


```

IIIIII  MM      MM      GGGGGGGG  DDDDDDDD  EEEEEEEEEE  CCCCCCCC  OOOOOO  DDDDDDDD  EEEEEEEEEE
IIIIII  MM      MM      GGGGGGGG  DDDDDDDD  EEEEEEEEEE  CCCCCCCC  OOOOOO  DDDDDDDD  EEEEEEEEEE
  II     MMMM  MMMM  GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MMMM  MMMM  GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG          DD          DD     EEEEEEEEE  CC          OO          OO  DD          DD  EEEEEEEEE
  II     MM   MM   MM   GG          DD          DD     EEEEEEEEE  CC          OO          OO  DD          DD  EEEEEEEEE
  II     MM   MM   MM   GG  GGGGGG  DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG  GGGGGG  DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG          GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
  II     MM   MM   MM   GG          GG          DD          DD     EE          CC          OO          OO  DD          DD  FF          FF
IIIIII  MM      MM      GGGGGG  DDDDDDDD  EEEEEEEEEE  CCCCCCCC  OOOOOO  DDDDDDDD  EEEEEEEEEE
IIIIII  MM      MM      GGGGGG  DDDDDDDD  EEEEEEEEEE  CCCCCCCC  OOOOOO  DDDDDDDD  EEEEEEEEEE

```

....
....
....
....

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLLLL IIIIII  SSSSSSSS

```



```

1 0001 0 XTITLE 'Get and Decode Image Header and Sections'
2 0002 0 MODULE IMG$DECODE (
3 0003 0     LANGUAGE (BLISS32),
4 0004 0     IDENT = 'V04-000'
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1
9 0009 1 *****
10 0010 1 *
11 0011 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 *  ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 *  TRANSFERRED.
21 0021 1 *
22 0022 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 *  CORPORATION.
25 0025 1 *
26 0026 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1
32 0032 1
33 0033 1 **
34 0034 1 FACILITY:      Exec, Shareable routines to decode image header and sections
35 0035 1
36 0036 1 ABSTRACT:
37 0037 1
38 0038 1     This module contains the routines to retrieve and decode
39 0039 1     an image header and the image section descriptors.
40 0040 1
41 0041 1 ENVIRONMENT:  VAX/VMS Operating System
42 0042 1
43 0043 1 AUTHOR:      Bob Grosso,      CREATION DATE: 16-Mar-1983
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1     V03-010 MSH0051      Michael S. Harvey      20-May-1984
48 0048 1     Convert old format image name string to new format.
49 0049 1
50 0050 1     V03-009 MSH0043      Michael S. Harvey      8-May-1984
51 0051 1     When converting x-linker image headers into a modern
52 0052 1     form, update the image IDs correspondingly.
53 0053 1
54 0054 1     V03-008 MSH0041      Michael S. Harvey      2-May-1984
55 0055 1     Add some beef to the bounds checking code to ensure that
56 0056 1     only valid images are run. These checks filter obviously
57 0057 1     bad image headers and images with bad ISD lists.

```

58	0058	1	
59	0059	1	
60	0060	1	V03-007 LJK0269 Lawrence J. Kenah 31-Mar-1984
61	0061	1	Miscellaneous cleanup.
62	0062	1	Do not perform consistence checks on TYPE 2 images. They are
63	0063	1	not necessarily produced by the linker.
64	0064	1	Make sure that a primitive length check is performed on the
65	0065	1	IHD and ISD sizes before the buffer is copied.
66	0066	1	
67	0067	1	V03-006 LJA0110 Laurie J. Anderson 6-Feb-1984
68	0068	1	Change the error messages returned from the image decode
69	0069	1	routines to be something more intelligent than "bad hdr".
70	0070	1	
71	0071	1	V03-005 WMC0001 Wayne Cardoza 24-Jan-1984
72	0072	1	Add support for cross-linker and V3 FT1 images.
73	0073	1	
74	0074	1	V03-004 LJK0243 Lawrence J. Kenah 23-Aug-1983
75	0075	1	Return IHDSQ_PRIVREQS of all privileges for old images,
76	0076	1	ones that do not contain a SYSVER field.
77	0077	1	
78	0078	1	V03-003 LJK0234 Lawrence J. Kenah 26-Jul-1983
79	0079	1	Fix code that transforms old image header into latest
80	0080	1	form of image header.
81	0081	1	
82	0082	1	V03-002 LJK0229 Lawrence J. Kenah 12-Jul-1983
83	0083	1	Treat the alias and offset as words. Treat the ISD
84	0084	1	size as a signed word.
85	0085	1	
86	0086	1	V03-001 LJK0223 Lawrence J. Kenah 6-Jul-1983
87	0087	1	Make IHD and ISD sizes into words so that the comparisons
88	0088	1	are made correctly.

```

: 90 0089 1 %SBTTL 'Definitions'
: 91 0090 1
: 92 0091 1
: 93 0092 1 !! INCLUDE FILES:
: 94 0093 1
: 95 0094 1
: 96 0095 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';           ! Define system data structures
: 97 0096 1
: 98 0097 1 REQUIRE 'LIB$:IMGMSGDEF.R32';           ! Get status code definitions
: 99 0183 1
100 0184 1
101 0185 1 !! PSECT DECLARATIONS:
102 0186 1
103 0187 1
104 0188 1 PSECT
105 0189 1     CODE      = YF$$$SYSIMGACT (WRITE),
106 0190 1     PLIT     = YF$$$SYSIMGACT (WRITE, 'EXECUTE');
107 0191 1
108 0192 1
109 0193 1 !! LITERALS
110 0194 1
111 0195 1
112 0196 1 LITERAL
113 0197 1     TRUE = 1,
114 0198 1     FALSE = 0,
115 0199 1
116 0200 1     IMGSC_BLOCKSIZ = 512;
117 0201 1
118 0202 1
119 0203 1 !! EXTERNAL REFERENCES:
120 0204 1
121 0205 1
122 0206 1 EXTERNAL LITERAL
123 0207 1
124 0208 1     EXESC_SYSEFN : UNSIGNED (6);           ! System event flag for QIO Wait read
125 0209 1
126 0210 1
127 0211 1 !! FORWARD ROUTINE REFERENCES
128 0212 1
129 0213 1
130 0214 1 FORWARD ROUTINE
131 0215 1     CONVERT_XLINK;
132 0216 1
133 0217 1
134 0218 1 !! Define VMS block structures
135 0219 1
136 0220 1 STRUCTURE
137 0221 1     BBLOCK [O, P, S, E; N] =
138 0222 1         [N]
139 0223 1         (BBLOCK + O) <P, S, E>;
: 140 0224 1

```

```

142 0225 1 XSBTTL 'IMG$DECODE_IHD Get Image Header'
143 0226 1 GLOBAL ROUTINE IMG$DECODE_IHD
144 0227 1 ( CHAN, BLK_BUFADR, IHD_BUFADR, VBN_ADR, OFFSET_ADR, HDRVER_ADR, ALIAS_ADR ) =
145 0228 1
146 0229 1 !++
147 0230 1 ! FUNCTIONAL DESCRIPTION:
148 0231 1 !
149 0232 1 ! FORMAL PARAMETERS:
150 0233 1 !
151 0234 1 ! Chan Channel on which image file is open
152 0235 1 ! Blk_bufadr Address of buffer to contain 1st block of image
153 0236 1 ! Ihd_bufadr Address of buffer to contain decoded IHD
154 0237 1 ! VBN_adr Address of VBN to be set to 1
155 0238 1 ! Offset_adr Address of Offset in which to return offset to 1st ISD
156 0239 1 !
157 0240 1 ! IMPLICIT INPUTS:
158 0241 1 ! NONE
159 0242 1 !
160 0243 1 ! IMPLICIT OUTPUTS:
161 0244 1 ! NONE
162 0245 1 !
163 0246 1 ! ROUTINE VALUE:
164 0247 1 ! COMPLETION CODES:
165 0248 1 ! NONE
166 0249 1 !
167 0250 1 ! SIDE EFFECTS:
168 0251 1 ! NONE
169 0252 1 !
170 0253 1 ! --
171 0254 1 !
172 0255 2 BEGIN
173 0256 2
174 0257 2 LITERAL
175 0258 2 IHDMAXSIZ = IHD$C_LENGTH + ! Maximum length for fixed portion of header
176 0259 2 IHASC_LENGTH +
177 0260 2 IHSSC_LENGTH +
178 0261 2 IHISC_LENGTH +
179 0262 2 IHPS_C_LENGTH ,
180 0263 2 IHI_S_IMGNAM = 16; ! Length of image name string prior to VMS V4
181 0264 2
182 0265 2 LOCAL
183 0266 2 B_IHD : REF BBLOCK, ! Buffer IHD
184 0267 2 IHD : REF BBLOCK,
185 0268 2 !OSB : BBLOCK [8], ! Quadword IO status block
186 0269 2 HDR_INSERT,
187 0270 2 IHI_INSERT,
188 0271 2 OFF2 : WORD,
189 0272 2 SIZE,
190 0273 2 STATUS; ! Status
191 0274 2
192 0275 2 BIND
193 0276 2 V4_MAJORID = UPLIT (XASCII'02'), ! Major ID for VMS V4 images
194 0277 2 V4_MINORID = UPLIT (XASCII'05'), ! Minor ID for VMS V4 images
195 0278 2 HEADER_VERSION = .HDRVER_ADR : WORD,
196 0279 2 LAST_WORD = .ALIAS_ADR : SIGNED WORD,
197 0280 2 OFFSET = .OFFSET_ADR : WORD,
198 0281 2 VBN = .VBN_ADR;

```



```

313 0396 3
314 0397 3
315 0398 3
316 0399 3
317 0400 3
318 0401 3
319 0402 3
320 0403 3
321 0404 3
322 0405 3
323 0406 3
324 0407 3
325 0408 3
326 0409 3
327 0410 3
328 0411 3
329 0412 4
330 0413 4
331 0414 4
332 0415 3
333 0416 3
334 0417 3
335 0418 3
336 0419 4
337 0420 4
338 0421 4
339 0422 4
340 0423 4
341 0424 4
342 0425 4
343 0426 4
344 0427 4
345 0428 3
346 0429 3
347 0430 3
348 0431 3
349 0432 4
350 0433 4
351 0434 4
352 0435 3
353 0436 3
354 0437 3
355 0438 3
356 0439 3
357 0440 4
358 0441 4
359 0442 4
360 0443 4
361 0444 4
362 0445 4
363 0446 4
364 0447 4
365 0448 4
366 0449 3
367 0450 3
368 0451 3
369 0452 3
!
! CHSMOVE (.B_IHD [IHD$W_SIZE], .B_IHD, .IHD_BUFADR);
!
! HDR_INSERT = 0; ! Length by which header will be pried open
! HEADER_VERSION = IHD$C_GEN_FIXUP; ! Default to most current
! IHD = .IHD_BUFADR;
!
! Calculate the degree by which the fixed portion of this header
! differs from the current format of the fixed part of an image header.
! Then, expand the fixed portion of the header by the required amount,
! thus converting it to the current format as if the image had been
! relinked.
!
! IF $BYTEOFFSET (IHD$L_LNKFLAGS) GEQ .IHD [IHD$W_ACTIVOFF]
! THEN ! Link flags were not present
! BEGIN ! so insert a longword
! HDR_INSERT = 4;
! HEADER_VERSION = IHD$C_GEN_NATIVE;
! END;
!
! IF $BYTEOFFSET (IHD$L_SYSVER) GEQ .IHD [IHD$W_ACTIVOFF]
! THEN ! System version and Ident were not present
! BEGIN ! so insert two blank longwords
! BIND
! PRIVILEGE_MASK = IHD [IHD$Q_PRIVREQS] : VECTOR [2];
!
! HDR_INSERT = .HDR_INSERT + 8;
! HEADER_VERSION = IHD$C_GEN_LNKFLG;
! PRIVILEGE_MASK [0] = -1; ! Insure that image privilege mask
! PRIVILEGE_MASK [1] = -1; ! indicates that all privileges are set
! END;
!
! IF $BYTEOFFSET (IHD$L_IAFVA) GEQ .IHD [IHD$W_ACTIVOFF]
! THEN ! Relative virtual address of fixup vector
! BEGIN ! not present so insert a blank longword
! HDR_INSERT = .HDR_INSERT + 4;
! HEADER_VERSION = IHD$C_GEN_SYSVER;
! END;
!
! IF .HDR_INSERT NEQ 0
! THEN ! Shift non-fixed portion of image
! BEGIN ! to insert missing part of fixed section
! CHSMOVE (
! (.IHD [IHD$W_SIZE] - .IHD [IHD$W_ACTIVOFF]), ! Shift the portion beginning at the
! (.IHD + .IHD [IHD$W_ACTIVOFF]), ! point located by the first offset
! (.IHD + .IHD [IHD$W_ACTIVOFF] + .HDR_INSERT)); ! by the amount to be inserted
!
! CHSFILL (0, .HDR_INSERT, ! Fill the space created for the insert
! .IHD + .IHD [IHD$W_ACTIVOFF]);
! END;
!
! Determine the extent that the image ident area differs in size from

```

```

370 0453 3      ! the current format. Expand the image ident area by the required
371 0454 3      amount, thus converting to the current format without relinking.
372 0455 3
373 0456 3      IHI_INSERT = 0; ! Assume no conversion required
374 0457 4      IF (.IHD [IHDSW_MAJORID] LSSU .V4_MAJORID)
375 0458 3      OR
376 0459 4      (
377 0460 5      (.IHD [IHDSW_MAJORID] EQL .V4_MAJORID)
378 0461 4      AND
379 0462 5      (.IHD [IHDSW_MINORID] LSSU .V4_MINORID)
380 0463 4      )
381 0464 3      THEN
382 0465 3      !
383 0466 3      ! The image name string grew between VMS V3 and V4. Split the
384 0467 3      ! image ident area after the old image name string and expand
385 0468 3      ! the string to the current maximum size, zero filled.
386 0469 3
387 0470 4      BEGIN
388 0471 4      IHI_INSERT = IHISS_IMGNAM - IHI_S_IMGNAM; ! Calculate size difference
389 0472 4      CH$MOVE (
390 0473 4      (.IHD [IHDSW_SIZE] - (.IHD [IHDSW_IMGIDOFF] + IHI_S_IMGNAM)),
391 0474 4      (.IHD + .IHD [IHDSW_IMGIDOFF] + IHI_S_IMGNAM),
392 0475 4      (.IHD + .IHD [IHDSW_IMGIDOFF] + IHISS_IMGNAM) );
393 0476 4      CH$FILL (0, .IHI_INSERT,
394 0477 4      (.IHD + .IHD [IHDSW_IMGIDOFF] + IHI_S_IMGNAM) );
395 0478 3      END;
396 0479 3      !
397 0480 3      ! Correct all the offsets to compensate for the insertion(s). Note that two of
398 0481 3      ! the offsets locate optional parts of the image header and are only updated
399 0482 3      ! if the associated areas are present in the image (offsets are nonzero).
400 0483 3
401 0484 4      IF (.HDR_INSERT NEQ 0)
402 0485 3      OR
403 0486 4      (.IHI_INSERT NEQ 0)
404 0487 3      THEN
405 0488 4      BEGIN
406 0489 4      IHD [IHDSW_SIZE] = .IHD [IHDSW_SIZE] + .HDR_INSERT + .IHI_INSERT;
407 0490 4      IHD [IHDSW_ACTIVOFF] = .IHD [IHDSW_ACTIVOFF] + .HDR_INSERT;
408 0491 4      IHD [IHDSW_IMGIDOFF] = .IHD [IHDSW_IMGIDOFF] + .HDR_INSERT;
409 0492 4
410 0493 4      IF .IHD [IHDSW_SYMDBGOFF] NEQU 0
411 0494 4      THEN
412 0495 4      IHD [IHDSW_SYMDBGOFF] = .IHD [IHDSW_SYMDBGOFF] + .HDR_INSERT;
413 0496 4
414 0497 4      IF .IHD [IHDSW_PATCHOFF] NEQU 0
415 0498 4      THEN
416 0499 4      IHD [IHDSW_PATCHOFF] = .IHD [IHDSW_PATCHOFF] + .HDR_INSERT + .IHI_INSERT;
417 0500 4
418 0501 3      END;
419 0502 3
420 0503 3      RETURN SSS_NORMAL;
421 0504 2      END;
422 0505 2
423 0506 2      [INRANGE,OUTRANGE] :
424 0507 2
425 0508 2      RETURN IMG$_BADHDR; ! Unrecognizable or unsupported image type
426 0509 2

```


				68		03	B0	00140	MOVW	#3, (R8)	0425
				60		01	CE	00143	MNEGL	#1, (R0)	0426
		04		A0		01	CE	00146	MNEGL	#1, 4(R0)	0427
				50		A6	3C	0014A	MOVZWL	2(IHD), R0	0430
				2C		50	B1	0014E	CMPW	R0, #4	
						06	1A	00151	BGTRU	13\$	
				59		04	C0	00153	ADDL2	#4, HDR_INSERT	0433
				68		04	B0	00156	MOVW	#4, (R8)	0434
						5A	D4	00159	CLRL	R10	0438
						59	D5	0015B	TSTL	HDR_INSERT	
						17	13	0015D	BEQL	14\$	
						5A	D6	0015F	INCL	R10	
				51		66	3C	00161	MOVZWL	(IHD), R1	0443
				51		50	C2	00164	SUBL2	R0, R1	
			57	56		50	C1	00167	ADDL3	R0, IHD, R7	0444
			6947	67		51	28	0016B	MOVCS	R1, (R7), (HDR_INSERT)[R7]	0445
			00	6E		00	2C	00170	MOVCS	#0, (SP), #0, HDR_INSERT, (R7)	0448
						67		00175			
						58	D4	00176	CLRL	IHI_INSERT	0456
FE78	CF	0C	A6	10		00	ED	00178	CMPZV	#0, #16, 12(IHD), V4_MAJORID	0457
						14	1F	00180	BLSSU	15\$	
FE6E	CF	0C	A6	10		00	ED	00182	CMPZV	#0, #16, 12(IHD), V4_MAJORID	0460
						2B	12	0018A	BNEQ	16\$	
FE68	CF	0E	A6	10		00	ED	0018C	CMPZV	#0, #16, 14(IHD), V4_MINORID	0462
						21	1E	00194	BGEQU	16\$	
				58		18	D0	00196	MOVL	#24, IHI_INSERT	0471
				50		A6	3C	00199	MOVZWL	6(IHD), R0	0473
				51		66	3C	0019D	MOVZWL	(IHD), R1	
				51		5C	C2	001A0	SUBL2	R0, R1	
				51		10	C2	001A3	SUBL2	#16, R1	
			57	56		50	C1	001A6	ADDL3	R0, IHD, R7	0474
			66	A7		51	28	001AA	MOVCS	R1, 16(R7), 40(R7)	0475
58		28	00	6E		00	2C	001B0	MOVCS	#0, (SP), #0, IHI_INSERT, 16(R7)	0477
						A7		001B5			
				04		5A	E8	001B7	BLBS	R10, 17\$	0484
						58	D5	001BA	TSTL	IHI_INSERT	0486
						29	13	001BC	BEQL	19\$	
				50		66	3C	001BE	MOVZWL	(IHD), R0	0489
				50		59	C0	001C1	ADDL2	HDR_INSERT, R0	
			66	50		58	A1	001C4	ADDW3	IHI_INSERT, R0, (IHD)	
				02	A6	59	A0	001C8	ADDW2	HDR_INSERT, 2(IHD)	0490
				06	A6	59	A0	001CC	ADDW2	HDR_INSERT, 6(IHD)	0491
						A6	B5	001D0	ADDW2	HDR_INSERT, 6(IHD)	
						04	A6	001D0	TSTW	4(IHD)	0493
						04	13	001D3	BEQL	18\$	
				04	A6	59	A0	001D5	ADDW2	HDR_INSERT, 4(IHD)	0495
				50		A6	3C	001D9	MOVZWL	8(IHD), R0	0497
						08	13	001DD	BEQL	19\$	
				50		59	C0	001DF	ADDL2	HDR_INSERT, R0	0499
			08	50		58	A1	001E2	ADDW3	IHI_INSERT, R0, 8(IHD)	
				50		01	D0	001E7	MOVL	#1, R0	0503
						04	001EA	RET			0512

; Routine Size: 491 bytes, Routine Base: YF\$\$\$SYSIMGACT + 0008

; 430 0513 1

```

432 0514 1 %SBTTL 'IMG$GET NEXT_ISD Get Image Section Descriptor'
433 0515 1 GLOBAL ROUTINE IMG$GET_NEXT_ISD
434 0516 1 ( CHAN, BLK_BUFADR, IHD_BUFADR, VBN_ADR, OFFSET_ADR, ISD_BUFADR, HEADER_VERSION ) =
435 0517 1
436 0518 1 ++
437 0519 1 FUNCTIONAL DESCRIPTION:
438 0520 1
439 0521 1 FORMAL PARAMETERS:
440 0522 1
441 0523 1 Chan Channel on which image file is open
442 0524 1 Blk_bufadr Address of buffer which contains block of image header
443 0525 1 Ihd_bufadr Address of buffer containing decoded IHD
444 0526 1 Vbn_adr Address of VBN in blk_bufadr
445 0527 1 Offset_adr Address of Offset to ISD
446 0528 1 ISD_bufadr Address of buffer to contain decoded ISD
447 0529 1
448 0530 1 IMPLICIT INPUTS:
449 0531 1 NONE
450 0532 1
451 0533 1 IMPLICIT OUTPUTS:
452 0534 1 NONE
453 0535 1
454 0536 1 ROUTINE VALUE:
455 0537 1 COMPLETION CODES:
456 0538 1 NONE
457 0539 1
458 0540 1 SIDE EFFECTS:
459 0541 1 NONE
460 0542 1
461 0543 1 --
462 0544 1
463 0545 2 BEGIN
464 0546 2 LOCAL
465 0547 2 IOSB : BBLOCK [8], ! Quadword IO status block
466 0548 2 B_ISD : REF BBLOCK, ! ISD is header block buffer
467 0549 2 ISD : REF BBLOCK,
468 0550 2 ISD_SIZ : SIGNED WORD,
469 0551 2 SIZE,
470 0552 2 STATUS: ! Status
471 0553 2
472 0554 2 BIND
473 0555 2 IHD = .IHD_BUFADR : BBLOCK,
474 0556 2 OFFSET = .OFFSET_ADR : WORD,
475 0557 2 VBN = .VBN_ADR;
476 0558 2
477 0559 2 :
478 0560 2 Validate that offset and VBN are reasonable
479 0561 2
480 0562 2 IF .OFFSET GEQU
481 0563 2 (IF .VBN EQL 1
482 0564 2 THEN IMG$C_BLOCKSIZ - 2
483 0565 2 ELSE IMG$C_BLOCKSIZ)
484 0566 2 THEN
485 0567 2 RETURN IMG$_ISD_OFF;
486 0568 2
487 0569 2 IF .VBN GTR .IHD [IHD$B_HDRBLKCNT]
488 0570 2 THEN

```

```

489 0571 2 RETURN IMG$_ISD_VBN;
490 0572 2
491 0573 2
492 0574 2 Get next ISD
493 0575 2
494 0576 2 B_ISD = .BLK_BUFADR + .OFFSET;
495 0577 2 ISD_SIZ = .B_ISD [ISD_W_SIZE];
496 0578 2
497 0579 2
498 0580 2 See whether offset points off the block and we need to read the next block
499 0581 2
500 0582 2 IF .ISD_SIZ EQL -1
501 0583 2 THEN ! Read next block
502 0584 2 BEGIN
503 0585 2 VBN = .VBN + 1; ! Increment VBN
504 0586 2 OFFSET = 0;
505 0587 2 SIZE = IMG$_BLOCKSIZ;
506 0588 2
507 P 0589 2 STATUS = $QIOW
508 P 0590 2 (
509 P 0591 2 EFN = EX$_SYSEFN, ! Event flag
510 P 0592 2 CHAN = .CHAN, ! Channel
511 P 0593 2 FUNC = IO$_READVBLK, ! Read a virtual block
512 P 0594 2 IOSB = IOSB, ! I/O status block
513 P 0595 2 P1 = .BLK_BUFADR, ! Buffer to read in to
514 P 0596 2 P2 = .SIZE, ! Number of bytes to read
515 P 0597 2 P3 = .VBN ! Virtual block number to read
516 0598 2 );
517 0599 2
518 0600 2 IF .STATUS
519 0601 2 THEN
520 0602 2 STATUS = .IOSB [0,0,16,0]; ! Pick up final status
521 0603 2 IF NOT .STATUS
522 0604 2 THEN
523 0605 2 RETURN .STATUS;
524 0606 2
525 0607 2 B_ISD = .BLK_BUFADR;
526 0608 2 ISD_SIZ = .B_ISD [ISD_W_SIZE];
527 0609 2
528 0610 2 IF .ISD_SIZ EQL -1 ! Trap consecutive 'wrap' ISDs
529 0611 2 THEN
530 0612 2 RETURN IMG$_INCONISD;
531 0613 2
532 0614 2 END;
533 0615 2
534 0616 2
535 0617 2 See whether there are any ISDs left
536 0618 2
537 0619 2 IF .ISD_SIZ EQL 0
538 0620 2 THEN ! No more ISDs left
539 0621 2 RETURN IMG$_ENDOFHDR;
540 0622 2
541 0623 2
542 0624 2 Validate that the ISD size is reasonable
543 0625 2
544 0626 2 IF (.ISD_SIZ LSS ISD$_LENDZRO)
545 0627 2 OR

```

```

546 0628 3 (.ISD_SIZ GTR ISD$C_MAXLENGLBL)
547 0629 2 THEN
548 0630 2 RETURN IMG$_ISD_SIZ;
549 0631 2
550 0632 2
551 0633 2
552 0634 2
553 0635 2
554 0636 2
555 0637 2
556 0638 2
557 0639 2
558 0640 2
559 0641 2
560 0642 2
561 0643 2
562 0644 2
563 0645 2
564 0646 2
565 0647 2
566 0648 2
567 0649 2
568 0650 2
569 0651 2
570 0652 2
571 0653 2
572 0654 2
573 0655 2
574 0656 2
575 0657 2
576 0658 2
577 0659 2
578 0660 2
579 0661 2
580 0662 2
581 0663 2
582 0664 2
583 0665 1

```

(.ISD_SIZ GTR ISD\$C_MAXLENGLBL)
 THEN
 RETURN IMG\$_ISD_SIZ;
 :
 Make sure that ISD doesn't attempt to wrap around to the next block
 IF (.OFFSET + .ISD_SIZ) GTRU
 (IF .VBN EQL 1
 THEN IMG\$C_BLOCKSIZ - 2
 ELSE IMG\$C_BLOCKSIZ)
 THEN
 RETURN IMG\$_INCONISD;
 ISD = .ISD_BUFADR;
 CH\$MOVE (.ISD_SIZ, .B_ISD, .ISD); ! Copy from block to ISD buffer
 OFFSET = .OFFSET + .ISD_SIZ;
 :
 Don't use page fault cluster size for cross-linker images
 IF .HEADER_VERSION EQL IH\$C_GEN_XLNKR
 THEN
 ISD [ISD\$B_PFC] = 0;
 :
 Some V3 images use IH\$SL_IAFVA to identify the fixup vectors
 IF .HEADER_VERSION EQL IH\$C_GEN_FIXUP
 THEN
 IF (.ISD [ISD\$V_VPN] * 512) EQL .IHD [IH\$SL_IAFVA]
 AND
 .ISD [ISD\$V_VPN] NEQ 0
 THEN
 ISD [ISD\$V_FIXUPVEC] = 1;
 RETURN S\$\$_NORMAL;
 END;

! IMG\$GET_NEXT_ISD routine

			03FC 0000	.ENTRY	IMG\$GET_NEXT_ISD, Save R2,R3,R4,R5,R6,R7,-	: 0515
					R8,R9	
	5E		08 C2 00002	SUBL2	#8, SP	
	59	0C	AC D0 00005	MOVL	IHD_BUFADR, R9	: 0555
	58	14	AC D0 00009	MOVL	OFFSET_ADR, R8	: 0556
	52	10	AC D0 0000D	MOVL	VBN_ADR, R2	: 0557
	01		62 D1 00011	CMPL	(R2), #1	: 0563
			07 12 00014	BNEQ	1\$	
	50	01FE	8F 3C 00016	MOVZWL	#510, R0	: 0564
			05 11 0001B	BRB	2\$	
	50	0200	8F 3C 0001D	MOVZWL	#512, R0	: 0563
50			10 00 ED 00022	CMPZV	#0, #16, (R8), R0	
			08 1F 00027	BLSSU	3\$	
	50	084D8CB4	8F D0 00029	MOVL	#139300020, R0	: 0567

62	10	A9	08	00	04 00C30	RET			
				08	ED 00031 3\$:	CMPZV	#0, #8, 16(R9), (R2)		0569
			50	084D8CBC	08 18 00037	BGEQ	4\$		
					8F D0 00039	MOVL	#139300028, R0		0571
			53		04 00040	RET			
			53	08	68 3C 00041 4\$:	MOVZWL	(R8), B ISD		05.6
			53		AC C0 00044	ADDL2	BLK BUFADR, B ISD		
			57		63 B0 00048	MOVW	(B ISD), ISD_SIZ		0577
		FFFF	8F		57 B1 0004B	CMPW	ISD_SIZ, #-1		0582
					40 12 00050	BNEQ	7\$		
					62 D6 00052	INCL	(R2)		0585
			50	0200	68 B4 00054	CLRW	(R8)		0586
					8F 3C 00056	MOVZWL	#512, SIZE		0587
					7E 7C 0005B	CLRW	-(SP)		0598
					7E D4 0005D	CLRL	-(SP)		
					62 DD 0005F	PUSHL	(R2)		
					50 DD 00061	PUSHL	SIZE		
				08	AC DD 00063	PUSHL	BLK BUFADR		
					7E 7C 00066	CLRW	-(SP)		
				20	AE 9F 00068	PUSHAB	IOSB		
					31 DD 0006B	PUSHL	#49		
				04	AC DD 0006D	PUSHL	CHAN		
			7E		00G 9A 00070	MOVZBL	S^EXESC SYSEFN, -(SP)		
		00000000G	00		0C FB 00073	CALLS	#12, SYSSQIOW		
			03		50 E9 0007A	BLBC	STATUS, 5\$		0600
			50		6E 3C 0007D	MOVZWL	IOSB, STATUS		0602
			01		50 E8 00080 5\$:	BLBS	STATUS, 6\$		0603
					04 00083	RET			
			53	08	AC D0 00084 6\$:	MOVL	BLK BUFADR, B ISD		0607
			57		63 B0 00088	MOVW	(B ISD), ISD_SIZ		0608
		FFFF	8F		57 B1 0008B	CMPW	ISD_SIZ, #-1		0610
					3F 13 00090	BEQL	13\$		
					57 B5 00092 7\$:	TSTW	ISD_SIZ		0619
					08 12 00094	BNEQ	8\$		
			50	084D8640	8F D0 00096	MOVL	#139298368, R0		0621
					04 0009D	RET			
			0C		57 B1 0009E 8\$:	CMPW	ISD_SIZ, #12		0626
					07 19 000A1	BLSS	9\$		
		0040	8F		57 B1 000A3	CMPW	ISD_SIZ, #64		0628
					08 15 000A8	BLEQ	10\$		
			50	084D8CC4	8F D0 000AA 9\$:	MOVL	#139300036, R0		0630
					04 000B1	RET			
			51		68 3C 000B2 10\$:	MOVZWL	(R8), R1		0635
			50		57 32 000B5	CVTWL	ISD_SIZ, R0		
			51		50 C0 000B8	ADDL2	R0, -R1		
			01		62 D1 000BB	CMPW	(R2), #1		0636
					07 12 000BE	BNEQ	11\$		
			50	01FE	8F 3C 000C0	MOVZWL	#510, R0		0637
					C5 11 000C5	BRB	12\$		
			50	0200	8F 3C 000C7 11\$:	MOVZWL	#512, R0		0636
			50		51 D1 000CC 12\$:	CMPW	R1, R0		
					08 1B 000CF	BLEQU	14\$		
			50	084D8CAC	8F D0 000D1 13\$:	MOVL	#139300012, R0		0640
					04 000D8	RET			
			56	18	AC D0 000D9 14\$:	MOVL	ISD_BUFADR, ISD		0642
		66	63		57 28 000DD	MOVW	ISD_SIZ, (B ISD), (ISD)		0643
			68		57 AC 000E1	ADDW2	ISD_SIZ, (R8)		0644


```

586 0667 1 %SBTTL 'CONVERT_XLINK Convert a cross-linker image header to standard format'
587 0668 1 ROUTINE CONVERT_XLINK
588 0669 1 ( BLK_BUFADR : REF $BBLOCK,
589 0670 1   IHD       : REF $BBLOCK ) =
590 0671 1
591 0672 1 ++
592 0673 1 FUNCTIONAL DESCRIPTION:
593 0674 1   An image header produced by the cross-linker is converted to the
594 0675 1   standard format.
595 0676 1
596 0677 1 FORMAL PARAMETERS:
597 0678 1
598 0679 1   Blk_bufadr   Address of buffer which contains first block of image header
599 0680 1   Ihd          Address of buffer to contain decoded IHD
600 0681 1
601 0682 1 IMPLICIT INPUTS:
602 0683 1   NONE
603 0684 1
604 0685 1 IMPLICIT OUTPUTS:
605 0686 1   NONE
606 0687 1
607 0688 1 ROUTINE VALUE:
608 0689 1 COMPLETION CODES:
609 0690 1   NONE
610 0691 1
611 0692 1 SIDE EFFECTS:
612 0693 1   NONE
613 0694 1
614 0695 1 --
615 0696 1
616 0697 2 BEGIN
617 0698 2
618 0699 2 BIND
619 0700 2   PRIV_MASK = IHD [IHDSQ PRIVREQS] : VECTOR [2],
620 0701 2   IHD_ACT_ADR = .IHD + IHDSK_LENGTH : VECTOR [3],
621 0702 2   IHX_ACT_ADR = BLK_BUFADR [IHXSQ_STARTADR] : VECTOR [2],
622 0703 2   IHS       = .IHD + IHDSK_LENGTH + IHASK_LENGTH : $BBLOCK;
623 0704 2
624 0705 2   Zero the one page buffer which will contain decoded IHD
625 0706 2
626 0707 2 CHSFILL (0, 512, .IHD);
627 0708 2
628 0709 2   Fill in offsets and directly transportable fields
629 0710 2
630 0711 2 IHD [IHDSW_ACTIVOFF] = IHDSK_LENGTH;
631 0712 2 IHD [IHDSW_SIZE] = IHDSK_LENGTH + IHASK_LENGTH + IHSSK_LENGTH;
632 0713 2 IHD [IHDSB_HDRBLKCNT] = .BLK_BUFADR [IHXSQ_HDRBLKCNT];
633 0714 2
634 0715 2   Convert image ID fields
635 0716 2
636 0717 2 IHD [IHDSW_MAJORID] = IHDSK_MAJORID;
637 0718 2 IHD [IHDSW_MINORID] = IHDSK_MINORID;
638 0719 2
639 0720 2   Assume all privileges
640 0721 2
641 0722 2 PRIV_MASK [0] = -1;
642 0723 2 PRIV_MASK [1] = -1;

```

```

: 643 0724 2 |
: 644 0725 2 | Add image activation data
: 645 0726 2 |
: 646 0727 2 | IHD_ACT_ADR [0] = .IHX_ACT_ADR [0];
: 647 0728 2 | IHD_ACT_ADR [1] = .IHX_ACT_ADR [1];
: 648 0729 2 |
: 649 0730 2 | Check for DEBUG data
: 650 0731 2 |
: 651 0732 2 | IF .BLK_BUFADR [IHX$W_MINORID] GEQ IHX$K_MINORID1
: 652 0733 2 | THEN
: 653 0734 2 | BEGIN
: 654 0735 2 | IHD [IHD$W_SYMDBGOFF] = IHD$K_LENGTH + IHAS$K_LENGTH;
: 655 0736 2 | IHD_ACT_ADR [2] = .BLK_BUFADR [IHX$L_TFRADR3];
: 656 0737 2 | IHS [IHSSL_DSTVBN] = .BLK_BUFADR [IHX$L_DSTVBN] ;
: 657 0738 2 | IHS [IHSSL_GSTVBN] = .BLK_BUFADR [IHX$L_GSTVBN] ;
: 658 0739 2 | IHS [IHSSW_DSTBLKS] = .BLK_BUFADR [IHX$W_DSTBLKS] ;
: 659 0740 2 | IHS [IHSSW_GSTRECS] = .BLK_BUFADR [IHX$W_GSTRECS] ;
: 660 0741 2 | END;
: 661 0742 2 |
: 662 0743 2 | RETURN SS$ _NORMAL;
: 663 0744 1 | END;

```

OFFC 00000 CONVERT_XLINK:

									Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 0668	
		56	08	AC	D0	00002			MOVW	IHD, R6	: 0700
		58	14	A6	9E	00006			MOVAB	20(R6), R11	
		58	30	A6	9E	0000A			MOVAB	48(R6), R8	: 0701
		57	04	AC	D0	0000E			MOVW	BLK_BUFADR, R7	: 0702
		5A	04	A7	9E	00012			MOVAB	4(R7), R10	
		59	44	A6	9E	00016			MOVAB	68(R6), R9	: 0703
0200	8F								MOVCS	#0, (SP), #0, #512, (R6)	: 0707
		66	00300058	8F	D0	00022			MOVW	#3145816, (R6)	: 0712
10	A6		02	A7	90	00029			MOVW	2(R7), 16(R6)	: 0713
0C	A6	35303230		8F	D0	0002E			MOVW	#892351024, 12(R6)	: 0717
	6B			01	CE	00036			MNEGL	#1, (R11)	: 0722
04	AB			01	CE	00039			MNEGL	#1, 4(R11)	: 0723
	68			6A	7D	0003D			MOVQ	(R10), (R8)	: 0727
3130	8F		0E	A7	B1	00040			CMPW	14(R7), #12592	: 0732
				13	1F	00046			BLSSU	1\$	
04	A6		44	8F	9B	00048			MOVZBW	#68, 4(R6)	: 0735
08	A8		34	A7	D0	0004D			MOVW	52(R7), 8(R8)	: 0736
	69		28	A7	7D	00052			MOVQ	40(R7), (R9)	: 0737
08	A9		30	A7	D0	00056			MOVW	48(R7), 8(R9)	: 0739
	50			01	D0	0005B	1\$:		MOVW	#1, R0	: 0743
				04	0005E				RET		: 0744

: Routine Size: 95 bytes, Routine Base: YF\$\$\$SYSIMGACT + 0306

IMG\$DECODE
V04-000

Get and Decode Image Header and Sections
CONVERT_XLINK Convert a cross-linker image head

M 4
16-Sep-1984 02:41:10
14-Sep-1984 13:12:35

VAX-11 Bliss-32 V4.0-742
[SYS.SRC]IMGDECODE.B32;1

Page 19
(6)

IMG\$
V04-

: 665 0745 1 END
: 666 0746 0 ELUDOM

.End of module IMGDECODE

PSECT SUMMARY

Name	Bytes	Attributes
YF\$\$\$SYSIMGACT	869	NOVEC, WRT, RD, EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	63 0	1000	00:01.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:IMGDECODE/OBJ=OBJ\$:IMGDECODE MSRC\$:IMGDECODE/UPDATE=(ENH\$:IMGDECODE)

: Size: 861 code + 8 data bytes
: Run Time: 00:19.1
: Elapsed Time: 00:22.0
: Lines/CPU Min: 2348
: Lexemes/CPU-Min: 16064
: Memory Used: 205 pages
: Compilation Complete

