

1000

```
PPPPPPPP      AAAAAA      SSSSSSSS      CCCCCCCC      000000      NN      NN      VV      VV      EEEEEEEEEEE      RRRRRRRR
PPPPPPPP      AAAAAA      SSSSSSSS      CCCCCCCC      000000      NN      NN      VV      VV      EEEEEEEEEEE      RRRRRRRR
PP      PP      AA      AA      SS      CC      00      00      NN      NN      VV      VV      EE      RR      RR
PP      FP      AA      AA      SS      CC      00      00      NN      NN      VV      VV      EE      RR      RR
PP      PP      AA      AA      SS      CC      00      00      NNNN      NN      VV      VV      EE      RR      RR
PP      PP      AA      AA      SS      CC      00      00      NNNN      NN      VV      VV      EE      RR      RR
PPPPPPPP      AA      AA      SSSSSS      CC      00      00      NN      NN      VV      VV      EEEEEEEEE      RRRRRRRR
PPPPPPPP      AA      AA      SSSSSS      CC      00      00      NN      NN      VV      VV      EEEEEEEEE      RRRRRRRR
PP      AAAAAAAAAA      SS      CC      00      00      NN      NNNN      VV      VV      EE      RR      RR
PP      AAAAAAAAAA      SS      CC      00      00      NN      NNNN      VV      VV      EE      RR      RR
PP      AA      AA      SS      CC      00      00      NN      NN      VV      VV      EE      RR      RR
PP      AA      AA      SS      CC      00      00      NN      NN      VV      VV      EE      RR      RR
PP      AA      AA      SSSSSSSS      CCCCCCCC      000000      NN      NN      VV      VV      EEEEEEEEEEE      RR      RR
PP      AA      AA      SSSSSSSS      CCCCCCCC      000000      NN      NN      VV      VV      EEEEEEEEEEE      RR      RR
                                         ....
                                         ....
                                         ....
                                         ....
```

```
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
```


(2)	46	DECLARATIONS
(3)	75	PASS\$CONVERT_L_BU = Convert signed longword to unsigned byte
(4)	118	PASS\$CONVERT_BU_L = Convert unsigned byte to signed longword
(5)	155	PASS\$CONVERT_WU_L = Convert unsigned word to signed longword
(6)	192	PASS\$CONVERT_L_L = Convert signed longword to signed longword
(7)	229	PASS\$CONVERT_LO_L = Convert unsigned longword to signed longword
(8)	271	PASS\$CONVERT_BOOL_L = Convert boolean to signed longword
(9)	309	PASS\$ROUND_F_LU = Round F_floating to unsigned longword
(10)	345	PASS\$ROUND_D_LU = Round D_floating to unsigned longword
(11)	381	PASS\$ROUND_G_LU = Round G_floating to unsigned longword
(12)	417	PASS\$ROUND_H_LU = Round H_floating to unsigned longword
(13)	453	PASS\$TRUNC_F_LU = Truncate F_floating to unsigned longword
(14)	489	PASS\$TRUNC_D_LU = Truncate D_floating to unsigned longword
(15)	525	PASS\$TRUNC_G_LU = Truncate G_floating to unsigned longword
(16)	561	PASS\$TRUNC_H_LU = Truncate H_floating to unsigned longword
(17)	597	PASS\$CONVERT_LU_F = Convert unsigned longword to F_floating
(18)	637	PASS\$CONVERT_F_F = Convert F_floating to F_floating
(19)	674	PASS\$CONVERT_H_F = Convert H_floating to F_floating
(20)	711	PASS\$CONVERT_LO_D = Convert unsigned longword to D_floating
(21)	750	PASS\$CONVERT_D_D = Convert D_floating to D_floating
(22)	787	PASS\$CONVERT_H_D = Convert H_floating to D_floating
(23)	824	PASS\$CONVERT_LO_G = Convert unsigned longword to G_floating
(24)	863	PASS\$CONVERT_G_G = Convert G_floating to G_floating
(25)	900	PASS\$CONVERT_H_G = Convert H_floating to G_floating
(26)	937	PASS\$CONVERT_L_H = Convert signed longword to H_floating
(27)	975	PASS\$CONVERT_LO_H = Convert unsigned longword to H_floating
(28)	1015	PASS\$CONVERT_F_H = Convert F_floating to H_floating
(29)	1053	PASS\$CONVERT_D_H = Convert D_floating to H_floating
(30)	1091	PASS\$CONVERT_G_H = Convert G_floating to H_floating
(31)	1129	PASS\$CONVERT_H_H = Convert H_floating to H_floating

```
0000 1 .TITLE PASSCONVERT - Pascal-specific conversion routines
0000 2 .IDENT /1-001/ ; File: PASCONVER.MAR Edit: SBL1001
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: Pascal Language Support
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 : This module contains all Pascal-specific arithmetic conversion
0000 35 : routines.
0000 36 :
0000 37 : ENVIRONMENT: Runs at any access mode, AST Reentrant
0000 38 :
0000 39 : AUTHOR: Steven B. Lionel, CREATION DATE: 04-NOV-1980
0000 40 :
0000 41 : MODIFIED BY:
0000 42 :
0000 43 : 1-001 - Original. SBL 4-NOV-1980
0000 44 :--
```



```

0000 46      .SBTTL  DECLARATIONS
0000 47      :
0000 48      : LIBRARY MACRO CALLS:
0000 49      :
0000 50      : $$$DEF      ; System error code definitions
0000 51      :
0000 52      : EXTERNAL DECLARATIONS:
0000 53      :
0000 54      : .DSABL  GBL      ; Force all external symbols to be declared
0000 55      : .EXTRN  LIB$SIGNAL ; Signal non-fatal error
0000 56      :
0000 57      : MACROS:
0000 58      :
0000 59      :     NONE
0000 60      :
0000 61      : EQUATED SYMBOLS:
0000 62      :
0000 63      :     NONE
0000 64      :
0000 65      : OWN STORAGE:
0000 66      :
0000 67      :     NONE
0000 68      :
0000 69      : PSECT DECLARATIONS:
0000 70      :
0000 71      : .PSECT _PASS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 72      :     EXE, RD, NOWRT, LONG
0000 73

```



```

0000 75 .SBTTL PASS$CONVERT_L_BU - Convert signed longword to unsigned byte
0000 76 :++
0000 77 : FUNCTIONAL DESCRIPTION:
0000 78 :
0000 79 : Converts a signed longword to an unsigned byte with possible
0000 80 : overflow.
0000 81 :
0000 82 : CALLING SEQUENCE:
0000 83 :
0000 84 : Result.wbu.v = PASS$CONVERT_L_BU (long.rl.r)
0000 85 :
0000 86 : FORMAL PARAMETERS:
0000 87 :
0000 88 : long - signed longword argument
0000 89 :
0000 90 : IMPLICIT INPUTS:
0000 91 :
0000 92 : NONE
0000 93 :
0000 94 : IMPLICIT OUTPUTS:
0000 95 :
0000 96 : NONE
0000 97 :
0000 98 : ROUTINE VALUE:
0000 99 :
0000 100 : The unsigned byte value of the argument
0000 101 :
0000 102 : SIDE EFFECTS:
0000 103 :
0000 104 : SS$_INTOVF - If argument negative or greater than 255. If continued,
0000 105 : the low 8 bits are stored.
0000 106 :
0000 107 :--
0000 108 :
4000 0000 .ENTRY PASS$CONVERT_L_BU, ^M<IV> ; Entry point
0002 110 :
0000 111 CMPL @4(AP), #255 ; Will value overflow?
0000 112 BLEQU 10$ ; No
0000 113 MOVZWL #SS$_INTOVF, -(SP)
0000 114 CALLS #1, G^LIB$SIGNAL ; Signal integer overflow
0000 115 10$: MOVZBL @4(AP), R0 ; Convert
0000 116 RET ; End of routine PASS$CONVERT_L_BU

```


[illegible]

PASS\$CONVERT
1-001

E 16
- Pascal-specific conversion routines 16-SEP-1984 01:23:05 VAX/VMS Macro V04-00 Page 6
PASS\$CONVERT_L_L = Convert signed longwor 6-SEP-1984 11:30:12 [PASRTL.SRC]PASCONVER.MAR;1 (6)

```
002B 192 .SBTTL PASS$CONVERT_L_L = Convert signed longword to signed longword
002B 193 :++
002B 194 : FUNCTIONAL DESCRIPTION:
002B 195 :
002B 196 :     Converts a signed longword to a signed longword.
002B 197 :
002B 198 : CALLING SEQUENCE:
002B 199 :
002B 200 :     Result.wl.v = PASS$CONVERT_L_L (Long.rl.r)
002B 201 :
002B 202 : FORMAL PARAMETERS:
002B 203 :
002B 204 :     Long    - Signed longword argument
002B 205 :
002B 206 : IMPLICIT INPUTS:
002B 207 :
002B 208 :     NONE
002B 209 :
002B 210 : IMPLICIT OUTPUTS:
002B 211 :
002B 212 :     NONE
002B 213 :
002B 214 : ROUTINE VALUE:
002B 215 :
002B 216 :     The argument sign-extended to a longword
002B 217 :
002B 218 : SIDE EFFECTS:
002B 219 :
002B 220 :     NONE
002B 221 :
002B 222 : --
002B 223 :
002B 224 .ENTRY PASS$CONVERT_L_L, ^M<IV> ; Entry point
002D 225
50 04 BC D0 002D 226      MOVL    @4(AP), R0      ; Convert
04 0031 227      RET                ; End of routine PASS$CONVERT_L_L
```



```

0032 229      .SBTTL PAS$CONVERT_LU_L = Convert unsigned longword to signed longword
0032 230      :++
0032 231      : FUNCTIONAL DESCRIPTION:
0032 232      :
0032 233      :     Converts an unsigned longword to a signed longword with possible
0032 234      :     overflow.
0032 235      :
0032 236      : CALLING SEQUENCE:
0032 237      :
0032 238      :     Result.wl.v = PAS$CONVERT_LU_L (Long.rlu.r)
0032 239      :
0032 240      : FORMAL PARAMETERS:
0032 241      :
0032 242      :     Long      - Unsigned longword argument
0032 243      :
0032 244      : IMPLICIT INPUTS:
0032 245      :
0032 246      :     NONE
0032 247      :
0032 248      : IMPLICIT OUTPUTS:
0032 249      :
0032 250      :     NONE
0032 251      :
0032 252      : ROUTINE VALUE:
0032 253      :
0032 254      :     The argument zero-extended to a longword
0032 255      :
0032 256      : SIDE EFFECTS:
0032 257      :
0032 258      :     SS$_INTOVF - If argument is greater than 2**31-1. If continued,
0032 259      :     the argument is stored directly into the result.
0032 260      :
0032 261      :--
0032 262      :
0032 263      :.ENTRY PAS$CONVERT_LU_L, ^M<IV> ; Entry point
0034 264      :
0034 265      :   MOVL    @4(AP), R0          ; Fetch argument
0038 266      :   BGEQ    10$             ; If not negative, ok
003A 267      :   MOVZWL  #SS$_INTOVF, -(SP) ; Signal integer overflow
003F 268      :   CALLS   #1, G^LIB$SIGNAL
0046 269 10$:   RET                  ; End of routine PAS$CONVERT_LU_L

```

4000

50 04 BC D0 0034 265

7E 047C 8F 3C 003A 267

00000000'GF 01 FB 003F 268

04 0046 269 10\$:


```

0047 271      .SBTTL PASS$CONVERT_BOOL_L - Convert boolean to signed longword
0047 272      :++
0047 273      : FUNCTIONAL DESCRIPTION:
0047 274      :
0047 275      :     Convert boolean to signed longword.
0047 276      :
0047 277      : CALLING SEQUENCE:
0047 278      :
0047 279      :     Result.wl.v = PASS$CONVERT_BOOL_L (Bool.rv.r)
0047 280      :
0047 281      : FORMAL PARAMETERS:
0047 282      :
0047 283      :     Bool      - Boolean argument
0047 284      :
0047 285      : IMPLICIT INPUTS:
0047 286      :
0047 287      :     NONE
0047 288      :
0047 289      : IMPLICIT OUTPUTS:
0047 290      :
0047 291      :     NONE
0047 292      :
0047 293      : ROUTINE VALUE:
0047 294      :
0047 295      :     The low bit of the argument zero-extended to a longword.
0047 296      :
0047 297      : SIDE EFFECTS:
0047 298      :
0047 299      :     NONE
0047 300      :
0047 301      :--
0047 302
4000 0047 303      .ENTRY PASS$CONVERT_BOOL_L, ^M<IV>      ; Entry point
0049 304
50   04 BC   9A 0049 305      MOVZBL  @4(AP), R0      ; Get low byte
50   FE 8F   8A 004D 306      BICB2   #^XFE, R0      ; Convert
                                04 0051 307      RET      ; End of routine PASS$CONVERT_BOOL_L

```



```

0052 309      .SBTTL PASS$ROUND_F_LU - Round F_floating to unsigned longword
0052 310      :++
0052 311      : FUNCTIONAL DESCRIPTION:
0052 312      :
0052 313      : Round an F_floating to an unsigned longword, without overflow.
0052 314      :
0052 315      : CALLING SEQUENCE:
0052 316      :
0052 317      : Result.wlu.v = PASS$ROUND_F_LU (Single.rf.r)
0052 318      :
0052 319      : FORMAL PARAMETERS:
0052 320      :
0052 321      : Single - F_floating argument
0052 322      :
0052 323      : IMPLICIT INPUTS:
0052 324      :
0052 325      : NONE
0052 326      :
0052 327      : IMPLICIT OUTPUTS:
0052 328      :
0052 329      : NONE
0052 330      :
0052 331      : ROUTINE VALUE:
0052 332      :
0052 333      : The argument rounded to the nearest integer modulus 2**32.
0052 334      :
0052 335      : SIDE EFFECTS:
0052 336      :
0052 337      : NONE
0052 338      :
0052 339      :--
0000 0052 340      .ENTRY PASS$ROUND_F_LU, ^M<>      ; IV must be disabled
50 04 BC 4B 0054 341
04 0054 342      CVTRFL @4(AP), R0      ; Round
0058 343      RET      ; End of routine PASS$ROUND_F_LU

```



```

0059 345 .SBTTL PASS$ROUND_D_LU - Round D_floating to unsigned longword
0059 346 :++
0059 347 : FUNCTIONAL DESCRIPTION:
0059 348 :
0059 349 : Round a D_floating to an unsigned longword, without overflow.
0059 350 :
0059 351 : CALLING SEQUENCE:
0059 352 :
0059 353 : Result.wlu.v = PASS$ROUND_D_LU (Double.rd.r)
0059 354 :
0059 355 : FORMAL PARAMETERS:
0059 356 :
0059 357 : Double - D_floating argument
0059 358 :
0059 359 : IMPLICIT INPUTS:
0059 360 :
0059 361 : NONE
0059 362 :
0059 363 : IMPLICIT OUTPUTS:
0059 364 :
0059 365 : NONE
0059 366 :
0059 367 : ROUTINE VALUE:
0059 368 :
0059 369 : The argument rounded to the nearest integer modulus 2**32.
0059 370 :
0059 371 : SIDE EFFECTS:
0059 372 :
0059 373 : NONE
0059 374 :
0059 375 :--
0000 0059 376 .ENTRY PASS$ROUND_D_LU, ^M<> ; IV must be disabled
50 04 BC 6B 005B 377
04 005B 378 CVTRDL @4(AP), R0 ; Round
RET ; End of routine PASS$ROUND_D_LU
005F 379

```



```

0060 381      .SBTTL PASS$ROUND_G_LU - Round G_floating to unsigned longword
0060 382      :++
0060 383      : FUNCTIONAL DESCRIPTION:
0060 384      :
0060 385      :     Round a G_floating to an unsigned longword, without overflow.
0060 386      :
0060 387      : CALLING SEQUENCE:
0060 388      :
0060 389      :     Result.wlu.v = PASS$ROUND_G_LU (Double.rg.r)
0060 390      :
0060 391      : FORMAL PARAMETERS:
0060 392      :
0060 393      :     Double - G_floating argument
0060 394      :
0060 395      : IMPLICIT INPUTS:
0060 396      :
0060 397      :     NONE
0060 398      :
0060 399      : IMPLICIT OUTPUTS:
0060 400      :
0060 401      :     NONE
0060 402      :
0060 403      : ROUTINE VALUE:
0060 404      :
0060 405      :     The argument rounded to the nearest integer modulus 2**32.
0060 406      :
0060 407      : SIDE EFFECTS:
0060 408      :
0060 409      :     NONE
0060 410      :
0060 411      :--
0000 0060 412      .ENTRY PASS$ROUND_G_LU, ^M<>      ; IV must be disabled
50   04 BC 4BFD 0062 413      CVTRGL @4(AP), R0      ; Round
04   0067 414      RET                                ; End of routine PASS$ROUND_G_LU
04   0067 415
  
```



```

0068 417      .SBTTL PASS$ROUND_H_LU - Round H_floating to unsigned longword
0068 418      :++
0068 419      : FUNCTIONAL DESCRIPTION:
0068 420      :
0068 421      :     Round an H_floating to an unsigned longword, without overflow.
0068 422      :
0068 423      : CALLING SEQUENCE:
0068 424      :
0068 425      :     Result.wlu.v = PASS$ROUND_H_LU (Quad.rh.r)
0068 426      :
0068 427      : FORMAL PARAMETERS:
0068 428      :
0068 429      :     Quad      - H_floating argument
0068 430      :
0068 431      : IMPLICIT INPUTS:
0068 432      :
0068 433      :     NONE
0068 434      :
0068 435      : IMPLICIT OUTPUTS:
0068 436      :
0068 437      :     NONE
0068 438      :
0068 439      : ROUTINE VALUE:
0068 440      :
0068 441      :     The argument rounded to the nearest integer modulus 2**32.
0068 442      :
0068 443      : SIDE EFFECTS:
0068 444      :
0068 445      :     NONE
0068 446      :
0068 447      :--
0000 0068 448      .ENTRY PASS$ROUND_H_LU, ^M<>      ; IV must be disabled
50   04 BC 6BFD 006A 449      CVTRHL @4(AP), R0      ; Round
      04 006F 450      RET                          ; End of routine PASS$ROUND_H_LU
      006F 451

```

```

0070 453 .SBTTL PAS$TRUNC_F_LU - Truncate F_floating to unsigned longword
0070 454 :++
0070 455 : FUNCTIONAL DESCRIPTION:
0070 456 :
0070 457 : Truncate an F_floating to an unsigned longword, without overflow.
0070 458 :
0070 459 : CALLING SEQUENCE:
0070 460 :
0070 461 : Result.wlu.v = PAS$TRUNC_F_LU (Single.rf.r)
0070 462 :
0070 463 : FORMAL PARAMETERS:
0070 464 :
0070 465 : Single - F_floating argument
0070 466 :
0070 467 : IMPLICIT INPUTS:
0070 468 :
0070 469 : NONE
0070 470 :
0070 471 : IMPLICIT OUTPUTS:
0070 472 :
0070 473 : NONE
0070 474 :
0070 475 : ROUTINE VALUE:
0070 476 :
0070 477 : The argument truncated to the nearest integer modulus 2**32.
0070 478 :
0070 479 : SIDE EFFECTS:
0070 480 :
0070 481 : NONE
0070 482 :
0070 483 :--
0000 0070 484 .ENTRY PAS$TRUNC_F_LU, ^M<> ; IV must be disabled
50 04 BC 4A 0072 485
04 0072 486 CVTFL @4(AP), R0 ; Truncate
0076 487 RET ; End of routine PAS$TRUNC_F_LU

```



```

0077 489      .SBTTL PAS$TRUNC_D_LU - Truncate D_floating to unsigned longword
0077 490      :++
0077 491      : FUNCTIONAL DESCRIPTION:
0077 492      :
0077 493      :     Truncate a D_floating to an unsigned longword, without overflow.
0077 494      :
0077 495      : CALLING SEQUENCE:
0077 496      :
0077 497      :     Result.wlu.v = PAS$TRUNC_D_LU (Double.rd.r)
0077 498      :
0077 499      : FORMAL PARAMETERS:
0077 500      :
0077 501      :     Double - D_floating argument
0077 502      :
0077 503      : IMPLICIT INPUTS:
0077 504      :
0077 505      :     NONE
0077 506      :
0077 507      : IMPLICIT OUTPUTS:
0077 508      :
0077 509      :     NONE
0077 510      :
0077 511      : ROUTINE VALUE:
0077 512      :
0077 513      :     The argument truncated to the nearest integer modulus 2**32.
0077 514      :
0077 515      : SIDE EFFECTS:
0077 516      :
0077 517      :     NONE
0077 518      :
0077 519      :--
0000 0077 520      .ENTRY PAS$TRUNC_D_LU, ^M<>      ; IV must be disabled
0079 521
50 04 BC 6A 0079 522      CVTDL @4(AP), R0      ; Truncate
04 007D 523      RET      ; End of routine PAS$TRUNC_D_LU

```



```
007E 525      .SBTTL PAS$TRUNC_G_LU - Truncate G_floating to unsigned longword
007E 526      :++
007E 527      : FUNCTIONAL DESCRIPTION:
007E 528      :
007E 529      :     Truncate a G_floating to an unsigned longword, without overflow.
007E 530      :
007E 531      : CALLING SEQUENCE:
007E 532      :
007E 533      :     Result.wlu.v = PAS$TRUNC_G_LU (Double.rg.r)
007E 534      :
007E 535      : FORMAL PARAMETERS:
007E 536      :
007E 537      :     Double - G_floating argument
007E 538      :
007E 539      : IMPLICIT INPUTS:
007E 540      :
007E 541      :     NONE
007E 542      :
007E 543      : IMPLICIT OUTPUTS:
007E 544      :
007E 545      :     NONE
007E 546      :
007E 547      : ROUTINE VALUE:
007E 548      :
007E 549      :     The argument truncated to the nearest integer modulus 2**32.
007E 550      :
007E 551      : SIDE EFFECTS:
007E 552      :
007E 553      :     NONE
007E 554      :
007E 555      :--
0000 007E 556      .ENTRY PAS$TRUNC_G_LU, ^M<>      ; IV must be disabled
0080 557
50 04 BC 4AFD 0080 558      CVTGL @4(AP), R0      ; Truncate
04 0085 559      RET      ; End of routine PAS$TRUNC_G_LU
```



```
0086 561 .SBTTL PAS$TRUNC_H_LU - Truncate H_floating to unsigned longword
0086 562 :++
0086 563 : FUNCTIONAL DESCRIPTION:
0086 564 :
0086 565 : Truncate an H_floating to an unsigned longword, without overflow.
0086 566 :
0086 567 : CALLING SEQUENCE:
0086 568 :
0086 569 : Result.wlu.v = PAS$TRUNC_H_LU (Quad.rh.r)
0086 570 :
0086 571 : FORMAL PARAMETERS:
0086 572 :
0086 573 : Quad - H_floating argument
0086 574 :
0086 575 : IMPLICIT INPUTS:
0086 576 :
0086 577 : NONE
0086 578 :
0086 579 : IMPLICIT OUTPUTS:
0086 580 :
0086 581 : NONE
0086 582 :
0086 583 : ROUTINE VALUE:
0086 584 :
0086 585 : The argument truncated to the nearest integer modulus 2**32.
0086 586 :
0086 587 : SIDE EFFECTS:
0086 588 :
0086 589 : NONE
0086 590 :
0086 591 :--
0000 0086 592 .ENTRY PAS$TRUNC_H_LU, ^M<> ; IV must be disabled
50 04 BC 6AFD 0088 593
04 0088 594 CVTHL @4(AP), R0 ; Truncate
008D 595 RET ; End of routine PAS$TRUNC_H_LU
```

```

008E 597      .SBTTL PASS$CONVERT_LU_F = Convert unsigned longword to F_floating
008E 598      ;++
008E 599      : FUNCTIONAL DESCRIPTION:
008E 600      :
008E 601      :     Converts an unsigned longword to an F_floating, rounded.
008E 602      :
008E 603      : CALLING SEQUENCE:
008E 604      :
008E 605      :     Result.wf.v = PASS$CONVERT_LU_F (Long.rlu.r)
008E 606      :
008E 607      : FORMAL PARAMETERS:
008E 608      :
008E 609      :     Long      - Unsigned longword argument
008E 610      :
008E 611      : IMPLICIT INPUTS:
008E 612      :
008E 613      :     NONE
008E 614      :
008E 615      : IMPLICIT OUTPUTS:
008E 616      :
008E 617      :     NONE
008E 618      :
008E 619      : ROUTINE VALUE:
008E 620      :
008E 621      :     The argument rounded to the nearest F_floating value.
008E 622      :
008E 623      : SIDE EFFECTS:
008E 624      :
008E 625      :     NONE
008E 626      :
008E 627      :--
008E 628
4000 008E 629      .ENTRY PASS$CONVERT_LU_F, ^M<IV> ; Entry point
0090 630
50   04 BC  6E 0090 631      CVTLD  @4(AP), R0          ; Convert to D_floating
0094 632      BGEQ  10$          ; If not negative, don't compensate
50   00000000 00005080 8F 60 0096 633      ADDD2  #^X5080, R0      ; Add 2**32
50   50  76 00A1 634 10$:  CVTDF  R0, R0          ; Round to F_floating
04   00A4 635      RET          ; End of routine PASS$CONVERT_LU_F
  
```



```

00A5 637 .SBTTL PASS$CONVERT_F_F = Convert F_floating to F_floating
00A5 638 :++
00A5 639 : FUNCTIONAL DESCRIPTION:
00A5 640 :
00A5 641 :     Converts an F_floating to an F_floating.
00A5 642 :
00A5 643 : CALLING SEQUENCE:
00A5 644 :
00A5 645 :     Result.wf.v = PASS$CONVERT_F_F (Single.rf.r)
00A5 646 :
00A5 647 : FORMAL PARAMETERS:
00A5 648 :
00A5 649 :     Single - F_floating argument
00A5 650 :
00A5 651 : IMPLICIT INPUTS:
00A5 652 :
00A5 653 :     NONE
00A5 654 :
00A5 655 : IMPLICIT OUTPUTS:
00A5 656 :
00A5 657 :     NONE
00A5 658 :
00A5 659 : ROUTINE VALUE:
00A5 660 :
00A5 661 :     The argument converted to F_floating
00A5 662 :
00A5 663 : SIDE EFFECTS:
00A5 664 :
00A5 665 :     SS$_ROPRAND - if the argument is a reserved operand
00A5 666 :
00A5 667 :--
00A5 668
4000 00A5 669 .ENTRY PASS$CONVERT_F_F, ^M<IV> ; Entry point
00A7 670
50 04 BC 50 00A7 671 MOVF @4(AP), R0 ; Convert
04 00AB 672 RET ; End of routine PASS$CONVERT_F_F

```

```

00AC 674      .SBTTL PASS$CONVERT_H_F = Convert H_floating to F_floating
00AC 675      :++
00AC 676      : FUNCTIONAL DESCRIPTION:
00AC 677      :
00AC 678      :     Converts an H_floating to an F_floating.
00AC 679      :
00AC 680      : CALLING SEQUENCE:
00AC 681      :
00AC 682      :     Result.wf.v = PASS$CONVERT_H_F (Quad.rh.r)
00AC 683      :
00AC 684      : FORMAL PARAMETERS:
00AC 685      :
00AC 686      :     Quad      - H_floating argument
00AC 687      :
00AC 688      : IMPLICIT INPUTS:
00AC 689      :
00AC 690      :     NONE
00AC 691      :
00AC 692      : IMPLICIT OUTPUTS:
00AC 693      :
00AC 694      :     NONE
00AC 695      :
00AC 696      : ROUTINE VALUE:
00AC 697      :
00AC 698      :     The argument converted to F_floating
00AC 699      :
00AC 700      : SIDE EFFECTS:
00AC 701      :
00AC 702      :     $$$_ROPRAND - if the argument is a reserved operand
00AC 703      :
00AC 704      :--
00AC 705
4000 00AC 706      .ENTRY PASS$CONVERT_H_F, ^M<IV> ; Entry point
00AE 707
50  04 BC F6FD 00AE 708      CVTHF @4(AP), R0 ; Convert
      04 00B3 709      RET ; End of routine PASS$CONVERT_H_F
  
```



```

00B4 711      .SBTTL PASS$CONVERT_LU_D = Convert unsigned longword to D_floating
00B4 712      :++
00B4 713      : FUNCTIONAL DESCRIPTION:
00B4 714      :
00B4 715      :     Converts an unsigned longword to a D_floating.
00B4 716      :
00B4 717      : CALLING SEQUENCE:
00B4 718      :
00B4 719      :     Result.wd.v = PASS$CONVERT_LU_D (Long.rlu.r)
00B4 720      :
00B4 721      : FORMAL PARAMETERS:
00B4 722      :
00B4 723      :     Long      - Unsigned longword argument
00B4 724      :
00B4 725      : IMPLICIT INPUTS:
00B4 726      :
00B4 727      :     NONE
00B4 728      :
00B4 729      : IMPLICIT OUTPUTS:
00B4 730      :
00B4 731      :     NONE
00B4 732      :
00B4 733      : ROUTINE VALUE:
00B4 734      :
00B4 735      :     The argument converted to a D_floating value.
00B4 736      :
00B4 737      : SIDE EFFECTS:
00B4 738      :
00B4 739      :     NONE
00B4 740      :
00B4 741      :--
00B4 742
4000 00B4 743      .ENTRY PASS$CONVERT_LU_D, ^M<IV> ; Entry point
00B6 744
50   04 BC 6E 00B6 745      CVTLD  @4(AP), R0          ; Convert to D_floating
00B8 746      BGEQ  10$          ; If not negative, don't compensate
50   00000000 00005080 8F 60 00BC 747      ADDD2  #^X5080, R0      ; Add 2**32
04   00C7 748 10$:      RET          ; End of routine PASS$CONVERT_LU_D
  
```

```

00C8 750      .SBTTL PASS$CONVERT_D_D = Convert D_floating to D_floating
00C8 751      :++
00C8 752      : FUNCTIONAL DESCRIPTION:
00C8 753      :
00C8 754      :     Converts an D_floating to a D_floating.
00C8 755      :
00C8 756      : CALLING SEQUENCE:
00C8 757      :
00C8 758      :     Result.wd.v = PASS$CONVERT_D_D (Double.rd.r)
00C8 759      :
00C8 760      : FORMAL PARAMETERS:
00C8 761      :
00C8 762      :     Double - D_floating argument
00C8 763      :
00C8 764      : IMPLICIT INPUTS:
00C8 765      :
00C8 766      :     NONE
00C8 767      :
00C8 768      : IMPLICIT OUTPUTS:
00C8 769      :
00C8 770      :     NONE
00C8 771      :
00C8 772      : ROUTINE VALUE:
00C8 773      :
00C8 774      :     The argument converted to D_floating
00C8 775      :
00C8 776      : SIDE EFFECTS:
00C8 777      :
00C8 778      :     $$$_ROPRAND - if the argument is a reserved operand
00C8 779      :
00C8 780      :--
00C8 781
4000 00C8 782      .ENTRY PASS$CONVERT_D_D, ^M<IV> ; Entry point
00CA 783
50  04 BC 70 00CA 784      MOVD    @4(AP), R0      ; Convert
04  00CE 785      RET                ; End of routine PASS$CONVERT_D_D
  
```



```

00CF 787 .SBTTL PASS$CONVERT_H_D = Convert H_floating to D_floating
00CF 788 :++
00CF 789 : FUNCTIONAL DESCRIPTION:
00CF 790 :
00CF 791 :     Converts an H_floating to a D_floating.
00CF 792 :
00CF 793 : CALLING SEQUENCE:
00CF 794 :
00CF 795 :     Result.wd.v = PASS$CONVERT_H_D (Quad.rh.r)
00CF 796 :
00CF 797 : FORMAL PARAMETERS:
00CF 798 :
00CF 799 :     Quad    - H_floating argument
00CF 800 :
00CF 801 : IMPLICIT INPUTS:
00CF 802 :
00CF 803 :     NONE
00CF 804 :
00CF 805 : IMPLICIT OUTPUTS:
00CF 806 :
00CF 807 :     NONE
00CF 808 :
00CF 809 : ROUTINE VALUE:
00CF 810 :
00CF 811 :     The argument converted to D_floating
00CF 812 :
00CF 813 : SIDE EFFECTS:
00CF 814 :
00CF 815 :     SSS$ROPRAND - if the argument is a reserved operand
00CF 816 :
00CF 817 : --
00CF 818 :
4000 00CF 819 .ENTRY PASS$CONVERT_H_D, ^M<IV> ; Entry point
00D1 820
50 04 BC F7FD 00D1 821 CVTHD @4(AP), R0 ; Convert
04 00D6 822 RET ; End of routine PASS$CONVERT_H_D

```

```

00D7 824 .SBTTL PASS$CONVERT_LU_G = Convert unsigned longword to G_floating
00D7 825 :++
00D7 826 : FUNCTIONAL DESCRIPTION:
00D7 827 :
00D7 828 :     Converts an unsigned longword to a G_floating.
00D7 829 :
00D7 830 : CALLING SEQUENCE:
00D7 831 :
00D7 832 :     Result.wg.v = PASS$CONVERT_LU_G (Long.rlu.r)
00D7 833 :
00D7 834 : FORMAL PARAMETERS:
00D7 835 :
00D7 836 :     Long      - Unsigned longword argument
00D7 837 :
00D7 838 : IMPLICIT INPUTS:
00D7 839 :
00D7 840 :     NONE
00D7 841 :
00D7 842 : IMPLICIT OUTPUTS:
00D7 843 :
00D7 844 :     NONE
00D7 845 :
00D7 846 : ROUTINE VALUE:
00D7 847 :
00D7 848 :     The argument converted to a G_floating value.
00D7 849 :
00D7 850 : SIDE EFFECTS:
00D7 851 :
00D7 852 :     NONE
00D7 853 :
00D7 854 : --
00D7 855 :
4000 00D7 856 .ENTRY PASS$CONVERT_LU_G, ^M<IV> ; Entry point
00D9 857
50 04 BC 4EFD 00D9 858 CCTLG @4(AP), R0 ; Convert to G_floating
OC 18 00DE 859 BGEQ 10$ ; If not negative, don't compensate
50 00000000 00004210 8F 40FD 00E0 860 ADDG2 #^X4210, R0 ; Add 2**32
04 00EC 861 10$: RET ; End of routine PASS$CONVERT_LU_G

```



```

00ED 863      .SBTTL PASS$CONVERT_G_G = Convert G_floating to G_floating
00ED 864      :++
00ED 865      : FUNCTIONAL DESCRIPTION:
00ED 866      :
00ED 867      :     Converts a G_floating to a G_floating.
00ED 868      :
00ED 869      : CALLING SEQUENCE:
00ED 870      :
00ED 871      :     Result.wg.v = PASS$CONVERT_G_G (Double.rg.r)
00ED 872      :
00ED 873      : FORMAL PARAMETERS:
00ED 874      :
00ED 875      :     Double - G_floating argument
00ED 876      :
00ED 877      : IMPLICIT INPUTS:
00ED 878      :
00ED 879      :     NONE
00ED 880      :
00ED 881      : IMPLICIT OUTPUTS:
00ED 882      :
00ED 883      :     NONE
00ED 884      :
00ED 885      : ROUTINE VALUE:
00ED 886      :
00ED 887      :     The argument converted to G_floating
00ED 888      :
00ED 889      : SIDE EFFECTS:
00ED 890      :
00ED 891      :     SS$_ROPRAND - if the argument is a reserved operand
00ED 892      :
00ED 893      : --
00ED 894      :
4000 00ED 895      .ENTRY PASS$CONVERT_G_G, ^M<IV> ; Entry point
00EF 896
50 04 BC 50FD 00EF 897      MOVG    @4(AP), R0      ; Convert
04 00F4 898      RET          ; End of routine PASS$CONVERT_G_G
  
```

```

00F5 900 .SBTTL PASS$CONVERT_H_G = Convert H_floating to G_floating
00F5 901 :++
00F5 902 : FUNCTIONAL DESCRIPTION:
00F5 903 :
00F5 904 :     Converts an H_floating to a G_floating.
00F5 905 :
00F5 906 : CALLING SEQUENCE:
00F5 907 :
00F5 908 :     Result.wg.v = PASS$CONVERT_H_G (Quad.rh.r)
00F5 909 :
00F5 910 : FORMAL PARAMETERS:
00F5 911 :
00F5 912 :     Quad      - H_floating argument
00F5 913 :
00F5 914 : IMPLICIT INPUTS:
00F5 915 :
00F5 916 :     NONE
00F5 917 :
00F5 918 : IMPLICIT OUTPUTS:
00F5 919 :
00F5 920 :     NONE
00F5 921 :
00F5 922 : ROUTINE VALUE:
00F5 923 :
00F5 924 :     The argument converted to G_floating
00F5 925 :
00F5 926 : SIDE EFFECTS:
00F5 927 :
00F5 928 :     SS$_ROPRAND - if the argument is a reserved operand
00F5 929 :
00F5 930 : --
00F5 931 :
00F5 932 .ENTRY PASS$CONVERT_H_G, ^M<IV> ; Entry point
00F7 933
00F7 934 CVTHG @4(AP), R0 ; Convert
00FC 935 RET ; End of routine PASS$CONVERT_H_G

```



```

00FD 937      .SBTTL PASS$CONVERT_L_H = Convert signed longword to H_floating
00FD 938      :++
00FD 939      : FUNCTIONAL DESCRIPTION:
00FD 940      :
00FD 941      :     Converts a signed longword to an H_floating.
00FD 942      :
00FD 943      : CALLING SEQUENCE:
00FD 944      :
00FD 945      :     CALL PASS$CONVERT_L_H (Result.wh.r, Long.rl.r)
00FD 946      :
00FD 947      : FORMAL PARAMETERS:
00FD 948      :
00FD 949      :     Result  - H_floating result
00FD 950      :     Long    - Signed longword argument
00FD 951      :
00FD 952      : IMPLICIT INPUTS:
00FD 953      :
00FD 954      :     NONE
00FD 955      :
00FD 956      : IMPLICIT OUTPUTS:
00FD 957      :
00FD 958      :     NONE
00FD 959      :
00FD 960      : ROUTINE VALUE:
00FD 961      :
00FD 962      :     NONE
00FD 963      :
00FD 964      : SIDE EFFECTS:
00FD 965      :
00FD 966      :     NONE
00FD 967      :
00FD 968      :--
00FD 969      :
00FD 970      :.ENTRY PASS$CONVERT_L_H, ^M<IV> ; Entry point
00FF 971
04 BC 08 BC 6EFD 00FF 972      CVTLH  @8(AP), @4(AP) ; Convert
04 0105 973      RET ; End of routine PASS$CONVERT_L_H
  
```

```

0106 975      .SBTTL PASS$CONVERT_LU_H = Convert unsigned longword to H_floating
0106 976      :++
0106 977      : FUNCTIONAL DESCRIPTION:
0106 978      :
0106 979      :     Converts an unsigned longword to an H_floating.
0106 980      :
0106 981      : CALLING SEQUENCE:
0106 982      :
0106 983      :     CALL PASS$CONVERT_LU_H (Result.wh.t, Long.rlu.r)
0106 984      :
0106 985      : FORMAL PARAMETERS:
0106 986      :
0106 987      :     Result - H_floating result
0106 988      :     Long   - Unsigned longword argument
0106 989      :
0106 990      : IMPLICIT INPUTS:
0106 991      :
0106 992      :     NONE
0106 993      :
0106 994      : IMPLICIT OUTPUTS:
0106 995      :
0106 996      :     NONE
0106 997      :
0106 998      : ROUTINE VALUE:
0106 999      :
0106 1000     :     NONE
0106 1001     :
0106 1002     : SIDE EFFECTS:
0106 1003     :
0106 1004     :     NONE
0106 1005     :
0106 1006     : --
0106 1007     :
0106 1008     .ENTRY PASS$CONVERT_LU_H, ^M<IV> ; Entry point
0108 1009
0108 1010     CVTLH  @8(AP), @4(AP)      ; Convert to H_floating
010E 1011     BGEQ   10$                ; If not negative, don't compensate
0110 1012     ADDH2  #^X4021, @4(AP)    ; Add 2**32
0112
011F
0125 1013 10$: RET                      ; End of routine PASS$CONVERT_LU_H
  
```

4000

04 BC 08 BC 6EFD

15 18

60FD

00000000 00000000 00004021 8F

04 BC 00000000

04


```

0126 1015      .SBTTL PASS$CONVERT_F_H = Convert F_floating to H_floating
0126 1016      :++
0126 1017      : FUNCTIONAL DESCRIPTION:
0126 1018      :
0126 1019      :     Converts an F_floating to an H_floating.
0126 1020      :
0126 1021      : CALLING SEQUENCE:
0126 1022      :
0126 1023      :     CALL PASS$CONVERT_F_H (Result.wh.r, Single.rf.r)
0126 1024      :
0126 1025      : FORMAL PARAMETERS:
0126 1026      :
0126 1027      :     Result - H_floating result
0126 1028      :     Single - F_floating argument
0126 1029      :
0126 1030      : IMPLICIT INPUTS:
0126 1031      :
0126 1032      :     NONE
0126 1033      :
0126 1034      : IMPLICIT OUTPUTS:
0126 1035      :
0126 1036      :     NONE
0126 1037      :
0126 1038      : ROUTINE VALUE:
0126 1039      :
0126 1040      :     NONE
0126 1041      :
0126 1042      : SIDE EFFECTS:
0126 1043      :
0126 1044      :     SS$_ROPRAND - if the argument is a reserved operand
0126 1045      :
0126 1046      :--
0126 1047
0126 1048      .ENTRY PASS$CONVERT_F_H, ^M<IV> ; Entry point
0128 1049
0128 1050      CVTFH @8(AP), @4(AP) ; Convert
012E 1051      RET ; End of routine PASS$CONVERT_F_H
  
```

04 BC 08 BC 98FD 4000


```

012F 1053      .SBTTL PASS$CONVERT_D_H = Convert D_floating to H_floating
012F 1054      :++
012F 1055      : FUNCTIONAL DESCRIPTION:
012F 1056      :
012F 1057      :     Converts a D_floating to an H_floating.
012F 1058      :
012F 1059      : CALLING SEQUENCE:
012F 1060      :
012F 1061      :     CALL PASS$CONVERT_D_H (Result.wh.r, Double.rd.r)
012F 1062      :
012F 1063      : FORMAL PARAMETERS:
012F 1064      :
012F 1065      :     Result - H_floating result
012F 1066      :     Double  - D_floating argument
012F 1067      :
012F 1068      : IMPLICIT INPUTS:
012F 1069      :
012F 1070      :     NONE
012F 1071      :
012F 1072      : IMPLICIT OUTPUTS:
012F 1073      :
012F 1074      :     NONE
012F 1075      :
012F 1076      : ROUTINE VALUE:
012F 1077      :
012F 1078      :     NONE
012F 1079      :
012F 1080      : SIDE EFFECTS:
012F 1081      :
012F 1082      :     SS$_ROPRAND - if the argument is a reserved operand
012F 1083      :
012F 1084      : --
012F 1085      :
012F 1086      : .ENTRY PASS$CONVERT_D_H, ^M<IV> ; Entry point
0131 1087
04 BC 08 BC 32FD 0131 1088      CVTDH  @8(AP), @4(AP) ; Convert
04 0137 1089      RET ; End of routine PASS$CONVERT_D_H
  
```



```

0138 1091      .SBTTL PASS$CONVERT_G_H = Convert G_floating to H_floating
0138 1092      :++
0138 1093      : FUNCTIONAL DESCRIPTION:
0138 1094      :
0138 1095      :     Converts a G_floating to an H_floating.
0138 1096      :
0138 1097      : CALLING SEQUENCE:
0138 1098      :
0138 1099      :     CALL PASS$CONVERT_G_H (Result.wh.r, Double.rg.r)
0138 1100      :
0138 1101      : FORMAL PARAMETERS:
0138 1102      :
0138 1103      :     Result - H_floating result
0138 1104      :     Double - G_floating argument
0138 1105      :
0138 1106      : IMPLICIT INPUTS:
0138 1107      :
0138 1108      :     NONE
0138 1109      :
0138 1110      : IMPLICIT OUTPUTS:
0138 1111      :
0138 1112      :     NONE
0138 1113      :
0138 1114      : ROUTINE VALUE:
0138 1115      :
0138 1116      :     NONE
0138 1117      :
0138 1118      : SIDE EFFECTS:
0138 1119      :
0138 1120      :     SS$_ROPRAND - if the argument is a reserved operand
0138 1121      :
0138 1122      :--
0138 1123
0138 1124      .ENTRY PASS$CONVERT_G_H, ^M<IV> ; Entry point
013A 1125
013A 1126      CVTGH @8(AP), @4(AP) ; Convert
0140 1127      RET ; End of routine PASS$CONVERT_G_H
  
```

04 BC 08 BC 56FD 04 4000

```

- Pascal-specific conversion routinesE 2 16-SEP-1984 01:23:05 VAX/VMS Macro V04-00 Page 31
PASS$CONVERT_H_H = Convert H_floating to 6-SEP-1984 11:30:12 [PASRTL.SRC]PASCONVER.MAR;1 (31)

```

```

0141 1129 .SBTTL PASS$CONVERT_H_H = Convert H_floating to H_floating
0141 1130 :++
0141 1131 : FUNCTIONAL DESCRIPTION:
0141 1132 :
0141 1133 :     Converts an H_floating to an H_floating.
0141 1134 :
0141 1135 : CALLING SEQUENCE:
0141 1136 :
0141 1137 :     CALL PASS$CONVERT_H_H (Result.wh.r, Quad.rh.r)
0141 1138 :
0141 1139 : FORMAL PARAMETERS:
0141 1140 :
0141 1141 :     Result - H_floating result
0141 1142 :     Quad   - H_floating argument
0141 1143 :
0141 1144 : IMPLICIT INPUTS:
0141 1145 :
0141 1146 :     NONE
0141 1147 :
0141 1148 : IMPLICIT OUTPUTS:
0141 1149 :
0141 1150 :     NONE
0141 1151 :
0141 1152 : ROUTINE VALUE:
0141 1153 :
0141 1154 :     NONE
0141 1155 :
0141 1156 : SIDE EFFECTS:
0141 1157 :
0141 1158 :     SS$_ROPRAND - if the argument is a reserved operand
0141 1159 :
0141 1160 :--
0141 1161 :
0141 1162 .ENTRY PASS$CONVERT_H_H, ^M<IV> ; Entry point
0143 1163
0143 1164 MOVH    @8(AP), @4(AP)           ; Convert
0149 1165 RET                               ; End of routine PASS$CONVERT_H_H
014A 1166
014A 1167 .END                               ; End of module PASS$CONVERT

```

PAS
1-0

PASS\$CONVERT
Symbol table

- Pascal-specific conversion routines

16-SEP-1984 01:23:05 VAX/VMS Macro V04-00 Page 32
6-SEP-1984 11:30:12 [PASRTL.SRC]PASCONVER.MAR;1 (31)

```
LIB$SIGNAL
PASS$CONVERT_BOOL_L
PASS$CONVERT_BU_L
PASS$CONVERT_D_D
PASS$CONVERT_D_H
PASS$CONVERT_F_F
PASS$CONVERT_F_H
PASS$CONVERT_G_G
PASS$CONVERT_G_H
PASS$CONVERT_H_D
PASS$CONVERT_H_F
PASS$CONVERT_H_G
PASS$CONVERT_H_H
PASS$CONVERT_LD_D
PASS$CONVERT_LU_F
PASS$CONVERT_LU_G
PASS$CONVERT_LU_H
PASS$CONVERT_LU_L
PASS$CONVERT_L_BU
PASS$CONVERT_L_H
PASS$CONVERT_L_L
PASS$CONVERT_WD_L
PASS$ROUND_D_LU
PASS$ROUND_F_LU
PASS$ROUND_G_LU
PASS$ROUND_H_LU
PASS$TRUNC_D_LU
PASS$TRUNC_F_LU
PASS$TRUNC_G_LU
PASS$TRUNC_H_LU
SS$_INTOVF
***** X 00
00000047 RG 02
0000001D RG 02
000000C8 RG 02
0000012F RG 02
000000A5 RG 02
00000126 RG 02
000000ED RG 02
00000138 RG 02
000000CF RG 02
000000AC RG 02
000000F5 RG 02
00000141 RG 02
000000B4 RG 02
0000008E RG 02
000000D7 RG 02
00000106 RG 02
00000032 RG 02
00000000 RG 02
000000FD RG 02
0000002B RG 02
00000024 RG 02
00000059 RG 02
00000052 RG 02
00000060 RG 02
00000068 RG 02
00000077 RG 02
00000070 RG 02
0000007E RG 02
00000086 RG 02
= 0000047C
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 (0.)	00 (0.)	NOPIC USR
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR
_PASS\$CODE	0000014A (330.)	02 (2.)	PIC USR

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	10	00:00:00.09	00:00:00.97
Command processing	72	00:00:00.60	00:00:03.83
Pass 1	200	00:00:04.92	00:00:18.87
Symbol table sort	0	00:00:00.67	00:00:01.82
Pass 2	190	00:00:02.35	00:00:07.95
Symbol table output	5	00:00:00.04	00:00:00.09
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	481	00:00:08.70	00:00:33.56

The working set limit was 1200 pages.
30776 bytes (61 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 437 non-local and 6 local symbols.
1167 source lines were read in Pass 1, producing 97 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4

469 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:PASCONVER/OBJ=OBJ\$:PASCONVER MSRC\$:PASCONVER/UPDATE=(ENH\$:PASCONVER)

PAS
VAX

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

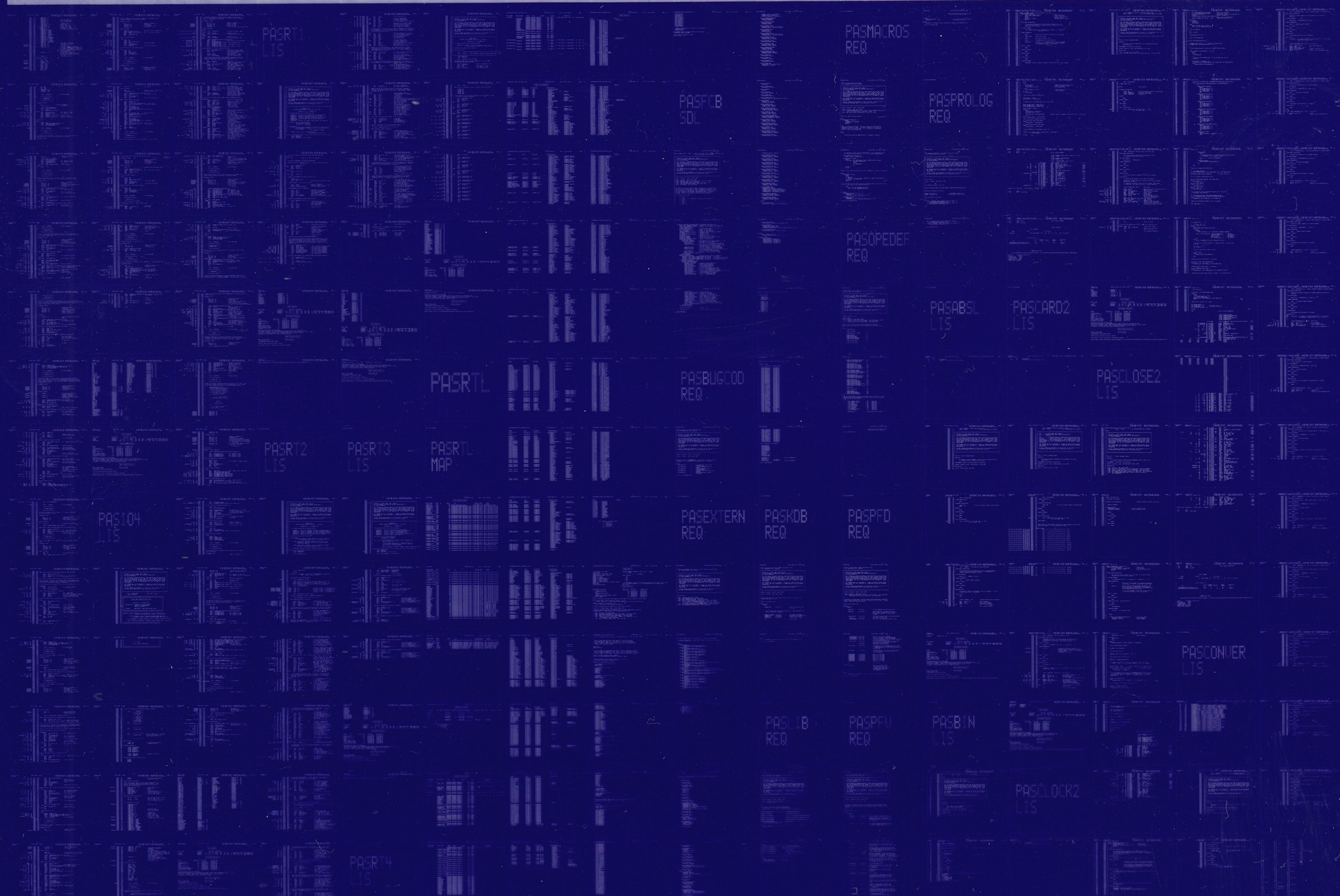
The
746
The
458
0 p

Mac

_\$2
0 G
The
MAC

0293 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0294 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY