```
PPPPPPPPPPPP      LLL                IIIIIIIII    RRRRRRRRRRRR      TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
```

```
PPPPPPPP    LL            IIIIII      CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR
PPPPPPPP    LL            IIIIII      CCCCCCCC  HH      HH    AAAAAA    RRRRRRRR
PP      PP  LL              II      CC          HH      HH  AA      AA  RR      RR
PP      PP  LL              II      CC          HH      HH  AA      AA  RR      RR
PP      PP  LL              II      CC          HH      HH  AA      AA  RR      RR
PP      PP  LL              II      CC          HH      HH  AA      AA  RR      RR
PPPPPPPP    LL              II      CC          HHHHHHHHHH  AA      AA  RRRRRRRR
PPPPPPPP    LL              II      CC          HHHHHHHHHH  AA      AA  RRRRRRRR
PP          LL              II      CC          HH      HH  AAAAAAAAAA  RR  RR
PP          LL              II      CC          HH      HH  AAAAAAAAAA  RR  RR
PP          LL              II      CC          HH      HH  AA      AA  RR    RR      ....
PP          LL              II      CC          HH      HH  AA      AA  RR    RR      ....
PP          LLLLLLLLLL    IIIIII      CCCCCCCC  HH      HH  AA      AA  RR      RR    ....
PP          LLLLLLLLLL    IIIIII      CCCCCCCC  HH      HH  AA      AA  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
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
                        .title  pli$char - pl1 runtime character routines
                        .ident  /1-002/                                         ; Edit CGN1003
                                                                                ; Edit WHM1002
                ;
                ;*****************************************************************************
                ;*                                                                           *
                ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                ;*  ALL RIGHTS RESERVED.                                                     *
                ;*                                                                           *
                ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
                ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
                ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
                ;*  TRANSFERRED.                                                             *
                ;*                                                                           *
                ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
                ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
                ;*  CORPORATION.                                                             *
                ;*                                                                           *
                ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
                ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                ;*                                                                           *
                ;*                                                                           *
                ;*****************************************************************************
                ;
                ;++
                ; facility:
                ;
                ;       VAX/VMS PL1 runtime library.
                ;
                ; abstract:
                ;
                ;       This module contains runtime routines for character built-in functions.
                ;
                ; author: r. heinen 26-feb-1979
                ;
                ; Modifications:
                ;
                ;       1-002   Bill Matthews   29-September-1982
                ;
                ;               Invoke macros $defdat and rtshare instead of $defopr and share.
                ;
                ;       1-003   Add the pli$search routine to provide runtime support for
                ;               the PL/I search builtin function.
                ;
                ;--

                ;
                ; external definitions
                ;

                        $defdat                         ; define data types
                ;
                ; local data
```

```
0000     58 ;
0000     59
0000     60          rtshare
```

```
                        0000     62          .sbttl  subroutines
                        0000     63 ;++
                        0000     64 ; set_opnd1 - setup operand one
                        0000     65 ;
                        0000     66 ; functional description:
                        0000     67 ;
                        0000     68 ; This routine interprets operand 1 and it's dope vector and returns
                        0000     69 ; the information.
                        0000     70 ;
                        0000     71 ; inputs:
                        0000     72 ;
                        0000     73 ;       common stack frame
                        0000     74 ;
                        0000     75 ; outputs:
                        0000     76 ;
                        0000     77 ;       r3 = size of operand
                        0000     78 ;       r4 = address of the data
                        0000     79 ;--
                        0000     80 set_opnd1:
50    08 AC     D0      0000     81          movl    8(ap),r0                ; address dope vector
54    04 AC     D0      0004     82          movl    4(ap),r4                ; address string
   80    0A     B1      0008     83          cmpw    #dat_k_char,(r0)+       ; character type?
         04     13      000B     84          beql    10$                     ; if eql then character
   53    84     3C      000D     85          movzwl  (r4)+,r3                ; get size and address string
                05      0010     86          rsb
   53    60     3C      0011     87 10$:     movzwl  (r0),r3                 ; get size and address data
                05      0014     88          rsb
```

```
                                                L 12
                     0015     90 ;++
                     0015     91 ; set_opnd2 - setup operand two
                     0015     92 ;
                     0015     93 ; functional description:
                     0015     94 ;
                     0015     95 ; This routine interprets operand 2 and it's dope vector and returns
                     0015     96 ; the information.
                     0015     97 ;
                     0015     98 ; inputs:
                     0015     99 ;
                     0015    100 ;       common stack frame
                     0015    101 ;
                     0015    102 ; outputs:
                     0015    103 ;
                     0015    104 ;       r5 = size of operand
                     0015    105 ;       r6 = address of the data
                     0015    106 ;--
                     0015    107 set_opnd2:
50   10 AC    D0     0015    108         movl    16(ap),r0               ; address dope vector
56   0C AC    D0     0019    109         movl    12(ap),r6               ; address string
   80   0A    B1     001D    110         cmpw    #dat_k_char,(r0)+       ; character type?
        04    13     0020    111         beql    10$                     ; if eql then character
   55   86    3C     0022    112         movzwl  (r6)+,r5                ; get size and address string
              05     0025    113         rsb
   55   60    3C     0026    114 10$:    movzwl  (r0),r5                 ; get size and address data
              05     0029    115         rsb
```

```
                                                M 12
                     002A   118 ;++
                     002A   119 ; set_opnd3 - setup operand three
                     002A   120 ;
                     002A   121 ; functional description:
                     002A   122 ;
                     002A   123 ; This routine interprets operand 3 and it's dope vector and returns
                     002A   124 ; the information.
                     002A   125 ;
                     002A   126 ; inputs:
                     002A   127 ;
                     002A   128 ;       common stack frame
                     002A   129 ;
                     002A   130 ; outputs:
                     002A   131 ;
                     002A   132 ;       r7 = size of operand
                     002A   133 ;       r8 = address of the data
                     002A   134 ;--
                     002A   135 set_opnd3:
50   18 AC    D0     002A   136         movl    24(ap),r0               ; address dope vector
58   14 AC    D0     002E   137         movl    20(ap),r8               ; address string
   80   0A    B1     0032   138         cmpw    #dat_k_char,(r0)+       ; character type?
        04    13     0035   139         beql    10$                     ; if eql then character
   57   88    3C     0037   140         movzwl  (r8)+,r7                ; get size and address string
              05     003A   141         rsb
   57   60    3C     003B   142 10$:    movzwl  (r0),r7                 ; get size and address data
              05     003E   143         rsb
```

```
                                        003F   145          .sbttl  pli$movtranchar - move translated
                                        003F   146 ;++
                                        003F   147 ; pli$movtranchar - move translated characters
                                        003F   148 ;
                                        003F   149 ; functional description:
                                        003F   150 ;
                                        003F   151 ; This routine augments the in-line code for the translate bif.
                                        003F   152 ; The pl1 translate bif is more functional than the movtc instruction.
                                        003F   153 ;
                                        003F   154 ; string1 = translate(string2,string3,string4);
                                        003F   155 ;
                                        003F   156 ;               (two operand translate bifs are done by inline code)
                                        003F   157 ;
                                        003F   158 ;
                                        003F   159 ;       string1(i) = substr(string3,index(string4,string2(i))
                                        003F   160 ;
                                        003F   161 ; inputs:
                                        003F   162 ;
                                        003F   163 ;       r1 = address to return string of size 4(ap)
                                        003F   164 ;
                                        003F   165 ;       0(ap) = 6
                                        003F   166 ;       4(ap) = address of source
                                        003F   167 ;       8(ap) = address of source dope
                                        003F   168 ;       12(ap) = address of the translate table
                                        003F   169 ;       16(ap) = address of translate table dope
                                        003F   170 ;       20(ap) = address of the translate string
                                        003F   171 ;       24(ap) = address of the translate string dope
                                        003F   172 ;
                                        003F   173 ; outputs:
                                        003F   174 ;
                                        003F   175 ;       result string is filled
                                        003F   176 ;
                                        003F   177 ;--
                                  01FC  003F   178          .entry pli$movtranchar,^m<r2,r3,r4,r5,r6,r7,r8>
                              52  51    D0   0041   179          movl    r1,r2                   ; save target address
                                FFB9    30   0044   180          bsbw    set_opnd1               ; setup operand 1
                                FFCB    30   0047   181          bsbw    set_opnd2               ; setup operand 2
                                FFDD    30   004A   182          bsbw    set_opnd3               ; setup operand 3
                                  53    D7   004D   183 10$:     decl    r3                      ; count string
                                  20    19   004F   184          blss    30$                     ; br if done
                          68    57    84   3A   0051   185          locc    (r4)+,r7,(r8)           ; look for character in table
                                  14    13   0055   186          beql    20$                     ; if eql then not found
                          50    57    50   C3   0057   187          subl3   r0,r7,r0                ; get offset in table
                                55    50   B1   005B   188          cmpw    r0,r5                   ; in range?
                                  05    1F   005E   189          blssu   15$                     ; if gtru then insert fill
                                82    20   90   0060   190          movb    #^a/ /,(r2)+            ; insert character in output
                                  E8    11   0063   191          brb     10$                     ; continue
                             82   6640   90   0065   192 15$:     movb    (r6)[r0],(r2)+          ; insert in output
                                  E2    11   0069   193          brb     10$                     ; continue
                           82   FF A4   90   006B   194 20$:     movb    -1(r4),(r2)+            ; insert source string char
                                  DC    11   006F   195          brb     10$                     ; continue
                                        04   0071   196 30$:     ret
```

```
                                0072   198          .sbttl  pli$verify - verify built-in function
                                0072   199 ;++
                                0072   200 ; pli$verify - pl1 verify built-in function
                                0072   201 ;
                                0072   202 ; functional description:
                                0072   203 ;
                                0072   204 ; This routine performs the verfiy built-in function.
                                0072   205 ;
                                0072   206 ; fixbin = verify(string1,string2);
                                0072   207 ;
                                0072   208 ;       fixbin = 0 if for each string1(i), string1(i) is in string2.
                                0072   209 ;       fixbin = i for the lowest i such that string1(i) is not in string2.
                                0072   210 ;
                                0072   211 ; inputs:
                                0072   212 ;
                                0072   213 ;       0(ap) = 4
                                0072   214 ;       4(ap) = source 1 address
                                0072   215 ;       8(ap) = address of the dope for source 1
                                0072   216 ;       12(ap) = address of source 2
                                0072   217 ;       16(ap) = address of the dope for source 2
                                0072   218 ;
                                0072   219 ; outputs:
                                0072   220 ;
                                0072   221 ;       r0 = index
                                0072   222 ;--
                          007C  0072   223          .entry pli$verify,^m<r2,r3,r4,r5,r6>
                    FF89   30   0074   224          bsbw    set_opnd1                       ; setup operand 1
                    FF9B   30   0077   225          bsbw    set_opnd2                       ; setup operand 2
                      54   DD   007A   226          pushl   r4                              ; save string starting address
                      53   D7   007C   227 10$:     decl    r3                              ; adjust count
                      0B   19   007E   228          blss    20$                             ; if lss then all match, fixbin = 0
                66    55   84   3A   0080   229          locc    (r4)+,r5,(r6)                   ; look for next character
                           F6   12   0084   230          bneq    10$                             ; if found continue
                50    54   6E   C3   0086   231          subl3   (sp),r4,r0                      ; calc index
                                04   008A   232          ret
                           50   D4   008B   233 20$:     clrl    r0                              ; return
                                04   008D   234          ret                             ;
```

```
                                008E   236          .sbttl  pli$search - search built-in function
                                008E   237 ;++
                                008E   238 ; pli$search - pl1 search built-in function
                                008E   239 ;
                                008E   240 ; functional description:
                                008E   241 ;
                                008E   242 ; This routine performs the search built-in function.
                                008E   243 ;
                                008E   244 ; fixbin = search(string1,string2);
                                008E   245 ;
                                008E   246 ;       fixbin = 0 if for each string1(i), no string1(i) is in string2.
                                008E   247 ;       fixbin = i for the lowest i such that string1(i) is in string2.
                                008E   248 ;
                                008E   249 ; inputs:
                                008E   250 ;
                                008E   251 ;       0(ap) = 4
                                008E   252 ;       4(ap) = source 1 address
                                008E   253 ;       8(ap) = address of the dope for source 1
                                008E   254 ;       12(ap) = address of source 2
                                008E   255 ;       16(ap) = address of the dope for source 2
                                008E   256 ;
                                008E   257 ; outputs:
                                008E   258 ;
                                008E   259 ;       r0 = index
                                008E   260 ;--
                         007C   008E   261          .entry pli$search,^m<r2,r3,r4,r5,r6>
                    FF6D   30   0090   262          bsbw    set_opnd1                       ; setup operand 1
                    FF7F   30   0093   263          bsbw    set_opnd2                       ; setup operand 2
                      54   DD   0096   264          pushl   r4                              ; save string starting address
                      53   D7   0098   265 10$:     decl    r3                              ; adjust count
                      0B   19   009A   266          blss    20$                             ; if lss then none match, fixbin = 0
                66    55   84   3A   009C   267          locc    (r4)+,r5,(r6)               ; look for next character
                           F6   13   00A0   268          beql    10$                             ; if not found continue
                50    54   6E   C3   00A2   269          subl3   (sp),r4,r0                  ; calc index
                                04   00A6   270          ret
                           50   D4   00A7   271 20$:     clrl    r0                          ; return
                                04   00A9   272          ret                             ;
```

```
00AA   274          .end
```

```
DAT_K_CHAR      = 0000000A
PLI$MOVTRANCHAR   0000003F RG    01
PLI$SEARCH        0000008E RG    01
PLI$VERIFY        00000072 RG    01
SET_OPND1         00000000 R     01
SET_OPND2         00000015 R     01
SET_OPND3         0000002A R     01
```

Psect synopsis

```
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
_PLI$CODE                       000000AA  (  170.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.02	00:00:02.75
Command processing	78	00:00:00.47	00:00:03.95
Pass 1	68	00:00:00.77	00:00:05.33
Symbol table sort	0	00:00:00.02	00:00:00.02
Pass 2	53	00:00:00.51	00:00:03.69
Symbol table output	1	00:00:00.02	00:00:00.02
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	210	00:00:01.83	00:00:15.79

```
The working set limit was 750 pages.
4513 bytes (9 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 33 non-local and 11 local symbols.
274 source lines were read in Pass 1, producing 16 object records in Pass 2.
3 pages of virtual memory were used to define 2 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	2

```
32 GETS were required to define 2 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLICHAR/OBJ=OBJ$:PLICHAR MSRC$:PLICHAR/UPDATE=(ENH$:PLICHAR)+LIB$:PLIRTMAC/LIB
```

PLICASERR
LIS

PLICONDIT
LIS

DATDEF
SDL

FILEDEF
IN

PLIRTDEF
MAR

PLICHRSTR
LIS

PLICHAR
LIS

PLICLOSE
LIS

PLIRTL

PLIFILDSP
IN

PLIBYTSIZ
LIS

PLIRTL
MAP

ENVCODES
IN

PLIBIT
LIS