```
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
```

```
UU       UU  BBBBBBBB      AAAAAA
UU       UU  BBBBBBBB      AAAAAA
UU       UU  BB      BB   AA      AA
UU       UU  BB      BB   AA      AA
UU       UU  BB      BB   AA      AA
UU       UU  BB      BB   AA      AA
UU       UU  BBBBBBBB     AA      AA
UU       UU  BBBBBBBB     AA      AA
UU       UU  BB      BB   AAAAAAAAAA
UU       UU  BB      BB   AAAAAAAAAA
UU       UU  BB      BB   AA      AA      ....
UU       UU  BB      BB   AA      AA      ....
UUUUUUUUUU   BBBBBBBB     AA      AA      ....
UUUUUUUUUU   BBBBBBBB     AA      AA      ....

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
```

```
UBA

011
011
011
011
012

PRO

  0
  1
  2
  3

ENT

  0

VAR

  3
  3
 AP

ARR

  3
  3

LAB

  1
```

```
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
0028    c       Author  Brian Porter                    Creation date   21-JUL-1981
0029
0030    c++
0031    c       Functional description:
0032    c
0033    c       This file is a collection of routines that support the VAX-11
0034    c       family of UBA adapters.  The routines are used by device-specific
0035    c       modules to display datapath and map register information.
0036    c
0037    c       Modified by:
0038    c
0039    c       V03-003 EAD0109         Elliott A. Drayton       5-Mar-1984
0040    c               Deleated 'common mrd' from DW780_MAPPING.
0041    c
0042    C       V03-002 SAR0158         Sharon A. Reynolds,      13-Oct-1983
0043    C               Added an SYE update that makes all register heralds
0044    C               generic.
0045    C
0046    C       V03-001 SAR0101         Sharon A. Reynolds,      20-Jun-1983
0047    C               Changed the carriage control in the 'format' statements
0048    C               for use with ERF.
0049    C
0050    c       v02-003 BP0003          Brian Porter,            28-SEP-1981
0051    c               Corrected 11/7zz mapping register decoding.
0052    c
0053    c       v02-002 BP0002          Brian Porter,            31-AUG-1981
0054    c               Added call to compress4 in appropriate places.  Corrected
0055    c               dw730 mapping register code.
0056    c
0057    c       v02-001 BP0001          Brian Porter,            24-AUG-1981
```

```
0058    c               Added 11/7zz support.
0059    c**
0060    c--
0061
0062
0063
0064            subroutine uba_datapath (lun,datapath_number,datapath_register)
0065
0066
0067
0068    c++
0069    c       This routine dispatches to the correct display module for
0070    c       UBA datapath registers.
0071    c--
0072
0073
0074            include 'src$:msghdr.for /nolist'
0133
0134
0135            byte            lun
0136
0137            integer*4       datapath_register
0138
0139            integer*4       datapath_number
0140
0141            integer*4       compress4
0142
0143
0144
0145            if (
0146            1 lib$extzv(24,8,emb$l_hd_sid) .eq. 255
0147            1 .or.
0148            1 lib$extzv(24,8,emb$l_hd_sid) .eq. 1
0149            1 ) then
0150
0151            call dw780_datapath (lun,datapath_number,datapath_register)
0152
0153            else if (lib$extzv(24,8,emb$l_hd_sid) .eq. 2) then
0154
0155            call dw750_datapath (lun,datapath_number,datapath_register)
0156
0157            else if (lib$extzv(24,8,emb$l_hd_sid) .eq. 3) then
0158
0159            call dw7zz_datapath (lun,datapath_number)
0160    c
0161    c       for future UBA support the ELSE-IF-THEN should be expanded at
0162    c       this point.
0163    c
0164
0165            else
0166
0167            call linchk (lun,2)
0168
0169            write(lun,5) 'DATAPATH #',datapath_number,'.',
0170            1 datapath_register
0171    5       format(/' ',a,i<compress4 (datapath_number)>,a,t24,z8.8)
0172            endif
```

```
0173
0174            return
0175
0176            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	242	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	41	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	84	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	879	

ENTRY POINTS

Address	Type	Name
0-00000000		UBA_DATAPATH

VARIABLES

Address	Type	Name	Address	Type	Name
2-0000000@	I*4	DATAPATH_NUMBER	AP-0000000C@	I*4	DATAPATH_REGISTER
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000004@	L*1	LUN

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label
1-00000017	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		DW750_DATAPATH		DW780_DATAPATH
	DW7ZZ_DATAPATH	I*4	LIB$EXTZV		LINCHR

```
0001
0002
0003
0004
0005            subroutine dw780_datapath (lun,datapath_number,datapath_register)
0006
0007
0008
0009    c++
0010    c       This routine displays the 11/780 DW780 datapath register.
0011    c--
0012
0013
0014
0015            byte            lun
0016
0017            integer*4       datapath_register
0018
0019            integer*4       datapath_number
0020
0021            character*22    v1dpr(30:31)
0022
0023            data            v1dpr(30)       /'BUFFER TRANSFER ERROR*'/
0024
0025            data            v1dpr(31)       /'BUFFER NOT EMPTY*'/
0026
0027            integer*4       compress4
0028
0029            integer*4       lib$extzv
0030
0031            integer*4       selected_map
0032
0033
0034
0035            call linchk (lun,2)
0036
0037            write(lun,5) ''DW'' DPR #',datapath_number,'.',datapath_register
0038    5       format(/' ',t8,a,i<compress4 (datapath_number)>,a,t24,z8.8)
0039
0040            selected_map = lib$extzv (7,9,datapath_register)
0041
0042            call linchk (lun,1)
0043
0044            write(lun,10) 'MAPPING REGISTER #',selected_map,'.'
0045    10      format(' ',t40,a,i<compress4 (selected_map)>,a)
0046
0047            do 20,i = 0,7
0048
0049            if (lib$extzv ((i+16),1,datapath_register) .eq. 1) then
0050
0051            call linchk (lun,1)
0052
0053            write(lun,15) 'BUFFER #',i,'. NOT EMPTY'
0054    15      format(' ',t40,a,i1.1,a)
0055            endif
0056
0057    20      continue
```

```
0058
0059            if (lib$extzv (29,1,datapath_register) .eq. 1) then
0060
0061            call linchk (lun,1)
0062
0063            write(lun,25) 'DMA READ'
0064     25     format(' ',t40,a)
0065            endif
0066
0067            call output (lun,datapath_register,v1dpr,30,30,31,'0')
0068
0069            return
0070
0071            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	409	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	138	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	264	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	811	

ENTRY POINTS

Address	Type	Name
0-00000000		DW780_DATAPATH

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000034@	I*4	DATAPATH_NUMBER	AP-0000000C@	I*4	DATAPATH_REGISTER
2-00000030	I*4	I	AP-00000004@	L*1	LUN
2-0000002C	I*4	SELECTED_MAP			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1DPR	44	(30:31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000056	5'	1-0000006A	10'	1-00000078	15'	**	20	1-00000083	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004            subroutine dw750_datapath (lun,datapath_number,datapath_register)
0005
0006
0007    c++
0008    c       This routine displays the 11/750 DW750 datapath register.
0009    c--
0010
0011
0012            byte            lun
0013
0014            integer*4       datapath_number
0015
0016            integer*4       datapath_register
0017
0018            integer*4       compress4
0019
0020            character*6     v1dpr(0:0)
0021
0022            data            v1dpr(0)        /'^URGE*'/
0023
0024            character*20    v2dpr(29:31)
0025
0026            data            v2dpr(29)       /'ERROR*'/
0027
0028            data            v2dpr(30)       /'NON-EXISTENT MEMORY*'/
0029
0030            data            v2dpr(31)       /'UNCORRECTABLE ERROR*'/
0031
0032
0033
0034            call linchk (lun,2)
0035
0036            write(lun,5) ''DW'' BDP #',datapath_number,'.',datapath_register
0037    5       format(/' ',t8,a,i<compress4 (datapath_number)>,a,t24,z8.8)
0038
0039            call output (lun,datapath_register,v1dpr,0,0,0,'0')
0040
0041            call output (lun,datapath_register,v2dpr,29,29,31,'0')
0042
0043            return
0044
0045            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	155	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	49	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	244	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	448	

ENTRY POINTS

Address	Type	Name
0-00000000		DW750_DATAPATH

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000044@	I*4	DATAPATH_NUMBER	AP-0000000C@	I*4	DATAPATH_REGISTER
AP-00000004@	L*1	LUN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1DPR	6	(0:0)
2-00000006	CHAR	V2DPR	60	(29:31)

LABELS

Address	Label
1-0000001D	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		LINCHK		OUTPUT

```
0001
0002
0003
0004            subroutine dw7zz_datapath (lun,datapath_number)
0005
0006
0007    c++
0008    c       This routine displays the 11/7zz DW7zz datapath register.
0009    c--
0010
0011
0012            byte            lun
0013
0014            integer*4       datapath_number
0015
0016            integer*4       compress4
0017
0018
0019
0020            call linchk (lun,3)
0021
0022            write(lun,5) ''DW'' BDP #',datapath_number,'. ASSIGNED TO QIO REQUEST'
0023    5       format(/' ',t8,a,i<compress4 (datapath_number)>,a,/)
0024
0025            return
0026
0027            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	106	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	55	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	201	

ENTRY POINTS

Address	Type	Name
0-00000000		DW7ZZ_DATAPATH

VARIABLES

Address	Type	Name	Address	Type	Name
2-0000000@	I*4	DATAPATH_NUMBER	AP-0000004@	L*1	LUN

**F

LABELS

Address	Label
1-00000027	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
I*4	COMPRESS4		LINCHK

```
0001
0002
0003
0004
0005            subroutine uba_mapping (lun,map_number,mapping_register)
0006
0007
0008
0009
0010    c++
0011    c       This routine dispatches to the correct display module for
0012    c       UBA mapping registers.
0013    c--
0014
0015
0016
0017            include 'src$:msghdr.for /nolist'
0076
0077
0078            byte            lun
0079
0080            integer*4       map_number
0081
0082            integer*4       mapping_register
0083
0084            integer*4       compress4
0085
0086
0087
0088            if (
0089            1 lib$extzv (24,8,emb$l_hd_sid) .eq. 255
0090            1 .or.
0091            1 lib$extzv (24,8,emb$l_hd_sid) .eq. 1
0092            1 ) then
0093
0094            call dw780_mapping (lun,map_number,mapping_register)
0095
0096            else if (lib$extzv (24,8,emb$l_hd_sid) .eq. 2) then
0097
0098            call dw750_mapping (lun,map_number,mapping_register)
0099
0100            else if (lib$extzv (24,8,emb$l_hd_sid) .eq. 3) then
0101
0102            call dw7zz_mapping (lun,map_number,mapping_register)
0103
0104
0105    c
0106    c       for future UBA support the ELSE-IF-THEN should be expanded at
0107    c       this point.
0108    c
0109
0110            else
0111
0112            call linchk (lun,2)
0113
0114            write(lun,5) 'MAP REGISTER #',map_number,'.',mapping_register
0115    5       format(/' ',a,i<compress4 (map_number)>,a,t24,z8.8)
```

```
0116            endif
0117
0118            return
0119
0120            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	247	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	45	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	72	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	876	

ENTRY POINTS

Address	Type	Name
0-00000000		UBA_MAPPING

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000004@	L*1	LUN
AP-0000000C@	I*4	MAPPING_REGISTER	2-00000000@	I*4	MAP_NUMBER

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label
1-0000001B	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		DW750_MAPPING		DW780_MAPPING		DW7ZZ_MAPPING	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004
0005            subroutine dw780_mapping (lun,map_number,mapping_register)
0006
0007
0008    c++
0009    c       This routine displays the 11/780 DW780 mapping register.
0010    c--
0011
0012
0013            byte            lun
0014
0015            integer*4       map_number
0016
0017            integer*4       mapping_register
0018
0019            integer*4       pfn
0020
0021            integer*4       dpr
0022
0023            integer*4       compress4
0024
0025            character*12    v1mr(25:25)
0026
0027            common  /mrd/   v1mr
0028
0029            data            v1mr(25)        /'BYTE OFFSET*'/
0030
0031            character*19    v2mr(31:31)
0032
0033            common  /mrd/   v2mr
0034
0035            data            v2mr(31)        /'MAP REGISTER VALID*'/
0036
0037
0038            call linchk (lun,1)
0039
0040            if (map_number .eq. -1) then
0041
0042            write(lun,3) ''DW'' MPR #???',mapping_register
0043    3       format(' ',t8,a,t24,z8.8)
0044            else
0045
0046            write(lun,5) ''DW'' MPR #',map_number,'.',mapping_register
0047    5       format(' ',t8,a,i<compress4 (map_number)>.a,t24,z8.8)
0048            endif
0049
0050            if (lib$extzv (31,1,mapping_register) .eq. 1) then
0051
0052            pfn = lib$extzv (0,21,mapping_register)
0053
0054            if (pfn .ne. 0) then
0055
0056            call linchk (lun,1)
0057
```

```
0058            if ((pfn/2)*2 .eq. pfn) then
0059
0060            write(lun,10) 'SBI PAGE ADDRESS ',pfn/2,'.K'
0061    10      format(' ',t40,a,i<compress4 (pfn/2)>,a)
0062            else
0063
0064            write(lun,15) 'SBI PAGE ADDRESS ',(floatj(pfn))/2,'.K'
0065    15      format(' ',t40,a,f<(compress4(pfn/2)+2)>.1,a)
0066            endif
0067            endif
0068
0069            dpr = lib$extzv (21,4,mapping_register)
0070
0071            call linchk (lun,1)
0072
0073            if (dpr .ne. 0) then
0074
0075            write(lun,20) 'DATAPATH REGISTER #',dpr,'.'
0076    20      format(' ',t40,a,i<compress4 (dpr)>,a)
0077            else
0078
0079            write(lun,25) 'DIRECT DATAPATH'
0080    25      format(' ',t40,a)
0081            endif
0082
0083            call output (lun,mapping_register,v1mr,25,25,25,'0')
0084
0085            call output (lun,mapping_register,v2mr,31,31,31,'0')
0086            endif
0087
0088            return
0089
0090            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	609	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	184	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	280	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	MRD	31	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1104	

ENTRY POINTS

Address	Type	Name
0-00000000		DW780_MAPPING

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	DPR	AP-00000004@	L*1	LUN
AP-0000000C@	I*4	MAPPING_REGISTER	2-00000008@	I*4	MAP_NUMBER
2-00000000	I*4	PFN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	CHAR	V1MR	12	(25:25)
3-0000000C	CHAR	V2MR	19	(31:31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000067	3'	1-00000073	5'	1-00000086	10'	1-00000094	15'	1-000000A3	20'	1-000000B1	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004            subroutine dw750_mapping (lun,map_number,mapping_register)
0005
0006
0007    c++
0008    c       This routine displays the 11/750 DW750 mapping register.
0009    c--
0010
0011
0012            byte            lun
0013
0014            integer*4       map_number
0015
0016            integer*4       mapping_register
0017
0018            integer*4       pfn
0019
0020            integer*4       bdp
0021
0022            integer*4       compress4
0023
0024            common          mrd
0025
0026            character*12    v1mr(25:25)
0027
0028            common  /mrd/   v1mr
0029
0030            character*19    v2mr(31:31)
0031
0032            common  /mrd/   v2mr
0033
0034
0035
0036            call linchk (lun,1)
0037
0038            if (map_number .eq. -1) then
0039
0040            write(lun,3) ''DW'' MPR #???',mapping_register
0041    3       format(' ',t8,a,t24,z8.8)
0042            else
0043
0044            write(lun,5) ''DW'' MPR #',map_number,'.',mapping_register
0045    5       format(' ',t8,a,i<compress4 (map_number)>,a,t24,z8.8)
0046            endif
0047
0048            if (lib$extzv (31,1,mapping_register) .eq. 1) then
0049
0050            pfn = lib$extzv (0,15,mapping_register)
0051
0052            if (pfn .ne. 0) then
0053
0054            call linchk (lun,1)
0055
0056            if ((pfn/2)*2 .eq. pfn) then
0057
```

```
0058                write(lun,10) 'PAGE ADDRESS ',pfn/2,'.K'
0059        10      format(' ',t40,a,i<compress4 (pfn/2)>,a)
0060                else
0061
0062                write(lun,15) 'PAGE ADDRESS ',(floatj(pfn))/2,'.K'
0063        15      format(' ',t40,a,f<(compress4(pfn/2)+2)>.1,a)
0064                endif
0065                endif
0066
0067                bdp = lib$extzv (21,2,mapping_register)
0068
0069                call linchk (lun,1)
0070
0071                if (bdp .ne. 0) then
0072
0073                write(lun,20) 'BUFFERED DATAPATH REGISTER #',bdp,'.'
0074        20      format(' ',t40,a,i<compress4 (bdp)>,a)
0075                else
0076
0077                write(lun,25) 'DIRECT DATAPATH'
0078        25      format(' ',t40,a)
0079                endif
0080
0081                call output (lun,mapping_register,v1mr,25,25,25,'0')
0082
0083                call output (lun,mapping_register,v2mr,31,31,31,'0')
0084                endif
0085
0086                return
0087
0088                end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	609	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	193	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	280	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	$BLANK	4	PIC OVR REL GBL SHR NOEXE RD WRT LONG
4	MRD	31	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1117	

ENTRY POINTS

Address	Type	Name
0-00000000		DW750_MAPPING

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	BDP	AP-00000004@	L*1	LUN
AP-0000000C@	I*4	MAPPING_REGISTER	2-00000008@	I*4	MAP_NUMBER
3-00000000	I*4	MRD	2-00000000	I*4	PFN

ARRAYS

Address	Type	Name	Bytes	Dimensions
4-00000000	CHAR	V1MR	12	(25:25)
4-0000000C	CHAR	V2MR	19	(31:31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000070	3'	1-0000007C	5'	1-0000008F	10'	1-0000009D	15'	1-000000AC	20'	1-000000BA	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004            subroutine dw7zz_mapping (lun,map_number,mapping_register)
0005
0006
0007    c++
0008    c       This routine displays the 11/7zz DW7zz mapping register.
0009    c--
0010
0011
0012            byte            lun
0013
0014            integer*4       map_number
0015
0016            integer*4       mapping_register
0017
0018            integer*4       pfn
0019
0020            integer*4       compress4
0021
0022            common          mrd
0023
0024            character*12    v1mr(25:25)
0025
0026            common  /mrd/   v1mr
0027
0028            character*19    v2mr(31:31)
0029
0030            common  /mrd/   v2mr
0031
0032
0033
0034            call linchk (lun,1)
0035
0036            if (map_number .eq. -1) then
0037
0038            write(lun,3) ''DW'' MPR #???',mapping_register
0039    3       format(' ',t8,a,t24,z8.8)
0040            else
0041
0042            write(lun,5) ''DW'' MPR #',map_number,'.',mapping_register
0043    5       format(' ',t8,a,i<compress4 (map_number)>,a,t24,z8.8)
0044            endif
0045
0046            if (lib$extzv (31,1,mapping_register) .eq. 1) then
0047
0048            pfn = lib$extzv (0,15,mapping_register)
0049
0050            if (pfn .ne. 0) then
0051
0052            call linchk (lun,1)
0053
0054            if ((pfn/2)*2 .eq. pfn) then
0055
0056            write(lun,10) 'PAGE ADDRESS ',pfn/2,'.K'
0057    10      format(' ',t40,a,i<compress4 (pfn/2)>,a)
```

```
0058            else
0059
0060            write(lun,15) 'PAGE ADDRESS ',(floatj(pfn))/2,'.K'
0061    15      format(' ',t40,a,f<(compress4(pfn/2)+2)>.1,a)
0062            endif
0063            endif
0064
0065            call output (lun,mapping_register,v1mr,25,25,25,'0')
0066
0067            call output (lun,mapping_register,v2mr,31,31,31,'0')
0068            endif
0069
0070            return
0071
0072            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	466	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	121	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	236	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	$BLANK	4	PIC OVR REL GBL SHR NOEXE RD WRT LONG
4	MRD	31	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	858	

ENTRY POINTS

Address	Type	Name
0-00000000		DW7ZZ_MAPPING

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-0000000C@	I*4	MAPPING_REGISTER
2-00000004@	I*4	MAP_NUMBER	3-00000000	I*4	MRD
2-00000000	I*4	PFN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
4-00000000	CHAR	V1MR	12	(25:25)
4-0000000C	CHAR	V2MR	19	(31:31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-0000003D	3'	1-00000049	5'	1-0000005C	10'	1-0000006A	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		OUTPUT

COMMAND QUALIFIERS

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

/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:          7.94 seconds
Elapsed Time:      18.78 seconds
Page Faults:       186
Dynamic Memory:    179 pages
```

0154 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TIMCMP.LIS
TUTAPE.LIS
SBI.LIS
STSEVENT.LIS
TRNS_BITS.LIS
UBAERR.LIS
TIMRB.LIS
UBA.LIS
TU81SENSE.LIS
UNDEFINED.LIS
SYSPWRFL.LIS
SUMMARY.LIS
UNKN_DISP.LIS
SHRVECTOR.LIS
SYSTARTUP.LIS
UBAINT.LIS
TOF.LIS
UNKNOWN.LIS
TSTAPE.LIS