

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPP	

_s
Va
--
000
000
000
7F1
7F1
7F1
7F1
7F1
7F1
7F1
7F1

```

UU      UU  EEEEEEEEE  TTTTTTTTT  DDDDDDD  RRRRRRR  11      WW      WW  000000  000000
UU      UU  EEEEEEEEE  TTTTTTTTT  DDDDDDD  RRRRRRR  11      WW      WW  000000  000000
UU      UU  EE          TT          DD      DD  RR      RR  1111    WW      WW  00      00  00      00
UU      UU  EE          TT          DD      DD  RR      RR  1111    WW      WW  00      00  00      00
UU      UU  EE          TT          DD      DD  RR      RR  11      WW      WW  00      0000  00      0000
UU      UU  EE          TT          DD      DD  RR      RR  11      WW      WW  00      0000  00      0000
UU      UU  EEEEEEEEE  TT          DD      DD  RRRRRRR  11      WW      WW  00  00  00  00  00  00
UU      UU  EEEEEEEEE  TT          DD      DD  RRRRRRR  11      WW      WW  00  00  00  00  00  00
UU      UU  EE          TT          DD      DD  RR      RR  11      WW      WW  0000  00  0000  00
UU      UU  EE          TT          DD      DD  RR      RR  11      WW      WW  0000  00  0000  00
UU      UU  EE          TT          DD      DD  RR      RR  11      WWW     WWW  00      00  00      00
UU      UU  EE          TT          DD      DD  RR      RR  11      WWW     WWW  00      00  00      00
UUUUUUUU  EEEEEEEEE  TT          DDDDDDD  RR      RR  111111  WW      WW  000000  000000
UUUUUUUU  EEEEEEEEE  TT          DDDDDDD  RR      RR  111111  WW      WW  000000  000000

```

```

LL      11111  SSSSSSS
LL      11111  SSSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SSSSSS
LL      11      SSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LLLLLLLL  11111  SSSSSSS
LLLLLLLL  11111  SSSSSSS

```

(2)	72	Declarations
(4)	207	Read-Only Data
(7)	433	Read/Write Data
(8)	526	RMS-32 Data Structures
(9)	568	Test and Device Initialization
(12)	861	Test the DR11-W
(16)	1068	Routine to Dump Debugging Info
(17)	1158	DR11-W AST Receiver
(18)	1192	Restore Original DR11-W Characteristics
(19)	1250	Timer Expiration Routine
(20)	1285	System Service Exception Handler
(21)	1414	RMS Error Handler
(22)	1478	CTRL/C Handler
(23)	1523	Error Exit
(24)	1584	Exit Handler

```

0000 1 .TITLE UETDR1W00 - VAX/VMS UETP DR11-W EXERCISER
0000 2 .IDENT 'V04-000'
0000 3 .ENABLE SUPPRESSION
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY:
0000 31 This module will be distributed with VAX/VMS under the [SYSTEST]
0000 32 account.
0000 33
0000 34 ABSTRACT:
0000 35 Using $QIO System Services, this program exercises the maintenance mode
0000 36 functions of a DR11-W.
0000 37
0000 38 ENVIRONMENT:
0000 39 This program will run in user access mode, with ASTs enabled except
0000 40 during error processing. The program requires an AST limit of 6, a
0000 41 buffered I/O limit of 10(10) and the PHY_IO and DIAGNOSE privileges.
0000 42
0000 43 --
0000 44
0000 45 AUTHOR: Richard N. Holstein, CREATION DATE: August, 1981
0000 46
0000 47 MODIFIED BY:
0000 48
0000 49 V03-005 RNH0006 Richard N. Holstein, 15-Feb-1984
0000 50 Take advantage of the new UETP message codes. Fix SSERROR
0000 51 interaction with RMS_ERROR.
0000 52
0000 53 V03-004 RNH0005 Richard N. Holstein, 19-Dec-1983
0000 54 Give correct sentinels to Test Controller. Use LIB$SIGNAL or
0000 55 $PUTMSG throughout, instead of LIB$PUT_OUTPUT.
0000 56
0000 57 V03-003 RNH0004 Richard N. Holstein, 11-Mar-1983

```

```
0000 58 : Don't signal ending message in EXIT_HANDLER.
0000 59 :
0000 60 : V03-002 RNH0003 Richard W. Holstein, 23-Feb-1983
0000 61 : Allow for longer device names.
0000 62 :
0000 63 : V03-001 RNH0002 Richard N. Holstein, 15-Oct-1982
0000 64 : Miscellaneous fixes listed in the V3B UETP Workplan.
0000 65 :
0000 66 : V02-001 RNH0001 Richard N. Holstein, 4-Dec-1981
0000 67 : Fix problems in reset routine which caused errors when the
0000 68 : turnaround connector wasn't installed.
0000 69 :
0000 70 : **
```

```

0000 72      .SBTTL Declarations
0000 73      :
0000 74      : INCLUDE FILES:
0000 75      :
0000 76      :     SYS$LIBRARY:LIB.MLB      for general definitions
0000 77      :     SHRLIB$:UETP.MLB      for UETP definitions
0000 78      :
0000 79      : MACROS:
0000 80      :
0000 81      :     $CHFDEF      : Condition handler frame definitions
0000 82      :     $DEVDEF      : Device definitions
0000 83      :     $DIBDEF      : Device Information Block
0000 84      :     $DVIDEF      : $GETDVI ITMLST item codes
0000 85      :     $IODEF      : I/O functions codes, etc.
0000 86      :     $QIODEF     : $QIO offsets and NARGS
0000 87      :     $SHRDEF     : Shared messages
0000 88      :     $SSDEF      : System Service status codes
0000 89      :     $STSDEF     : Status return
0000 90      :     $UETUNTDEF  : UETP unit block offset definitions
0000 91      :     $UETPDEF   : UETP
0000 92      :     $XADEF     : DR11-W
0000 93      :
0000 94      : EQUATED SYMBOLS:
0000 95      :
0000 96      : Facility number definitions:
0000 97      :     RMSS_FACILITY = 1
0000 98      :
0000 99      : SHR message definitions:
00740000 0000 100 : UETP = UETPS_FACILITY@STSSV FAC_NO ; Define the UETP facility code
007410E0 0000 101 : UETPS_ABENDD = UETP!SHRS_ABENDD ; Define the UETP message codes
00741038 0000 102 : UETPS_BEGINN = UETP!SHRS_BEGINN
00741080 0000 103 : UETPS_ENEDDD = UETP!SHRS_ENEDDD
00741098 0000 104 : UETPS_OPENIN = UETP!SHRS_OPENIN
0074113C 0000 105 : UETPS_TEXT  = UETP!SHRS_TEXT
0000 106      :
0000 107      : Internal flag bits...:
0000 108      :     TEST_OVERV  = 1      ; Set when test is over
0000 109      :     SAFE_TO_UPDV = 2      ; Set if it's safe to update UETINIDEV
0000 110      :     BEGIN_MSGV  = 3      ; Set if 'BEGIN' msg has been printed
0000 111      :     ONE_SHOTV   = 4      ; Set if running in one-shot mode
0000 112      :     DUMP_MODEV  = 5
0000 113      :     NO_MESSAGEV = 6      ; Set if bad data msg given after $QIO
0000 114      : ...and corresponding masks:
0000 115      :     TEST_OVERM  = 1@TEST_OVERV
0000 116      :     SAFE_TO_UPDM = 1@SAFE_TO_UPDV
0000 117      :     BEGIN_MSGM  = 1@BEGIN_MSGV
0000 118      :     ONE_SHOTM   = 1@ONE_SHOTV
0000 119      :     DUMP_MODEM  = 1@DUMP_MODEV
0000 120      :     NO_MESSAGEM = 1@NO_MESSAGEV
0000 121      :
0000 122      : Miscellany:
0000 123      :     LC_BITM     = ^X20   ; Mask to convert lower case to upper
0000 124      :     REC_SIZE    = 40     ; UETINIDEV.DAT record size
0000 125      :     TEXT_BUFFER = 500    ; Internal text buffer size
0000 126      :     EFN2        = 4      ; EFN used for three minute timer
0000 127      :     SS_SYNCH_EFN = 3     ; Synch miscellaneous system services
0000 128      :     MAX_PROC_NAME = 15   ; Longest possible process name

```

0000000A 0000 129
00000005 0000 130

MAX_DEV_DESIG = 10
MAX_UNIT_DESIG= 5

; Longest possible controller name
; Longest possible unit number

```

00000005 0000 132 ; DR11-W specific definitions:
000003E8 0000 133 ; QIO_EFN = EFN2+1 ; EFN for DR11-W I/O
000003E8 0000 134 ; DWT_SIZE = 1000 ; Typical DR11-W transfer size in bytes
00000258 0000 135 ; .IIF NE DWT_SIZE&1, ERROR DWT_SIZE ; DWT_SIZE must be an even number!
00000258 0000 136 ; .IIF GE DWT_SIZE-65535, ERROR DWT_SIZE ; DWT_SIZE must be less than 65535!
00000258 0000 137 ; MINIMUM = 30000/<DWT_SIZE/2> ; Min. acceptable $QIOs for normal run
00000258 0000 138 ; ; = transfer-rate-in-words-per-second/transfer-size-in-words
00000258 0000 139 ; ; The test lasts well over a second and not all $QIOs transfer so many
00000258 0000 140 ; ; bytes, so this should be quite a minimal value!
00000258 0000 141 ;
00000258 0000 142 ; Note well that the DR11-W transfers words, but VMS counts bytes, and that in
00000258 0000 143 ; maintenance mode, "the DR11-W does alternating DATI/DATO transfers at
00000258 0000 144 ; consecutive locations; a DATI from location X followed by a DATO to location
00000258 0000 145 ; X+2, followed by a DATI from location X+4 and so on." - DR11-W Specification,
00000258 0000 146 ; September 1980 revision. The word count is decremented twice for each word
00000258 0000 147 ; of data, once for the DATI and once for the DATO.
000003E8 0000 148 ; WRITE_SIZE = DWT_SIZE ; Buffer size in bytes
000003E8 0000 149 ;
0000190B 0000 150 ; For conciseness later on, define here bit masks for I/O function codes.
0000110B 0000 151 ; XAW_RESET_CYCLE = IOS_WRITEPBLK!IOSM_DIAGNOSTIC!IOSM_RESET!IOSM_CYCLE
0000130B 0000 152 ; XAW_CYCLE = IOS_WRITEPBLK!IOSM_DIAGNOSTIC!IOSM_CYCLE
0000130B 0000 153 ; XAW_SETFNCT_CYCLE = IOS_WRITEPBLK!IOSM_DIAGNOSTIC!IOSM_SETFNCT!IOSM_CYCLE
0000118B 0000 154 ; XAW_TIMED_CYCLE = IOS_WRITEPBLK!IOSM_DIAGNOSTIC!IOSM_TIMED!IOSM_CYCLE
0000118B 0000 155 ;
0000118B 0000 156 ;
0000118B 0000 157 ; For each unit, there will be a data structure set up, called a
0000118B 0000 158 ; node. These nodes will be linked together in a self-relative queue whose
0000118B 0000 159 ; header is UNIT_LIST. The first part of each node will be the standard
0000118B 0000 160 ; definition from $UETUNDEF. Following that will come the device test
0000118B 0000 161 ; dependent stuff, defined below. NOTE THAT THIS DEFINITION IS DONE WITH AN
0000118B 0000 162 ; ABSOLUTE PSECT. This means that what look like declarations are really
0000118B 0000 163 ; definitions and the labels are really just offsets into a given node on the
0000118B 0000 164 ; queue. (A not necessarily obvious consequence of using an ABS PSECT is that
0000118B 0000 165 ; space must be reserved with .BLKx operations, since .BYTE, etc., attempt to
0000118B 0000 166 ; store data.)
0000118B 0000 167 ;
0000118B 0000 168 ; .PSECT DEVDEP_STR_DEF,ABS,NOEXE,NOWRT,PAGE ; Note ABS attribute!
0000118B 0000 169 ;
000001A4 0000 170 ; .BLKB UETUNT$C_DEVDEP ; Skip over standard UETUNT block
000001A4 01A4 171 ;
000001AC 01A4 172 XA_Q_IOSB: ; IOSB for our DR11-W
000001AC 01A4 173 ; .BLKQ 1
000001AC 01AC 174 ;
000001B4 01AC 175 XA_Q_CHARAC: ; Characteristics buffer for IOS_SETCHAR
000001B4 01AC 176 ; .BLKQ 1
000001B4 01B4 177 ;
000001BC 01B4 178 XA_Q_ORIGINAL: ; DR11-W characteristics before starting tes
000001BC 01B4 179 ; .BLKQ 1
000001BC 01BC 180 ;
000001F0 01BC 181 XA_K_QIO: ; Space for $QIO_G argument list...
000001F0 01BC 182 ; .BLKL QIOS_NARGS+1 ; ...and the argument list counter
000001F0 01F0 183 ;
000001F0 01F0 184 ;
000001F0 01F0 185 ; All the code which uses the following two items is heavily dependent on its
000001F0 01F0 186 ; dealing with quadword pairs (i.e., 16-byte) of time stamps and on the
000001F0 01F0 187 ; number of quadword pairs fitting into a byte.
000001F0 01F0 188 ;

```

```

00000100 01F0 189          TIME_STAMP_LEN = 256          ; 2*count of start/finish quadword pairs
          01F0 190 XA_B_TSI:                          ; Index into the following...
000001F1 01F0 191          .BLKB 1
          01F1 192 XA_K_TSTAMP:                        ; ...circular list of time stamps of...
000009F1 01F1 193          .BLKB TIME_STAMP_LEN        ; ...last start/finish of $QIOs
          09F1 194
0000084D 09F1 195          DEVDEP_SIZE = .-UETUNT$C_DEVDEP ; Device dependent part size of node
          09F1 196          ; Note that this excludes buffers
          09F1 197
          09F1 198          .ALIGN WORD                ; Unibus DDP requires word alignment
          09F2 199 XA_K_BUF:                            ; I/O buffer for both reads and writes
00000DDA 09F2 200          .BLKB WRITE_SIZE
          ODDA 201
          ODDA 202          PAGES = <<UETUNT$C_INDSIZ+-    ; Add together all of the pieces...
          ODDA 203          DEVDEP_SIZE+-              ; ...which make up a UETP unit block...
          ODDA 204          WRITE_SIZE+-              ; ...to give to the $EXPREG service...
00000007 ODDA 205          511>/512>                    ; ...later

```

```

      ODDA      207      .SBTTL  Read-Only Data
00000000      208      .PSECT  RODATA,NOEXE,NOWRT,PAGE
      0000      209
53 45 54 53 59 53 00000008'010E0000' 0000 210 ACNT_NAME:           ; Process name on exit
      54 000E      211      .ASCID  /SYSTEST/
      000F      212
31 52 44 54 45 55 00000017'010E0000' 000F 213 TEST_NAMEF:           ; This test name
      30 30 57 001D 214      .ASCID  /UETDR1W00/
      0020      215
50 55 53 54 45 55 00000028'010E0000' 0020 216 SUPDEV_GBLSEC:         ; How we access UETSUPDEV.DAT
      56 45 44 002E 217      .ASCID  /UETSUPDEV/
      0031      218
41 4E 4C 52 54 43 00000039'010E0000' 0031 219 CONTROLLER:           ; Logical name of controller
      45 4D 003F 220      .ASCID  /CTRLNAME/
      0041      221
      45 44 4F 4D 00000049'010E0000' 0041 222 MODE:                   ; Run mode logical name
      004D 223      .ASCID  /MODE/
      004D 224
      00000000' 004D 225 NO_RMS_AST TABLE:           ; List of errors for which...
      00000000' 0051 226      .LONG   RMS$_BLN           ; ...RMS cannot deliver an AST...
      00000000' 0055 227      .LONG   RMS$_BUSY          ; ...even if one has an ERR= arg
      00000000' 0059 228      .LONG   RMS$_CDA           ; Note that we can search table...
      00000000' 005D 229      .LONG   RMS$_FAB           ; ...via MATCHC since <31:16>...
      00000014 0061 230      .LONG   RMS$_RAB           ; ...pattern can't be in <15:0>
      0061 231 NRAT_LENGTH = -.NO_RMS_AST_TABLE
      0061 232
4E 49 24 53 59 53 00000069'010E0000' 0061 233 SYSS$INPUT:             ; Name of device from which...
      54 55 50 006F 234      .ASCID  /SYSS$INPUT/           ; ...the test can be aborted
      0072 235
      0020 0040 0072 236 INPUT_ITMLST:           ; $GETDVI arg list for SYSS$INPUT
0000000C'00000014' 0076 237      .WORD   64,DVIS$ DEVNAM          ; We need the equivalence name
      00000000 007E 238      .LONG   BUFFER,BUFFER_PTR
      0082 239      .LONG   0
      0082 240
21 20 42 58 32 21 0000008A'010E0000' 0082 241 CS1:                   ; Device class and type control string
      20 42 58 32 0090 242      .ASCID  /!2XB !2XB /
      0094 243
2A 20 42 58 32 21 0000009C'010E0000' 0094 244 CS3:                   ; Device class-only control string
      2A 00A2 245      .ASCID  /!2YB **/
      00A3 246
65 74 72 6F 62 41 000000AB'010E0000' 00A3 247 CNTRLMSG:             ;
72 65 73 75 20 61 20 61 69 76 20 64 00B1 248      .ASCID  \Aborted via a user CTRL/C\
      43 2F 4C 52 54 43 20 00BD
      00C4 249
6E 6F 63 20 6F 4E 0000C0CC'010E0000' 00C4 250 NO_CTRLNAME:           ;
63 65 70 73 20 72 65 6C 6C 6F 72 74 00D2 251      .ASCID  /No controller specified./
      2E 64 65 69 66 69 00DE
      00E4 252

```

```

20 74 27 6E 61 43 000000EC'010E0000' 00E4 253 DEAD_CTRLNAME:
6C 6F 72 74 6E 6F 63 20 74 73 65 74 00E4 254 .ASCID /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
72 61 6D 20 2C 53 41 21 20 72 65 6C 00F2
61 73 75 6E 75 20 73 61 20 64 65 6B 00FE
4E 49 54 45 55 20 6E 69 20 65 6C 62 010A
2E 54 41 44 2E 56 45 44 49 0116
0122
0128 255
0128 256 NOUNIT_SELECTED:
0128 257 .ASCID /No units selected for testing./
0139
0145
0151 258
0151 259 ILLEGAL_REC:
0151 260 .ASCID /Illegal record format in file UETINIDEV.DAT!/
015F
016B
0177
0183
0185 261
0185 262 PASS_MSG:
0185 263 .ASCID /End of pass !UL with !UL iterations at !%D./
0193
019F
01AB
01B7
01B8 264
01B8 265 INIDEV_UPDERR: ; Error during exit handler
01B8 266 .ASCID /Error updating UETINIDEV.DAT./
01C6
01D2
01DD 267
01DD 268 THREEMIN: ; 3 minute delta time
01DD 269 .LONG -10*1000*1000*180,-1
01E5 270
01E5 271 UNIT_DESC: ; Descriptor used to convert unit #
01E5 272 .LONG 5
01E9 273 .ADDRESS BUFFER+6
01ED 274
01ED 275 CONT_DESC: ; Descriptor used to convert controller...
01ED 276 .WORD REC_SIZE,0 ; ...from lowercase to uppercase
01F1 277 .ADDRESS BUFFER
01F5 278
01F5 279 FILE: ; Fills in RMS_ERR_STRING
01F5 280 .ASCID /file/
0201 281
0201 282 RECORD: ; Fills in RMS_ERR_STRING
0201 283 .ASCID /record/
020F 284
020F 285 RMS_ERR_STRING: ; Announces an RMS error
020F 286 .ASCID /RMS !AS error in file !AD/
021D
0229
0230 287
0230 288 PROMPT:
0230 289 .ASCII /Controller designation?: /
023C

```

```

                20 0248
00000019 0249 290
                0249 291
                0249 292
73 69 20 53 41 21 00000251'010E0000' 0249 293
62 61 6C 69 61 76 61 20 74 6F 6E 20 0257
69 74 73 65 74 20 72 6F 66 20 65 6C 0263
                2E 67 6E 026F
                0272 294
                0272 295
6E 6F 20 43 41 21 0000027A'010E0000' 0272 296
64 65 74 65 6C 70 6D 6F 63 20 79 6C 0280
                2E 73 4F 49 51 24 20 4C 55 21 20 028C
                0297 297
65 20 4F 49 51 24 0000029F'010E0000' 0297 298
69 76 65 64 20 6E 6F 20 72 6F 72 72 02A5 299
                2E 43 41 21 20 65 63 02B1
                02B8 300
6F 20 4F 49 51 24 000002C0'010E0000' 02B8 301
64 65 6C 69 61 66 20 43 41 21 20 6E 02C6 302
21 20 6E 6F 69 74 63 6E 75 66 20 2C 02D2
4C 58 21 20 42 53 4F 49 20 2C 4C 58 02DE
                2E 4C 58 21 20 02EA
                02EF 303
                02EF 304
75 62 65 44 2F 21 000002F7'010E0000' 02EF 305
20 30 52 20 20 3A 6F 66 6E 69 20 67 02FD
5F 24 4F 49 51 20 2C 4C 58 21 20 3D 0309
21 20 3D 20 29 30 31 52 28 4E 46 45 0315
                2C 4C 58 0321
28 4E 41 48 43 5F 24 4F 49 51 2F 21 0324 306
20 2C 4C 58 21 20 3D 20 29 30 31 52 0330
31 52 28 43 4E 55 46 5F 24 4F 49 51 033C
                2C 4C 58 21 20 3D 20 29 30 0348
44 41 54 53 41 5F 24 4F 49 51 2F 21 0351 307
4C 58 21 20 3D 20 29 30 31 52 28 52 035D
52 50 54 53 41 5F 24 4F 49 51 20 2C 0369
4C 58 21 20 3D 20 29 30 31 52 28 4D 0375
                2C 0381
31 52 28 31 50 5F 24 4F 49 51 2F 21 0382 308
49 51 20 2C 4C 58 21 20 3D 20 29 30 038E
3D 20 29 30 31 52 28 32 50 5F 24 4F 039A
                2C 4C 58 21 20 03A6
31 52 28 33 50 5F 24 4F 49 51 2F 21 03AB 309
49 51 20 2C 4C 58 21 20 3D 20 29 30 03B7
3D 20 29 30 31 52 28 34 50 5F 24 4F 03C3
50 5F 24 4F 49 51 20 2C 4C 58 21 20 03CF
4C 58 21 20 3D 20 29 30 31 52 28 35 03DB
                2C 03E7
28 42 53 4F 49 5F 51 5F 41 58 2F 21 03E8 310
58 20 2C 4C 58 21 20 3D 20 29 36 52 03F4
52 28 34 2B 42 53 4F 49 5F 51 5F 41 0400
                4C 58 21 20 3D 20 29 36 040C
41 52 41 48 43 5F 51 5F 41 58 2F 21 0414 311
2C 4C 58 21 20 3D 20 29 36 52 28 43 0420

```

```

                PMTSIZ = .-PROMPT
DEVALLOC: ; Warns if DR11-W already assigned
                .ASCID /!AS is not available for testing./
SLOW_DR11W: ; Warns if DR11-W didn't do min. I/O
                .ASCID /!AC only completed !UL $QIOs./
QIO_ERROR: ; Message if $QIO itself fails
                .ASCID /$QIO error on device !AC./
QIO_FUNC_FAIL: ; Message if $QIO function failed
                .ASCID \ $QIO on !AC failed, function !XL, IOSB !XL !XL.\
DEBUG_MSG: ; Useful stuff if test dies
                .ASCID \!/Debug info: R0 = !XL, QIOS_EFN(R10) = !XL,\-
                \!/QIOS_CHAN(R10) = !XL, QIOS_FUNC(R10) = !XL,\-
                \!/QIOS_ASTADR(R10) = !XL, QIOS_ASTPRM(R10) = !XL,\-
                \!/QIOS_P1(R10) = !XL, QIOS_P2(R10) = !XL,\-
                \!/QIOS_P3(R10) = !XL, QIOS_P4(R10) = !XL, QIOS_P5(R10) = !XL,\-
                \!/XA_Q_IOSB(R6) = !XL, XA_Q_IOSB+4(R6) = !XL\-\
                \!/XA_Q_CHARAC(R6) = !XL, XA_Q_CHARAC+4(R6) = !XL\

```

```

43 41 52 41 48 43 5F 51 5F 41 58 20 042C
4C 58 21 20 3D 20 29 36 52 28 34 2B 0438
                                0444
                                0444
67 61 69 44 2F 21 0000044C '010E0000' 0444
65 66 66 75 62 20 63 69 74 73 6F 6E 0452
2F 21 29 4C 58 39 28 36 21 20 3A 72 045E
21 20 3D 20 50 4D 54 52 53 43 2F 21 046A
3D 20 50 4D 54 52 41 42 20 2C 4C 58 0476
20 20 20 52 53 43 20 2C 4C 58 21 20 0482
20 52 49 45 20 2C 4C 58 21 20 3D 20 048E
                                049A
21 20 3D 20 20 20 20 52 44 49 2F 21 04A2
3D 20 20 20 20 52 41 42 20 2C 4C 58 04AE
20 20 20 52 43 57 20 2C 4C 58 21 20 04BA
4F 52 52 45 20 2C 4C 58 21 20 3D 20 04C6
                                04D2
21 20 3D 20 6D 75 6E 52 50 44 2F 21 04DA
3D 20 6E 6F 63 52 50 44 20 2C 4C 58 04E6
20 20 52 50 4D 46 20 2C 4C 58 21 20 04F2
52 50 4D 50 20 2C 4C 58 21 20 3D 20 04FE
                                050A
21 20 3D 20 72 61 70 52 50 44 2F 21 0512
29 4C 58 39 28 38 21 2F 21 4C 58 051E

```

```

312
313 DIAG_MSG: ; Contents of $QIO P6 buffer
314 .ASCID \!/Diagnostic buffer: !6(9XL)!/\-
315 \!/CSRTMP = !XL, BARTMP = !XL, CSR = !XL, EIR = !XL\!-
316 \!/IDR = !XL, BAR = !XL, WCR = !XL, ERROR = !XL\!-
317 \!/DPRnum = !XL, DPRcon = !XL, FMPR = !XL, PMPR = !XL\!-
318 \!/DPRpar = !XL!/\!8(9XL)\

```

```

0529 320 :
0529 321 : The TABLE_GEN macro is used to coherently and consistently lay out the
0529 322 : parameters which will change from $QIO to $QIO when testing the DR11-W.
0529 323 : Each line is a call to the LINE_GEN macro. The LINE_GEN macro will be
0529 324 : expanded to fill in a set of parallel tables from which the parameters
0529 325 : will be taken when the $QIO is performed.
0529 326 :
0529 327 : Because these tables introduce an additional level of indirection in the
0529 328 : arguments, the typical $QIO_S form of the system service becomes unuseable.
0529 329 : We will use the $QIO_G form. The argument list will be reserved space in
0529 330 : the node on UNIT LIST for the DR11-W; we can't use the $QIO macro there
0529 331 : because of ABS .PSECT restrictions. Define a dummy argument list now with
0529 332 : the items which can be supplied at assembly time:
0529 333 :
0529 334 DUMMY_QIO:
0529 335     $QIO     EFN = QIO_EFN, P6 = DIAG_BUF
0529 336 :
055D 337 : The rest of the arguments will be supplied as the node is allocated (those
055D 338 : which are static across $QIOs) or as an individual $QIO is done.
055D 339 :-
055D 340 :
055D 341 .MACRO TABLE_GEN
055D 342 :
055D 343 : The function codes used below are all defined earlier in the DR11-W Specific
055D 344 : definitions area.
055D 345 :
055D 346 : First, try some basic functions: in maintenance mode, do various length word
055D 347 : and block mode transfers. Since logical, virtual and physical I/O are the
055D 348 : same to the DR11-W, doing all I/O in physical mode is sufficient. Physical
055D 349 : mode I/O is necessary to access the DR11-W in maintenance mode.
055D 350 :
055D 351     LINE_GEN IO$_SETCHAR, 0, XA_Q_CHARAC, 0, 0, 0, 0
055D 352 :
055D 353     LINE_GEN     FUNC, ASTADR,     P1,     P2, P3, P4, P5
055D 354     LINE_GEN XAW__RESET_CYCLE, 0, XA_K_BUF,     4, 0, 0, 0
055D 355     LINE_GEN XAW__RESET_CYCLE, 0, XA_K_BUF,     4, 0, 0, 0
055D 356     LINE_GEN XAW__CYCLE, 0, XA_K_BUF,     4, 0, 0, 0
055D 357     LINE_GEN XAW__CYCLE, 0, XA_K_BUF, DWT_SIZE, 0, 0, 0
055D 358     LINE_GEN XAW__CYCLE, 0, XA_K_BUF, DWT_SIZE, 0, 0, 0
055D 359 :
055D 360 : Play with FNCT and STATUS bits. The set won't transfer any
055D 361 : data, per se, but will cause the DR11-W IDR and ODR to be accessed.
055D 362 : NOTE: This function cancelled because it works only if the turnaround
055D 363 : connector is installed.
055D 364 :
055D 365     LINE_GEN     FUNC, ASTADR,     P1, P2, P3, P4,     P5
055D 366     LINE_GEN XAW__SETFNCT_CYCLE, 0, XA_K_BUF, 0, 0, 5, <^XA72E>
055D 367 :
055D 368 : Do some word and block mode transfers with a timeout parameter. Get an
055D 369 : AST when the device finishes.
055D 370 :
055D 371     LINE_GEN     FUNC, ASTADR,     P1,     P2, P3, P4, P5
055D 372     LINE_GEN XAW__TIMED_CYCLE, IOAST, XA_K_BUF, DWT_SIZE, 2, 0, 0
055D 373 :
055D 374 .ENDM TABLE_GEN

```

```

055D 376 :+
055D 377 : We now need to generate the set of parallel tables from which the $QIOs
055D 378 : will take their arguments. Define the LINE_GEN macro twice, the first time
055D 379 : to count the number of calls, and the second time to fill the tables. In
055D 380 : between, allocate the space for the tables.
055D 381 :-
055D 382 .MACRO LINE_GEN FUNC,ASTADR,P1,P2,P3,P4,P5
055D 383 LINE_GEN_COUNT = LINE_GEN_COUNT+1
055D 384 .ENDM LINE_GEN
055D 385
00000000 055D 386 LINE_GEN_COUNT = 0
055D 387 TABLE_GEN ; This one counts LINE_GEN calls
055D 388
055D 389 .ALIGN LONG ; May as well speed things up a bit
0560 390
0000057C 0560 391 FUNC_TABLE: ; $QIO function code
0560 392 .BLKL LINE_GEN_COUNT
057C 393
00000598 057C 394 ASTADR_TABLE: ; AST routine when I/O completes
057C 395 .BLKL LINE_GEN_COUNT
0598 396
000005B4 0598 397 P1_TABLE: ; Data buffer, characteristics buffer
0598 398 .BLKL LINE_GEN_COUNT ; or attention AST service routine
05B4 399
000005D0 05B4 400 P2_TABLE: ; Byte size of data buffer
05B4 401 .BLKL LINE_GEN_COUNT
05D0 402
000005EC 05D0 403 P3_TABLE: ; Timeout in seconds or AST access mode
05D0 404 .BLKL LINE_GEN_COUNT
05EC 405
00000608 05EC 406 P4_TABLE: ; CSR FNCT bits (2-0 only)
05EC 407 .BLKL LINE_GEN_COUNT
0608 408
00000624 0608 409 P5_TABLE: ; Value (word) to load into ODR
0608 410 .BLKL LINE_GEN_COUNT
0624 411
0624 412 .MACRO LINE_GEN FUNC,ASTADR,P1,P2,P3,P4,P5
0624 413 . = FUNC_TABLE+<4*LINE_GEN_COUNT>
0624 414 .LONG FUNC
0624 415 . = ASTADR_TABLE+<4*LINE_GEN_COUNT>
0624 416 .ADDRESS ASTADR
0624 417 . = P1_TABLE+<4*LINE_GEN_COUNT>
0624 418 .ADDRESS P1
0624 419 . = P2_TABLE+<4*LINE_GEN_COUNT>
0624 420 .LONG P2
0624 421 . = P3_TABLE+<4*LINE_GEN_COUNT>
0624 422 .LONG P3
0624 423 . = P4_TABLE+<4*LINE_GEN_COUNT>
0624 424 .LONG P4
0624 425 . = P5_TABLE+<4*LINE_GEN_COUNT>
0624 426 .LONG P5
0624 427 LINE_GEN_COUNT = LINE_GEN_COUNT+1
0624 428 .ENDM LINE_GEN
0624 429
00000000 0624 430 LINE_GEN_COUNT = 0
0624 431 TABLE_GEN ; This one fills the above tables

```

```

0624 433 .SBTTL Read/Write Data
00000000 434 .PSECT RWDATA,WRT,NOEXE,PAGE
0000 435
0000 436 TTCHAN: ; Channel associated with ctrl. term.
0000 437 .WORD 0
0002 438
0002 439 FLAG: ; Miscellaneous flag bits
0000 440 .WORD 0 ; (See Equated Symbols for definitions)
0004 441
0004 442 FAO_BUF: ; FAO output string descriptor
0000 01F4 443 .WORD TEXT_BUFFER,0
00000014 444 .ADDRESS BUFFER
000C 445
000C 446 BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 01F4 447 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
00000014 448 .ADDRESS BUFFER
0014 449
0014 450 BUFFER: ; FAO output and other misc. buffer
00000208 451 .BLKB TEXT_BUFFER
0208 452
0208 453 DEV_DSC: ; Device name descriptor
0000 000A 454 .WORD MAX_DEV_DESIG,0
00000227 455 .ADDRESS DEV_NAME
0210 456
0210 457 PROCESS_NAME: ; Process name
57 31 52 44 00000218 010E0000 458 .ASCID /DR1W/
0000000B 021C 459 PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
00000227 021C 460 .BLKB PROCESS_NAME_FREE
0227 461
0227 462 DEV_NAME: ; Device name buffer
00000236 0227 463 .BLKB MAX_DEV_DESIG+MAX_UNIT_DESIG
0000000F 0236 464 NAME_LEN = :-DEV_NAME
0236 465
0236 466 DIB: ; Device Information Block
0000 0074 0236 467 .WORD DIBSK_LENGTH,0
0000023E 023A 468 .ADDRESS DIBBUF
023E 469
000002B2 023E 470 .BLKB DIBSK_LENGTH
02B2 471
02B2 472 ERROR_COUNT: ; Cumulative error count at runtime
00000000 02B2 473 .LONG 0
02B6 474
02B6 475 STATUS: ; Status value on program exit
00000000 02B6 476 .LONG 0
02BA 477
00000000 00000000 02BA 478 QUAD_STATUS: ; IO status block for misc sys. svcs.
02BA 479 .QUAD 0
02C2 480
00000000 00000000 02C2 481 INADDRESS: ; $CRMPSC address storage
02C2 482 .LONG 0,0
02CA 483
02CA 484 OUTADDRESS:
00000000 00000000 02CA 485 .LONG 0,0
02D2 486
0000 02D2 487 DEVNAM_LEN: ; Current device name length
02D2 488 .WORD 0
02D4 489

```

```

02D4 490 ; RANDOM1 and RANDOM2 may be combined to produce a set of pseudo-random numbers
02D4 491 RANDOM1: ; Random word #1
AAAAAAA 02D4 492 .LONG ^XAAAAAAA
02D8 493
02D8 494 RANDOM2: ; Random word #2
A72EA72E 02D8 495 .LONG ^XA72EA72E
02DC 496
02DC 497 ITERATION: ; # of times all tests were executed
00000000 02DC 498 .LONG 0
02E0 499
02E0 500 PASS: ; Pass count
00000000 02E0 501 .LONG 0
02E4 502
02E4 503 MSG_BLOCK: ; Auxiliary $GEIMSG info
000002E8 02E4 504 .BLKB 4
02E8 505
02E8 506 EXIT_DESC: ; Exit handler descriptor
00000000 02E8 507 .LONG 0
00000AC2 02EC 508 .ADDRESS EXIT_HANDLER
00000001 02F0 509 .LONG 1
000002B6 02F4 510 .ADDRESS STATUS
02F8 511
02F8 512 ARG_COUNT: ; Argument counter used by ERROR_EXIT
00000000 02F8 513 .LONG 0
02FC 514
02FC 515 .ALIGN QUAD ; For self-relative queue of unit blocks
0300 516
0300 517 UNIT_LIST: ; Head of unit block circular list
00000000 00000000 0300 518 .QUAD 0
0308 519
0308 520 NEW_NODE: ; Newly acquired node address
00000000 00000000 0308 521 .QUAD 0
0310 522
0310 523 DIAG_BUF: ; $QIC P6 diagnostic buffer
000003D8 0310 524 .BLKL 50

```

```

03D8 526      .SBTTL RMS-32 Data Structures
03D8 527      .ALIGN LONG
03D8 528
03D8 529 SYSIN_FAB:      ; Allocate FAB for SYSS$INPUT
03D8 530      $FAB-
03D8 531      FNM = <SYSS$INPUT>
0428 532
0428 533 SYSIN_RAB:      ; Allocate RAB for SYSS$INPUT
0428 534      $RAB-
0428 535      FAB = SYSIN_FAB,-
0428 536      ROP = PMT,-
0428 537      PBF = PROMPT,-
0428 538      PSZ = PMTSIZ,-
0428 539      UBF = DEV_NAME,-
0428 540      USZ = NAME_LEN
046C 541
046C 542 INI_FAB:      ; Allocate FAB for UETINIDEV
046C 543      $FAB-
046C 544      FAC = <GET,PUT,UPD>,-
046C 545      RAT = CR,-
046C 546      SHR = <GET,PUT,UPI>,-
046C 547      FNM = <UETINIDEV.DAT>
04BC 548
04BC 549 INI_RAB:      ; Allocate RAB for UETINIDEV
04BC 550      $RAB-
04BC 551      FAB = INI_FAB,-
04BC 552      RBF = BUFFER,-
04BC 553      UBF = BUFFER,-
04BC 554      USZ = REC_SIZE
0500 555
00000506 0500 556 DDB_RFA:      ; RFA storage for INI_RAB
0500 557      .BLKB 6
0506 558
0506 559      .ALIGN LONG
0508 560 SUP_FAB:      ; Allocate FAB for UETSUPDEV
0508 561      $FAB-
0508 562      FAC = GET,-
0508 563      SHR = <UPI,GET>,-
0508 564      RAT = CR,-
0508 565      FOP = UFO,-
0508 566      FNM = <UETSUPDEV.DAT>

```

```

00000558 568 .SBTTL Test and Device Initialization
00000000 569 .PSECT DR11W,EXE,NOWRT,PAGE
0000 570
0000 571 .DEFAULT DISPLACEMENT,WORD
0000 572
0000 573 :+
0000 574 :+
0000 575 :+
0000 576 :+
0000 577 :+
0000 578 :-
0000 579
0000 580 .ENTRY UETDR1W00,^M<> ; Entry mask
6D 08B9'CF DE 0002 581
0002 582 MOVAL SSERROR,(FP) ; Declare exception handler
0007 583 $$SETSFM_S ENBFLG = #1 ; Enable system service failure mode
0010 584 $DCLEXH_S DESBLK = EXIT_DESC ; Declare an exit handler
001B 585
001B 586 $OPEN FAB = SYSIN FAB,- ; Open SYSS$INPUT
001B 587 ERR = RMS ERROR
002A 588 $CONNECT RAB = SYSIN RAB,- ; Connect RAB to SYSS$INPUT
002A 589 ERR = RMS ERROR
1E 0418'CF E1 0039 590 BBC S^#DEV$V TRM,- ; BR if SYSS$INPUT is NOT a terminal
003B 591 SYSIN FAB+FAB$L DEV,10$
003F 592 $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
003F 593 RSLLEN = DEVNAM_LEN,- ; ...a logical name...
003F 594 RSLBUF = DEVDSC ; ...for the controller to test
01 50 D1 0058 595 CMPL RO,#SS$ NORMAL ; Was a controller specified?
2E 13 005B 596 BEQL PROC_CONT_NAME ; BR if it was - go process it
005D 597 10$:
005D 598
005D 599 $GET RAB = SYSIN RAB,- ; Read SYSS$INPUT...
006C 600 MOVW SYSIN RAB+RAB$W_RSZ,- ; ...for the controller name
0070 601 DEVNAM_LEN ; Save the name length
0073 602 BNEQ PROC_CONT_NAME ; BR if we got something
02B6'CF 14 D0 0075 603 MOVL #SS$_BADPARAM,STATUS ; Save an exit status if not
00C4'CF DF 007A 604 PUSHAL NO_CTRLNAME ; Prepare for message...
01 DD 007E 605 PUSHL #1 ; ...
00741132 8F DD 0080 606 PUSHL #UETP$_TEXT!STSS$K_ERROR ; ...
03 DD 0086 607 PUSHL #3 ; ...
09B7 31 0088 608 BRW ERROR_EXIT ; ...to tell of bad setup
0088 609
0088 610 PROC_CONT_NAME:
0208'CF 02D2'CF 3C 0088 611 MOVZWL DEVNAM_LEN,DEVDSC ; Set the device name length
0208'CF DF 0092 612 PUSHAL DEVDSC ; Make sure...
0208'CF DF 0096 613 PUSHAL DEVDSC ; ...that the specified controller...
00000000'GF 02 FB 009A 614 CALLS #2,G^STR$UPCASE ; ...is all uppercase for later comparison
52 0208'CF 01 C1 00A1 615 ADDL3 #1,DEVVSC,R2 ; Estimate the eventual...
0210'CF 52 A0 00A7 616 ADDW2 R2,PROCESS_NAME ; ...process name length (incl. '"')
DE 00AC 617 MOVAL PROCESS_NAME+8- ; Locate first available byte...
00AD 618 +MAX_PROC_NAME- ; ...in process name handle...
50 021C'CF 00AD 619 -PROCESS_NAME_FREE,R0 ; ...for device name
08 C3 00B1 620 SUBL3 #PROCESS_NAME_FREE,- ; Will the device name fit...
51 52 00B3 621 R2,R1 ; ...in the remaining space?
08 15 00B5 622 BLEQ 10$ ; BR if it will
50 51 C2 00B7 623 SUBL2 R1,R0 ; Overwrite handle otherwise...
0210'CF 0F B0 00BA 624 MOVW #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length

```

				00BF	625	10\$:			
60	0227'CF	80 5F 8F	0208'CF	90 00BF	626		MOVB	#^A/ /,(R0)+	; Separate handle from device name
				28 00C3	627		MOVCS	DEVDS,DEV_NAME,(R0)	; Concatenate handle with device name
				D4 00CB	628		CLRL	-(SP)	; Set the time stamp flag
				000F'CF	629		PUSHAL	TEST_NAME	; Set the test name
				02	630		PUSHL	#2	; Push the argument count
				00741039 8F	631		PUSHL	#UETPS BEGIN!STSSK_SUCCESS	; Set the message code
				00000000'GF 04	632		CALLS	#4,G^LIB\$SIGNAL	; Print the startup message
				0002'CF 08	633		BISW2	#BEGIN MSGM,FLAG	; Set flag so we don't print it again
				A8 00E0	634		\$SETPRN_S	PRCNAM = PROCESS_NAME	; Set the process name to UETDR1W00_x
				00F0	635				
				02	636		BBC	S^#DEV\$V TRM,-	; BR if SYSS\$INPUT is NOT a terminal
				66 0418'CF	637		\$GETDVI_S	DEVNAM = SYSS\$INPUT,-	; Get the name of...
					638			EFN = #SS SYNCH EFN,-	; ...device which may abort test
					639			ITMLST = INPUT ITMLST,-	
					640			IOSB = QUAD_STATUS	
					641				
				45 02BA'CF	642		BLBC	QUAD STATUS,20\$; Avoid CTRL/C handler if any error
					643		\$ASSIGN_S	DEVNAM = BUFFER_PTR,-	; Set up for CTRL/C AST handler
					644			CHAN = TTCHAN	
					645		\$QIOW_S	CHAN = TTCHAN,-	; Enable CTRL/C AST's...
					646			FUNC = #IOS\$ SETMODE!IOSM_CTRLCAST,-	
					647			P1 = CCASTHAND	
					648		PUSHAL	PROCESS_NAME	; ...and tell the user...
					649		PUSHL	#1	
					650		PUSHL	#UETPS ABORTC!STSSK_SUCCESS	; ...how to abort gracefully...
					651		CALLS	#3,G^LIB\$SIGNAL	; ...
					652	20\$:			
					653		\$TRNLOG_S	LOGNAM = MODE,-	; Get the run mode
					654			RSLLEN = BUFFER_PTR,-	
					655			RSLBUF = FAO BUF	
					656		BICB2	#LC BITM,BUFFER	; Convert to upper case
					657		CMPB	#^A70/,BUFFER	; Is this a one shot?
					658		BNEQ	25\$; BR if not
					659		BISW2	#ONE_SHOTM,FLAG	; Set flag for one-shot mode
					660	25\$:			
					661		CMPB	#^A/DUMP/,BUFFER	; Special dump mode info wanted?
					662		BNEQ	27\$; BR if not
					663		BISW2	#DUMP_MODEM,FLAG	; Set flag for dump mode messages
					664	27\$:			
					0197				

```

0197 666 :
0197 667 : From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
0197 668 : and unit configuration and lets us know if the setup to run this test was
0197 669 : done correctly.
0197 670 :
0197 671 $OPEN FAB = INI_FAB,- ; Open file 'UETINIDEV.DAT'
0197 672 ERR = RMS_ERROR
01A6 673 $CONNECT RAB = INI_RAB,- ; Connect the RAB and FAB
01A6 674 ERR = RMS_ERROR
01B5 675 $MGBLSC_S INADR = INADDRESS,- ; Connect to UETSUPDEV global section
01B5 676 RETADR = OUTADDRESS,-
01B5 677 GSDNAM = SUPDEV_GBLSEC,-
01B5 678 FLAGS = #SECSM_EXPREG
00000978 8F 50 D1 01D4 679 CMPL RO,#SS$_NOSUCHSEC ; Was the section already there?
37 12 01DB 680 BNEQ 30$ ; BR if it was...
01DD 681 $OPEN FAB = SUP_FAB,- ; ...else open 'UETSUPDEV.DAT'
01DD 682 ERR = RMS_ERROR
01EC 683 $CRMPSC_S CHAN = SUP_FAB+FAB$SL_STV,- ; Create the global section
01EC 684 INADR = INADDRESS,-
01EC 685 RETADR = OUTADDRESS,-
01EC 686 GSDNAM = SUPDEV_GBLSEC,-
01EC 687 FLAGS = #SECSM_EXPREG!SECSM_GBL
56 02CE'CF 02CA'CF C3 0214 688 30$:
0214 689 SUBL3 OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
021C 690
021C 691 FIND_IT:
021C 692 $GET RAB = INI_RAB,- ; Get the first record
021C 693 ERR = RMS_ERROR
01ED'CF DF 022B 694 PUSHAL CONT_DESC ; Make sure...
01ED'CF DF 022F 695 PUSHAL CONT_DESC ; ...that the controller name...
00000000'GF 02 FB 0233 696 CALLS #2,G*STR$UPCASE ; ...is all uppercase letters
0014'CF 44 8F 91 023A 697 CMPB #^A/D/,BUFFER ; Is this a DDB?
27 13 0240 698 BEQL 10$ ; Go on if not
0014'CF 45 8F 91 0242 699 CMPB #^A/E/,BUFFER ; Is this the end of the file?
D2 12 0248 700 BNEQ FIND_IT ; Continue on if not
0208'CF DF 024A 701 PUSHAL DEV$DC ; Push device not supported message
0210'CF DF 024E 702 PUSHAL PROCESS_NAME ; Parameters on the stack
02 02 0252 703 PUSHL #2
00748333 8F DD 0254 704 PUSHL #UETPS_DENOSU
02 FO 025A 705 INSV #ST$K_ERROR,- ; Set the severity code...
00 025C 706 #ST$V_SEVERITY,-
6E 03 025D 707 #ST$S_SEVERITY,(SP)
02B6'CF 6E D0 025F 708 MOVL (SP),STATUS ; ...and save it as the exit status
04 DD 0264 709 PUSHL #4
07D9 31 0266 710 BRW ERROR_EXIT ; Exit in error
0269 711 10$:
0227'CF 001A'CF 02D2'CF 29 0269 712 CMPC DEVDNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
A7 12 0273 713 BNEQ FIND_IT ; BR if not
0500'CF 04CC'CF 06 28 0275 714 MOVC3 #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the record file address
0018'CF 54 8F 91 027D 715 CMPB #^A/T/,BUFFER+4 ; Can we test this controller?
2F 13 0283 716 BEQL FOUND_IT ; BR if we can...
0285 717 $FAO_S CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
0285 718 OUTLEN = BUFFER_PTR,-
0285 719 OUTBUF = FAO_BUF,-
0285 720 P1 = #DEV$DC
02B6'CF 14 D0 029E 721 MOVL #SS$_BADPARAM,STATUS ; Set return status
000C'CF DF 02A3 722 PUSHAL BUFFER_PTR ; ...

```



```

0210'CF DF 0395 780 PUSHAL PROCESS_NAME ; Parameters on the stack
02 DD 0399 781 PUSHL #2 ; Push the argument count
00748333 8F DD 039B 782 PUSHL #UETPS_DENOSU
02 FO 03A1 783 INSV #STSSK_ERROR,-
00 03A3 784 #STSSV_SEVERITY,-
6E 03 03A4 785 #STSSS_SEVERITY,(SP) ; Set the severity code...
02B6'CF 6E DO 03A6 786 (SP),STATUS ; ...and save it as the exit status
04 DD 03AB 787 PUSHL #4 ; Push the partial arg count...
0692 31 03AD 788 BRW ERROR_EXIT ; ...and split this scene
03B0 789 60$:
03B0 790 $EXPREG_S PAGCNT = #PAGES,- ; Get a new node of demand zero memory
03B0 791 RETADR = NEW_NODE
0300'CF 0308'DF 5D 03C1 792 INSQTI @NEW_NODE,UNIT_LIST ; Put the new node in the unit list
56 0308'CF DO 03C8 793 MOVL NEW_NODE,R6 ; Save a copy of its address
08 A6 01 90 03CD 794 MOVW #1,DETUNTSB_TYPE(R6) ; Set the structure type
09F1 8F B0 03D1 795 MOVW #UETUNTSC INDSIZ+DEVDEP_SIZE,-
09 A6 03D5 796 UETUNT$W_SIZE(R6) ; Set the structure size
14 A6 0208'CF 90 03D7 797 MOVW DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
020C'DF 0208'CF 28 03DD 798 MOVW3 DEVDSC,@DEVVSC+4,- ; Save the device name
15 A6 03E4 799 UETUNT$T_FILSPC+1(R6)
02 88 03E6 800 BISB2 #UETUNT$M_TESTABLE,- ; Assume DR11-W testable
08 A6 03E8 801 UETUNT$B_FLAGS(R6)
03EA 802 $ASSIGN_S DEVNAM = DEVVSC,- ; Get the DR11-W for our exclusive use
03EA 803 CHAN = UETUNT$W_CHAN(R6)
02B6'CF 3E 50 E8 03FA 804 BLBS R0,70$ ; We're OK if we got the device
50 DO 03FD 805 MOVL R0,STATUS ; Save the failure code as exit status
02 FO 0402 806 INSV #STSSK_ERROR,-
00 0404 807 #STSSV_SEVERITY,-
02B6'CF 03 0405 808 #STSSS_SEVERITY,STATUS ; Set the severity code
02 8A 0409 809 BICB2 #UETUNT$M_TESTABLE,- ; We can't test DR11-W
08 A6 040B 810 UETUNT$B_FLAGS(R6)
040D 811 $FAO_S CTRSTR = DEVALLOC,- ; Otherwise bitch somewhat
040D 812 OUTLEN = BUFFER_PTR,-
040D 813 OUTBUF = FAO_BUF,-
040D 814 P1 = #DEVVSC
000C'CF DF 0426 815 PUSHAL BUFFER_PTR ; ...
01 DD 042A 816 PUSHL #1 ; ...
00741132 8F DD 042C 817 PUSHL #UETPS_TEXT!STSSK_ERROR ; ...
02B6'CF DD 0432 818 PUSHL STATUS ; ...
04 DD 0436 819 PUSHL #4 ; ...
0607 31 0438 820 BRW ERROR_EXIT
043B 821 70$:
01BC C6 34 28 043B 822 MOVW3 #4*<QIOS NARGS+1>,- ; Fill static $QIO_G args...
0529'CF 0C A6 3C 0443 823 MOVZWL UETUNT$W_CHAN(R6),- ; ...with those which...
01C4 C6 0446 825 XA_K_QIO+QIOS_CHAN(R6)
01A4 C6 7E 0449 826 MOVAQ XA_Q_IOSB(R6),- ; ...can't be filled at assembly
01CC C6 044D 827 XA_K_QIO+QIOS_IOSB(R6)
0242'CF 7D 0450 828 MOVW DIBBUF+DIB$B_DEVCLASS,- ; Save original characs
01B4 C6 0454 829 XA_Q_ORIGINAL(R6)
FE5A 31 0457 830 BRW FOUND_IT ; Do the next UCB

```

```

045A 832 :
045A 833 : Arrive here when we have the device configuration. In normal or loop forever
045A 834 : mode, set a timer far enough in the future such that we can do a reasonable
045A 835 : set of tests before the timer expires, but if our device gets hung, the
045A 836 : program won't waste too much time before noticing. Let one-shot mode be a
045A 837 : special case.
045A 838 :
045A 839 ALL_SET:
0300'CF D5 045A 840 TSTL UNIT_LIST ; Anything to test?
16 12 045E 841 BNEQ 10$ ; BR if yes
012B'CF DF 0460 842 PUSHAL NOUNIT_SELECTED ; Else set up the error message...
01 DD 0464 843 PUSHL #1 ; ...argument count...
00741132 8F DD 0466 844 PUSHL #UETPS_TEXT!STSSK_ERROR ; ...signal name...
03 DD 046C 845 PUSHL #3 ; ...and parameter count
02B6'CF 14 D0 046E 846 MOVL #SS$ BADPARAM,STATUS ; Set return status
05CC 31 0473 847 BRW ERROR_EXIT ; ...and give up, complaining
0476 848 10$:
0002'CF 04 A8 0476 849 BISW2 #SAFE TO UPDM,FLAG ; OK, safe to update UETINIDEV.DAT now
047B 850 ; MOVL DEVNAM_LEN,DEVVSC ; DEVVSC will describe device name
05 0002'CF 04 E1 047B 851 BBC #ONE_SROTV,FLAG,TIME_IT ; BR if in normal loop forever modes
0002'CF 02 A8 0481 852 BISW2 #TEST_OVERM,FLAG ; One-shot mode, stop after one shot!
0486 853 ; Because not all $QIOs have a timeout parameter, this test will always fall
0486 854 ; into TIME_IT to do a $SETIMR.
0486 855
0486 856 TIME_IT:
0486 857 $SETIMR_S DAYTIM = THREEMIN,- ; Set timer AST to 3 minutes
0486 858 ASTADR = TIME_OUT,-
0486 859 EFN = #EFN2

```

```

0499 861 .SBTTL Test the DR11-W
0499 862 RESTART:
0499 863 :
0499 864 : At this point the device designation is in location DEV_NAME pointed to by
0499 865 : descriptor DEV_DSC. The device is known to be supported by this test.
0499 866 :
56 0300'CF DE 0499 867 MOVAL UNIT_LIST,R6 ; R6 will point to the current node
049E 868 TEST_LOOP:
56 56 66 C0 049E 869 ADDL2 (R6),R6 ; Point to the next possible node
56 00000300'8F D1 04A1 870 CMPL #UNIT_LIST,R6 ; Back at the head of the queue?
03 12 04A8 871 BNEQ 10$ ; BR if not
01DE 31 04AA 872 BRW 90$ ; Exit test portion if we are
04AD 873 10$:
04AD 874 BBC #UETUNTSV_TESTABLE,- ; Skip this unit if can't test it
EC 0B A6 04AF 875 UETUNTSB_FLAGS(R6),TEST_LOOP
04B2 876
58 000000FA 8F D0 04B2 877 MOVL #<WRITE SIZE/4>,R8 ; Fill alternate words (byte count/4)...
59 09F2 C6 DE 04B9 878 MOVAL XA_K_BUF(R6),R9 ; ...of the read/write buffer...
02D4'CF 02D8'CF C0 04BE 879 20$:
89 02D4'CF 3C 04C5 880 ADDL2 RANDOM2,RANDOM1 ; ...with random...
F1 58 F5 04CA 881 MOVZWL RANDOM1,(R9)+ ; ...words...
04CD 882 SOBGTR R8,20$ ; ...until it's full
04CD 883 :
04CD 884 : Set up DR11-W quadword characteristics buffer for future IOS_SETMODEs.
04CD 885 : Copy device type, class and (bogus) buffer size and enabling for Unibus BDP
04CD 886 : when we want it.
04CD 887 :
01B4 C6 7D 04CD 888 MOVQ XA_Q_ORIGINAL(R6),- ; Class, type & transfer count
01AC C6 04D1 889 XA_Q_CHARAC(R6)
01B0 C6 02 88 04D4 890 BISB2 #XASM_LINK,- ; Add other characteristics...
04D6 891 XA_Q_CHARAC+4(R6) ; ... (but avoid XASM_DATAPATH)
04D9 892 :
04D9 893 : As described previously, the TABLE_GEN and LINE_GEN macros set up a set of
04D9 894 : parallel tables with parameters to be used in $QIOs to the DR11-W. Go
04D9 895 : through those tables. For each $QIO which transfers data, check that the
04D9 896 : data were passed correctly and clear the words into which data were written.
04D9 897 :
04D9 898 : Although the data structures of this test would permit multi-unit,
04D9 899 : asynchronous testing of DR11-W's, the way VMS treats DR11-W's (one unit per
04D9 900 : logical controller) means that we will test one one unit per test invocation.
04D9 901 : Since operations will turn out synchronous anyway, we may as well be honest
04D9 902 : and use the $QIOW system service to synchronize control.
04D9 903 :
57 D4 04D9 904 CLRL R7 ; Set up counter to go thru tables
5A 01BC C6 DE 04DB 905 MOVAL XA_K_QIO(R6),R10 ; Point to $QIO arglist for clarity
04E0 906 30$:
2C AA 0608'CF47 D0 04E0 907 MOVL P5_TABLE[R7],QIOS_P5(R10) ; Set up those $QIO_G args...
28 AA 05EC'CF47 D0 04E7 908 MOVL P4_TABLE[R7],QIOS_P4(R10) ; ...which must be done for...
24 AA 05D0'CF47 D0 04EE 909 MOVL P3_TABLE[R7],QIOS_P3(R10) ; ...each individual $QIO
20 AA 05B4'CF47 D0 04F5 910 MOVL P2_TABLE[R7],QIOS_P2(R10) ; ...
1C AA 0598'CF47 56 C1 04FC 911 ADDL3 R6,P1_TABLE[R7],QIOS_P1(R10) ; ...
18 AA 57 D0 0504 912 MOVL R7,QIOS_ASTPRM(R10) ; ...
14 AA 057C'CF47 D0 0508 913 MOVL ASTADR_TABLE[R7],QIOS_ASTADR(R10) ; ...
0C AA 0560'CF47 D0 050F 914 MOVL FUNC_TABLE[R7],QIOS_FUNC(R10) ; ...
0516 915 $QIOW_G (R10) ; Do one function of the DR11-W
10 A6 D6 051D 916 INCL UETUNT$L_ITER(R6) ; Count $QIOs done

```

51	01F0 C6	9A	0520	918	MOVZBL	XA_B_TSI(R6),R1	: Put index value into index register
01F1 C641	0310'CF	7D	0525	919	MOVQ	DIAG_BUF,XA_K_TSTAMP(R6)[R1]	: Store \$QIO starting time stamp
	51	D6	052D	920	INCL	R1	: Bump up index
01F1 C641	0318'CF	7D	052F	921	MOVQ	DIAG_BUF+8,XA_K_TSTAMP(R6)[R1]	: Store \$QIO ending time stamp
	51	D6	0537	922	INCL	R1	: Bump up index
	01F0 C6	90	0539	923	MOVB	R1,XA_B_TSI(R6)	: Keep index modulus 2**8
			053E	924			
	38 50	E8	053E	925	BLBS	R0,40\$: BR if the \$QIO itself worked
0788'CF	00	FB	0541	926	CALLS	#0,DEBUG_DUMP	: Print diagnostic info if it failed
02B6'CF	50	DD	0546	927	MOVL	R0,STATUS	: Save a record of what failed...
58	14 A6	DE	054B	928	MOVAL	UETUNT\$T_FILSPC(R6),R8	: ...
			054F	929	\$FAO_S	CTRSTR = QIO_ERRCR,-	: ... and consider it a fatal error
			054F	930		OUTLEN = BUFFER_PTR,-	
			054F	931		OUTBUF = FAO_BUF,-	
			054F	932		P1 = R8	
	02B6'CF	DD	0564	933	PUSHL	STATUS	: ...
	000C'CF	DF	0568	934	PUSHAL	BUFFER_PTR	: ...
	01	DD	056C	935	PUSHL	#1	: ...
00741132	8F	DD	056E	936	PUSHL	#UETP\$TEXT!STSS\$K_ERROR	: ...
	04	DD	0574	937	PUSHL	#4	: ...
	04C9	31	0576	938	BRW	ERROR_EXIT	: ...
			0579	939			
	71 01A4 C6	E8	0579	940	BLBS	XA_Q_IOSB(R6),50\$: BR if the function of the \$QIO worked
0788'CF	00	FB	057E	941	CALLS	#0,DEBUG_DUMP	: Type special info if in DUMP mode
58	14 A6	DE	0583	942	MOVAL	UETUNT\$T_FILSPC(R6),R8	
			0587	943	\$FAO_S	CTRSTR = QIO_FUNC_FAIL,-	: Report the problem otherwise
			0587	944		OUTLEN = BUFFER_PTR,-	
			0587	945		OUTBUF = FAO_BUF,-	
			0587	946		P1 = R8,-	
			0587	947		P2 = FUNC_TABLE[R7],-	
			0587	948		P3 = XA_Q_IOSB(R6),-	
			0587	949		P4 = XA_Q_IOSB+4(R6)	
7E	01A4 C6	3C	05A9	950	MOVZWL	XA_Q_IOSB(R6),-(SP)	: ...
	000C'CF	DF	05AE	951	PUSHAL	BUFFER_PTR	: ...
	000F0001	8F	05B2	952	PUSHL	#^XF0001	: ...
	00741130	8F	05B8	953	PUSHL	#UETP\$TEXT	: ...
	01A4 C6	F0	05BE	954	INSV	XA_Q_IOSB(R6),-	: Set the severity code
	00		05C2	955		#STSSV_SEVERITY,-	
	6E 03		05C3	956		#STSS\$SEVERITY,(SP)	
	02B2'CF	D6	05C5	957	INCL	ERROR_COUNT	: Keep a running count...
	02B2'CF	DD	05C9	958	PUSHL	ERROR_COUNT	: ...of the errors we've gotten
	0210'CF	DF	05CD	959	PUSHAL	PROCESS_NAME	
	00010002	8F	05D1	960	PUSHL	#^X10002	
	00748022	8F	05D7	961	PUSHL	#UETP\$ERBOXPROC!STSS\$K_ERROR	: Have the error stand out
00000000'GF	08	FB	05DD	962	CALLS	#8,G^LIB\$SIGNAL	: Bitch, bitch, bitch
	02	8A	05E4	963	BICB2	UETUNT\$M_TESTABLE,-	: Indicate that this unit is no good
	08 A6		05E6	964		UETUNT\$B_FLAGS(R6)	
0002'CF	0040 8F	A8	05E8	965	BISW2	#NO_MESSAGE,FLAG	: Indicate message already printed
			05EF	966			

	05B4'CF47	D5	05EF	968	TSTL	P2 TABLE[R7]	:	If zero length data transfer (words)...
		13	05F4	969	BEQL	80\$:	...skip the check and reset
58	05B4'CF47	FE	8F	78	05F6	970	:	Convert byte count to every-other-word cou
59	0598'CF47	56	C1	05FE	971	ADDL3	:	R6,P1_TABLE[R7],R9 ; Check that the data buffer...
				0605	972	60\$:		
	5B	69	89	AD	0605	973	:	XORW3 (R9)+,(R9),R11 ; ...got filled correctly
			65	13	0609	974	:	BEQL 70\$; BR if it did
	SF	0002'CF	06	E0	060B	975	:	BBS #NO MESSAGEV,FLAG,70\$; Avoid extra messages
		0788'CF	00	FB	0611	976	:	CALLS #0,DEBUG DUMP ; Type special info if in DUMP mode
5B	5B	10	00	EA	0616	977	:	FFS #0,#16,RT1,R11 ; Find the first bit of bad data
	5B	5B	FD	78	061B	978	:	ASHL #-3,R11,R11 ; Convert bit-position to byte-in-word
		7E	694B	9A	0620	979	:	MOVZBL (R9)[R11],-(SP) ; Save the bad byte,...
		7E	FE	9A	0624	980	:	MOVZBL -2(R9)[R11],-(SP) ; ...the corresponding good byte,...
		7E	59	C3	0629	981	:	SUBL3 R6,R9,-(SP) ; ...where the bad data was...
			6E	C0	062D	982	:	ADDL2 R11,(SP) ; ...
6E	0598'CF47			C2	0630	983	:	SUBL2 P1_TABLE[R7],(SP) ; ...in our buffer,...
				DF	0636	984	:	PUSHAL DEVDISC ; ...the device name,...
				DD	063A	985	:	PUSHL #^XF0004 ; ...the argument count,...
				DD	0640	986	:	PUSHL #UETP\$_DATADEVERR!STSSK_ERROR ; ...and the error type
				D6	0646	987	:	INCL ERROR_COUNT ; Keep a running count...
				DD	064A	988	:	PUSHL ERROR_COUNT ; ...of the errors we've gotten
				DF	064E	989	:	PUSHAL PROCESS_NAME ; ...
				DD	0652	990	:	PUSHL #^X10002 ; ...
				DD	0658	991	:	PUSHL #UETP\$_ERBOXPROC!STSSK_ERROR ; Have the error stand out
				FB	065E	992	:	CALLS #10,G^CIBSSIGNAL ; Bitch, bitch, bitch
				8A	0665	993	:	BICB2 #UETUNTSM_TESTABLE,- ; Indicate that this unit is no good
					0667	994	:	UETUNT\$B_FLAGS(R6) ; ...
0002'CF	0040	8F	A8	0669	995	70\$:	:	BISW2 #NO_MESSAGE,FLAG ; Indicate message already printed
					0670	996	:	
				B4	0670	997	:	CLRW (R9)+ ; Clear word to which DR11-W does DATO
				F5	0672	998	:	SOBGTR R8,60\$; Loop through the whole buffer
					0675	999	:	80\$:
0002'CF	0040	8F	AA	0675	1000	80\$:	:	BICW2 #NO_MESSAGE,FLAG ; New buffer rates another message
				E0	067C	1001	:	BBS #TEST_OVERV,FLAG,90\$; Exit quickly if the test is over
FE58	57	01	06	9D	0682	1002	:	ACBB #LINE_GEN_COUNT-1,#1,R7,30\$; Loop until all \$QIOs are done
					0688	1003	:	
				31	0688	1004	:	BRW TEST_LOOP ; Loop for next DR11-W
					068B	1005	:	
					0683	1006	:	90\$:
				30	068B	1007	:	BSBW RESET_DR11WS ; Do \$QIOs to reset original characs

```

56 0300'CF 00000300'8F 02DC'CF D6 068E 1009 INCL ITERATION ; Increment iteration count
C1 0692 1010 ADDL3 #UNIT_LIST,UNIT_LIST,R6 ; Go through UNIT_LIST to...
069C 1011 100$: BBS #UETUNTSV TESTABLE,- ; ...see if any testable units are left
E0 069C 1012 UETUNTSB_FLAGS(R6),110$ ; ...and BR if at least one is
069E 1013 (R6),R6 ; This one isn't. Is there another?
56 00000300'8F 56 66 C0 06A1 1014 ADDL2 #UNIT_LIST,R6
D1 06A4 1015 CMPL #UNIT_LIST,R6
EF 12 06AB 1016 BNEQ 100$ ; BR if there are others to try
0002'CF 02 A8 06AD 1017 BISW2 #TEST_OVERM,FLAG ; None testable so indicate test over
06B2 1018 110$: BITW #TEST_OVERM,FLAG ; Is the test over?
0002'CF 02 B3 06B2 1019 BNEQ SUC_EXIT ; BR if yes
03 12 06B7 1020 BRW RESTART ; Loop until the test is over
FDDD 31 06B9 1021 06BC 1022
06BC 1023 SUC_EXIT:
4C 0002'CF 04 E0 06BC 1024 BBS #ONE_SHOTV,FLAG,30$ ; Skip minimum I/O check if one-shot
56 0300'CF DE 06C2 1025 MOVAL UNIT_LIST,R6 ; Check the queue...
06C7 1026 20$: ADDL2 (R6),R6 ; ...to see if...
56 00000300'8F D1 06CA 1028 CMPL #UNIT_LIST,R6 ; ...each DR11-W...
3B 13 06D1 1029 BEQL 30$ ; ...tested...
10 A6 00000258 8F D1 06D3 1030 CMPL #MINIMUM,UETUNTSL_ITER(R6) ; ...has done the minimum I/O...
EA 15 06DB 1031 BLEQ 20$ ; ...to not be considered hung
57 14 A6 DE 06DD 1032 MOVAL UETUNT$T_FILSPC(R6),R7
06E1 1033 $FAO_S CTRSTR = SLOW DR11W,- ; Otherwise bitch somewhat
06E1 1034 OUTLEN = BUFFER_PTR,-
06E1 1035 OUTBUF = FAO_BUF,-
06E1 1036 P1 = R7,-
06E1 1037 P2 = UETUNTSL_ITER(R6)
000C'CF DF 06F9 1038 PUSHAL BUFFER_PTR ; ...
01 DD 06FD 1039 PUSHL #1 ; ...
00741130 8F DD 06FF 1040 PUSHL #UETPS_TEXT!STSSK_WARNING ; ...
00000000'GF 03 FB 0705 1041 CALLS #3,G^LIB$SIGNAL ; ...
B9 11 070C 1042 BRB 20$
070E 1043 30$: STRNLOG_S LOGNAM = MODE,-
070E 1044 RSLLEN = BUFFER_PTR,-
070E 1045 RSLBUF = FAO_BUF ; Get the run mode
0014'CF 20 8A 0727 1047 BICB2 #LC_BITM,BUFFER ; Convert to upper case
0014'CF 4C 8F 91 072C 1048 CMPB #^A7L/,BUFFER ; Is this a loop for ever?
40 12 0732 1049 BNEQ 10$ ; BR if not
0002'CF 02 AA 0734 1050 BICW2 #TEST_OVERM,FLAG ; Reset the termination flag
02E0'CF D6 0739 1051 INCL PASS ; Bump the pass count
073D 1052 $FAO_S CTRSTR = PASS MSG,-
073D 1053 OUTLEN = BUFFER_PTR,-
073D 1054 OUTBUF = FAO_BUF,-
073D 1055 P1 = PASS,-
073D 1056 P2 = ITERATION,-
073D 1057 P3 = #0 ; Make the end of pass message
000C'CF DF 075A 1058 PUSHAL BUFFER_PTR ; Push the string desc.
01 DD 075E 1059 PUSHL #1 ; Push arg count
00741133 8F DD 0760 1060 PUSHL #UETPS_TEXT!STSSK_INFO ; Push the signal name
00000000'GF 03 FB 0766 1061 CALLS #3,G^LIB$SIGNAL ; Print the end of pass message
02DC'CF D4 076D 1062 CLRL ITERATION ; Reset the iteration count
FD12 31 0771 1063 BRW TIME_IT ; Do the next pass
02B6'CF 10000001 8F D0 0774 1064 10$: MOVL #SS$_NORMAL!STSSM_INHIB_MSG,STATUS ; Set successful exit status
0774 1065

```

UETDR1W00
V04-000

- VAX/VMS UETP DR11-W EXERCISER L 6
Test the DR11-W

16-SEP-1984 01:25:57 VAX/VMS Macro V04-00
5-SEP-1984 04:25:15 [UETP.SRC]UETDR1W00.MAR;1

Page 26
(15)

JE
VO

077D 1066

\$EXIT_S STATUS

; Exit with the status

```

0788 1068      .SBTTL Routine to Dump Debugging Info
0788 1069      :++
0788 1070      : FUNCTIONAL DESCRIPTION:
0788 1071      : This routine will be called only if the logical name MODE translates to
0788 1072      : the string 'DUMP', and then only for those situations where a dump of
0788 1073      : info pertaining to the last $QIO would be useful.
0788 1074      :
0788 1075      : CALLING SEQUENCE:
0788 1076      : CALLS #0,DEBUG_DUMP
0788 1077      :
0788 1078      : INPUT PARAMETERS:
0788 1079      : NONE
0788 1080      :
0788 1081      : IMPLICIT INPUTS:
0788 1082      : R0 has the $QIO (in)completion status.
0788 1083      : R7 is an index into the $QIO tables.
0788 1084      : R10 points to the $QIO argument list. (redundant with R6)
0788 1085      : XA_Q_IOSB(R6) is the IOSB for the last $QIO.
0788 1086      :
0788 1087      : OUTPUT PARAMETERS:
0788 1088      : NONE
0788 1089      :
0788 1090      : IMPLICIT OUTPUTS:
0788 1091      : Message to SYSS$OUTPUT
0788 1092      : R0 and R1 returned unscathed!
0788 1093      :
0788 1094      : COMPLETION CODES:
0788 1095      : NONE
0788 1096      :
0788 1097      : SIDE EFFECTS:
0788 1098      : BUFFER and BUFFER_PTR modified after $FA0 call
0788 1099      :
0788 1100      :--
0788 1101      :
0788 1102      : DEBUG_DUMP:
0788 1103      : .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
078A 1104      :
01 0002'CF 05  E0 078A 1105      BBS #DUMP_MODEV,FLAG,10$ ; Dump mode info wanted?
04 0790 1106      RET ; Nope - call is a no-op
0791 1107      10$:
03  BB 0791 1108      PUSHR #^M<R0,R1> ; Save all registers!
0793 1109      $FA0_S CTRSTR = DEBUG MSG,- ; All the news that fits, we print
0793 1110      OUTLEN = BUFFER_PTR,-
0793 1111      OUTBUF = FA0_BUF,-
0793 1112      P1 = R0,-
0793 1113      P2 = QIOS_EFN(R10),-
0793 1114      P3 = QIOS_CHAN(R10),-
0793 1115      P4 = QIOS_FUNC(R10),-
0793 1116      P5 = QIOS_ASTADR(R10),-
0793 1117      P6 = QIOS_ASTPRM(R10),-
0793 1118      P7 = QIOS_P1(R10),-
0793 1119      P8 = QIOS_P2(R10),-
0793 1120      P9 = QIOS_P3(R10),-
0793 1121      P10 = QIOS_P4(R10),-
0793 1122      P11 = QIOS_P5(R10),-
0793 1123      P12 = XA_Q_IOSB(R6),-
0793 1124      P13 = XA_Q_IOSB+4(R6),-

```

			0793	1125		P14	=	XA_Q_CHARAC(R6),-	
			0793	1126		P15	=	XA_Q_CHARAC+4(R6)	
			07D6	1127				BUFFER_PTR	
			07DA	1128				#^X10001	
			07E0	1129				#UETP\$ TEXT!ST\$K_INFO	
			07E6	1130				#3,G^LIB\$SIGNAL	
			07ED	1131				XA_B_TSI(R6),R1	; Index of next time stamp quad pair
			07F2	1132				#2-RT	; Step back to current time stamp quad pair
			07F5	1133				#<19*4>,R2	; First available longword in DIAG_BUF
			07FC	1134				#2,R3	; Number of quad pairs for DIAG_MSG
			07FF	1135	20\$:				
			07FF	1136				#2,R1	; Step back one quad pair
			0802	1137				30\$; BR if not below first entry
			0804	1138				#TIME_STAMP_LEN-2,R1	; Point to top of quad pair list if it is
			080B	1139	30\$:				
			080B	1140				XA_K_TSTAMP(R6)[R1],DIAG_BUF(R2)	; Position one time stamp for \$FAO
			0813	1141				#8,R2	; Next available longword in DIAG_BUF
			0816	1142				R1	; Bump index
			0818	1143				XA_K_TSTAMP(R6)[R1],DIAG_BUF(R2)	; Position one time stamp for \$FAO
			0820	1144				#8,R2	; Next available longword in DIAG_BUF
			0823	1145				R1	; Bump index back where expected
			0825	1146				R3,20\$	
			0828	1147				CTRSTR = DIAG_MSG,-	
			0828	1148				OUTLEN = BUFFER_PTR,-	
			0828	1149				OUTBUF = FAO_BUF,-	
			0828	1150				PRMLST = DIAG_BUF	
			083F	1151				BUFFER_PTR	
			0843	1152				#^X10001	
			0849	1153				#UETP\$ TEXT!ST\$K_INFO	
			084F	1154				#3,G^LIB\$SIGNAL	
			0856	1155				#^M<R0,R1>	; Restore registers pristine
			0858	1156					


```

0889 1285 .SBTTL System Service Exception Handler
0889 1286 :++
0889 1287 : FUNCTIONAL DESCRIPTION:
0889 1288 : This routine is executed if a software or hardware exception occurs or
0889 1289 : if a LIB$SIGNAL system service is used to output a message.
0889 1290 :
0889 1291 : CALLING SEQUENCE:
0889 1292 : Entered via an exception from the system
0889 1293 :
0889 1294 : INPUT PARAMETERS:
0889 1295 : ERROR_COUNT = previous cumulative error count
0889 1296 :
0889 1297 : AP ---->
0889 1298 :
0889 1299 :
0889 1300 :
0889 1301 :
0889 1302 :
0889 1303 :
0889 1304 :
0889 1305 :
0889 1306 :
0889 1307 :
0889 1308 :
0889 1309 :
0889 1310 :
0889 1311 :
0889 1312 :
0889 1313 :
0889 1314 :
0889 1315 :
0889 1316 :
0889 1317 :
0889 1318 :
0889 1319 :
0889 1320 :
0889 1321 :
0889 1322 :
0889 1323 :
0889 1324 : IMPLICIT INPUTS:
0889 1325 : NONE
0889 1326 :
0889 1327 : OUTPUT PARAMETERS:
0889 1328 : NONE
0889 1329 :
0889 1330 : IMPLICIT OUTPUTS:
0889 1331 : NONE
0889 1332 :
0889 1333 : COMPLETION CODES:
0889 1334 : $$$_NORMAL if it's a UETP condition or RMS error.
0889 1335 : Error status from exception, otherwise.
0889 1336 :
0889 1337 : SIDE EFFECTS:
0889 1338 : May branch to ERROR_EXIT.
0889 1339 : May print a message.
0889 1340 :--
0889 1341 :

```

2	
SIGNL ARY PNT	
MECH ARY PNT	
4	
ESTABLISH FP	
DEPTH	Mechanism Array
R0	
R1	
N	
CONDITION NAME	
N-3 ADDITIONAL LONG WORD ARGS	Signal Array
PC	
PSL	

```

OFFC 08B9 1342 SSERROR:
      08B9 1343 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
      082B 1344
      08BB 1345 $SETAST_S ENBFLG = #0 ; Disable AST delivery
50 01 DD 08C4 1346 PUSHL #1 ; Assume ASTs were enabled
    09 D1 08C6 1347 CMPL S^#SS$_WASSET,R0 ; Were ASTs enabled?
    02 13 08C9 1348 BEQL 10$ ; BR if they were
    6E D4 08CB 1349 CLRL (SP) ; Set ASTs to remain disabled
      08CD 1350 10$:
      08CD 1351 $SETSFM_S ENBFLG = #0 ; Disable SS failure mode
50 01 DD 08D6 1352 PUSHL #1 ; Assume SS failure mode was enabled
    09 D1 08D8 1353 CMPL S^#SS$_WASSET,R0 ; Was SS failure mode enabled?
    02 13 08DB 1354 BEQL 20$ ; BR if it was
    6E D4 08DD 1355 CLRL (SP) ; Set SS failure mode to remain off
      08DF 1356 20$:
56 04 AC D0 08DF 1357 MOVL CHF$_SIGARGLST(AP),R6 ; Get the signal array pointer
59 04 A6 7D 08E3 1358 MOVQ CHF$_SIG_NAME(R6),R9 ; Get NAME in R9 and ARG1 in R10
    10 ED 08E7 1359 CMPZV #STSS$_FAC_NO,- ; Is this a message from LIB$SIGNAL?
    0C 08E9 1360 #STSS$_FAC_NO,-
00000074 8F 59 08EA 1361 R9,#UETP$_FACILITY
    14 12 08F0 1362 BNEQ 30$ ; BR if this is not a UETP exception
    66 02 C2 08F2 1363 SUBL2 #2,CHF$_SIG_ARGS(R6) ; Drop the PC and PSL
    21 11 08F5 1364 $PUTMSG_S MSGVEC = CHF$_SIG_ARGS(R6) ; Print the message
    0904 1365 BRB 40$ ; Restore ASTs and SS fail mode
59 0000045C 8F D1 0906 1366 30$:
    32 12 090D 1367 CMPL #SS$_SSFAIL,R9 ; RMS failures are SysSvc failures
    10 ED 090F 1368 BNEQ 50$ ; BR if this can't be an RMS failure
    0C 0911 1369 CMPZV #STSS$_FAC_NO,- ; Is it an RMS failure?
    01 5A 0912 1370 R10,#RMS$_FACILITY
5A F0000000 8F CA 0914 1371 BNEQ 50$ ; BR if not
    08 A6 04 39 0916 1372 BICL2 #^XF0000000,R10 ; Strip control bits from status code
    14 091D 1373 MATCHC #4,CHF$_SIG_ARG1(R6),- ; Is it an RMS failure for which...
    004D'CF 14 0921 1374 #NRAT_LENGTH,-
    1A 13 0922 1375 NO_RMS_AST_TABLE ; ...no AST can be delivered?
    0927 1376 BEQL 50$ ; BR if so - must give error here
    01 BA 0927 1377 40$:
    0929 1378 POPR #^M<R0> ; Restore SS failure mode...
    01 BA 0929 1379 $SETSFM_S ENBFLG = R0 ; ...
    0932 1380 POPR #^M<R0> ; Restore AST enable...
    50 01 D0 0934 1381 $SETAST_S ENBFLG = R0 ; ...
    04 093D 1382 MOVL S^#SS$_NORMAL,R0 ; Supply a standard status for exit
    0941 1383 RET ; Resume processing (or goto RMS_ERROR)
    02B6'CF 59 D0 0941 1384 50$:
59 0000045C 58 D4 0946 1385 MOVL R9,STATUS ; Save the status
    8F D1 0948 1386 CLRL R8 ; Assume for now it's not SS failure
    38 12 094F 1387 CMPL #SS$_SSFAIL,R9 ; But is it a System Service failure?
    0951 1388 BNEQ 70$ ; BR if not - no special case message
    0951 1389 $GETMSG_S MSGID = R10,- ; Get SS failure code associated text
    0951 1390 MSGLEN = BUFFER_PTR,-
    0951 1391 BUFADR = FAO_BUF,-
    0951 1392 FLAGS = #14,-
    0951 1393 OUTADR = MSG_BLOCK
    02E5'CF 95 0968 1394 TSTB MSG_BLOCK+1 ; Get FAO arg count for SS failure code
    16 13 096C 1395 BEQL 60$ ; Don't use $GETMSG if no $FAO args...
    000C'CF DF 096E 1396 PUSHAL BUFFER_PTR ; ...else build up...
    01 DD 0972 1397 PUSHL #1 ; ...a message describing...
    0972 1398

```



```

OAOB 1478 .SBTTL CTRL/C Handler
OAOB 1479 :++
OAOB 1480 : FUNCTIONAL DESCRIPTION:
OAOB 1481 : This routine handles CTRL/C AST's
OAOB 1482 :
OAOB 1483 : CALLING SEQUENCE:
OAOB 1484 : Called via AST
OAOB 1485 :
OAOB 1486 : INPUT PARAMETERS:
OAOB 1487 : NONE
OAOB 1488 :
OAOB 1489 : IMPLICIT INPUTS:
OAOB 1490 : NONE
OAOB 1491 :
OAOB 1492 : OUTPUT PARAMETERS:
OAOB 1493 : NONE
OAOB 1494 :
OAOB 1495 : IMPLICIT OUTPUTS:
OAOB 1496 : NONE
OAOB 1497 :
OAOB 1498 : COMPLETION CODES:
OAOB 1499 : NONE
OAOB 1500 :
OAOB 1501 : SIDE EFFECTS:
OAOB 1502 : NONE
OAOB 1503 :
OAOB 1504 :--
OAOB 1505 :
OAOB 1506 CCASTHAND:
OFFC OAOB 1507 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OAOB 1508
00A3'CF DF OAOB 1509 PUSHAL CNTRLCMSG ; Set message pointer
01 DD OAOB 1510 PUSHL #1 ; Set arg count
00741130 8F DD OAOB 1511 PUSHL #UETPS_TEXT!STSSK_WARNING ; Set signal name
00 DD OAOB 1512 PUSHL #0 ; Indicate an abnormal termination
0210'CF DF OAOB 1513 PUSHAL PROCESS_NAME ; ...
02 DD OAOB 1514 PUSHL #2 ; ...
007410E0 8F DD OAOB 1515 PUSHL #UETPS_ABENDD!STSSK_WARNING ; ...
00000000'GF 07 FB OAOB 1516 CALLS #7,G^LIBSSIGNAL ; Output the message
OAOB 1517 MOVL #<STSSM INHIB_MSG!- ; Set the exit status
OAOB 1518 SSS CONTROLC=-
OAOB 1519 STSSK_SUCCESS+STSSK_WARNING>,-
02B6'CF 10000650 8F OAOB 1520 STATUS
OAOB 1521 $EXIT_S STATUS ; Terminate program cleanly

```

```

0A42 1523      .SBTTL Error Exit
0A42 1524      :++
0A42 1525      : FUNCTIONAL DESCRIPTION:
0A42 1526      :   This routine prints an error message and exits.
0A42 1527      :
0A42 1528      : CALLING SEQUENCE:
0A42 1529      :   MOVx error status value,STATUS
0A42 1530      :   PUSHx error specific information on the stack
0A42 1531      :   PUSHL current argument count
0A42 1532      :   BRW ERROR_EXIT
0A42 1533      :
0A42 1534      : INPUT PARAMETERS:
0A42 1535      :   Arguments to LIB$SIGNAL, as above
0A42 1536      :
0A42 1537      : IMPLICIT INPUTS:
0A42 1538      :   NONE
0A42 1539      :
0A42 1540      : OUTPUT PARAMETERS:
0A42 1541      :   Message to SYSS$OUTPUT and SYSS$ERROR
0A42 1542      :
0A42 1543      : IMPLICIT OUTPUTS:
0A42 1544      :   Program exit
0A42 1545      :
0A42 1546      : COMPLETION CODES:
0A42 1547      :   NONE
0A42 1548      :
0A42 1549      : SIDE EFFECTS:
0A42 1550      :   NONE
0A42 1551      :
0A42 1552      :--
0A42 1553      :
0A42 1554      ERROR_EXIT:
0A42 1555      :
0A42 1556      $SETAST_S ENBFLG = #0 ; ASTs can play havoc with messages
15 0002'CF 03 E0 0A4B 1557 BBS #BEGIN_MSGV,FLAG,10$ ; BR if 'begin' msg already printed
      7E D4 0A51 1558 CLRL -(SP) ; Set the time stamp flag
      000F'CF DF 0A53 1559 PUSHAL TEST_NAME ; Set the test name
      02 DD 0A57 1560 PUSHL #2 ; Push the argument count
00741039 8F DD 0A59 1561 PUSHL #UETP$_BEGIN!STSSK_SUCCESS ; Set the message code
00000000'GF 04 FB 0A5F 1562 CALLS #4,G^LIB$SIGNAL ; Print the startup message
0A66 1563 10$:
02F8'CF 08 8E C1 0A66 1564 ADDL3 (SP)+,#8,ARG_COUNT ; Get total # args, pop partial count
      02B2'CF D6 0A6C 1565 INCL ERROR_COUNT ; Keep running error count
      00 DD 0A70 1566 PUSHL #0 ; Push the time parameter
      0210'CF DF 0A72 1567 PUSHAL PROCESS_NAME ; Push test name...
000F0002 8F DD 0A76 1568 PUSHL #^XF0002 ; ...arg count...
007410E2 8F DD 0A7C 1569 PUSHL #UETP$_ABFNDD!STSSK_ERROR ; ...and signal name
      02B2'CF DD 0A82 1570 PUSHL ERROR_COUNT ; Finish off arg list...
      0210'CF DF 0A86 1571 PUSHAL PROCESS_NAME ; ...
00010002 8F DD 0A8A 1572 PUSHL #^X10002 ; ...
00748022 8F DD 0A90 1573 PUSHL #UETP$_ERBOXPROC!STSSK_ERROR ; ...for error box message
00000000'GF 02F8'CF FB 0A96 1574 CALLS ARG_COUNT,G^LIB$SIGNAL ; Truly bitch
      0A9F 1575
      02B6'CF D5 0A9F 1576 TSTL STATUS ; Did we exit with an error code?
      09 12 0AA3 1577 BNEQ 20$ ; BR if we did
007410E2 8F D0 0AA5 1578 MCVL #UETP$_ABENDD!STSSK_ERROR,- ; Supply a generic one otherwise
      02B6'CF 0AAB 1579 STATUS

```



```

5B 0300'CF 0000300'8F C1 0B08 1641 ADDL3 #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
59 D4 0B12 1642 CLRRL R9 ; Init a counter
01 E1 0B14 1643 UNIT_LOOP: BBC #UETUNT$V TESTABLE,- ; BR if this unit is not testable
02 0B AB 59 D6 0B16 1645 UETUNT$B_FLAGS(R11),10$ ;
59 D6 0B19 1646 INCL R9 ; Count testable units
5B 6B C0 0B1B 1647 10$: ADDL2 (R11),R11 ; Next unit block
00000300'8F 5B D1 0B1E 1649 CMPL R11,#UNIT_LIST ; Are we full circle in the list?
5B D1 0B1E 1649 CMPL R11,#UNIT_LIST ; BR if not
ED 12 0B25 1650 BNEQ UNIF_LOOP- ; BR if not
59 D5 0B27 1651 TSTL R9 ; Any testable units?
12 12 0B29 1652 BNEQ 20$ ; BR if yes...
0018'CF 4E 8F 90 0B2B 1653 MOVB #^A/N/,BUFFER+4 ; ...else disable the DDB record...
3C 50 E9 0B3A 1655 $UPDATE RAB = (R10) ; ...here
0B3D 1656 20$: BLBC R0,UPDATE_FAILED ; If error then forget it
5B 6B C0 0B3D 1657 ADDL2 (R11),R11 ; Next unit block
00000300'8F 5B D1 0B40 1658 CMPL R11,#UNIT_LIST ; Are we full circle in the list?
4E 13 0B47 1659 BEQL END_UPDATE ; BR if yes
24 50 E9 0B49 1660 $GET RAB=(R10) ; Get a record
0014'CF 20 8A 0B52 1661 BLBC R0,UPDATE_FAILED ; If error then forget it
0014'CF 55 8F 91 0B5A 1662 BICB2 #LC_BITM,BUFFER ; Convert to uppercase
35 12 0B60 1664 BNEQ END_UPDATE ; Is it a UCB record?
01 E0 0B62 1665 BBS #UETUNT$V TESTABLE,- ; BR if not
0018'CF D6 0B AB 0B64 1666 UETUNT$B_FLAGS(R11),20$ ; BR if this unit is testable...
4E 8F 90 0B67 1667 MOVB #^A/N/,BUFFER+4 ; ...else disable the UCB record...
0B6D 1668 $UPDATE RAB = (R10) ; ...here
C4 50 E8 0B76 1669 BLBS R0,20$ ; Look at the next record if no error
0C AA DD 0B79 1671 UPDATE_FAILED: PUSHL RAB$L_STV(R10) ; Do a simple message...
50 DD 0B7C 1672 PUSHL R0 ; ...to tell of the failure
01BB'CF DF 0B7E 1673 PUSHAL INIDEV_UPDERR
01 DD 0B82 1674 PUSHL #1
00 EF 0B84 1675 EXTZV #STSSV_SEVERITY,- ; Copy the severity from RMS status...
7E 50 03 0B86 1676 #STSS$SEVERITY,R0,-(SP) ;
6E 00741130 8F C8 0B89 1677 BISL2 #UETP$TEXT,(SP) ; ...to our message
00000000'GF 05 FB 0B90 1678 CALLS #5,G^LIB$SIGNAL
0B97 1679 END_UPDATE:
03 0002'CF 01 E0 0B97 1680 BBS #TEST_OVERV,FLAG,10$ ; Did the test complete normally?
FCBC 30 0B9D 1681 BSBW RESET_DR11WS ; Reset original DR11-W characs if not
00 DD 0BA0 1682 10$: PUSHL #0 ; Set the time flag
000F'CF DF 0BA2 1684 PUSHAL TEST_NAME ; Push the test name
02 DD 0BA6 1685 PUSHL #2 ; Push arg count
00 EF 0BA8 1686 EXTZV #STSSV_SEVERITY,- ; Push the proper exit severity...
03 0BA A 1687 #STSS$SEVERITY,-
7E 02B6'CF 03 0BAA 1688 STATUS,-(SP)
6E 00741080 8F C8 0BAF 1689 BISL2 #UETP$ENDEDD,(SP) ; ...and use it in our message code
04 DD 0BB6 1690 PUSHL #4
51 5E D0 0BB8 1691 MOVL SP,R1
0B8B 1692 $PUTMSG_S MSGVEC = (R1) ; Output the message
0BCA 1693 $SETPRN_S PRCNAM = ACNT_NAME ; Reset the process name
04 0BD5 1694 RET ; That's all folks!
0BD6 1695
0BD6 1696 .END UETDR1W00

```

UE
PR

EN
VAI
LAI
FUI
CCI

\$\$TAB	=	00000508	R	04	FABSB_BID	=	00000000		
\$\$TABEND	=	00000558	R	04	FABSB_FNS	=	00000034		
\$\$TMP	=	00000002			FABSC_BID	=	00000003		
\$\$TMP1	=	00000001			FABSC_BLN	=	00000050		
\$\$TMP2	=	0000006A			FABSC_SEQ	=	00000000		
\$\$TMPX	=	00000016	R	05	FABSC_VAR	=	00000002		
\$\$TMPX1	=	0000000D			FABSL_ALQ	=	00000010		
\$\$ARGS	=	0000000C			FABSL_DEV	=	00000040		
\$\$T1	=	00000000			FABSL_FNA	=	0000002C		
\$\$T2	=	00000006			FABSL_FOP	=	00000004		
ACNT_NAME		00000000	R	03	FABSL_STS	=	00000008		
ALL_SET		0000045A	R	06	FABSL_STV	=	0000000C		
ARG_COUNT		000002F8	R	04	FABSV_CHAN_MODE	=	00000002		
ASTADR_TABLE		0000057C	R	03	FABSV_CR	=	00000001		
BEGIN_MSGM	=	00000008			FABSV_FILE_MODE	=	00000004		
BEGIN_MSGV	=	00000003			FABSV_GET	=	00000001		
BUFFER		00000014	R	04	FABSV_LNM_MODE	=	00000000		
BUFFER_PTR		0000000C	R	04	FABSV_PUT	=	00000000		
CCASTHAND		00000A0B	R	06	FABSV_UFO	=	00000011		
CHFSL_SIGARGLST	=	00000004			FABSV_UPD	=	00000003		
CHFSL_SIG_ARG1	=	00000008			FABSV_UPI	=	00000006		
CHFSL_SIG_ARGS	=	00000000			FABSW_GBC	=	00000048		
CHFSL_SIG_NAME	=	00000004			FAO_BUF		00000004	R	04
CNTRLMSG		000000A3	R	03	FILE		000001F5	R	03
COMMON		000009D2	R	06	FIND_IT		0000021C	R	06
CONTROLLER		00000031	R	03	FLAG		00000002	R	04
CONT_DESC		000001ED	R	03	FOUND_IT		000002B4	R	06
CS1		00000082	R	03	FUNC_TABLE		00000560	R	03
CS3		00000094	R	03	ILLEGAL_REC		00000151	P	03
DDB_RFA		00000500	R	04	INADDRESS		000002C2	R	04
DEAD_CTRLNAME		000000E4	R	03	INIDEV_UPDERR		000001B8	R	03
DEBUG_DUMP		00000788	R	06	INI_FAB		0000046C	R	04
DEBUG_MSG		000002EF	R	03	INI_RAB		0000048C	R	04
DEVSV_TRM	=	00000002			INPUT_ITMLST		00000072	R	03
DEVALCOC		00000249	R	03	IOSM_CTRLCAST	=	00000100		
DEVDEP_SIZE	=	0000084D			IOSM_CYCLE	=	00001000		
DEVDS		00000208	R	04	IOSM_DIAGNOSTIC	=	00000100		
DEVNAM_LEN		000002D2	R	04	IOSM_RESE1	=	00000800		
DEV_NAME		00000227	R	04	IOSM_SETFNCT	=	00000200		
DIAG_BUF		00000310	R	04	IOSM_TIMED	=	00000080		
DIAG_MSG		00000444	R	03	IOS_SETCHAR	=	0000001A		
DIB		00000236	R	04	IOS_SETMODE	=	00000023		
DIBSB_DEVCLASS	=	00000004			IOS_WRITEPBLK	=	0000000B		
DIBSB_DEVTYPE	=	00000005			IOAST		00000859	R	06
DIBSK_LENGTH	=	00000074			ITERATION		000002DC	R	04
DIBBUF		0000023E	R	04	LC_BITM	=	00000020		
DUMMY_QIO		00000529	R	03	LIBSSIGNAL		*****	X	06
DUMP_MODEM	=	00000020			LINE_GEN_COUNT	=	00000007		
DUMP_MODEV	=	00000005			MAX_DEV_DESIG	=	0000000A		
DVIS_DEVNAM	=	00000020			MAX_PROC_NAME	=	0000000F		
DWT_SIZE	=	000003E8			MAX_UNIT_DESIG	=	00000005		
EFN2	=	00000004			MINIMUM	=	00000258		
END_UPDATE		00000B97	R	06	MODE		00000041	R	03
ERROR_COUNT		000002B2	R	04	MSG_BLOCK		000002E4	R	04
ERROR_EXIT		00000A42	R	06	NAME_LEN	=	0000000F		
EXIT_DESC		000002E8	R	04	NEW_NODE		00000308	R	04
EXIT_HANDLER		00000AC2	R	06	NOUNIT_SELECTED		0000012B	R	03

NO_CTRLNAME	= 000000C4	R	03	RESET DR11WS	0000085C	R	06
NO_MESSAGE	= 00000040			RESTART	00000499	R	06
NO_MESSAGEV	= 00000006			RMS\$BLN	*****	X	03
NO_RMS_AST_TABLE	= 00000040	R	03	RMS\$BUSY	*****	X	03
NRAT_LENGTH	= 00000014			RMS\$CDA	*****	X	03
ONE_SHOTM	= 00000010			RMS\$FAB	*****	X	03
ONE_SHOTV	= 00000004			RMS\$FACILITY	= 00000001		
OUTADDRESS	000002CA	R	04	RMS\$RAB	*****	X	03
P1_TABLE	00000598	R	03	RMS_ERROR	0000099C	R	06
P2_TABLE	000005B4	R	03	RMS_ERR_STRING	0000020F	R	03
P3_TABLE	000005D0	R	03	SAFE_TO_UPDM	= 00000004		
P4_TABLE	000005EC	R	03	SAFE_TO_UPDV	= 00000002		
P5_TABLE	00000608	R	03	SEC\$M_EXPREG	*****	X	06
PAGES	= 00000007			SEC\$M_GBL	*****	X	06
PASS	000002E0	R	04	SHR\$ABENDD	= 000010E0		
PASS_MSG	00000185	R	03	SHR\$BEGIN	= 00001038		
PMTSTZ	= 00000019			SHR\$ENDEDD	= 00001080		
PROCESS_NAME	00000210	R	04	SHR\$OPENIN	= 00001098		
PROCESS_NAME_FREE	= 0000000B			SHR\$TEXT	= 00001130		
PROC_CONT_NAME	0000008B	R	06	SLOW-DR11W	00000272	R	03
PROMPT	00000230	R	03	SS\$BADPARAM	= 00000014		
QIOS_ASTADR	= 00000014			SS\$CONTROL	= 00000651		
QIOS_ASTPRM	= 00000018			SS\$NORMAL	= 00000001		
QIOS_CHAN	= 00000008			SS\$NOSUCHSEC	= 00000978		
QIOS_EFN	= 00000004			SS\$SSFAIL	= 0000045C		
QIOS_FUNC	= 0000000C			SS\$WASSET	= 00000009		
QIOS_IOSB	= 00000010			SSEERROR	000008B9	R	06
QIOS_NARGS	= 0000000C			SS_SYNCH_EFN	= 00000003		
QIOS_P1	= 0000001C			STATUS	00000286	R	04
QIOS_P2	= 00000020			STR\$UPCASE	*****	X	06
QIOS_P3	= 00000024			STSSK_ERROR	= 00000002		
QIOS_P4	= 00000028			STSSK_INFO	= 00000003		
QIOS_P5	= 0000002C			STSSK_SUCCESS	= 00000001		
QIOS_P6	= 00000030			STSSK_WARNING	= 00000000		
QIO_EFN	= 00000005			STSSM-INHIB MSG	= 10000000		
QIO_ERROR	00000297	R	03	STSS\$FAC NO	= 0000000C		
QIO_FUNC_FAIL	000002B8	R	03	STSS\$SEVERITY	= 00000003		
QUAD STATUS	000002BA	R	04	STSSV_FAC NO	= 00000010		
RAB\$B_PSZ	= 00000034			STSSV_SEVERITY	= 00000000		
RAB\$B_RAC	= 0000001E			SUC_EXIT	000006BC	R	06
RAB\$C_BID	= 00000001			SUPDEV_GBLSEC	00000020	R	03
RAB\$C_BLN	= 00000044			SUP_FAB	00000508	R	04
RAB\$C_RFA	= 00000002			SYSS\$ASSIGN	*****	GX	06
RAB\$C_SEQ	= 00000000			SYSS\$CONNECT	*****	GX	06
RAB\$L_CIX	= 00000018			SYSS\$CRMPSC	*****	GX	06
RAB\$L_FAB	= 0000003C			SYSS\$DCLEXH	*****	GX	06
RAB\$L_PBF	= 00000030			SYSS\$EXIT	*****	GX	06
RAB\$L_ROP	= 00000004			SYSS\$EXPREG	*****	GX	06
RAB\$L_STS	= 00000008			SYSS\$FAO	*****	X	06
RAB\$L_STV	= 0000000C			SYSS\$FAOL	*****	GX	06
RAB\$V_PMT	= 0000001E			SYSS\$GET	*****	GX	06
RAB\$W_RFA	= 00000010			SYSS\$GETDEV	*****	GX	06
RAB\$W_RSZ	= 00000022			SYSS\$GETDVI	*****	GX	06
RANDOM1	000002D4	R	04	SYSS\$GETMSG	*****	GX	06
RANDOM2	000002E8	R	04	SYSS\$INPUT	00000061	R	03
RECORD	00000201	R	03	SYSS\$MGBLSC	*****	GX	06
REC_SIZE	= 00000028			SYSS\$OPEN	*****	GX	06

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DEVDEP_STR_DEF	00000DDA (3546.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RODATA	00000624 (1572.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000558 (1368.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
\$RMSNAM	00000023 (35.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DR11W	00000BD6 (3030.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.07	00:00:00.63
Command processing	131	00:00:00.77	00:00:05.53
Pass 1	723	00:00:24.42	00:00:55.59
Symbol table sort	0	00:00:02.57	00:00:05.04
Pass 2	412	00:00:06.00	00:00:11.24
Symbol table output	35	00:00:00.27	00:00:00.74
Psect synopsis output	5	00:00:00.05	00:00:00.08
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1342	00:00:34.16	00:01:18.85

The working set limit was 2000 pages.
135028 bytes (264 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1724 non-local and 49 local symbols.
1696 source lines were read in Pass 1, producing 38 object records in Pass 2.
67 pages of virtual memory were used to define 56 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	2
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	49
TOTALS (all libraries)	51

2074 GETS were required to define 51 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:UETDR1W00/OBJ=OBJ\$:UETDR1W00 MSRCS:UETDR1W00/UPDATE=(ENHS:UETDR1W00)+EXECMLS/LIB+LIBS:UETP/LIB

0411 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 100 small terminal windows, each showing a different screen of a VAX/VMS operating system. The screens contain various system messages, command prompts, and data listings. Some screens are more legible than others, showing text like 'LETFORT03 LIS', 'LETDR1400 LIS', and 'LETNETS00 LIS'. The overall appearance is that of a multi-processor system's console output.