K 6

```
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
```

N 14

```
**FILE**ID**DUP11
```

```
DDDDDDDD    UU      UU  PPPPPPPP        11          11
DDDDDDDD    UU      UU  PPPPPPPP        11          11
DD      DD  UU      UU  PP      PP    1111        1111
DD      DD  UU      UU  PP      PP    1111        1111
DD      DD  UU      UU  PP      PP      11          11
DD      DD  UU      UU  PP      PP      11          11
DD      DD  UU      UU  PPPPPPPP        11          11
DD      DD  UU      UU  PPPPPPPP        11          11
DD      DD  UU      UU  PP              11          11
DD      DD  UU      UU  PP              11          11
DD      DD  UU      UU  PP              11          11
DD      DD  UU      UU  PP              11          11          ....
DDDDDDDD    UUUUUUUUUU  PP            111111      111111        ....
DDDDDDDD    UUUUUUUUUU  PP            111111      111111        ....
                                                                ....

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
```

```
DUP

027
027
027
027
027
027
027
028
028
028
028
028
028
028
028
028
028
029
029
029
029
029
029
029
029
029
029
030
030
030
030
030
030
030
030
030
030
031
031
031
031
031
031
031
031
031
031
032
032
032
032
032
032
032
032
032
032
```

```
        SUBROUTINE DUP11 (lun,parcsr,rxcsr,rxdbuf,txcsr,txdbuf)
C
C Version:      'V04-000'
C
C*****************************************************************************
C*                                                                           *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
C*  ALL RIGHTS RESERVED.                                                     *
C*                                                                           *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
C*  TRANSFERRED.                                                             *
C*                                                                           *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
C*  CORPORATION.                                                             *
C*                                                                           *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
C*                                                                           *
C*                                                                           *
C*****************************************************************************
C
C
C
C       Author: Sharon Reynolds                 Creation date: 8-Sept-80
C
C
C
C       Functional description:
C
C       This module decodes the Dup11 register contents. It is called
C       by the DUP3271 module.
C
C       Modified by:
C
C       V03-002 SAR0112         Sharon A. Reynolds,             23-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
C       v03-001 BP0001          Brian Porter,                   20-AUG-1982
C               Minor edit.
C**
C--




        Byte            lun

        Integer*4       addrs
        Integer*4       char
        Integer*4       field
```

```
              Integer*4         temp
              Integer*4         rxcsr
              Integer*4         rxdbuf
              Intege-*4         txcsr
              Intege-*4         txdbuf
              Integer*4         parcsr
              Integer*4         compressC
              Integer*4         compress4

              logical*1         diagnostic_mode

              Character*23      protocol(0:1)
              Character*12      oper_mode(0·1)
              Character*18      rxcsr_1(0:0)
              Character*26      rxcsr_2(3:8)
              Character*24      rxcsr_3(10:11)
              Character*18      rxcsr_4(14:15)
              Character*26      rxdbuT_1(8:10)
              Character*19      rxdbuf_2(12:12)
              Character*17      rxdbuf_3(14:15)
              Characte:*5       txcsr_T(4:4)
              Character*29      txcsr_2(6:9)
              Character*28      txcsr_3(15:15)
              Character*27      txdbuT_1(8:10)
              Character*12      parcsr_1(9:9)
              Character*26      dtr_status(0:1)
              Character*22      rts_status(0:1)
              Character*21      dsr_status(0:1)
              Character*21      car_status(0:1)
              Character*20      cts_status(0:1)



C
C             Define text for bits in the RXCSR register
C

              Data    rxcsr_1(0)       /'DATA SET CHANGE B*'/
              Data    rxcsr_2(3)       /'SECONDARY TRANSMIT DATA*'/
              Data    rxcsr_2(4)       /'RECEIVER ENABLE*'/
              Data    rxcsr_2(5)       /'DATA SET INTERRUPT ENABLE*'/
              Data    rxcsr_2(6)       /'RECEIVER INTERRUPT ENABLE*'/
              Data    rxcsr_2(7)       /'RECEIVER DONE*'/
              Data    rxcsr_2(8)       /'STRIP SYNC CHARACTER*'/
              Data    rxcsr_3(10:      /'SECONDARY RECEIVED DATA*'/
              Data    rxcsr_3(11)      /'RECEIVER ACTIVE*'/
              Data    rxcsr_4(14)      /'RING*'/
              Data    rxcsr_4(15)      /'DATA SET CHANGE A*'/


C
C             Define bits for MODEM/LINE STATUS from rxcsr register
C

              Data    dtr_status(0)    /'DATA TERMINAL READY = OFF*'/
              Data    dtr_status(1)    /'DATA TERMINAL READY = ON*'/

```

```
        Data    rts_status(0)   /'REQUEST TO SEND = OFF*'/
        Data    rts_status(1)   /'REQUEST TO SEND = ON*'/

        Data    dsr_status(0)   /'DATA SET READY = OFF*'/
        Data    dsr_status(1)   /'DATA SET READY = ON*'/

        Data    car_status(0)   /'CARRIER DETECT = OFF*'/
        Data    car_status(1)   /'CARRIER DETECT = ON*'/

        Data    cts_status(0)   /'CLEAR TO SEND = OFF*'/
        Data    cts_status(1)   /'CLEAR TO SEND = ON*'/


C
C       Define text for bits in the RXDBUF register
C

        Data    rxdbuf_1(8)     /'START OF RECEIVED MESSAGE*'/
        Data    rxdbuf_1(9)     /'END OF RECEIVED MESSAGE*'/
        Data    rxdbuf_1(10)    /'RECEIVER ABORT*'/
        Data    rxdbuf_2(12)    /'RECEIVER CRC ERROR*'/
        Data    rxdbuf_3(14)    /'RECEIVER OVERRUN*'/
        Data    rxdbuf_3(15)    /'RECEIVER ERROR*'/


C
C       Define text for bits in the TXCSR register
C

        Data    oper_mode(0)    /'FULL DUPLEX*'/
        Data    oper_mode(1)    /'HALF DUPLEX*'/
        Data    txcsr_1(4)      /'SEND*'/
        Data    txcsr_2(6)      /'TRANSMITTER INTERRUPT ENABLE*'/
        Data    txcsr_2(7)      /'TRANSMITTER DONE*'/
        Data    txcsr_2(8)      /'DEVICE RESET*'/
        Data    txcsr_2(9)      /'TRANSMITTER ACTIVE*'/
        Data    txcsr_3(15)     /'TRANSMITTER DATA LATE ERROR*'/


C
C       Define text for bits in the TXDBUF register
C

        Data    txdbuf_1(8)     /'TRANSMIT START OF MESSAGE*'/
        Data    txdbuf_1(9)     /'END OF TRANSMITTED MESSAGE*'/
        Data    txdbuf_1(10)    /'TRANSMIT ABORT*'/


C
C       Define text for bits in the PARCSR register
C

        Data    parcsr_1(9)     /'CRC INHIBIT*'/
        Data    protocol(0)     /'BIT ORIENTED PROTOCOL*'/
        Data    protocol(1)     /'BYTE ORIENTED PROTOCOL*'/





        diagnostic_mode = .false.

        if (lib$extzv(11,2,txcsr) .ne. 0) diagnostic_mode = .true.
C
C       Decode receiver control and status register
C

        Call LINCHK (lun,2)
        Write (lun,20) rxcsr
20      Format (/' ',T8,'RXCSR',T24,Z8.4)

        if (.not. diagnostic_mode) then

        Call LINCHK (lun,1)
        Write (lun,30) ('*',I=1,34)
30      Format (' ',T40,34A1)

        Call LINCHK (lun,1)
        Write (lun,40)
40      Format (' ',T47,'MODEM/LINE STATUS')

        Field=LIB$EXTZV (1,1,rxcsr)

        Call LINCHK (lun,1)
        Write (lun,50) dtr_status(field)
50      Format (' ',T40,A<compressC (dtr_status(field))>)


        Field=LIB$EXTZV (2,1,rxcsr)

        Call LINCHK (lun,1)
        Write (lun,60) rts_status(field)
60      Format (' ',T40,A<compressC (rts_status(field))>)


        Field=LIB$EXTZV (9,1,rxcsr)

        Call LINCHK (lun,1)
        Write (lun,70) dsr_status(field)
70      Format (' ',T40,A<compressC (dsr_status(field))>)


        Field=LIB$EXTZV (12,1,rxcsr)

        Call LINCHK (lun,1)
        Write (lun,80) car_status(field)
80      Format (' ',T40,A<compressC (car_status(field))>)


        Field=LIB$EXTZV (13,1,rxcsr)

        Call LINCHK (lun,1)
        Write (lun,90) cts_status(field)
90      Format (' ',T40,A<compressC (cts_status(field))>)
```

```

        Call LINCHK (lun,1)
        Write (lun,100) ('*',I=1,34)
100     Format (' ',T40,34A1)

        Call OUTPUT (lun,rxcsr,rxcsr_1,0,0,0,'0')

        Call OUTPUT (lun,rxcsr,rxcsr_2,3,3,8,'0')

        Call OUTPUT (lun,rxcsr,rxcsr_3,10,10,11,'0')

        Call OUTPUT (lun,rxcsr,rxcsr_4,14,14,15,'0')
        endif

C
C       Decode receiver data buffer register
C

        Call LINCHK (lun,1)
        Write (lun,110) rxdbuf
110     Format (' ',T8,'RXDBUF',T24,Z8.4)

        if (.not. diagnostic_mode) then

        Field=LIB$EXTZV(0,8,rxdbuf)

        Call LINCHK (lun,1)
        Write (lun,120) field
120     Format (' ',T40,'RECEIVER DATA BUFFER = ',
        1 I<compress4 (field)>,'.')

        Call OUTPUT (lun,rxdbuf,rxdbuf_1,8,8,10,'0')

        Call OUTPUT (lun,rxdbuf,rxdbuf_2,12,12,12,'0')

        Call OUTPUT (lun,rxdbuf,rxdbuf_3,14,14,15,'0')
        endif

C
C       Decode tranmitter control and status register
C

        Call LINCHK (lun,1)
        Write (lun,130) txcsr
130     Format (' ',T8,'TXCSR',T24,Z8.4)

        if (.not. diagnostic_mode) then

        Field=LIB$EXTZV(3,1,txcsr)

        Call LINCHK (lun,1)
        Write (lun,140) oper_mode(field)
140     Format (' ',T40,'OPERATION MODE = ',
        1 A<compressC (oper_mode(field))>)

        Call OUTPUT (lun,txcsr,txcsr_1,4,4,4,'0')

```

```
                Call OUTPUT (lun,txcsr,txcsr_2,6,6,9,'0')

                Call OUTPUT (lun,txcsr,txcsr_3,15,15,15,'0')
                else

                Call LINCHK (lun,1)
                Write (lun,150)
        150     Format (' ',T40,'DIAGNOSTIC MODE')
                endif

        C
        C       Decode transmitter data buffer register
        C

                Call LINCHK (lun,1)
                Write (lun,160) txdbuf
        160     Format (' ',T8,'TXDBUF',T24,Z8.4)

                if (.not. diagnostic_mode) then

                Field=LIB$EXTZV(0,8,txdbuf)

                Call LINCHK (lun,1)
                Write (lun,170) field
        170     Format (' ',T40,'TRANSMITTER DATA BUFFER = ',
                1 I<compress4 (field)>,'.')

                Call OUTPUT (lun,txdbuf,txdbuf_1,8,8,10,'0')
                endif

        C
        C       Decode and output PARCSR register contents
        C



                Call LINCHK (lun,1)
                Write (lun,180) parcsr
        180     Format (' ',T8,'PARCSR',T24,Z8.4,/,
                1 T40,'**  LOADED WITH  **')

                if (.not. diagnostic_mode) then

                Call OUTPUT (lun,parcsr,parcsr_1,9,9,9,'0')

                Temp=LIB$EXTZV (12,1,parcsr)

                Field=LIB$EXTZV (15,1,parcsr)

                Call LINCHK (lun,1)
                Write (lun,190) protocol(field)
        190     Format (' ',T40, A<compressC (protocol(field))> )

                Call LINCHK (lun,1)
                If (field .eq. 0 .and. temp .eq.0) then

                Write (lun,200)
```

```
200     Format (' ',T40,'OPERATING AS A PRIMARY STATION')

        Else if (field .eq. 0 .and. temp .eq. 1) then

        Write (lun,210)
210     Format (' ',T40,'OPERATING AS A SECONDARY STATION')

        Addrs=LIB$EXTZV (0,8,parcsr)

        Call LINCHK (lun,1)
        Write (lun,220) addrs
220     Format (' ',T40,'SECONDARY STATION ADDRESS = ',
        1 I<Compress4 (addrs)>,'.')

        Else

        Char=LIB$EXTZV (0,8,parcsr)

        Write (lun,230) char
230     Format (' ',T40,'EXPECTED SYNC CHAR = ',
        1 I<Compress4 (char)>,'.')


        Endif
        endif

        Return


        END
```

```
PROGRAM SECTIONS

    Name                                         Bytes   Attributes

  0 $CODE                                         2023   PIC CON REL LCL    SHR    EXE    RD NOWRT LONG
  1 $PDATA                                         592   PIC CON REL LCL    SHR NOEXE    RD NOWRT LONG
  2 $LOCAL                                        2024   PIC CON REL LCL NOSHR NOEXE    RD   WRT LONG

    Total Space Allocated                         4639


ENTRY POINTS

    Address   Type  Name

  0-00000000        DUP11


VARIABLES

    Address   Type  Name                                        Address   Type  Name

  2-0000039C   I*4  ADDRS                                     2-000003A0   I*4  CHAR
  2-00000399   L*1  DIAGNOSTIC_MODE                           2-000003A4   I*4  FIELD
  2-000003AC   I*4  I                                        AP-00000004@ L*1  LUN
 AP-00000008@ I*4  PARCSR                                   AP-0000000C@ I*4  RXCSR
 AP-00000010@ I*4  RXDBUF                                     2-000003A8   I*4  TEMP
 AP-00000014@ I*4  TXCSR                                    AP-00000018@ I*4  TXDBUF


ARRAYS

    Address   Type  Name                     Bytes  Dimensions

  2-00000347   CHAR CAR_STATUS                  42  (0:1)
  2-00000371   CHAR CTS_STATUS                  40  (0:1)
  2-0000031D   CHAR DSR_STATUS                  42  (0:1)
  2-000002BD   CHAR DTR_STATUS                  52  (0:1)
  2-0000002E   CHAR OPER_MODE                   24  (0:1)
  2-000002B1   CHAR PARCSR_1                    12  (9:9)
  2-00000000   CHAR PROTOCOL                    46  (0:1)
  2-000002F1   CHAR RTS_STATUS                  44  (0:1)
  2-00000046   CHAR RXCSR_1                     18  (0:0)
  2-00000058   CHAR RXCSR_2                    156  (3:8)
  2-000000F4   CHAR RXCSR_3                     48  (10:11)
  2-00000124   CHAR RXCSR_4                     36  (14:15)
  2-00000148   CHAR RXDBUF_1                    78  (8:10)
  2-00000196   CHAR RXDBUF_2                    19  (12:12)
  2-000001A9   CHAR RXDBUF_3                    34  (14:15)
  2-000001CB   CHAR TXCSR_1                      5  (4:4)
  2-000001D0   CHAR TXCSR_2                    116  (6:9)
  2-00000244   CHAR TXCSR_3                     28  (15:15)
  2-00000260   CHAR TXDBUF_1                    81  (8:10)
```

```
LABELS

    Address    Label        Address    Label        Address    Label        Address    Label        Address    Label        Address    Label

  1-0000003B  20'         1-0000004E  30'         1-00000058  40'         1-00000071  50'         1-0000007D  60'         1-00000089  70'
  1-00000095  80'         1-000000A1  90'         1-000000AD  100'        1-000000B7  110'        1-000000CA  120'        1-000000F2  130'
  1-00000104  140'        1-00000123  150'        1-0000013A  160'        1-0000014D  170'        1-00000178  180'        1-000001A3  190'
  1-000001AF  200'        1-000001D5  210'        1-000001FD  220'        1-0000022A  230'


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name               Type  Name               Type  Name               Type  Name               Type  Name

   I*4  COMPRESS4           I*4  COMPRESSC           I*4  LIB$EXTZV                LINCHK                   OUTPUT


COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:DUP11/OBJ=OBJ$:DUP11 MSRC$:DUP11

  /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:           5.30 seconds
  Elapsed Time:       13.85 seconds
  Page Faults:        201
  Dynamic Memory:     200 pages
```

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY