```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

FILEID**OUTPUT

```
  OOOOOO    UU      UU  TTTTTTTTTT  PPPPPPPP    UU      UU  TTTTTTTTTT
  OOOOOO    UU      UU  TTTTTTTTTT  PPPPPPPP    UU      UU  TTTTTTTTTT
OO      OO  UU      UU      TT      PP      PP  UU      UU      TT
OO      OO  UU      UU      TT      PP      PP  UU      UU      TT
OO      OO  UU      UU      TT      PP      PP  UU      UU      TT
OO      OO  UU      UU      TT      PP      PP  UU      UU      TT
OO      OO  UU      UU      TT      PPPPPPPP    UU      UU      TT
OO      OO  UU      UU      TT      PPPPPPPP    UU      UU      TT
OO      OO  UU      UU      TT      PP          UU      UU      TT
OO      OO  UU      UU      TT      PP          UU      UU      TT
OO      OO  UU      UU      TT      PP          UU      UU      TT          ....
OO      OO  UU      UU      TT      PP          UU      UU      TT          ....
  OOOOOO    UUUUUUUUUU      TT      PP          UUUUUUUUUU      TT          ....
  OOOOOO    UUUUUUUUUU      TT      PP          UUUUUUUUUU      TT          ....


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
```

```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C*****************************************************************
0005    C*                                                               *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                      *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.       *
0008    C*  ALL RIGHTS RESERVED.                                         *
0009    C*                                                               *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0015    C*  TRANSFERRED.                                                 *
0016    C*                                                               *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0019    C*  CORPORATION.                                                 *
0020    C*                                                               *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.      *
0023    C*                                                               *
0024    C*                                                               *
0025    C*****************************************************************
0026    C
0027    C
0028    C
0029    C       AUTHOR  BRIAN PORTER            CREATION DATE   07-MAR-79
0030    C
0031    C
0032
0033    C++
0034    C       Functional description:
0035    C
0036    C       This routine outputs the bit-to-text translation for a given
0037    C       register bit field.  The bit-to-text translation is controlled by
0038    C       argument FLAG.  If FLAG is '0' then all set bits are translated.
0039    C       If FLAG is '1' then all clear bits are translated.  If FLAG is '2'
0040    C       then all bits are translated according to their polarity.
0041    C
0042    C       Modified by:
0043    C
0044    C       v03-001 BP0001          Brian Porter,           20-AUG-1982
0045    C               Added some definitions.
0046    C--
0047    C--
0048
0049
0050
0051            subroutine output (lun,register_mask,text_array,text_array_base,
0052            1 translation_start_bit,translation_end_bit,translation_control_flag)
0053
0054
0055
0056
0057            byte            lun
```

```
0058
0059            byte            translation_control_flag
0060
0061            integer*4       text_array_base
0062
0063            integer*4       translation_start_bit
0064
0065            integer*4       translation_end_bit
0066
0067            integer*4       register_mask
0068
0069            integer*4       polarity
0070
0071            integer*4       lib$extzv
0072
0073            integer*4       compressc
0074
0075            character*(*)   text_array(text_array_base:translation_end_bit,0:1)
0076
0077
0078
0079
0080            do 100,i = translation_start_bit,translation_end_bit
0081
0082            polarity = lib$extzv (i,1,register_mask)
0083
0084            if ((translation_control_flag .eq. '0'
0085            1 .and.
0086            1 polarity .eq. 1)
0087            1 .or.
0088            1 (translation_control_flag .eq. '1'
0089            1 .and.
0090            1 polarity .eq. 0)) then
0091
0092            call linchk (lun,1)
0093
0094            write(lun,10) text_array(i,0)
0095      10    format(' ',t40,a<compressc (text_array(i,0))>)
0096
0097            else if (translation_control_flag .eq. '2')  then
0098
0099            call linchk (lun,1)
0100
0101            write(lun,20) text_array(i,polarity)
0102      20    format(' ',t40,a<compressc (text_array(i,polarity))>)
0103            endif
0104
0105      100   continue
0106
0107            return
0108
0109            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	375	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	28	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	108	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	511	

ENTRY POINTS

Address	Type	Name
0-00000000		OUTPUT

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	I	AP-00000004@	L*1	LUN
2-00000000	I*4	POLARITY	AP-00000008@	I*4	REGISTER_MASK
AP-00000010@	I*4	TEXT_ARRAY_BASE	AP-0000001C@	L*1	TRANSLATION_CONTROL_FLAG
AP-00000018@	I*4	TRANSLATION_END_BIT	AP-00000014@	I*4	TRANSLATION_START_BIT

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000008@	CHAR	TEXT_ARRAY	**	(*:*, 0:1)

LABELS

Address	Label	Address	Label	Address	Label
1-00000004	10'	1-00000010	20'	**	100

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESSC	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine OUTPUT_MLINES (lun,string,sub_string,field_length)
0005
0006
0007
0008    C++
0009    C       Author: Sharon Reynolds                 Creation date:  19-Mar-1981
0010    C
0011    C
0012    C       Functional description:
0013    C
0014    C       This module will take the passed character string and separate
0015    C       it at the delimiting character(s) and output it. If there are
0016    C       no delimiting characters found, the string will be output
0017    C       on one line.
0018    C       Because this routine is an interim solution for outputting
0019    C       multiple lines of text, there are two restrictions; (1) the
0020    C       passed string will only be separated into six lines and (2) the
0021    C       passed string can only be 192 characters in length (six lines for
0022    C       32 character field). These restrictions may be changed by manipulating
0023    C       the size of 'array' and 'c_string'.
0024    C
0025    C       Modified by:
0026    C--
0027    C--
0028
0029
0030            Byte            lun
0031
0032
0033            Integer*4       field_length
0034            Integer*4       cnt
0035            Integer*4       num_lines
0036            Integer*4       sub_string_pos
0037            Integer*4       ret_status
0038            Integer*4       rstatus
0039            Integer*4       I
0040
0041            integer*4       str$left
0042
0043            integer*4       str$right
0044
0045            Character*(*)   string
0046
0047            Character*80    array(0:5)
0048            Character*1     sub_string
0049            Character*192   c_string
0050
0051
0052    C
0053    C       Initialize necessary variables
0054    C
0055
0056            Cnt = 0
0057
```

```
0058              C_string = string
0059
0060     C
0061     C        Locate the first occurence of the delimiter symbol in the string
0062     C
0063
0064              Sub_string_pos = INDEX (c_string,sub_string)
0065
0066     C
0067     C        If there was no delimiter symbol in the string then output it
0068     C        on one line using the field length that was passed via the call
0069     C
0070
0071              If (sub_string_pos .eq. 0) then
0072
0073              Call LINCHK (lun,1)
0074              Write (lun,10) string
0075     10       Format (' ',T40,A<field_length>)
0076
0077              Return
0078              Endif
0079
0080     C
0081     C        There was a delimiter symbol found in the string; separate the
0082     C        string at the delimiter
0083     C
0084
0085     20       Ret_status = STR$LEFT (array(cnt),c_string,(sub_string_pos - 1))
0086
0087              Cnt = cnt + 1
0088
0089              Rstatus = STR$RIGHT (array(cnt),c_string,(sub_string_pos + 1))
0090
0091     C
0092     C        See if there is another delimiter symbol in the second portion
0093     C        of the string and make a copy of the string
0094     C
0095
0096              Sub_string_pos = INDEX (array(cnt),sub_string)
0097
0098              C_String = array(cnt)
0099
0100     C
0101     C        Check to see if the string has been separated into the maximum
0102     C        number of lines and if so, output the strings
0103     C
0104
0105              If (cnt .eq. 5) then
0106
0107              Goto 25
0108              Endif
0109
0110     C
0111     C        Check to see if a deiimiter symbol was found in the second
0112     C        portion of the string and if so, go separate the string
0113     C        again
0114     C
```

```
0115
0116            If (sub_string_pos .ne. 0) then
0117
0118            Goto 20
0119
0120            Endif
0121
0122    C
0123    C       The strings are ready to be output so do it.
0124    C
0125
0126    25      Num_lines = cnt
0127            Cnt = 0
0128
0129            Call LINCHK (lun,(num_lines))
0130
0131            Write (lun,27) array(cnt)
0132    27      Format (' ',T40,A<field_length>)
0133
0134            Do 40, I=1, num_lines
0135
0136            Cnt = cnt + 1
0137
0138            Write (lun,30) array(cnt)
0139    30      Format (' ',T40,'- ',A<field_length>)
0140
0141    40      Continue
0142
0143    100     Return
0144            END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	414	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	44	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	804	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1262	

ENTRY POINTS

Address	Type	Name
0-00000000		OUTPUT_MLINES

VARIABLES

Address	Type	Name	Address	Type	Name
2-000002A0	I*4	CNT	2-000001E0	CHAR	C_STRING
2-000002B8	I*4	FIELD_LENGTH	2-000002B4	I*4	I
AP-00000004	L*1	LUN	2-000002A4	I*4	NUM_LINES
2-000002AC	I*4	RET_STATUS	2-000002B0	I*4	RSTATUS
AP-00000008	CHAR	STRING	AP-0000000C	CHAR	SUB_STRING
2-000002A8	I*4	SUB_STRING_POS			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	ARRAY	480	(0:5)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000004	10'	0-0000006E	20	0-000000EE	25	1-00000010	27'	1-0000001C	30'	**	40
**	100										

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	LIB$INDEX		LINCHK	I*4	STR$LEFT	I*4	STR$RIGHT

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:OUTPUT/OBJ=OBJ$:OUTPUT MSRC$:OUTPUT

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19
```

COMPILATION STATISTICS

```
  Run Time:           2.62 seconds
  Elapsed Time:       8.29 seconds
  Page Faults:        130
  Dynamic Memory:     174 pages
```

PADRIVER.LIS

OUTPUT.LIS

NEW_RTN.LIS

MT_DISMT.LIS

OPNOUTFIL.LIS

OUTPUT0.LIS

NEWFILE.LIS