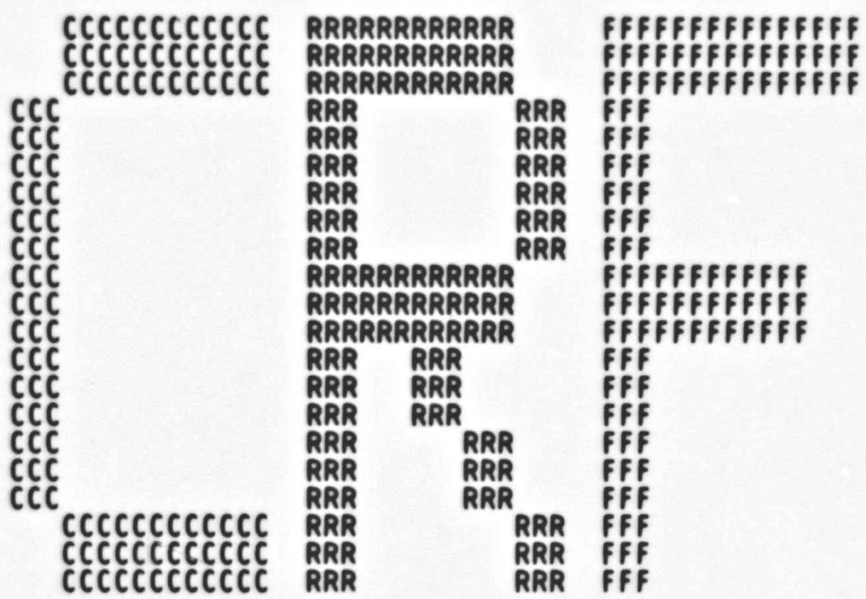
CCCCCCCCCCCC RRRRRRRRRRRR FFFFFFFFFFFFFFF
CCCCCCCCCCCC RRRRRRRRRRRR FFFFFFFFFFFFFFF
CCCCCCCCCCCC RRRRRRRRRRRR FFFFFFFFFFFFFFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRRRRRRRRRR FFFFFFFFFFF
CCC RRRRRRRRRRR FFFFFFFFFFF
CCC RRRRRRRRRRR FFFFFFFFFFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCC RRR RRR FFF
CCCCCCCCCCCC RRR RRR FFF
CCCCCCCCCCCC RRR RRR FFF
CCCCCCCCCCCC RRR RRR FFF
CCCCCCCCCCCC RRR RRR FFF
CCCCCCCCCCCC RRR RRR FFF

```
  CCCCCCCC  RRRRRRR    FFFFFFFFFF    SSSSSSSS  UU      UU  BBBBBBBB
  CCCCCCCC  RRRRRRRR   FFFFFFFFFF    SSSSSSSS  UU      UU  BBBBBBBB
CC          RR      RR FF          SS          UU      UU  BB      BB
CC          RR      RR FF          SS          UU      UU  BB      BB
CC          RR      RR FF          SS          UU      UU  BB      BB
CC          RR      RR FF          SS          UU      UU  BB      BB
CC          RRRRRRRR   FFFFFFFF      SSSSSS    UU      UU  BBBBBBBB
CC          RRRRRRRR   FFFFFFFF      SSSSSS    UU      UU  BBBBBBBB
CC          RR   RR    FF                  SS  UU      UU  BB      BB
CC          RR  RR     FF                  SS  UU      UU  BB      BB
CC          RR    RR   FF                  SS  UU      UU  BB      BB    ....
CC          RR    RR   FF                  SS  UU      UU  BB      BB    ....
  CCCCCCCC  RR      RR FF          SSSSSSSS    UUUUUUUUUU  BBBBBBBB      ....
  CCCCCCCC  RR      RR FF          SSSSSSSS    UUUUUUUUUU  BBBBBBBB      ....

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

```
0000     1          .title  crfsub  subroutines for cross reference
0000     2          .ident  'V04-000'
0000     3  ;
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26  ;
0000    27  ;++
0000    28  ;   FACILITY
0000    29  ;
0000    30  ;       Cross reference
0000    31  ;
0000    32  ;   ABSTRACT
0000    33  ;
0000    34  ;       Routines to manipulate key hash table
0000    35  ;
0000    36  ;   ENVIRONMENT
0000    37  ;
0000    38  ;       Native mode, user mode
0000    39  ;
0000    40  ;   AUTHOR
0000    41  ;
0000    42  ;       Benn Schreiber, 3-Dec-1979
0000    43  ;
0000    44  ;   MODIFIED BY
0000    45  ;
0000    46  ;--
```

```
                                        0000     48          .sbttl  Declarations
                                        0000     49 ;
                                        0000     50 ;       Symbol defintions
                                        0000     51 ;
                                        0000     52
                                        0000     53          $keydef                         ;Key offsets
                                        0000     54          $crfdef                         ;Cref control table offsets
                                        0000     55
                                    00000000     56          .psect  $code$,exe,nowrt
                                        0000     57
00000028 0000000D 00000004 00000001     0000     58 steps:   .long   1, 4, 13, 40, 121, 364, 1093, 3280, 9841, 32767 ;Steps for shellsort
00000CD0 00000445 0000016C 00000079     0010
                  00007FFF 00002671     0020
                           0000000A     0028     59      numsteps = .-steps/4
                                        0028     60
                                        0028     61          .default displacement,word
```

```
                        0028     63          .sbttl  sort_hash_table sort the hash table into a list
                        0028     64
                        0028     65 ;++
                        0028     66 ;
                        0028     67 ; Inputs:
                        0028     68 ;
                        0028     69 ;        4(ap)   number of entries in hash table
                        0028     70 ;        8(ap)   Address of vector to hold addresses of sorted entries
                        0028     71 ;        r11     Address of cross-reference control table
                        0028     72 ;
                        0028     73 ; Outputs:
                        0028     74 ;
                        0028     75 ;        the hash table is sorted and the vector is filled in with the
                        0028     76 ;        addresses of the key blocks
                        0028     77 ;--
                        0028     78
                  07FC  0028     79          .entry  sort_hash_table,^m<r2, r3, r4, r5, r6, r7, r8, r9, r10>
                        002A     80
                        002A     81 ;
                        002A     82 ; determine highest step to use
                        002A     83 ;
              5A    D4  002A     84          clrl    r10                             ;index starts at 0
        50    0C AB   D0  002C     85          movl    crf$l_entries(r11),r0           ;get number of keys
     50    D4 AF4A    D1  0030     86 10$:     cmpl    steps+8[r10],r0                 ;this step high enough?
              09      18  0035     87          bgeq    20$                             ;if geq yes
FFF3 5A    01    07    F1  0037     88          acbl    #<numsteps-3>,#1,r10,10$;no--look through all - 3
              5A    07  D0  003D     89          movl    #<numsteps-3>,r10               ;lots of symbols--use all steps
   7E    08 AC    04    C3  0040     90 20$:     subl3   #4,8(ap),-(sp)                  ;set table address-4 on stack
              50    DD  0045     91          pushl   r0                              ;set # of entries on stack
                        0047     92 ;
                        0047     93 ; now copy addresses of all entries into the table
                        0047     94 ;
        59    6B    D0  0047     95          movl    crf$l_hasht(r11),r9             ;get address of hash table
     58    08 AC    D0  004A     96          movl    8(ap),r8                        ;get address of sorted list vector
     55    04 AC    D0  004E     97          movl    4(ap),r5                        ;get no. of entries in hash table
                        0052     98
        57    89    D0  0052     99 30$:     movl    (r9)+,r7                        ;next hash table entry
              08    13  0055    100          beql    50$                             ;if eql then empty bucket so skip it
        88    57    D0  0057    101 40$:     movl    r7,(r8)+                        ;put entry into sorted list
        57    67    D0  005A    102          movl    key$l_next(r7),r7               ;chain down the hash bucket
              F8    12  005D    103          bneq    40$                             ;and insert them all into the sorted list
           F0 55    F5  005F    104 50$:     sobgtr  r5,30$                          ;loop over all hash entries
        01    6E    D1  0062    105          cmpl    (sp),#1                         ;If there are not at least two entries
              60    1B  0065    106          blequ   sort_exit                       ; then quit now
                        0067    107 ;
                        0067    108 ; now do the shell sort on the list.  The shell sort is described in
                        0067    109 ; Knuth Vol. 3 and is also referred to as the Diminishing Increment Sort.
                        0067    110 ;
                        0067    111 shell_sort:
  59    95 AF4A    D0  0067    112 10$:     movl    steps[r10],r9                   ;get step value for this "t"
     58    01 A9    9E  006C    113          movab   1(r9),r8                        ;set up loop for step+1 to index
  56    04 BE48    D0  0070    114 20$:     movl    @4(sp)[r8],r6                   ;get address of key block for j'th key
  57    58    59    C3  0075    115          subl3   r9,r8,r7                        ;i=j-h
     55    08 A6    D0  0079    116 30$:     movl    key$l_keyadr(r6),r5             ;get j'th key address
  54    04 BE47    D0  007D    117          movl    @4(sp)[r7],r4                   ;get address of key block for i'th key
     53    08 A4    D0  0082    118          movl    key$l_keyadr(r4),r3             ;get key address
        14 08 AB    E8  0086    119          blbs    crf$b_keytype(r11),40$  ;branch if binary keys
```

```
                        50   85   9A  008A   120          movzbl  (r5)+,r0                ;get key lengths
                        51   83   9A  008D   121          movzbl  (r3)+,r1                ;...
                             30   BB  0090   122          pushr   #^m<r4,r5>
          63   51   00   65   50   2D  0092   123          cmpc5   r0,(r5),#0,r1,(r3)      ;compare keys
                             30   BA  0098   124          popr    #^m<r4,r5>
                             07   1E  009A   125          bgequ   50$
                             10   11  009C   126          brb     60$
                        53   55   D1  009E   127 40$:     cmpl    r5,r3                   ;compare binary keys
                             0B   1F  00A1   128          blssu   60$
                   50   59   57   C1  00A3   129 50$:     addl3   r7,r9,r0                ;compute i+h
                   04 BE40   56   D0  00A7   130          movl    r6,@4(sp)[r0]           ;ids(i+h) = val
                             10   11  00AC   131          brb     70$
                   50   59   57   C1  00AE   132 60$:     addl3   r7,r9,r0                ;ids(i+h) = ids(i)
                   04 BE40   54   D0  00B2   133          movl    r4,@4(sp)[r0]
                        57   59   C2  00B7   134          subl2   r9,r7                   ;i=i-h
                             BD   14  00BA   135          bgtr    30$
                             E5   11  00BC   136          brb     50$                     ;go set ids(i+h)=val
             FFAC 58   01   6E   F1  00BE   137 70$:     acbl    (sp),#1,r8,20$          ;loop for all entries in table
                        A0 5A   F4  00C4   138 80$:     sobgeq  r10,10$                 ;loop for all steps
                                   00C7   139 sort_exit:
                             04    00C7   140          ret
                                   00C8   141
                                   00C8   142          .END
```

```
CRF$B_KEYTYPE                   = 00000008
CRF$L_ENTRIES                   = 0000000C
CRF$L_HASHT                     = 00000000
KEY$C_LENGTH                      00000018
KEY$K_LENGTH                      00000018
KEY$L_DEFNAM                      00000014
KEY$L_KEYADR                      00000008
KEY$L_NEXT                        00000000
KEY$L_REFLIST                     00000004
KEY$L_VALADR                      0000000C
KEY$W_DEFFLG                      00000012
KEY$W_VALFLG                      00000010
NUMSTEPS                        = 0000000A
SHELL_SORT                        00000067 R      02
SORT_EXIT                         000000C7 R      02
SORT_HASH_TABLE                   00000028 RG     02
STEP$                             00000000 R      02
```

Psect synopsis

```
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
. ABS .                         00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000018  (   24.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT   NOVEC BYTE
$CODE$                          000000C8  (  200.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	43	00:00:00.07	00:00:00.27
Command processing	168	00:00:00.57	00:00:02.72
Pass 1	136	00:00:01.42	00:00:04.84
Symbol table sort	0	00:00:00.04	00:00:00.04
Pass 2	42	00:00:00.43	00:00:01.29
Symbol table output	4	00:00:00.02	00:00:00.04
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	396	00:00:02.58	00:00:09.22

The working set limit was 1050 pages.
5569 bytes (11 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 54 non-local and 13 local symbols.
142 source lines were read in Pass 1, producing 15 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

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

Macro library name                                Macros defined
------------------                                --------------
_$255$DUA28:[CRF.OBJ]CRF.MLB;1                               2
_$255$DUA28:[SYSLIB]STARLET.MLB;2                            3
TOTALS (all libraries)                                       5

117 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:CRFSUB/OBJ=OBJ$:CRFSUB MSRC$:CRFSUB/UPDATE=(ENH$:CRFSUB)+LIB$:CRF/LIB
```

CRFOR
LIS

CRFMEM
LIS

CRFMACROS
MAR

CLITABDEF
SDL

FILINPUT
LIS

KEYS
LIS

CRFTFRVEC
LIS

DCLDEF
MDL

CRFMDL
MDL

CREF
LIS

CRFERRMSG
LIS

CRF
SDL

CRF

CRFSUB
LIS

FILOUTPUT
LIS

CRFSHR
MAP

CRFGLOBAL
LIS

CRFMAC
REQ

DCL

DCL
MAP