQ_HIRT_CDRP		000002E0	RG	02
CDRPSL_ABCNT	= FFFFFFFE0			DUSTEST_HIRT_RWAITCNT		00000205	RG	02
CDRPSL_ARB	= FFFFFFFF8			DUSUNLOCK_HIRT		00000178	RG	02
CDRPSL_AST	= FFFFFFFB0			DUTUSINIT_MSCP_MSG_UNIT		*****	X	02
CDRPSL_ASTPRM	= FFFFFFFB4			DUTUSINSERT_RESTARTQ		*****	X	02
CDRPSL_BCNT	= FFFFFFFD2			DUTUSKILL_TRIS_THREAD		*****	X	02
CDRPSL_CDT	= 00000024			DUTUSL_CDDB_LISTHEAD		00000000		
CDRPSL_DIAGBUF	= FFFFFFFEC			DUTUSPOST_CDRP		*****	X	02
CDRPSL_DUTUFLAGS	= 00000040			DUTUSSEND_MSCP_MSG		*****	X	02
CDRPSL_EXTEND	= FFFFFFFF4			DUTUSTEST_CANCEL_CDRP		*****	X	02
CDRPSL_FR3	= 00000010			DYN\$C_CDRP	=	00000039		
CDRPSL_FR4	= 00000014			DYN\$C_CD_BBRPG	=	00000002		
CDRPSL_IOQBL	= FFFFFFFA4			DYN\$C_CLASSDRV	=	00000064		
CDRPSL_IOQFL	= FFFFFFFA0			DYN\$C_IRP	=	0000000A		
CDRPSL_IOSB	= FFFFFFFC4			END_INIT_HIRT		000000B1	R	02
CDRPSL_IOST1	= FFFFFFFD8			EXESALONONPAGED		*****	X	02
CDRPSL_IOST2	= FFFFFFFDC			EXESFORK_WAIT		*****	X	02
CDRPSL_JNL_SEQNO	= FFFFFFFE8			FILL_RCT_PACKET		00000A7C	R	02
CDRPSL_LBUFH_AD	= 0000002C			FIXEDSIZ	=	00000023		
CDRPSL_MEDIA	= FFFFFFFD8			FKBSK_LENGTH	=	00000018		
CDRPSL_MSG_BUF	= 0000001C			FKBSL_FPC	=	0000000C		
CDRPSL_OBCNT	= FFFFFFFE4			FKBSL_FQBL	=	00000004		
CDRPSL_PID	= FFFFFFFAC			FKBSL_FQFL	=	00000000		
CDRPSL_RSPID	= 00000020			FKBSL_FR3	=	00000010		
CDRPSL_RWCPTX	= 00000028			FUNCSTZ	=	00000007		
CDRPSL_SAVD_RTN	= 00000018			GRANT_HIRT		00000120	R	02
CDRPSL_SEGVBN	= FFFFFFFE8			HASH_CBN		00000BD6	R	02
CDRPSL_SEQNUM	= FFFFFFFF0			HIRERSK_DEVNAMSIZ	=	00000012		
CDRPSL_SVAPTE	= FFFFFFFFC			HIRERSK_MSGSIZE	=	00000048		

DUHIRT
Symbol table

D 1
HOST INITIATED REPLACEMENT FOR THE DISK 16-SEP-1984 00:58:58 VAX/VMS Macro V04-00
5-SEP-1984 00:13:32 [DRIVER.SRC]DUHIRT.MAR;1

Page 60
(23)

DU
VO

HIRERSK_RCTFULL	=	00000003		HIRTSW_BOFF2	00000048	R	03
HIRERSK_READ	=	00000001		HIRTSW_BOFF3	0000004A	R	03
HIRERSK_REPFALL	=	00000004		HIRTSW_IOST	00000008	R	03
HIRERSK_WRITE	=	00000002		HIRTSW_IOWORST	0000000A	R	03
HIRERSM_ONLINE	=	00001000		HIRTSW_PAGENO	00000056	R	03
HIRERSM_REPLACE	=	00000000		HIRTSW_PG0CNTNT	0000004C	R	03
HIRERSM_STEP	=	000000FF		HIRTSW_PG1CNTNT	0000004E	R	03
HIRERSM_TYPE	=	00000F00		HIRTSW_PG2CNTNT	00000050	R	03
HIRERSS_STEP	=	00000008		HIRTSW_PG3CNTNT	00000052	R	03
HIRERSS_TYPE	=	00000004		HIRTSW_SECTORNO	00000054	R	03
HIRERSV_ONLINE	=	0000000C		HIRTSW_STS	0000000E	R	03
HIRERSV_STEP	=	00000000		HIR_ERR_ONLINE	0000002D	R	04
HIRERSV_TYPE	=	00000008		HIR_ERR_REPLACE	00000025	R	04
HIRTSK_SUBSTKLN	=	00000005		HIR_ERR_SEG1	00000034	R	04
HIRTSL_BADRBN		00000060	R 03	HIR_ERR_SEG2	00000044	R	04
HIRTSL_CDRP		00000020	R 03	HIR_ERR_SEG3	0000004F	R	04
HIRTSL_LBN		00000018	R 03	HIR_ERR_TYPES	00000000	R	04
HIRTSL_LOOPCNT		00000010	R 03	IOCSBROADCAST	*****	X	02
HIRTSL_MATCHRBN		0000005C	R 03	IOCSVT_DEVNAM	*****	X	02
HIRTSL_OFFSET		0000006C	R 03	IRPSB_CARCON	= 0000003C		
HIRTSL_OWNUCB		00000014	R 03	IRPSB_EFN	= 00000022		
HIRTSL_PAGEOPTR		00000024	R 03	IRPSB_PRI	= 00000023		
HIRTSL_PAGE1PTR		00000028	R 03	IRPSB_RMOD	= 0000000B		
HIRTSL_PAGE2PTR		0000002C	R 03	IRPSB_TYPE	= 0000000A		
HIRTSL_PAGE3PTR		00000030	R 03	IRPSK_LENGTH	= 000000C4		
HIRTSL_RBN		00000058	R 03	IRPSL_ABCNT	= 00000040		
HIRTSL_RCTBLOCK		00000068	R 03	IRPSL_ARB	= 00000058		
HIRTSL_RPLQFL		00000000	R 03	IRPSL_AST	= 00000010		
HIRTSL_RPLQTP		00000004	R 03	IRPSL_ASTPRM	= 00000014		
HIRTSL_SAVDCDRP		0000001C	R 03	IRPSL_BCNT	= 00000032		
HIRTSL_STARTBLK		00000064	R 03	IRPSL_CDT	= 00000084		
HIRTSL_STKPTR		00000070	R 03	IRPSL_DIAGBUF	= 0000004C		
HIRTSL_SUBSTACK		00000074	R 03	IRPSL_EXTEND	= 00000054		
HIRTSL_SWAPTE0		00000034	R 03	IRPSL_FQFL	= 00000060		
HIRTSL_SWAPTE1		00000038	R 03	IRPSL_IQBL	= 00000004		
HIRTSL_SWAPTE2		0000003C	R 03	IRPSL_IQFL	= 00000000		
HIRTSL_SWAPTE3		00000040	R 03	IRPSL_IOSB	= 00000024		
HIRTSM_ACTIVE	=	00000001		IRPSL_IOST1	= 00000038		
HIRTSM_BUSY	=	00000002		IRPSL_IOST2	= 0000003C		
HIRTSM_EMPTYTYPE	=	00000010		IRPSL_JNL_SEQNO	= 00000048		
HIRTSM_ERLOGIP	=	00000080		IRPSL_MEDIA	= 00000038		
HIRTSM_FE	=	00000004		IRPSL_OBCNT	= 00000044		
HIRTSM_MATCH	=	00000008		IRPSL_PID	= 0000000C		
HIRTSM_RCTFE	=	00000100		IRPSL_SEGVBN	= 00000048		
HIRTSM_RCTFULL	=	00000040		IRPSL_SEQNUM	= 00000050		
HIRTSM_RESCAN	=	00000020		IRPSL_SWAPTE	= 0000002C		
HIRTSV_ACTIVE	=	00000000		IRPSL_TT_TERM	= 0000003C		
HIRTSV_BUSY	=	00000001		IRPSL_UCB	= 0000001C		
HIRTSV_EMPTYTYPE	=	00000004		IRPSL_WIND	= 0000C018		
HIRTSV_ERLOGIP	=	00000007		IRPSQ_NT_PRVMSK	= 00000040		
HIRTSV_FE	=	00000002		IRPSW_ABCNT	= 00000040		
HIRTSV_MATCH	=	00000003		IRPSW_BCNT	= 00000032		
HIRTSV_RCTFE	=	00000008		IRPSW_BOFF	= 00000030		
HIRTSV_RCTFULL	=	00000006		IRPSW_CHAN	= 00000028		
HIRTSV_RESCAN	=	00000005		IRPSW_FUNC	= 00000020		
HIRTSW_BOFF0		00000044	R 03	IRPSW_OBCNT	= 00000044		
HIRTSW_BOFF1		00000046	R 03	IRPSW_SIZE	= 00000008		

IRPSW_STS	= 0000002A			READ_RCT_BLOCK	000009C3	R	02
MAP_PAGE	00000AAB	R	02	REPLACE_CONNECT_FAILURE	0000080B	R	02
MMG\$GL_SPTBASE	*****	X	02	SCSS\$DEAL_RSPID	*****	X	02
MSCPSB_BUFFER	= 00000010			SCSS\$FIND_RDT	*****	X	02
MSCPSB_FLAGS	= 00000009			SCSS\$UNSTALLUCB	*****	X	02
MSCPSB_OPCODE	= 00000008			SEARCH_FAIL	00000B72	R	02
MSCPSK_OP_READ	= 00000021			SEARCH_RCT	00000ADB	R	02
MSCPSK_OP_REPLC	= 00000014			SEARCH_RTN	00000B7B	R	02
MSCPSK_OP_WRITE	= 00000022			SIZ...	= 00000001		
MSCPSK_ST_DATA	= 00000008			SIZE	= 00000045		
MSCPSK_ST_SUCC	= 00000000			SS\$_BADRCT	= 0000216C		
MSCPSK_ST_WRTPR	= 00000006			SS\$_CANCEL	= 00000830		
MSCPSL_BYTE_CNT	= 0000000C			SS\$_NORMAL	= 00000001		
MSCPSL_CMD_REF	= 00000000			SS\$_WRITLCK	= 0000025C		
MSCPSL_FRST_BAD	= 0000001C			STEP10	00000526	R	02
MSCPSL_LBN	= 0000001C			STEP11	00000562	R	02
MSCPSL_RBN	= 0000000C			STEP12	00000609	R	02
MSCPSM_MD_COMP	= 00004000			STEP13	00000680	R	02
MSCPSM_MD_ERROR	= 00001000			STEP14	000006D4	R	02
MSCPSM_MD_EXPRS	= 00008000			STEP15_A	000006EF	R	02
MSCPSM_MD_PRMR	= 00000001			STEP15_B	00000717	R	02
MSCPSM_MD_SECOR	= 00000200			STEP16	00000783	R	02
MSCPSM_MD_SEREC	= 00000100			STEP17	000007C5	R	02
MSCPSM_ST_MASK	= 0000001F			STEP18	000007E6	R	02
MSCPSS_ST_MASK	= 00000005			STEP5	00000399	R	02
MSCPSV_EF_BBLKR	= 00000007			STEP6	000003B2	R	02
MSCPSV_EF_ERLOG	= 00000005			STEP7	000003FB	R	02
MSCPSV_ST_MASK	= 00000000			STEP8	000004B4	R	02
MSCPSW_MODIFIER	= 0000000A			STEP9	000004FE	R	02
MSCPSW_STATUS	= 0000000A			STEPSIZ	= 00000002		
NEXT	00000B2D	R	02	TEST_PATTERN	= B6DBCB6D		
OPASUCBO	*****	X	02	TEST_RCT_ENTRY	00000885	R	02
PDTSL_ALLOCMG	= 00000014			TYPSTZ	= 00000007		
PDTSL_DEALLOMSG	= 00000020			UCBSB_DU_RBNPTRK	*****	X	02
PDTSL_MAPIRP	= 00000034			UCBSB_DU_RCTCPYS	*****	X	02
PDTSL_RCHMSGBUF	= 00000044			UCBSL_CDDB	= 000000BC		
PDTSL_UNMAP	= 00000064			UCBSL_CDT	= 000000C8		
RCTSL_BAD_RBN	= 00000014			UCBSL_DU_USIZE	*****	X	02
RCTSL_LBN	= 0000000C			UCBSV_MSCP_WRTP	= 0000000D		
RCTSL_RBN	= 00000010			UCBSW_DEVSTS	= 00000068		
RCTSM_ALLOCATED	= 20000000			UCBSW_DU_LBNPTRK	*****	X	02
RCTSM_BR	= 00002000			UCBSW_DU_RCTSIZE	*****	X	02
RCTSM_FE	= 00000080			UCBSW_RWAITCNT	= 00000056		
RCTSM_NONPRIME	= 00000000			VAS_VPN	= 00000015		
RCTSM_RP1	= 00008000			VASV_VPN	= 00000009		
RCTSM_RP2	= 00004000			WRITE_RCT_BLOCK	00000909	R	02
RCTSM_UNUSABLE	= 40000000						
RCTSS_LBN	= 0000001C						
RCTSV_ALLOCATED	= 0000001D						
RCTSV_BR	= 0000000D						
RCTSV_FE	= 00000007						
RCTSV_LBN	= 00000000						
RCTSV_NULL	= 0000001F						
RCTSV_RP1	= 0000000F						
RCTSV_RP2	= 0000000E						
RCTSW_FLAGS	= 00000008						
RDSL_CDRP	= 00000000						

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	000001F8 (504.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000C9E (3230.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$300_HIRT	00000088 (136.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$301_HIR_ERRORS	00000056 (86.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.33
Command processing	139	00:00:00.44	00:00:02.55
Pass 1	724	00:00:22.06	00:01:13.75
Symbol table sort	0	00:00:03.17	00:00:09.39
Pass 2	407	00:00:05.60	00:00:41.25
Symbol table output	1	00:00:00.20	00:00:00.47
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1310	00:00:31.58	00:02:07.79

The working set limit was 2700 pages.
187469 bytes (367 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2953 non-local and 117 local symbols.
2652 source lines were read in Pass 1, producing 29 object records in Pass 2.
61 pages of virtual memory were used to define 59 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[DRIVER.OBJ]DUTULIB.MLB;1	8
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	31
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	48

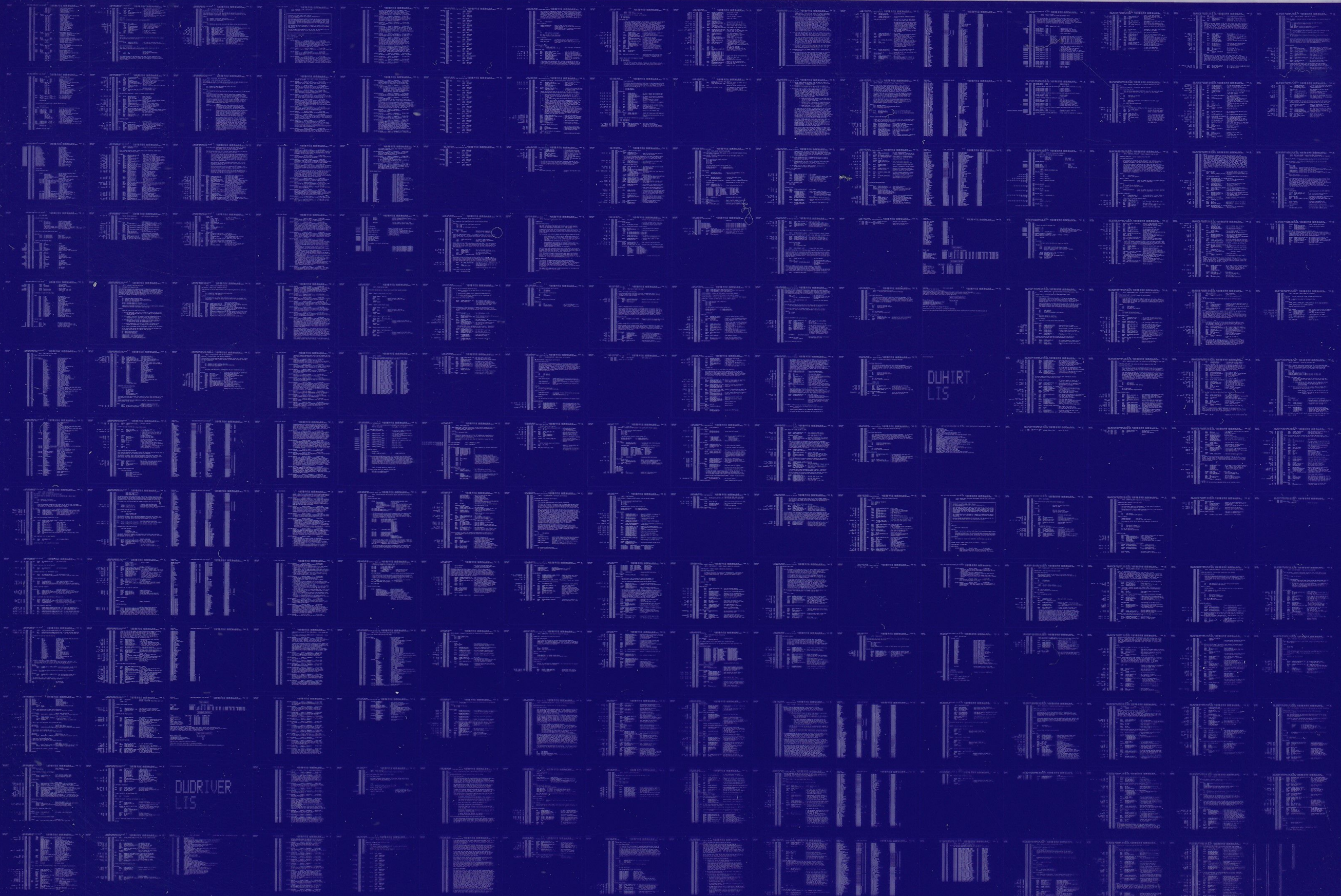
3143 GETS were required to define 48 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DUHIRT/OBJ=OBJ\$:DUHIRT MSRC\$:DUHIRT/UPDATE=(ENH\$:DUHIRT)+EXECMLS/LIB+LIB\$:DUTULIB/LIB

0110 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0111 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

