

EY-0019E-TP-0101

VAX/VMS System Programmer

Tests and Exercises

Prepared by Educational Services
of
Digital Equipment Corporation

Copyright © 1982, Digital Equipment Corporation.
All Rights Reserved.

The reproduction of this material, in part or whole, is strictly prohibited. For copy information, contact the Educational Services Department, Digital Equipment Corporation, Bedford, Massachusetts 01730.

Printed in U.S.A.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may not be used or copied except in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

The following are trademarks of Digital Equipment Corporation, Maynard, Massachusetts:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECSYSTEM-20	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	RSTS
UNIBUS	VAX	RSX
	VMS	IAS

CONTENTS

1	WRITING PRIVILEGED CODE	1
2	WRITING USER-WRITTEN SYSTEM SERVICES.	28
3	WRITING COMMAND LANGUAGE INTERPRETERS	58
4	WRITING SYMBIONTS	79
5	WRITING AN AME.	86

WRITING PRIVILEGED CODE

TEST

Before starting the lab assignment, copy the MAKETQE and STOPTQE files from a directory specified by your instructor. Change all references in your copy of the files from EXESGL_SITESPEC to CTL\$GL_SITESPEC. In doing so, you can run your copy of the programs without interfering with other students running their copies of the programs.

1. Use the MAKETQE.COM command procedure discussed in the module to assemble and link all the programs needed to run the MAKETQE and STOPTQE programs. That command procedure also defines DELTA as the debugger to use, and incorporates it into the programs. It also produces a file (TQEDEF.OBJ) defining all the symbolic offsets for TQE symbols.

Listing of TQEDEF.MAR File

```
          :                               TQEDEF.MAR
          :
          :   This program defines the offsets in a TQE block.
          :
          :   STQEDEF GLOBAL
          :   .END
```

1. Run the MAKETQE program without the control of the DELTA debugger.
2. Use SDA to analyze the current system.
3. Display the contents of the site-specific longword, CTL\$GL_SITESPEC. It should contain the address of the TQE built by the MAKETQE program.
4. Make the TQE offsets known to SDA by reading in the TQEDEF.OBJ file created earlier. Note that the file SYSS\$SYSTEM:SYSDEF.STB can be read to define many of the system data structures. Similar files exist for RMS and DCL symbols (try a directory of SYSS\$SYSTEM:*.STB).
5. Format the block pointed to by CTL\$GL_SITESPEC as a TQE. You should notice that the RQTYPE field contains a 5 (for a REPEAT, SYSTEM SUBROUTINE request - TQESC SSREPT). The FPC field should contain a system address (pointing to the code in the block from nonpaged pool).

WRITING PRIVILEGED CODE

6. Use the FPC field displayed above to display the contents of the code block, and location being updated. Display a range of address starting at the FPC value (minus 12, hex C) for 20 hex bytes. You should see the INCL instruction (D6 FF ...), which was copied into the block by the MAKETQE program (check the MAKETQE listing file in the module).
7. Repeat the above instruction several times. You should find the value in the first location being incremented each time you repeat the command.
8. Try to find the TQE block on the timer queue. The timer queue listhead is EXESGL_TQFL. Note, however, that you may not be able to find the TQE, since you are examining a running system, and as you traverse forward links, the data structures may be re-used, and no longer be valid TQEs. You will know when you reach the end of the TQE queue when you find a forward link pointing to EXESGL_TQFL+008.
9. Exit from SDA.
10. Run the STOPTQE program without the control of the DELTA debugger.
11. Invoke SDA again for the currently running system.
12. Display the contents of the CTL\$GL_SITESPEC location. That location should now contain a 0.
13. Redisplay the contents of the data block several times. You should find that the first location is not changing. The data block should have been returned to nonpaged pool, so the first two longwords may be being used as forward and backward links. Depending on how busy the system is, that data block might already have been reused, or it may still contain the INCL instruction.
14. Again read the file containing TQE symbols.
15. Format the pool space previously containing the TQE as a TQE. Again, depending on system activity, this block may or may not have been reused, and the contents may or may not have changed.
16. Try to find the old TQE block on the timer queue again. This time, you shouldn't find it there.

WRITING PRIVILEGED CODE

Solution to Problem 1 (Page 1 of 3)

```

$ RUN/NODEBUG MAKETQE 1
$
$ ANALYZE/SYSTEM 2
VAX/VMS System analyzer

SDA> SHOW SYMBOL CTL$GL_SITESPEC 3
CTL$GL_SITESPEC = 7FFEFF60 : 8010B720
SDA>
SDA> READ TQDEF.OBJ 4
SDA>
SDA> FORMAT G10B720/TYPE=TQE 5
8010B720 TQE$L_TQFL 800A3EAO
8010B724 TQE$L_TQBL 80002870
8010B728 TQE$W_SIZE 0030
8010B72A TQE$B_TYPE OF
8010B72B TQE$B_RQTYPE 05 ←
8010B72C TQE$L_FPC 8011394C ←
8010B730 TQE$L_PID
8010B734 TQE$L_AST 004D0000
TQE$L_FR3
TQE$L_FR4 00030071
8010B738 TQE$Q_TIME 245B4460
8010B73C 008A72A2
8010B740 TQE$Q_DELTA 000F4240
8010B744 00000000
8010B748 TQE$B_RMOD 45
8010B749 TQE$B_EFN 46
8010B74A 5720
8010B74C TQE$L_RQPID 3A4B524F

SDA>
SDA> EXAMINE G11394C-C:G11394C+20 6
0001FFD6 00780020 00000000 000003A6 ..... .x..... 80113940
59530803 0000002C 00801139 40050000 ...@9.....SY 80113950
59532053 45544F4E 001C5345 544F4E53 SNOTES..NOTES SY 80113960
SDA>
SDA> $ 7
0001FFD6 00780020 00000000 000003E7 ..... .x..... 80113940
59530803 0000002C 00801139 40050000 ...@9.....SY 80113950
59532053 45544F4E 001C5345 544F4E53 SNOTES..NOTES SY 80113960
SDA>
SDA> $ 7
0001FFB6 00780020 00000000 0000040F ..... .x..... 80113940
59530803 0000002C 00801139 40050000 ...@9.....SY 80113950
59532053 45544F4E 001C5345 544F4E53 SNOTES..NOTES SY 80113960
SDA>

```

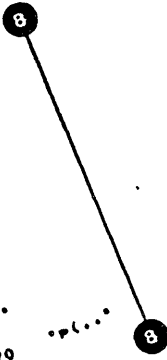
WRITING PRIVILEGED CODE

Solution to Problem 1 (Page 2 of 3)

```

SDA> SHOW SYMBOL EXE$GL_TQFL
EXE$GL_TQFL = 80002870 ; 8010B720
SDA> EX @.
8010B720: 800A3EA0 ..>..
SDA> &
800A3EA0: 80099F4D .....
SDA> &
8010E300: 8010E4E0 *8....
SDA> &
8010E4E0: 8010A640 *8....
SDA> &
8010A640: 8010DC40 *8....
SDA> &
8010DC40: 800028A0 *p(...
SDA> &
EXE$AL_TQENDREPT: 80002870 *....
SDA> &
EXE$GL_TQFL: 8010B720 *....
SDA> EXIT
* RUN STOPIDE
Value in CTL$GL_SITESPEC = 8010B720
Value in field = 00000AC8
Value in field = 00000ACA
Value in field = 00000ACC
*
* ANALYZE/SYSTEM
VAX/VMS System analyzer
SDA> SHOW SYMBOL CTL$GL_SITESPEC
CTL$GL_SITESPEC = 7FFEFF60 ; 00000000
SDA> EXAMINE G1139AC-C:G1139AC+20
0001FFD6 00780020 8010B720 8010CAA0 .....X.....
59530803 0000002C 00801139 40050000 ...89.....SY
59532053 4554AF4E 001C5345 544F4E53 SNOTES..NOTES SY
SDA> &
0001FFD6 00780020 8010B720 8010CAA0 .....X.....
59530803 0000002C 00801139 40050000 ...89.....SY
59532053 4554AF4E 001C5345 544F4E53 SNOTES..NOTES SY
SDA> READ TOEDEF.OBJ
SDA>

```



80113940
80113950
80113960

80113940
80113950
80113960

WRITING PRIVILEGED CODE

Solution to Problem 1 (Page 3 of 3)

```
SDA>
SDA> FORMAT 810B720/TYPE=TQE
8010B720 TQE$GL_TQFL 80113940
8010B724 TQE$GL_TQBL 8010C620
8010B728 TQE$W_SIZE 0015
8010B72A TQE$B_TYPE 13
8010B72B TQE$B_RQTYPE 00
8010B72C TQE$GL_FPC 00000000
TQE$GL_PID
8010B730 TQE$GL_AST 004D0000
TQE$GL_FR3
8010B734 TQE$GL_ASTPRM 00030041
TQE$GL_FR4
8010B738 TQE$Q_TIME A50AFE20
8010B73C 008A72A2
8010B740 TQE$Q_DELTA 000F4240
8010B744 00000000
8010B748 TQE$B_RMOD 43
8010B749 TQE$B_EFN 00
8010B74A 0000
8010B74C TQE$GL_RQPID 00010040
```

15

```
SDA>
SDA> EXAMINE EXE$GL_TQFL
EXE$GL_TQFL: 8009BFD4 *.....*
SDA> EX 8
800A3EA0: 8009BFD4 *.....*
SDA> 1
DZDRIVER+A34: 80002878 *x(...*
SDA> 1
EXE$GL_TQFL+008: 8010E300 *.....*
SDA>
SDA>
SDA> EXIT
$
```

16

16

WRITING PRIVILEGED CODE

2. Either print a copy of the MAKETQE and STOPTQE map files, or use the following section of those files to perform the following activities using the DELTA debugger. You will also need to consult the listing files in the module for MAKETQE and STOPTQE (or print copies for yourself).

Relevant Section of MAKETQE.MAP File

WORK:CMUIZNIKS.SYSPRG.PRIVCODEJMAKETQE.EXE:1 14-MAY-1982 16:28

Program Section Synopsis

<u>Psect Name</u>	<u>Module Name</u>	<u>Base</u>	<u>End</u>	<u>Length</u>	<u>Align</u>
NONSHARED_DATA	MAKETQE	00000200	00000212	00000013 (19.) LONG 2
		00000200	00000212	00000013 (19.) LONG 2
CODE	MAKETQE	00000400	000004AC	000000AD (173.) BYTE 0
		00000400	000004AC	000000AD (173.) BYTE 0
. BLANK .	MAKETQE	00000600	00000600	00000000 (0.) BYTE 0
	MAKETQE	00000600	00000600	00000000 (0.) BYTE 0
	SYS\$SDEF	00000600	00000600	00000000 (0.) BYTE 0
	SYS\$PI_VECTOR	00000600	00000600	00000000 (0.) BYTE 0

Relevant Section of STOPTQE.MAP File

WORK:CMUIZNIKS.SYSPRG.PRIVCODEJSTOPTQE.EXE:1 14-MAY-1982 16:28

Program Section Synopsis

<u>Psect Name</u>	<u>Module Name</u>	<u>Base</u>	<u>End</u>	<u>Length</u>	<u>Align</u>
NONSHARED_DATA	STOPTQE	00000200	000002AE	000000AF (175.) LONG 2
		00000200	000002AE	000000AF (175.) LONG 2
CODE	STOPTQE	00000400	00000568	00000169 (361.) BYTE 0
		00000400	00000568	00000169 (361.) BYTE 0
. BLANK .	STOPTQE	00000600	00000600	00000000 (0.) BYTE 0
	STOPTQE	00000600	00000600	00000000 (0.) BYTE 0
	SYS\$PI_VECTOR	00000600	00000600	00000000 (0.) BYTE 0

WRITING PRIVILEGED CODE

1. Run the MAKETOE program under the control of DELTA.
2. Set a breakpoint at the TSTL G^CTL\$GL_SITESPEC instruction (line 62 in listing file), and start the program.
3. When the breakpoint is reached, single step program execution for two instructions. Then, display the contents of location DELTA (line 32 in the listing file). You should confirm that the right value is present (000F4240), as indicated by the listing file.
4. Set a breakpoint at the RET instruction on line 146 in the listing file. Recall that DELTA cannot be used when debugging code that executes at any IPL greater than 0. Proceed execution from the breakpoint.
5. Examine the contents of R0, R1, and R2 when the breakpoint is reached. R0 should have a success code, and R1 and R2 will have system addresses. Proceed execution from the breakpoint.
6. When the program exits, exit from DELTA. (Be careful issuing the EXIT command. If you issue it at kernel mode, your process will be deleted, and when you hit <CR>, you will see a Username: prompt.)
7. Run the STOPTOE program under the control of DELTA.
8. Set a breakpoint at line 61 in the listing file, the call to \$LKWSET. Then, start program execution.
9. When the breakpoint is reached, issue the O instruction four times. You will notice that the arguments are pushed on the stack for the system service call (as the system service call macro is expanded). Notice that after the last O, the system service code is executed, and control is not returned until the system service completes. Set a breakpoint at line 79, the \$OUTPUT macro, and continue executing the program.
10. When this breakpoint is reached, issue the S instruction eleven times, while arguments are pushed on the stack. After the last S, you will notice that the call is not skipped over, but entered.
11. Resume execution, and when the program completes, exit from DELTA.

WRITING PRIVILEGED CODE

Solution to Problem 2

```

* RUN MAKETQE 1
DELTA Version X2.2

00000402/PUSHL #00 414:B:P 2

1 BRK AT 00000414
00000414/TSTL #07FFEFF60 S
0000041A/BEQL 00000424 S
00000424/MOVL #17,R1 200/000F4240 3
4A8:B:P 4

2 BRK AT 000004A8
000004A8/RET R0/00000001 R1/008A72A3 R2/80002870

IP 5

EXIT 00000001
80011D28/POPR #03 EXIT 6

*
* RUN/DEBUG STOPTQE 7
DELTA Version X2.2

00000402/PUSHL #00 414:B:P 8

1 BRK AT 00000414
00000414/PUSHL #00 0
00000416/PUSHL #00 0
0000041B/PUSHAQ 00000200 0
0000041E/CALLS #03,#07FFEDFA0 0
00000425/BLBS R0.00000429 4A9:B:P 9

2 BRK AT 000004A9
000004A9/CLRQ -(SP) S
000004AB/PUSHL #20 S
000004AD/PUSHL #00 S
000004AF/PUSHL 0000025F S
000004B5/PUSHAL 0000026B S
000004BB/CLRQ -(SP) S
000004BD/PUSHL #00 S
000004BF/MOVZWL #30,-(SP) S
000004C2/MOVZWL 00000208,-(SP) S
000004C9/PUSHL #00 S
000004CB/CALLS #0C,#07FFEDE00 S
7FFEDE02/CHKK #002C IP 10

Value in CTL$GL_SITESPEC = 80112AA0 11
Value in field = 0000030C
Value in field = 0000030F
Value in field = 00000311
EXIT 00000001
80011D28/POPR #03 EXIT 11

```

WRITING PRIVILEGED CODE

3. Write a program that will allocate and build an AST control block, and queue it to your own process.
 1. The program should determine its own process id, using the PCBSL_PID field.
 2. It should then allocate enough space for an AST control block from nonpaged pool, using the EXESALONONPAGED routine found in the module.
 3. It should initialize at least the following fields:
 - ACBSW_SIZE (using size from EXESALONONPAGED)
 - ACBSB_TYPE (using appropriate DYN\$... symbol)
 - ACBSL_PID (using PID previously determined)
 - ACBSL_AST (using address of AST routine)
 - ACBSB_RMOD (using USER mode, PSL\$C_USER)
 4. It should call on routine SCHSQAST to queue the AST (consult the supplied listing for required inputs and outputs). Note that this routine will raise IPL to SYNCH to queue the AST, and return at the same IPL at which it was called.
 5. The main program should then hibernate.
 6. The AST routine simply issues a \$WAKE system service, so the main program can exit.

WRITING PRIVILEGED CODE

Listing of ASTDEL Module (Page 1 of 3)

ASTDEL
V03-000

- AST ENQUEUE AND DELIVERY 27-APR-1982 01:08:27 VAX-11 Macro V03-00
SCHSQAST - ENQUEUE AST CONTROL BLOCK FOR 12-MAR-1982 17:09:55 _DBB0:CSYS.SRCJASTDEL.M

```

01AF 455 .SBTTL SCHSQAST - ENQUEUE AST CONTROL BLOCK FOR PROCESS
01AF 456 ;++
01AF 457 ; FUNCTIONAL DESCRIPTION:
01AF 458 ; SCHSQAST INSERTS THE AST CONTROL BLOCK SUPPLIED IN THE PROPER
01AF 459 ; POSITION BY ACCESS MODE IN THE AST QUEUE OF THE PROCESS SPECIF
01AF 460 ; BY THE PID FIELD OF THE AST CONTROL BLOCK. AN AST ARRIVAL EVE
01AF 461 ; IS THEN REPORTED FOR THE PROCESS TO REACTIVATE FROM A WAIT STA
01AF 462 ; IF APPROPRIATE. THE AST CONTROL BLOCK WILL BE RELEASED IMMEDI
01AF 463 ; IF THE PID SPECIFIES A NON-EXISTENT PROCESS.
01AF 464 ;
01AF 465 ; LOADABLE MULTI-PROCESSING CODE WILL REPLACE THIS ROUTINE WITH
01AF 466 ; ENTIRELY NEW CODE, AT NPMHQSAST.
01AF 467 ;
01AF 468 ; CALLING SEQUENCE:
01AF 469 ; BSB/JSB SCHSQAST
01AF 470 ;
01AF 471 ; INPUT PARAMETERS:
01AF 472 ; R2 - PRIORITY INCREMENT CLASS
01AF 473 ; R5 - POINTER TO AST CONTROL BLOCK
01AF 474 ;
01AF 475 ; IMPLICIT INPUTS:
01AF 476 ; PCB OF PROCESS IDENTIFIED BY PID FIELD
01AF 477 ;
01AF 478 ; OUTPUT PARAMETERS:
01AF 479 ; R0 - COMPLETION STATUS CODE
01AF 480 ; R4 - PCB ADDRESS OF PROCESS FOR WHICH AST WAS QUEUED
01AF 481 ;
01AF 482 ; SIDE EFFECTS:
01AF 483 ; THE PROCESS IDENTIFIED BY THE PID IN THE AST CONTROL BLOCK
01AF 484 ; WILL BE MADE EXECUTABLE IF NOT SUSPENDED.
01AF 485 ;
01AF 486 ; COMPLETION CODES:
01AF 487 ; $$$_NORMAL - NORMAL SUCCESSFUL COMPLETION STATUS
01AF 488 ; $$$_NONEXPR - NON-EXISTENT PROCESS
01AF 489 ;--
01AF 490 .ENABL LSB
01AF 491 QNONEXPR:
50 55 00 01AF 492 MOVL R5,R0 ; RELEASE AST CONTROL BLOCK
FE48 30 01B2 493 $SBW EXES0EANONPAGED ; IF NO SUCH PROCESS
50 08E8 8F 3C 01B5 494 MOVZWL $$$_NONEXPR,R0 ; SET ERROR STATUS CODE
43 11 01BA 495 BRB QEXIT ; AND EXIT
01BC 496
01BC 497 NPMHQSAST: ; MULTI-PROCESSING HOOK TO REP
01BC 498 SCHSQAST: ; ENQUEUE AST FOR PROCESS
50 0C A5 3C 01BC 499 MOVZWL ACB$$_PID(R5),R0 ; GET PROCESS INDEX FOR AST TA
01C0 500 DSBINT #IPL$_SYNCH ; DISABLE SYSTEM EVENTS
54 0000 DF40 00 01C6 501 MOVL $M^SCH$GL_PCBVECCR0J,R4 ; LOOK UP PCB ADDRESS
60 A4 0C A5 01 01CC 502 CNPL ACB$$_PID(R5),PCB$$_PID(R4) ; CHECK FOR MATCH IN PID
0C 12 01D1 503 BNEQ QNONEXPR ; PID MISMATCHES
50 04 01D3 504 CLRL R0 ; ASSUME KERNEL MODE AND CLEAR
10 A4 65 0E 01D5 505 IN$QUE (R5),PCB$$_ASTQFL(R4) ; ASSUME QUEUE IS EMPTY AND AT
2D 12 01D9 506 BNEQ 508 ; BR IF IT WAS NOT EMPTY
0B A5 95 01DB 507 TSTB ACB$$_RMODE(R5) ; CHECK FOR SPECIAL KERNEL AST
06 19 01DE 508 BLSS 108 ; BR IF YES
50 0B A5 FC 8F 8B 01E0 509 BICB3 4^C(3),ACB$$_RMODE(R5),R0 ; GET AST MODE
01E6 510 ;
01E6 511 ; THE PROCESS HEADER ADDRESS IS ALWAYS A SYSTEM SPACE ADDRESS (NEGATIV

```

WRITING PRIVILEGED CODE

Listing of ASTDEL Module (Page 2 of 3)

- AST ENQUEUE AND DELIVERY 27-APR-1982 01:08:27 VAX-11 Macro V03-00 Page 14
 SCH\$QAST - ENQUEUE AST CONTROL BLOCK FOR 12-MAR-1982 17:09:55 _0880:CSYS.SRCJASTDEL.MAR:1 (1)

```

01E6 512 : WHILE THE PROCESS HEADER IS RESIDENT. DURING THE OUTSWAP TRANSITION IT IS
01E6 513 : THE BALANCE SLOT INDEX, A SMALL POSITIVE NUMBER. FINALLY, AFTER OUTSWAP IT
01E6 514 : IS SET TO ZERO. HENCE, THE FOLLOWING TEST COMBINES THE FETCH OF THE PHD
01E6 515 : ADDRESS WITH THE TEST FOR PROCESS RESIDENCE.
01E6 516 :
51 64 A4 00 01E6 517 10%: MOVL PC$$_PHD(R4),R1 : POINT TO PROCESS HEADER
05 18 01EA 518 BGEQ 20% : DON'T SET ASTLVL IF NOT RESIDENT
00CB C1 50 90 01EC 519 MOVB R0,PHD$$_ASTLVL(R1) : SET ASTLVL IN PROCESS HEADER
54 0000 CF 01 01F1 520 20%: CMPL W$SCH$GL_CURPCB,R4 : IS PROCESS CURRENT PROCESS
08 13 01F6 521 BEQL 40% : YES,
01F8 522 RPTEVT AST : REPORT AST ARRIVAL
50 01 3C 01FC 523 30%: MOVZWL #SS$_NORMAL,R0 : SET SUCCESS STATUS CODE
01FF 524 QEXIT: ENBINT : ENABLE INTERRUPTS
05 0202 525 RSB : AND RETURN
0203 526 :
0203 527 : IF THE AST IS BEING ENQUEUED FOR THE CURRENT PROCESS, THEN THE REPORTING
0203 528 : OF THE AST EVENT CAN BE BYPASSED AND THE ASTLVL PROCESSOR REGISTER MUST BE
0203 529 : SET INSTEAD.
0203 530 :
13 50 0A 0203 531 40%: MTPR R0,#PR$_ASTLVL : ALSO SET ASTLVL REGISTER
F4 11 0206 532 BRB 30% :
0208 533 :
0208 534 : THE AST QUEUE WAS NOT EMPTY (ITS USUAL CONDITION) AND THE PROPER
0208 535 : POSITION FOR THE NEW AST MUST BE LOCATED. SINCE THE AST CONTROL
0208 536 : BLOCK HAS BEEN ERRONEOUSLY INSERTED ON THE QUEUE, IT MUST BE REMOVED
0208 537 : FIRST.
0209 538 :
55 65 0F 0209 539 50%: REMQUE (R5),R5 : ELSE CORRECT MISTAKE
51 10 A4 DE 0208 540 MOVAL PC$$_ASTQFL(R4),R1 : POINT TO QUEUE HEADER
53 61 00 020F 541 MOVL (R1),R3 : GET FIRST ENTRY ON QUEUE
08 A5 95 0212 542 TSTB ACB$$_RMOD(R5) : CHECK FOR SPECIAL KERNEL AST
0F 18 0215 543 BGEQ 70% : BR IF NOT
0217 544 :
0217 545 : THE NEW AST IS A SPECIAL KERNEL AST. IT WILL GO AFTER ALL OTHER SPECIAL
0217 546 : KERNEL ASTS OR AT THE HEAD OF THE QUEUE IF THERE ARE NONE.
0217 547 :
53 51 01 0217 548 60%: CMPL R1,R3 : CHECK FOR END OF QUEUE
27 13 021A 549 BEQL 110% : BR IF NOT
08 A3 95 021C 550 TSTB ACB$$_RMOD(R3) : CHECK FOR SPECIAL KERNEL IN QUEUE
22 13 021F 551 BGEQ 110% : BR IF NOT
53 63 00 0221 552 MOVL (R3),R3 : FLINK ON TO NEXT ACB
F1 11 0224 553 BRB 60% :
0226 554 :
0226 555 : THE NEW AST IS A NORMAL AST. IT WILL GO AFTER ALL SPECIAL KERNEL ASTS
0226 556 : AND ASTS WITH LOWER ACCESS MODE.
0226 557 :
50 08 A5 FC 8F 89 0226 558 70%: BICB3 #C<<3>,ACB$$_RMOD(R5),R0: GET AST MODE
53 51 01 022C 559 80%: CMPL R1,R3 : CHECK FOR END OF QUEUE
12 13 022F 560 BEQL 110% : INSERT IF AT END
02 00 ED 0231 561 CMPZV #ACB$$_MODE,#ACB$$_MODE,- :
50 08 A3 0234 562 ACB$$_RMOD(R3),R0 : COMPARE ACCESS MODES
05 14 0237 563 BGTR 100% : IF GTR AT RIGHT PLACE
53 63 00 0239 564 90%: MOVL (R3),R3 : FLINK ON TO NEXT ACB
EE 11 023C 565 BRB 80% :
08 A3 95 023E 566 100%: TSTB ACB$$_RMOD(R3) : IS THIS ENTRY A SPECIAL KAST?
F6 19 0241 567 BLSS 90% : YES, MUST GO AFTER THIS
0243 568 :

```

WRITING PRIVILEGED CODE

Listing of ASTDEL Module (Page 3 of 3)

ASTDEL
V03-000

- AST ENQUEUE AND DELIVERY 27-APR-1982 01:08:27 VAX-11 Macro V03-00
SCH\$QAST - ENQUEUE AST CONTROL BLOCK FOR 12-MAR-1982 17:09:55 _DB90:CSYS-SRCJASTDEL.MAR:

```

0243 569 : NOW THE CORRECT POSITION HAS BEEN LOCATED.  INSERT THE AST CONTROL BLOC
0243 570 : ON THE QUEUE AND COMPUTE THE NEW VALUE FOR ASTLVL BY INTERROGATING THE
0243 571 : MODE OF THE AST CONTROL BLOCK AT THE HEAD OF THE QUEUE.
0243 572 :
04 83 65 0E 0243 573 110$:  INSQUE (R5),@ACB$$_ASTQBL(R3)  : INSERT AFTER PREVIOUS
                    50 D4 0247 574          CLRRL R0          : ASSUME KERNEL MODE
51 10 A4 D0 0249 575          MOVL PCB$$_ASTQFL(R4),R1    : GET HEAD OF AST QUEUE
08 A1 95 0240 576          TSTB ACB$$_RMODE(R1)      : IS IT KAST?
                    94 19 0250 577          BLSS 10$          : BR IF YES TO SET ASTLVL
50 08 A1 FC 8F 88 0252 578          BICB3 @^C(3),ACB$$_RMODE(R1),R0 : GET AST MODE FOR HEAD OF QUEUE
                    8C 11 0258 579          BRB 10$          : GO SET ASTLVL
025A 580
025A 581          .DSABL LSB
025A 582          ASSUME ACB$$_MODE EQ 0
025A 583          ASSUME ACB$$_MODE EQ 2
025A 584          ASSUME ACB$$_KAST EQ 7
025A 585

```


WRITING PRIVILEGED CODE

Listing of OWNAST.MAR File

```

:
:                               OWNAST.MAR
:
:   This program allocates and builds an AST control block,
:   which it queues to itself.
:
:   Global symbols
:   $CTNDEF
:   $ACBDEF
:   $PPIDEF
:   $IPLDEF
:   $PSLDEF
:
:   Local Storage
:   .PSECT NONSHARED_DATA PIC, NOEXE, LCNB
PID:  .BLKL 1 : Location to store PID
AST:  .ADDRESS ASTADR : Address of AST routine
:
:   Program entry point
:   .PSECT CODE PIC, SHR, NCWRT
START: .WORD 0 : no registers need be saved
:   $CMKRNLS ROUTIN=10s : go queue AST to self
:   $LBS RO,7s : exit if error
:   RET
7s:  $MIBER_S : wait for AST to wake
:   RET
:
:   Queue AST to self
10s: .WORD $MCP2,R3,R4,R5 : save some registers
:   MOVL $SCHSGL_CURPCB,RO : get CURRENT PCB address
:   MOVL PCB$SL_PID(RO),PID : get PID from PCB field
:   MOVL $ACBSK_LENGTH, R1 : get length of AST block
:   SETIPL $IPL$ASTDEL : prevent process deletion
:   JSB $EXESALONONPAGED : get pool space
:   $LBS RO, 20s : exit if none
:   SETIPL $0 : lower IPL if error
:   RET
:
:   Initialize code block to look like ACB
20s: MOVW R1, ACBSW_SIZE(R2) : record size of block allocated
:   MOVW $DYN$C_ACB, ACBS$TYPE(R2) : and type
:   MOVL PID, ACBS$PID(R2) : got pid from GETJPI
:   MOVL AST, ACBS$AST(R2) : store AST address
:   MOVW $PSL$C_USER, ACBS$RM00(R2) : user mode AST
:
:   Prepare registers to queue AST ($SCH$QAST raises IPL to SYNCH)
:   MOVL R2, R5 : need address of AST block
:   MOVL $PRIS_TICOM, R2 : give big boost
:   JSB $SCH$QAST : queue AST
:   MOVL $SS$NORMAL, R0 : indicate success
:   SETIPL $0 : lower IPL before returning
:   RET
:
:   AST routine simply awakens main
ASTADR: .WORD 0 : no registers saved
:   $WAKE_S
:   $LBC RO, 40s
:   RET
40s: $EXIT_S CODE=RO
:
:   .END START

```

WRITING PRIVILEGED CODE

Sample Run

```
$ TYPE OWNAST.COM                                OWNAST.COM
$!
$!
$ MACRO OWNAST+SYS+LIBRARY:LIB/LIB
$ LINK OWNAST, SYS+SYSTEM:SYS.STB/SELECTIVE
$
$ @OWNAST
$
$ SET PROCESS/PRIV=(CHKRNL)
$
$ RUN OWNAST
$
```

WRITING PRIVILEGED CODE

4. If you receive MAIL while you are logged in, the system displays the following message on your terminal "New mail from xyzuser". Use the PATCH utility, as described below, to alter that message.
 1. Copy the file OLDMAIL.EXE to your directory from a directory specified by your instructor. (This file may not be exactly the same as SYSSSYSTEM:MAIL.EXE.) You will patch your own copy of this MAIL file, not the one that is being used on the system.
 2. Use the DUMP command to display the contents of the OLDMAIL.EXE file (you probably want to use the /OUTPUT= qualifier so you can print the output file). Also, use the ANALYZE/IMAGE/OUTPUT= command to obtain a listing of the image section descriptors.
 3. Locate the text of the "New mail from !AS" message. The descriptor for the string follows the text (string length, string address). If you didn't know where the descriptor was, once you found the string, you could use the SEARCH command to search the file for the string address, and decide if any of the occurrences could be part of the descriptor.
 4. If you only wanted to change the text (keeping the same number of characters in the string), you could simply deposit over the locations containing the text you want altered. For example, changing "New" to "Fan".
 5. If you want to change the message to have a different number of characters, you will have to insert a new text string in the image file, and update the descriptor to contain the new string length and address.
 6. To add a different string like "Junk mail from !AS", you must locate some unused space in the image file. Look in the image section containing the message for empty space (all 0's in the DUMP output). Note that the number of blocks in each image section is specified in the ANALYZE/IMAGE output.
 7. Run the PATCH utility, creating a journal file, and specify your copy of MAIL as input.
 1. Ask to CREATE a command procedure file with the PATCH commands you issue.
 2. SET the ECO level.

WRITING PRIVILEGED CODE

3. Issue DEPOSIT commands to add the ASCII text you want at an unused location in the image file. (Note that a BYTE value of 7 rings the bell. You may want to ring the bell before displaying the message.)
4. Overwrite the string descriptor with the length of the text string you chose to use, and the address of the text string you entered.
5. Issue the UPDATE command to make the changes in the file.
6. EXIT from PATCH.

8. Test your MAIL program by sending MAIL to yourself, and seeing if the right text string is displayed. You will require the following privileges to run your copy of MAIL: (WORLD, OPER, NETMBX, SYSPRV)
9. You should verify your PATCH session was a success by displaying the journal file and the produced command procedure. You can also issue the ANALYZE/IMAGE/PATCH command to examine the file you patched.

WRITING PRIVILEGED CODE

Relevant Section of ANALYZE/IMAGE Output

Analyze Image 14-MAY-1982 16:44:46.10 Page 1
WORK:CHUIZNIKES.SYSPRG.PRIVCODEJOLDMAIL.EXE11

IMAGE HEADER

Fixed Header Information

image format major id: 02, minor id: 04
header block count: 1
image type: executable (IHDSK_EXE)
I/O channel count: default
I/O page count: default
linker flags:
 (0) IHDSV_LNKDEBUG 0
 (1) IHDSV_LNKNOTFR 0
 (2) IHDSV_NOPOBUFS 0
 (3) IHDSV_PICING 1
 (4) IHDSV_POIMAGE 0

Image Activation Information

first transfer address: XX'00001810'
second transfer address: XX'00000000'
third transfer address: XX'00000000'

Global Symbol Table & Debug Symbol Table Information

debug symbol table VBN: 0, block count: 0
global symbol table VBN: 0, record count: 0

Image Identification Information

image name: 'MAIL'
image file identification: 'U03-001'
link date/time: 24-FEB-1982 09:34:06.12
linker identification: '03-14'

Patch Information

There are no patches at this time.

Image Section Descriptors (ISD)

1) image section descriptor (16 bytes)
page count: 3
base virtual address: XX'00000200' (PO space)
page fault cluster size: default
ISD flags:
 (0) ISDSV_GBL 0
 (1) ISDSV_CRF 0
 (2) ISDSV_DZRO 0
 (3) ISDSV_WRT 0
 (7) ISDSV_LASTCLU 1
 (8) ISDSV_COPYALWAY 0
 (9) ISDSV_BASED 0
 (10) ISDSV_FIXUPVEC 0
 (17) ISDSV_VECTOR 0
 (18) ISDSV_PROTECT 0

WRITING PRIVILEGED CODE

Relevant Section of DUMP Command Output (Page 1 of 2)

Dump of file WORK:\CHUIZNIKES.SYSPRG.PRIVCODE\OLDMAIL.EXE:1 on 14-MAY-1982 16:45:
53.02

File ID (3726.23.0) End of file block 36 / Allocated 36

Virtual block number 3 (00000003), 512 (0200) bytes

```

00000410 010E0005 0054454E 24535953 SYS$NET..... 000000
0000223D 4C49414D 223A3A3A 54454E5F _NET:::'MAIL'.. 000010
66204E52 55544552 20737365 72500A0D ..Press RETURN f 000020
0000001A 00002E2E 2E65726F 6D20726F or more..... 000030
00000007 00534121 202A2321 00000420 ...!@: !AS..... 000040
20206D6F 72462023 20202020 00000444 D... # From 000050
44202020 20202020 20202020 20202020 D 000060
6A627553 20202020 20202020 20637461 at Subject 000070
00000020 0000045A 0000002F 00746365 ect./...T... .. 000080
4C553521 00003A3A 0000048C 00000001 .....!SUL 000090
44412120 53413231 21204441 30322120 !20AD !12AS !AD 0000A0
49444524 4C49414D 0000049C 00000014 .....MAIL$EDI 0000B0
21402024 00000488 00000009 00000054 T.....$ @! 0000C0
00002253 41212220 22534121 22205341 AS '!AS' '!AS'.. 0000D0
54535953 24535953 000004CC 00000012 .....SYS$SYST 0000E0
004D4F43 2E544944 454C4941 4D3A4D45 EN:MAILEDIT.COM. 0000F0
6F792072 65746E45 000004E8 00000017 .....Enter wo 000100
776F6C65 62206567 61737365 6D207275 ur message below 000110
77205A2F 4C525443 20737365 7250202E . Press CTRL/Z w 000120
5443202C 6574656C 706D6F63 206E6568 hen complete, CT 000130
0000003A 74697571 206F7420 432F4C52 RL/C to quit:... 000140
00000003 00000001 00000508 00000045 E..... 000150
00000560 00000008 504C4548 4C49414D MAILHELP..... 000160
00000042 4C482E3A 504C4548 24535953 SYS$HELP:..HLB... 000170
00001764 0202000C 00000570 0000000D .....P.....d... 000180
00001778 00001770 031D0008 00001760 .....P.....x... 000190
7266206C 69616D20 77654E07 00000000 .....New mail fr 0001A0
000005A4 00000012 00005341 21206D6F on !AS..... 0001B0
00000009 00000054 4E495250 24535953 SYS$PRINT..... 0001C0
6E652057 35212042 6F4A2020 000005C0 .... Job !UW en 0001D0
21206575 65757120 6E6F2064 65726574 tered on queue ! 0001E0
72500A0D 000005D4 0000001E 00004341 AC.....Pr 0001F0

```

Descriptor for message to be changed
length = 12 (hex) = 18 decimal bytes
5A4 = starting address of string

WRITING PRIVILEGED CODE

Relevant Section of DUMP Command Output (Page 2 of 2)

Dump of file WORK:\MUIZNIKES.SYSPRG.PRIVCODE\JOLDMAIL.EXE:1 on 14-MAY-1982 16:45:
53.02

File ID (3726.23.0) End of file block 36 / Allocated 36

Virtual block number 4 (00000004), 512 (0200) bytes

```
6D20726F 66204E52 55544552 20737365  ess RETURN for a 000000
000005FC 0000001A 00002E2E 2E65726F  are..... 000010
0000000C 4C552123 20534121 202A2321  !* !AS #!UL... 000020
00000000 00000000 00000000 00000620  ..... 000030
00000000 00000000 00000000 00000000  ..... 000040
00000000 00000000 00000000 00000000  ..... 000050
00000000 00000000 00000000 00000000  ..... 000060
00000000 00000000 00000000 00000000  ..... 000070
00000000 00000000 00000000 00000000  ..... 000080
00000000 00000000 00000000 00000000  ..... 000090
00000000 00000000 00000000 00000000  ..... 0000A0
00000000 00000000 00000000 00000000  ..... 0000B0
00000000 00000000 00000000 00000000  ..... 0000C0
00000000 00000000 00000000 00000000  ..... 0000D0
00000000 00000000 00000000 00000000  ..... 0000E0
00000000 00000000 00000000 00000000  ..... 0000F0
00000000 00000000 00000000 00000000  ..... 000100
00000000 00000000 00000000 00000000  ..... 000110
00000000 00000000 00000000 00000000  ..... 000120
00000000 00000000 00000000 00000000  ..... 000130
00000000 00000000 00000000 00000000  ..... 000140
00000000 00000000 00000000 00000000  ..... 000150
00000000 00000000 00000000 00000000  ..... 000160
00000000 00000000 00000000 00000000  ..... 000170
00000000 00000000 00000000 00000000  ..... 000180
00000000 00000000 00000000 00000000  ..... 000190
00000000 00000000 00000000 00000000  ..... 0001A0
00000000 00000000 00000000 00000000  ..... 0001B0
00000000 00000000 00000000 00000000  ..... 0001C0
00000000 00000000 00000000 00000000  ..... 0001D0
00000000 00000000 00000000 00000000  ..... 0001E0
00000000 00000000 00000000 00000000  ..... 0001F0
```

← Unused space in image section - will be used to hold text of new message starting at 634 (hex)

WRITING PRIVILEGED CODE

Patching the New MAIL File Terminal Session

\$ PATCH/JOURNAL NEWMAIL.EXE

PATCH Version 3-00 15-Mar-1982

ZPATCH-I-NOLCL, imase does not contain local symbols
ZPATCH-I-NOGBL, some or all global symbols not accessible

PATCH>CREATE PATCHMAIL

PATCH>SET ECO 1

PATCH>DEPOSIT/ASCII ← Enter new message
LOC>~X634 in unused area

NEW>' Jun'

NEW>'k aa'

NEW>'il f'

NEW>'rga'

NEW>'IAS'

NEW>EXIT

```
old: 00000634: ''
old: 00000638: ''
old: 0000063C: ''
old: 00000640: ''
old: 00000644: ''
new: 00000634: ' Jun'
new: 00000638: 'k aa'
new: 0000063C: 'il f'
new: 00000640: 'rga'
new: 00000644: 'IAS'
```

PATCH>DEPOSIT/BYTE ← Make first character
LOC>~X634 ring terminal bell

NEW>7

NEW>EXIT

```
old: 00000634: 20
new: 00000634: 07
```

PATCH>DEPOSIT ←

LOC>5B8

NEW>~X13

NEW>~X634

NEW>EXIT

```
old: 000005B8: 00000012
old: 000005BC: 000005A4
new: 000005B8: 00000013
new: 000005BC: 00000634
```

Update descriptor to point
to new message. New message
length now 19 (decimal) =
13 (hex) bytes

PATCH>UPDATE

ZPATCH-I-WRTFIL, updating imase file WORK:[HUIZNIK.SYSPRG.PRIVCODE]NEWMAIL.EXE

12

PATCH>EXIT

\$

WRITING PRIVILEGED CODE

Sample Run To Test Patched File

```
$ SET PROCESS/PRIV=(WORLD,OPER,NETMBX,SYSPRV)
$
$ RUN NEWMAIL

MAIL> SEND
To: MUIZNIEKS
Subj: TESTING NEW MAIL VERSION
Enter your message below. Press CTRL/Z when complete, CTRL/C to quit:
This should be sent with JUNK MAIL message
^Z
Junk mail from MUIZNIEKS

                                MAIL> EXIT

$
$ MAIL

You have 1 new message.

MAIL> _____                                MAIL #53
From: MUIZNIEKS      14-MAY-1982 16:53
To: MUIZNIEKS
Subj: TESTING NEW MAIL VERSION

This should be sent with JUNK MAIL message

MAIL> DELETE

MAIL> SEND
To: MUIZNIEKS
Subj: STILL TESTING
Enter your message below. Press CTRL/Z when complete, CTRL/C to quit:
This message should still be sent with regular NEW MAIL indication
^Z
New mail from MUIZNIEKS

MAIL> READ MAIL                                MAIL #53
From: MUIZNIEKS      14-MAY-1982 16:54
To: MUIZNIEKS
Subj: STILL TESTING

This message should still be sent with regular NEW MAIL indication

MAIL> DEL

MAIL> EXIT
$
```

WRITING PRIVILEGED CODE

Examining Journal File

\$ TYPE NEWMAIL.JNL

PATCH Version 3-00 15-Mar-1982

IMAGE FILE BEING PATCHED: 'WORK:[CHUIZNIKS.SYSPRG.PRIVCODE]NEWMAIL.EXE;1'
JOURNAL FILE: 'WORK:[CHUIZNIKS.SYSPRG.PRIVCODE]NEWMAIL.JNL;1'
DATE/TIME OF PATCH: 14-MAY-1982 16:47:48.54

XPATCH-I-NOLCL, image does not contain local symbols
XPATCH-I-NOGBL, some or all global symbols not accessible

PATCH>CREATE PATCHMAIL
COMMAND FILE: 'WORK:[CHUIZNIKS.SYSPRG.PRIVCODE]PATCHMAIL.COM;2'

PATCH>SET ECD 1
PATCH>DEPOSIT/ASCII

LOC> ^X634
NEW> ' Jun'
NEW> 'k sa'
NEW> 'll f'
NEW> 'rom'
NEW> 'IAS'
NEW> EXIT
old: 00000634: ''
old: 00000638: ''
old: 0000063C: ''
old: 00000640: ''
old: 00000644: ''
new: 00000634: ' Jun'
new: 00000638: 'k sa'
new: 0000063C: 'll f'
new: 00000640: 'rom'
new: 00000644: 'IAS'

PATCH>DEPOSIT/BYTE

LOC> ^X634
NEW> 7
NEW> EXIT
old: 00000634: 20
new: 00000634: 07

PATCH>DEPOSIT

LOC> 5B8
NEW> ^X13
NEW> ^X634
NEW> EXIT
old: 000005B8: 00000012
old: 000005BC: 000005A4
new: 000005B8: 00000013
new: 000005BC: 00000634

PATCH>UPDATE

XPATCH-I-WRTFIL, updating image file WORK:[CHUIZNIKS.SYSPRG.PRIVCODE]NEWMAIL.EXE

!2
PATCH>EXIT

\$

WRITING PRIVILEGED CODE

Relevant Sections of ANALYZE/IMAGE/PATCH Output (Page 1 of 2)

Analyze Image 14-MAY-1982 16:59:22.98 Page 1
WORK:CMUIZNIKS.SYSPRG.PRIVCODEJNEWHAIL.EXE12

IMAGE HEADER

Fixed Header Information

image format major id: 02, minor id: 04
header block count: 1
image type: executable (IHDSK_EXE)
I/O channel count: default
I/O page count: default
linker flags:
(0) IHDSV_LNKDEBUG 0
(1) IHDSV_LMKNOTFR 0
(2) IHDSV_NOPBUF 0
(3) IHDSV_PICING 1
(4) IHDSV_POIMAGE 0

Image Activation Information

first transfer address: XX'00001810'
second transfer address: XX'00000000'
third transfer address: XX'00000000'

Global Symbol Table & Debug Symbol Table Information

debug symbol table VBN: 0, block count: 0
global symbol table VBN: 0, record count: 0

Image Identification Information

image name: 'MAIL'
image file identification: 'V03-001'
link date/time: 24-FEB-1982 09:34:06.12
linker identification: '03-14'

Patch Information

DEC eco levels 1- 98: XX'00000001', XX'00000000', XX'00000000'
user eco levels 99-132: XX'00000000'
read/write patch area address: XX'00000000', length: 0
read-only patch area address: XX'00000000', length: 0
patch command text VBN: 37
last patch date/time: 14-MAY-1982 16:47:48.54

Image Section Descriptors (ISD)

1) image section descriptor (16 bytes)
page count: 3
base virtual address: XX'00000200' (PO space)
page fault cluster size: default
ISD flags:
(0) ISDSV_GBL 0
(1) ISDSV_CRF 0
(2) ISDSV_DZRO 0
(3) ISDSV_WRT 0
(7) ISDSV_LASTCLU 1

WRITING PRIVILEGED CODE

Relevant Sections of ANALYZE/IMAGE/PATCH Output (Page 2 of 2)

Analyze Image 14-MAY-1982 17:01:03.85 Page 6
WORK:[CHUIZNIKS.SYSPRG.PRIVCODE]NEWMAIL.EXE:2

PATCH TEXT

```
/JOURNAL NEWMAIL.EXE
SE EC
~X00000001
D /AS
~X00000634
' Jun'
'k ss'
'il f'
'pss'
'IAS'
EXI
D /B
~X00000634
~X00000007
EXI
D
~X000005B8
~X00000013
~X00000634
EXI
```

The analysis uncovered NO errors.

ANALYZE/IMAGE/PATCH NEWMAIL.EXE
8

WRITING PRIVILEGED CODE

Relevant Sections of DUMP Output for Patched File (Page 1 of 2)

Dump of file WORK:CHUIZNIKS.SYSPRG.PRIVCODE\NEWMAIL.EXE12 on 14-MAY-1982 17:01:
20.06
File ID (7567,16,0) End of file block 37 / Allocated 39

Virtual block number 3 (00000003), 512 (0200) bytes

```
00000410 010E0005 0054454E 24535953 SYS$NET..... 000000
0000223D 4C49414D 223A3A3A 54454E5F _NET!!!MAIL.. 000010
66204E52 55544552 20737365 72500A0D ..Press RETURN f 000020
0000001A 00002E2E 2E65726F 6D20726F or more..... 000030
00000007 00534121 202A2321 00000420 ...!$$ !AS.... 000040
20206D6F 72442023 20202020 00000444 D... $ From 000050
44202020 20202020 20202020 20202020 D 000060
6A627553 20202020 20202020 20657461 ate SubJ 000070
00000020 00000454 0000002F 00746365 ect./...T... .. 000080
4C553521 00003A3A 0000048C 00000001 .....:..!SUL 000090
44412120 53413231 21204441 30322120 !20AD !12AS !AD 0000A0
49444524 4C49414D 0000049C 00000014 .....MAIL$EDI 0000B0
21402024 000004B8 00000009 00000054 T.....$ @! 0000C0
00002253 41212220 22534121 22205341 AS !AS° !AS!.. 0000D0
54535953 24535953 000004CC 00000012 .....SYS$SYST 0000E0
004D4F43 2E544944 454C4941 4D3A4D45 EN:MAILEDIT.COM. 0000F0
6F792072 65746E45 000004E8 00000017 .....Enter vo 000100
776F6C65 62206567 61737365 6D207275 ur message below 000110
77205A2F 4C525443 20737365 7250202E ..Press CTRL/Z w 000120
5443202C 6574656C 706D6F63 206E6568 hen complete, CT 000130
0000003A 74697571 206F7420 432F4C52 RL/C to quit:... 000140
00000003 00000001 00000508 00000045 E..... 000150
00000560 00000008 504C4548 4C49414D MAILHELP.... 000160
00000042 4C482E3A 504C4548 24535953 SYS$HELP:.HLB... 000170
00001764 0202000C 00000570 0000000D ....P.....d... 000180
00001778 00001770 031D0008 00001740 .....P.....x... 000190
7244204C 69416D2C 77654E07 00000000 .....New mail fr 0001A0
00000634 00000013 00005341 21206D6F om !AS.....4... 0001B0
00000009 00000054 4E495250 24535953 SYS$PRINT..... 0001C0
6E652057 55212042 6F4A2020 000005C0 .... Job !UM en 0001D0
21206575 65757120 6E6F2064 65726574 tered on queue ! 0001E0
72500A0D 000005D4 0000001E 00004341 AC.....Pr 0001F0
```

Original text
still there

Descriptor
updated

WRITING PRIVILEGED CODE

Relevant Sections of DUMP Output for Patched File (Page 2 of 2)

Dump of file WORK:\CHUIZNIKS.SYSPRG.PRIVCODE\JNEWSMAIL.EXE:2 on 14-MAY-1982 17:01:
20.06

File ID (7567.16.0) End of file block 37 / Allocated 39

Virtual block number 4 (00000004), 512 (0200) bytes

```
4D20726F 66204E52 55544552 20737365 ess RETURN for a 000000
000005FC 0000001A 00002E2E 2E65726F ore..... 000010
0000000C 4C552123 20534121 202A2321 !* !AS #!UL.... 000020
66206C69 616D206B 6E754A07 00000620 ....Junk mail f 000030 ← New text added
00000000 00000000 20534121 206D6F72 rom !AS ..... 000040
00000000 00000000 00000000 00000000 ..... 000050
00000000 00000000 00000000 00000000 ..... 000060
00000000 00000000 00000000 00000000 ..... 000070
00000000 00000000 00000000 00000000 ..... 000080
00000000 00000000 00000000 00000000 ..... 000090
00000000 00000000 00000000 00000000 ..... 0000A0
00000000 00000000 00000000 00000000 ..... 0000B0
00000000 00000000 00000000 00000000 ..... 0000C0
00000000 00000000 00000000 00000000 ..... 0000D0
00000000 00000000 00000000 00000000 ..... 0000E0
00000000 00000000 00000000 00000000 ..... 0000F0
00000000 00000000 00000000 00000000 ..... 000100
00000000 00000000 00000000 00000000 ..... 000110
00000000 00000000 00000000 00000000 ..... 000120
00000000 00000000 00000000 00000000 ..... 000130
00000000 00000000 00000000 00000000 ..... 000140
00000000 00000000 00000000 00000000 ..... 000150
00000000 00000000 00000000 00000000 ..... 000160
00000000 00000000 00000000 00000000 ..... 000170
00000000 00000000 00000000 00000000 ..... 000180
00000000 00000000 00000000 00000000 ..... 000190
00000000 00000000 00000000 00000000 ..... 0001A0
00000000 00000000 00000000 00000000 ..... 0001B0
00000000 00000000 00000000 00000000 ..... 0001C0
00000000 00000000 00000000 00000000 ..... 0001D0
00000000 00000000 00000000 00000000 ..... 0001E0
00000000 00000000 00000000 00000000 ..... 0001F0
```

WRITING PRIVILEGED CODE

Command Procedure Produced by PATCH

```
$ TYPE PATCHMAIL.COM  
/JOURNAL NEWMAIL.EXE  
SE EC  
~X00000001  
D /AS  
~X00000634  
' Jun'  
'k es'  
'il f'  
'rob '  
'IAS '  
EXI  
D /B  
~X00000634  
~X00000007  
EXI  
D  
~X000005B8  
~X00000013  
~X00000634  
EXI  
U  
EXI  
$
```

USER-WRITTEN SYSTEM SERVICES

TEST

Your assignment is to develop a number of programs, and user-written system service, that allow a process to declare TBIT exception handler. This handler will receive control immediately following a TBIT exception, without executing all of the VMS exception dispatching code.

The TBIT exception (Trace Trap) is generated following an instruction for which the Trace Bit (bit 4) is set in the Processor Status Word (PSW). This bit is often used by debugger (for single stepping instructions), or for performance monitoring/testing.

You will first have to write a dispatcher that can transfer control to a user-written handler following a TBIT exception. This dispatcher will be patterned after the change mode handler that can be declared using the \$DCLCMH system service. These handlers are entered following an exception (by the hardware using vectors in the SCB). They transfer control by looking for an address in a fixed location in P1 space, where the address of user-declared handler may be stored.

You will then need to write a system service that allows a user to declare a TBIT handler on a per-process basis. The system service will load the handler address in the fixed location in P1 space.

Finally, you will want to write a program that will restore the default system routine (EXESTBIT) for handling TBIT exceptions.

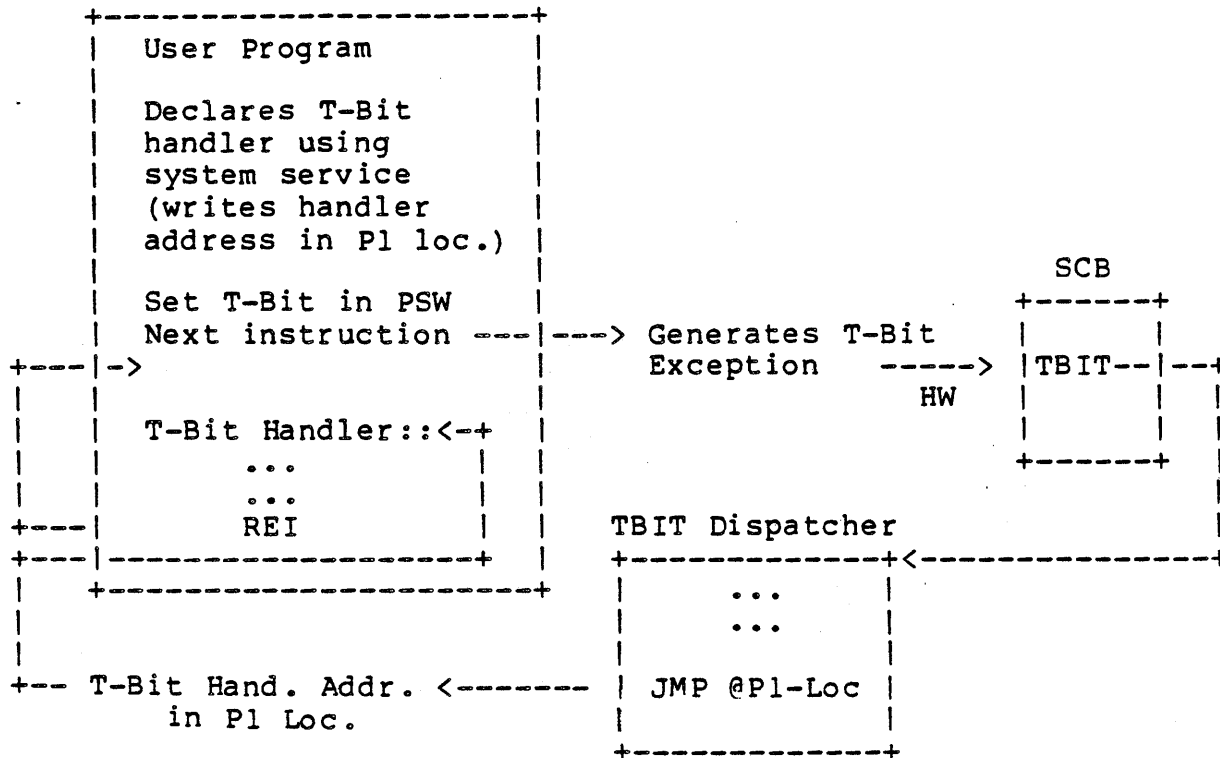
This exercise has been split into several parts. You should read the description of all the parts before starting to solve any of them. You will probably want to work on some of the parts "out of order", and use solutions provided by the instructor for some parts to test other parts you are writing. In particular, you may want to solve this exercise in the following order:

- Do Part 2 first, which involves writing a system service
- Use Part 5 to build and test the entire application (you may need to modify the command procedure somewhat)
- Do Part 3 next, since it is simpler than Part 1
- Then do Part 1
- Finally do Part 4

Note that it is quite unlikely that you will be able to complete all five parts in one day.

USER-WRITTEN SYSTEM SERVICES

Overview of Control Flow



USER-WRITTEN SYSTEM SERVICES

PART 1 - Loading TBIT Dispatcher (LOADTBIT.MAR)

In order to receive control after a TBIT exception, you first need to:

- Allocate a block of nonpaged pool to store the TBIT dispatcher
 - Use EXESALONONPAGED
 - Assume TBIT dispatcher block type is DYN_K_TBIT

- Copy the dispatch code to the pool block
 - The code must be PIC, since it is copied
 - Any register used by the routine must be saved, and later restored (perhaps by the user's handler). The supplied dispatcher does not use any registers.
 - Special considerations include:
 - Access mode at which handler entered

The supplied solution enters the handler in kernel mode. It is possible to change access mode (using REI) to the mode of the caller, but this takes more time, and introduces considerable complexity (probing accessibility of stack, so can copy arguments to that stack, etc.). To help prevent a system security problem (non-privileged users being able to write and execute kernel mode code), the system service will check to make sure the establisher of the TBIT handler has CMKRNL privilege.
 - If no handler specified, transfer control to system handler, EXESTBIT
 - Handler address will be specified in fixed location in P1 space (CTLSGL_SITESPEC). That way, separate TBIT handlers can be declared by each process in the system.
 - Transfer control to the TBIT handler with JMP instruction

- Modify the SCB to point to the dispatch code in the allocated block for TBIT exceptions

USER-WRITTEN SYSTEM SERVICES

- SCB address at EXESGL_SCB
- Assume TBIT offset is SCB_L_TBIT

In addition, it is necessary to test for the presence of XDELTA (since XDELTA uses the TBIT entry in the SCB). If XDELTA is present, you do not want to load your dispatcher. In such a case, exit the program and indicate that XDELTA is present.

- To test for the presence of XDELTA, examine the RPB\$V DEBUG bit in the RPB\$L_BOOTR5 offset in the Restart Parameter Block. When set, XDELTA is present.
- EXESGL_RPB contains the address of the RPB.

Since you later want to be able to restore the SCB, and deallocate the block of nonpaged pool containing the dispatcher, store the address of the block in EXESGL_SITESPEC. Before using this location, however, test to make sure it is not being used (i.e., it contains a 0). If it is being used, exit the program.

- Note that in order to synchronize access to the EXESGL_SITESPEC location, you must execute at IPL SYNCH from the time you first test the location, to the time it is modified.

The file TBITDEF.MAR contains several macros that may be of use to you, including:

```
_TBITDEF    Defines offsets in TBIT dispatcher block:

             TBIT_W_SIZE      for size of block
             TBIT_B_TYPE      for type of block
             TBIT_B_CODE      code is loaded beginning here
             TBIT_L_HANDLER   address of new TBIT dispatcher
                               (used for consistency checks)

_SCBDEF     Defines offset of TBIT vector (SCB_L_TBIT) in
             SCB.

_DYNDEF     Defines type code for TBIT dispatcher block
             (DYN_K_TBIT).
```

The file TBITMSG.MSG defines some specific error codes that you may wish to report:

```
TBIT__VECINUSE   Site specific longword already in use

TBIT__INSFMEM    Nonpaged pool allocation failed

TBIT__XDELTA     Dispatcher cannot be installed in
                 system with XDELTA
```

USER-WRITTEN SYSTEM SERVICES

LOADTBIT Program Solution (Page 1 of 5)

LOADTBIT
02

Create new path to TBIT handler

14-MAY-1982 18:22:43 VAX-11 Macro V03-00
14-MAY-1982 18:05:30 WORK:CMUIZINIEKS.SYSPRG.USSJ1

```

0000 1 :                                LOADTBIT.MAR
0000 2 :
0000 3 :      .title loadtbit      Create new path to TBIT handler
0000 4 :      .ident "V02"
0000 5 :++
0000 6 :
0000 7 :      Facility:
0000 8 :
0000 9 :      This code is dynamically patched into the running executive.
0000 10 :      It introduces a new dispatch method to handle TBIT exceptions,
0000 11 :      similar to the method used to handle CHMS, CHMU, and
0000 12 :      compatibility mode exceptions.
0000 13 :
0000 14 :      This program is designed to provide an environment for the
0000 15 :      user-written system service USSDCLTBIT, also included as a part
0000 16 :      of this example.
0000 17 :
0000 18 :      Environment:
0000 19 :
0000 20 :      This program must execute in kernel mode in order to write
0000 21 :      protected locations. IPL is raised to 7 (SYNCH) to synchronize
0000 22 :      access to the site specific location EXESGL_SITESPEC.
0000 23 :
0000 24 :      Author:
0000 25 :
0000 26 :      Larry Kenah
0000 27 :
0000 28 :      Creation Date:
0000 29 :
0000 30 :      19 May 1980
0000 31 :
0000 32 :      Revisions:
0000 33 :
0000 34 :      27 Mar. 1982                                Vik Muiznieks
0000 35 :
0000 36 :      Removed patch to EXESRUNDOWN
0000 37 :      Added synchronization for EXESGL_SITESPEC
0000 38 :
0000 39 :      12 June 1980                                Have new exception service routine execute
0000 40 :      in kernel mode, simplifying stack problems.
0000 41 :
0000 42 :--
0000 43 :
0000 44 :      Include Files:
0000 45 :
0000 46 :      _dyndef                                :define type field identifier
0000 47 :      _spldef                                :define ipl constants
0000 48 :      _spsldef                               :define fields in PSL
0000 49 :      _rpbdef                                :define offsets into Restart Param
0000 50 :      _scbdef                                :define system control block offset
0000 51 :      _tbitdef                               :this macro is defined especially
0000 52 :
0000 52 :      ; the set of programs in this exam

```

USER-WRITTEN SYSTEM SERVICES

LOADTBIT Program Solution (Page 2 of 5)

Create new path to TBIT handler
add fast dispatch method

14-MAY-1982 18:22:43 VAX-11 Macro V03-00 Page 2
14-MAY-1982 18:05:30 WORK:CMUIZNIERS.SYSPRG.USSJLOADTBIT(1)

```

0000 54      .subtitle      add fast dispatch method
0000 55
0000 56 :++
0000 57 :
0000 58 :   Functional Description:
0000 59 :
0000 60 :   The kernel mode procedure beginning at label 10$ allocates
0000 61 :   a block of nonpaged pool and copies the new dispatcher for
0000 62 :   TBIT exceptions to that block. The last instruction of the new
0000 63 :   dispatcher is a JMP to the normal dispatcher so that continuity
0000 64 :   is established.
0000 65 :
0000 66 :   If no debugger support was selected at initialization time,
0000 67 :   the address of the new dispatcher is simply moved into the
0000 68 :   TBIT vector in the system control block.
0000 69 :
0000 70 :   If debugger support was selected, then XDELTA has already
0000 71 :   intercepted the system control block vector. This dispatcher
0000 72 :   could steal the SCB vector contents and JMP to XDELTA as its
0000 73 :   last instruction but that hierarchy (first this, then XDELTA,
0000 74 :   and finally the normal dispatch path) is out of order. Since
0000 75 :   this new dispatch path exists on a per process basis, it should
0000 76 :   execute after XDELTA executes. However, due to the difficulty
0000 77 :   in patching XDELTA with its PC relative references to EXESTBIT,
0000 78 :   this program does nothing (except indicate an error) when
0000 79 :   XDELTA is included in the system.
0000 80 :
0000 81 :   Implicit Output:
0000 82 :
0000 83 :   1. A block of pool is allocated and a new TBIT exception
0000 84 :   dispatcher is loaded into the block of pool.
0000 85 :
0000 86 :   2. The new dispatcher is plugged into the executive by modifying
0000 87 :   the TBIT vector in the SCB if no XDELTA support was included
0000 88 :
0000 89 :   Completion Codes:
0000 90 :
0000 91 :   R0 low bit set => success
0000 92 :
0000 93 :           SS$_NCRMAL      Normal Successful Completion
0000 94 :
0000 95 :   R0 low bit clear => failure
0000 96 :
0000 97 :           TBIT__VECIINUSE  The site specific longword is already being used
0000 98 :
0000 99 :           TBIT__INSMEM    Insufficient dynamic memory is available. (This
0000 100 :                          status is only returned if the current process
0000 101 :                          has disabled resource wait mode.)
0000 102 :
0000 103 :           TBIT__XDELTA    The system included XDELTA at initialization
0000 104 :                          time. This program does nothing in this case.
0000 105 :
0000 106 :--

```

USER-WRITTEN SYSTEM SERVICES

LOADTBIT Program Solution (Page 3 of 5)

```

DADTBIT
02
                                Create new path to TBIT handler      14-MAY-1982 18:22:43  VAX-11 Macro V03-00
                                add fast dispatch method             14-MAY-1982 18:05:30  WORK:CMUIZIEKS.SYSPRG.USSJ

00000000 108      .psect nonshared_data  pic,nonexe,long
0000      109
00000029 0000 110 range: .address lock_begin
000000C3 0004 111      .address lock_end
0008      112
00000000 113      .psect tbit_code          pic,shr,nowrt
0000      114
0000 0000 115      .entry begin,0
0002      116      scwkrnl_s      -
0002      117      routin=10$
04 0011 118      ret
0012      119
003C 0012 120 10$: .word  ^m<r2,r3,r4,r5>          ;save volatile registers
0014      121      slkset_s  inadr=range      ;lock pages in working set
01 50  E8 0025 122      blbs  r0,lock_begin      ;continue on success
06 0028 123      ret
0029      124 lock_begin:
0029      125      setipl  $iplt_synch          ;synchronize access to sitespec
00000000 GF 05 002C 126      tstl  g^exe$gl_sitespec      ;see if site specific pointer
0032      127      ; is already being used
5C 12 0032 128      bnequ  inuse          ;error if already loaded
0034      129
51 00000000 GF D0 0034 130      movl  g^exe$gl_rpb,r1          ;get address of restart parameter
55 30 A1 D0 0038 131      movl  rpbsl_bootr5(r1),r5      ;r5 contains bootstrap flags
58 55 01 E0 003F 132      bbs  $rpb$debug,r5,xdelta      ;do nothing if XDELTA included
0043      133
51 0000001E 8F D0 0043 134      movl  $(<<handler_end-handler_start>+12),r1 ;size of block includ
004A      135      ; 12 byte header
00000000 GF 16 004A 136      jsb  g^exe$alononpaged
53 50  E9 0050 137      blbc  r0,nopool
00000000 GF 52 D0 0053 138      movl  r2,g^exe$gl_sitespec      ;save address of block for program
005A      139      ; that unloads this dispatcher
7E 0C A2 9E 005A 140      movab tbit_b_code(r2),-(sp)      ;save address of beginning of code
04 88 005E 141      pushr $^m<r2>          ; and address of beginning of bloc
82 7C 0060 142      clrq  (r2)>          ;clear out first 8 bytes of header
82 51 80 0062 143      movw  r1,(r2)+          ;save size of block for deallocati
82 7F 8F 98 0065 144      movzwb $dyn_k_tbit,(r2)>      ;store block type and clear spare
0069      145
0069      146 ; r2 now points to the portion of the block that is to be loaded with
0069      147 ; the new dispatch code. The address of the beginning of the block
0069      148 ; and the address of the portion of the block where the code is to be
0069      149 ; loaded are stored on the stack. No other register contents need to be
0069      150 ; saved across the MOVPC3 instruction.
0069      151
62 000000B1 EF 0012 8F 28 0069 152      movc3  $(<handler_end-handler_start>,handler_start,(r2)
0C 8A 0073 153      poor  $^m<r2,r3>          ;restore address of beginning of l
0075 154      ; into r2 and address of code into
51 00000000 GF D0 0075 155      movl  g^exe$gl_scb,r1          ;otherwise, simply modify SCB vec
28 A1 00C0 C3 9E 007C 156      movab tbit-handler_start(r3),scb_l_tbit(r1) ;load address of nei
0082      157      ; dispatcher into SCB
62 0000 C3 9E 0082 158      movab tbit-handler_start(r3),tbit_l_handler(r2) ; and also save
0087      159      ; of the header longwords in the
0087      160      setipl  #0          ;safe to allow process deletion n
50 0000 8F 3C 008A 161      movzwl $sss_normal,r0          ;indicate success
04 008F 162      ret
0090      163
0090      164 inuse:          ;site specific longword is already

```

USER-WRITTEN SYSTEM SERVICES

LOADTBIT Program Solution (Page 4 of 5)

```
                                14-MAY-1982 18:22:43 VAX-11 Macro V03-00 Page 4
                                14-MAY-1982 18:05:30 WORK:[MUIZNIKES.SYSPRG.USSJLOADTBI(1)

Create new path to TBIT handler
add fast dispatch method

50 000000C0*8F 00 0090 165      movl   #tbit__vecinuse,r0
                   0097 166      setipl #0
                   04 009A 167      ret
                   0095 168

50 00000000*8F 00 0098 169 xdelta: movl   #tbit__xdelta,r0      ;XDELTA was included at bootstrap time.
                   00A2 170      setipl #0
                   04 00A5 171      ret      ;pass back failure code
                   00A6 172
                   00A6 173 ncpool: ;insufficient nonpaged pool space
                   00A6 174      setipl #0
50 00000000*8F C0 00A9 175      movl   #tbit__insfmem,r0
                   04 00B0 176      ret
                   00B1 177
```

USER-WRITTEN SYSTEM SERVICES

LOADTBIT Program Solution (Page 5 of 5)

JADTBIT
32

Create new path to TBIT handler 14-MAY-1982 18:22:43 VAX-11 Macro V03-00
dispatch code for TBIT exceptions 14-MAY-1982 18:05:30 WORK:EMUIZNIEKS.SYSPRG.USS31

```

00B1 179          .sbttl dispatch code for TBIT exceptions
00B1 180
00B1 181 :++
00B1 182 :
00B1 183 : Functional Description:
00B1 184 :
00B1 185 : This code is designed to be the first code that executes following
00B1 186 : a TBIT exception (if no XDELTA). It checks whether a special TBIT
00B1 187 : handler has been declared by the process and, if so, dispatches to
00B1 188 : that dispatcher with only the exception PC and PSL on the stack.
00B1 189 :
00B1 190 : This design is closely modeled on the existing rapid dispatchers
00B1 191 : for change mode and compatibility mode exceptions. Such dispatcher:
00B1 192 : can be accessed quickly without the overhead of searching for
00B1 193 : addresses of condition handlers in exception vectors and in call fr
00B1 194 : Handlers accessed with this method are not procedures and can only
00B1 195 : be written in a language that allows J58 like linkage.
00B1 196 :
00B1 197 : The method of getting this dispatch scheme into the system imposes
00B1 198 : several constraints on this code.
00B1 199 :
00B1 200 : 1. The code must be position independent since it is simply
00B1 201 : copied into nonpaged pool with no attempt at dynamic relocation
00B1 202 :
00B1 203 : 2. Any registers used by this routine must first be saved.
00B1 204 : (None is currently used.) In addition,
00B1 205 : stack usage must be watched carefully.
00B1 206 :
00B1 207 : 3. The initial design of this dispatch method transfers control
00B1 208 : to the user specified dispatcher in kernel mode. One improvement
00B1 209 : which would slow up the dispatching and add stack concerns,
00B1 210 : would be to change to the access mode in which the TBIT
00B1 211 : exception occurred.
00B1 212 :
00B1 213 :--
00B1 214
00B1 215 handler_start:          :address of beginning of block of c
00B1 216                          : that is loaded into nonpaged pool
00B1 217
00B1 218 tbit:                :address of tbit handler
00000000*GF 00 00B1 219          pushl  G^ctl$ql_sitiespec  :store contents on stack
02 13 00B7 220          beql  10$          :if empty, no dispatcher
9E 17 00B9 221          jmp    2(sp)+       :transfer to user written dispatch
00B9 222
00B8 223 10$:            tstl  (sp)+       :clean stack
00000000*GF 17 00B0 224          jmp    .g^exestbit    :and transfer to VMS TBIT exceptior
00C3 225
00C3 226 handler_end:      :end of code that is being loaded
00C3 227                          : into nonpaged pool
00C3 228 lock_end:
00C3 229
00C3 230          .end    begin

```


USER-WRITTEN SYSTEM SERVICES

TBITMSG.MSG File

TBITMSG.MSG

This file defines specific error messages for the
TBIT dispatching programs

```
.title tbit_handler    Alternate TBIT dispatcher
.facility      TBIT,1/prefix=TBIT_
.severity     SEVERE

VECINUSE      <Site specific longword already in use>
INSPREM      <Nonpaged pool allocation failed>
XDELTA       <Dispatcher cannot be installed in system with XDELTA>
NOMANDLER    <Site specific longword contains no data block address>
BADHANDLER   <Data in block is inconsistent with LOADTBIT input>

.end
```

USER-WRITTEN SYSTEM SERVICES

PART 2 - User-Written System Service To Declare TBIT Handler (TBITDISP.MAR)

In order to incorporate a system service to declare a TBIT handler, you will first need to:

1. Copy SYS\$EXAMPLES:USSDISP.MAR to your directory.
2. Edit that file to remove the sample system services, the supplied rundown routine, and the executive mode dispatcher.
3. Add a DEFINE_SERVICE macro for your system service:
 - Call it USSDCLTBIT
 - You want 2 parameters:
 - First is address of TBIT handler to be established (or 0 to disable previous handler)
 - Second is address of longword to receive previous handler address, if any (this parameter is optional)
 - Service should execute in kernel mode
4. Include the offset to a RUNDOWN routine in the privileged library vector
5. Add code for USSDCLTBIT system service:
 - Test for CMKRNL privilege, since handler will execute in kernel mode (unless in Part 1 your dispatcher changed access mode).
 - Could test for presence of TBIT dispatcher loaded in Part 1.
 - Not really necessary since VMS does not use CTL\$GL_SITESPEC location.
 - However, TBIT handler declared will never be entered for a TBIT exception until program from Part 1 is run to establish TBIT dispatcher.
 - If user requested address of previous handler, make sure user can write to the address specified.

USER-WRITTEN SYSTEM SERVICES

- Copy address of handler being declared to location CTL\$GL_SITESPEC.
6. Add code for RUNDOWN routine
- Routine entered at image rundown time (with JSB instruction)
 - Executes in kernel mode
 - Should clear CTL\$GL_SITESPEC location, so next image not effected by handler declared in previous image
 - Should exit with RSB instruction

USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 1 of 6)

Declare TBIT Handler

14-MAY-1982 18:22:56 VAX-11 Macro V03-00 Page 1
 14-MAY-1982 17:54:52 WORK:CMUIZNIKS.SYSPRG.USSJTBITDIS(1)

```

0000 1 :                               TBITDISP.MAR
0000 2 :
0000 3 :   This file contains both an edited user-written system service
0000 4 :   dispatcher (from SYS$EXAMPLES:USSDISP.MAR) and the system
0000 5 :   service code itself for the load TBIT dispatcher system service.
0000 6 :
0000 7 :   Macro Definitions
0000 8 :
0000 9 :   DEFINE_SERVICE - A macro to make the appropriate entries in several
0000 10 :   different PSECTS required to define an EXEC or KERNEL
0000 11 :   mode service. These include the transfer vector,
0000 12 :   the case table for dispatching, and a table containing
0000 13 :   the number of required arguments.
0000 14 :
0000 15 :   DEFINE_SERVICE Name,Number_of_Arguments,Mode
0000 16 :
0000 17 :   .MACRO DEFINE_SERVICE,NAME,NARG=0,MODE=KERNEL
0000 18 :   .PSECT $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
0000 19 :   .ALIGN QUAD           : Align entry points for speed and style
0000 20 :   .TRANSFER           NAME           : Define name as universal symbol for entry
0000 21 :   .MASK NAME           : Use entry mask defined in main routine
0000 22 :   .IF IDN MODE,KERNEL
0000 23 :   CHMK #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and execute
0000 24 :   RET           : Return
0000 25 :   KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter
0000 26 :
0000 27 :   .PSECT KERNEL_NARG,BYTE,NOWRT,EXE,PIC
0000 28 :   .BYTE NARG           : Define number of required arguments
0000 29 :
0000 30 :   .PSECT USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
0000 31 :   .WORD 2+NAME-KCASE_BASE : Make entry in kernel mode CASE table
0000 32 :
0000 33 :   .IF
0000 34 :   CHME #<ECODE_BASE+EXEC_COUNTER> ; Change to executive mode and execute
0000 35 :   RET           : Return
0000 36 :   EXEC_COUNTER=EXEC_COUNTER+1     : Advance counter
0000 37 :
0000 38 :   .PSECT EXEC_NARG,BYTE,NOWRT,EXE,PIC
0000 39 :   .BYTE NARG           : Define number of required arguments
0000 40 :
0000 41 :   .PSECT USER_EXEC_DISP1,BYTE,NOWRT,EXE,PIC
0000 42 :   .WORD 2+NAME-ECASE_BASE : Make entry in exec mode CASE table
0000 43 :   .ENDC
0000 44 :   .ENOM DEFINE_SERVICE
0000 45 :
0000 46 :   Equated Symbols
0000 47 :
0000 48 :
0000 49 :   $PHODEF           : Define process header offsets
0000 50 :   $PLVDEF           : Define PLY offsets and values
0000 51 :   $PRDEF           : Define processor register numbers
0000 52 :
0000 53 :   Initialize counters for change mode dispatching codes
0000 54 :
00000000 0000 55 KERNEL_COUNTER=0           : Kernel code counter
00000000 0000 56 EXEC_COUNTER=0           : Exec code counter
0000 57
    
```

USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 2 of 6)

```

SSOCLTBIT
32
Declare TBIT Handler
14-MAY-1982 18:22:56 VAX-11 Macro V03-00
14-MAY-1982 17:54:52 WORK:CMUIZNIKES.SYSPRG.USSJT

0000 59 :
0000 59 :      Don Storage
0000 60 :
00000000 61      .PSECT  KERNEL_NARG, BYTE, NOWRT, EXE, PIC
0000 62  KERNEL_NARG:      : Base of byte table containing the
0000 63      :      : number of required arguments.
00000000 64      .PSECT  EXEC_NARG, BYTE, NOWRT, EXE, PIC
0000 65  EXEC_NARG:      : Base of byte table containing the
0000 66      :      : number of required arguments.
0000 67
0000 68
0000 69      DEFINE_SERVICE  USSOCLTBIT, 2, KERNEL      : Declare TBIT system servi
0002 70
0002 71 :
0002 72 : The base values used to generate the dispatching codes should be negative
0002 73 : user services and must be chosen to avoid overlap with any other privileg
0002 74 : shareable images that will be used concurrently. Their definition is
0002 75 : deferred to this point in the assembly to cause their use in the precedin
0002 76 : macro calls to be forward references that guarantee the size of the chang
0002 77 : mode instructions to be four bytes. This satisfies an assumption that is
0002 78 : made by for services that have to wait and be retried. The PC for retryi
0002 79 : the change mode instruction that invokes the service is assumed to be 4 b
0002 80 : less than that saved in the change mode exception frame. Of course, the
0002 81 : service routine determines whether this is possible.
0002 82 :
FFFFFE0C 0002 83  KCODE_BASE=-500      : Base CHMK code value for these se
FFFFFE0C 0002 84  ECODE_BASE=-500      : Base CHME code value for these se
0002 85 :
0002 86 :
00000000 87      .PSECT  USER_SERVICES, PAGE, VEC, PIC, NOWRT, EXE
0000 88
00000001 0000 89      .LONG  PLY$C_TYP_CMOD      : Set type of vector to change mode
00000000 0004 90      .LONG  SY$K_VERSION      : Identify system version
00000005 0008 91      .LONG  KERNEL_DISPATCH-   : Offset to kernel mode dispatcher
00000000 000C 92      .LONG  0                  : Offset to executive mode dispatch
00000024 0010 93      .LONG  RUNDOWN-         : Offset to RUNDOWN routine
00000000 0014 94      .LONG  0                  : Reserved.
00000000 0018 95      .LONG  0                  : No RMS dispatcher
00000000 001C 96      .LONG  0                  : Address check - PIC image

```

USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 3 of 6)

IIT

Declare TBIT Handler
Kernel Mode Dispatcher

14-MAY-1982 18:22:56 VAX-11 Macro V03-00
14-MAY-1982 17:54:52 WORK:CMUIZINIEKS.SYSPRG.USSJTBITDIS(1)

Page 3

```

0020 98 .SBTTL Kernel Mode Dispatcher
0020 99 :++
0020 100 : Input Parameters:
0020 101 :
0020 102 : (SP) - Return address if bad change mode value
0020 103 :
0020 104 : R0 - Change mode argument value.
0020 105 :
0020 106 : R4 - Current PCB Address. (Therefore R4 must be specified in all
0020 107 : register save masks for kernel routines.)
0020 108 :
0020 109 : AP - Argument pointer existing when the change
0020 110 : mode instruction was executed.
0020 111 :
0020 112 : FP - Address of minimal call frame to exit
0020 113 : the change mode dispatcher and return to
0020 114 : the original mode.
0020 115 :--
00000000 116 .PSECT USER_KERNEL_DISP0,BYTE,NOVRT,EXE,PIC
0000 117 KACCVIO: ; Kernel access violation
50 0000*8F 3C 0000 118 MOVZWL $$$_ACCVIO,R0 ; Set access violation status code
04 0005 119 RET ; and return
0006 120 KINSFARG: ; Kernel insufficient arguments.
50 0000*8F 3C 0006 121 MOVZWL $$$_INSFARG,R0 ; Set status code and
04 0008 122 RET ; return
05 000C 123 KNOTME: RSB ; RSB to forward request
0000 124
0000 125 KERNEL_DISPATCH:: ; Entry to dispatcher
51 01F4 C0 9E 0000 126 MOVAB W^<KCODE_BASE(R0),R1 ; Normalize dispatch code value
F8 19 0012 127 BLSS KNOTME ; Branch if code value too low
01 51 81 0014 128 CMPW R1,#KERNEL_COUNTER ; Check high limit
F3 1E 0017 129 SGEQU KNOTME ; Branch if out of range
0019 130 :
0019 131 : The dispatch code has now been verified as being handled by this dispatcher,
0019 132 : now the argument list will be probed and the required number of arguments
0019 133 : verified.
0019 134 :
51 0000*CF41 9A 0019 135 MOVZBL W^<KERNEL_MARG(R1),R1 ; Get required argument count
51 00000004 9F41 0E 001F 136 MOVAL @#4(R1),R1 ; Compute byte count including arg count
0027 137 IFNORD R1,(AP),KACCVIO ; Branch if arglist not readable
01F4*CF40 6C 91 002D 138 CMPB (AP),W^<KERNEL_MARG-KCODE_BASE>(R0) ; Check for required number
01 1F 0033 139 BLSSU KINSFARG ; of arguments
50 AF 0035 140 CASEW R0,- ; Case on change mode
0037 141 - ; argument value
0037 142 @KCODE_BASE,- ; Base value
00  FE0C 8F 0037 143 @<KERNEL_COUNTER-1> ; Limit value (number of entries)
0038 144 KCASE_BASE: ; Case table base address for DEFINE_SERVICE
0038 145 :
0038 146 : Case table entries are made in the PSECT USER_KERNEL_DISP1 by
0038 147 : invocations of the DEFINE_SERVICE macro. The three PSECTS,
0038 148 : USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-time.
0038 149 :
00000000 150 .PSECT USER_KERNEL_DISP2,BYTE,NOVRT,EXE,PIC
05 0000 151 RSB ; Return to reject out of
0001 152 ; range value
0001 153 .title ussdcltbit Declare TBIT Handler
0001 154 .ident *V02*

```

USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 4 of 6)

```

SSDCLTBIT
02
Declare TBIT Handler
Kernel Mode Dispatcher
14-MAY-1982 18:22:56 VAX-11 Macro V03-00
14-MAY-1982 17:54:52 WORK:CNUIZNIKES.SYSPRG.USSJ

0001 155
0001 156
0001 157 ;++
0001 158 ;
0001 159 ; Facility:
0001 160 ;
0001 161 ; This procedure allows a privileged process to declare a TBIT
0001 162 ; handler that is accessed quickly, bypassing the usual VMS
0001 163 ; exception dispatch mechanism. The procedure can either
0001 164 ; be called through the SYS$CHKRNL system service or installed
0001 165 ; as a user written system service.
0001 166 ;
0001 167 ; Environment:
0001 168 ;
0001 169 ; Kernel mode procedure that alters the CTL$GL_SITESPEC location.
0001 170 ;
0001 171 ; Author:
0001 172 ;
0001 173 ; Larry Kenah
0001 174 ;
0001 175 ; Creation Date:
0001 176 ;
0001 177 ; 4 August 1980
0001 178 ;
0001 179 ; Revisions:
0001 180 ;
0001 181 ; Vik Muiznieks 8 Apr 1982
0001 182 ;
0001 183 ; Added RUNDOWN routine, and updated privileged library
0001 184 ; vector
0001 185 ;
0001 186 ; Synchronized access to EXE$GL_SITESPEC
0001 187 ;--
0001 188 ;
0001 189 ; Include Files:
0001 190 ;
0001 191
0001 192
0001 193 sprvdef ;define privilege bits
0001 194 spcbdef ;define offsets into PCB
0001 195
0001 196 ;
0001 197 ; Argument List Offset Definitions
0001 198 ;
0001 199
00000004 0001 200 addres = 4 ;Address of new TBIT handler
00000008 0001 201 prvhd = 8 ;Address of longword to receive
0001 202 ; old handler address
0001 203

```


USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 5 of 6)

```

T
Declare TBIT Handler          14-MAY-1982 18:22:56 VAX-11 Macro V03-00          Page 5
Declare TBIT Handler          14-MAY-1982 17:54:52 WORK:CMUIZNIEKS.SYSPRG.USSJTBITDIS(1)

0001 205      .subtitle      Declare TBIT Handler
0001 206
0001 207 :=
0001 208 : This procedure allows a process to specify a TBIT handler that will
0001 209 : immediately receive control when a TBIT exception occurs, bypassing
0001 210 : the normal VMS exception dispatching mechanism.
0001 211 :
0001 212 : To avoid some tricky stack problems, the initial implementation of this
0001 213 : service executes in kernel mode. For this reason, the procedure can
0001 214 : either be called by means of the Change Mode to Kernel system service,
0001 215 : or the procedure can be made a part of a privileged shareable image,
0001 216 : accessed as a user written system service.
0001 217 :
0001 218 : Input Parameters:
0001 219 :
0001 220 :     adres(ap)      Address of handler that is being declared
0001 221 :
0001 222 :     prvhd(ap)      Address of longword that will receive the
0001 223 :                   address of a previous handler, if one existed.
0001 224 :
0001 225 : Implicit Input:
0001 226 :
0001 227 :     r4              PCB address of caller
0001 228 :
0001 229 :     ctlsq1_sitespec Location in which to store handler address.
0001 230 :                   The symbol is defined (and unused) by VMS.
0001 231 :
0001 232 : Note that this procedure does not test whether the fast TBIT
0001 233 : dispatcher has been installed. Such a test could easily be added
0001 234 : to this service.
0001 235 :
0001 236 : No harm is done if the P1 space location is loaded without
0001 237 : the dispatcher being installed because VMS does not use
0001 238 : the P1 space location in question. However, the TBIT handler
0001 239 : declared with this procedure will never be called until the TBIT
0001 240 : dispatcher is installed.
0001 241 :
0001 242 : Implicit Output:
0001 243 :
0001 244 :     ctlsq1_sitespec is loaded with the address of the new handler.
0001 245 :
0001 246 : Status Codes:
0001 247 :
0001 248 :     R0 low bit set implies success.
0001 249 :
0001 250 :     R0 low bit clear implies an error.
0001 251 :
0001 252 :     $$$_ACCVID     Previous handler address cannot be
0001 253 :                   written by caller's access mode
0001 254 :
0001 255 :     $$$_NCPRIV     The caller does not have CMKRNL privilege
0001 256 :-
0001 257
00000000 259      .osect  tbit_code      pic,shr,noert
0000 259
001C 0000 260      .entry  ussdcltbit,*(r2,r3,r4)
0002 261

```

USER-WRITTEN SYSTEM SERVICES

TBITDISP.MAR File (Page 6 of 6)

```

SSDCLTBIT
02
                                14-MAY-1982 18:22:56 VAX-11 Macro V03-00
                                14-MAY-1982 17:54:52 WORK:CMUIZNIKS.SYSPRG.USSJ

Declare TBIT Handler
Declare TBIT Handler

    0002 262      ifnpriv cmkrnl,nopriv      ;does caller have CMKRNL privilege
51 09 AC D0 0007 263      movl prvhd(ap),r1      ;get previous handler address
    0008 264      beql 10$      ;skip if none
    000D 265      ifnoert $4,(r1),accvio      ;is it accessible?
61 00000000^GF D0 0013 266      movl G^ctl$gl_sitespec,(r1) ;OK, so write old contents
00000000^GF 04 AC D0 001A 267 10$:      movl adres(ap),G^ctl$gl_sitespec ; and store new handler adre
50 0000^8F 3C 0022 268      movzel $sst_normal,r0      ;indicate success
    0027 269      ret      ;and return
    0028 270
    0028 271 nopriv:
50 0000^8F 3C 0028 272      movzel $sst_nopriv,r0      ;caller does not have CMKRNL
    002D 273      ret
    002E 274
    002E 275 accvio:
50 0000^8F 3C 002E 276      movzel $sst_accvio,r0      ;previous handler address inaccess
    0033 277      ret
    0034 278
    0034 279 :      This entry point is called at image rundown time, by the rundown
    0034 280 :      code, in KERNEL mode. Control is transferred via JSB.
    0034 281 :
    0034 282 :      The routine clears the location containing the TBIT handler
    0034 283 :      address, so the next image can start "fresh".
    0034 284
    0034 285 RUNDOWN:
00000000^GF 04 0034 286      clr1 G^ctl$gl_sitespec      ; Entered via JSB
    003A 287      rsb      ; clear handler address for next i
    003B 288      .end      ; return to rundown code

```

USER-WRITTEN SYSTEM SERVICES

PART 3 - Unloading TBIT Handler And Restoring SCB (REMOVTBIT.MAR)

This program needs to perform the following operations:

1. Include the `_DYNDEF` and `_SCBDEF` macros discussed in Part 1.
2. Test that the loader program has been run:
 - Avoid synchronization problems by running at IPL SYNCH
 - Examine `EXESGL_SITESPEC` to find TBIT dispatcher address
 - Check type field of TBIT block for consistency. Should find `DYN_K_TBIT`.
3. Restore SCB to original state
 - Use `EXESGL_SCB` as SCB base address
 - Use `SCB_L_TBIT` as offset to TBIT vector
 - Replace TBIT vector with address of `EXESTBIT`
4. Deallocate TBIT dispatcher block
5. Clear `EXESGL_SITESPEC`

You may want to use the following messages defined in the `TBITMSG.MSG` file:

`TBIT_NOHANDLER` Site specific longword contains no data
block address

`TBIT_BADHANDLR` Data in block is inconsistent with
`LOADTBIT` input

USER-WRITTEN SYSTEM SERVICES

REMOVTBIT.MAR File (Page 1 of 3)

REMOVE_TBIT
32

Remove special TBIT exception handler 14-MAY-1982 18:22:50 VAX-11 Macro V03-00
27-MAR-1982 17:10:13 WORK:CMUIZNIIEKS.SYSRPG.US

```

0000 1 : REMOVTBIT.MAR
0000 2 :
0000 3 : .title remove_tbit Remove special TBIT exception handler
0000 4 : .ident "V02"
0000 5 :
0000 6 :
0000 7 : **
0000 8 :
0000 9 : Facility:
0000 10 :
0000 11 : This program unloads the special TBIT exception handler that was
0000 12 : loaded by the LOADTBIT program.
0000 13 :
0000 14 : Environment:
0000 15 :
0000 16 : This program must execute in kernel mode in order to write
0000 17 : protected locations.
0000 18 :
0000 19 :
0000 20 : Author:
0000 21 :
0000 22 : Larry Kenah
0000 23 :
0000 24 : Creation Date:
0000 25 :
0000 26 : 19 May 1980
0000 27 :
0000 28 :
0000 29 : Revisions:
0000 30 :
0000 31 : Vik Muiznieks 27-May-1982
0000 32 :
0000 33 : Remove patches to SYSRUNDWN
0000 34 : Add synchronization for EXESGL_SITESPEC
0000 35 :
0000 36 :--
0000 37 :
0000 38 :
0000 39 : Include Files:
0000 40 :
0000 41 : _dyndef ;define type field identifier
0000 42 : $ipldef ;define ipl constants
0000 43 : $rpbdef ;define offsets into Restart Par
0000 44 : _scbdef ;define system control block off
0000 45 : _tbitdef ;this macro is defined especiall
0000 46 : ; the set of programs in this ex
0000 47 :
0000 48 :
0000 49 : Local Symbols:
0000 50 :

```

USER-WRITTEN SYSTEM SERVICES

REMOVETBIT.MAR File (Page 2 of 3)

```

T
Remove special TBIT exception handler      14-MAY-1982 18:22:50 VAX-11 Macro V03-00      Page 2
Remove special TBIT dispatcher            27-MAR-1982 17:10:13 WORK:CMUIZNIKS.SYSPRG.USSJREMOVETB(1)

0000 52      .subtitle      Remove special TBIT dispatcher
0000 53
0000 54 :++
0000 55 :
0000 56 : Functional Description
0000 57 :
0000 58 : The kernel mode procedure beginning at address 10$ locates the
0000 59 : address of the new dispatcher, performs a consistency check, and
0000 60 : removes the handler from the system.
0000 61 :
0000 62 : 1. The SCB entry for the TBIT exception is restored to
0000 63 : its original contents, the address of the routine EXE$TBIT.
0000 64 :
0000 65 : 2. The block of nonpaged pool that held the dispatcher and
0000 66 : other information is deallocated.
0000 67 :
0000 68 : Completion Codes:
0000 69 :
0000 70 : R0 low bit set => success
0000 71 :
0000 72 : SSS_NORMAL Normal Successful Completion
0000 73 :
0000 74 : R0 low bit clear => failure
0000 75 :
0000 76 : TBIT__NOHANDLER There is no handler currently in the system.
0000 77 : This error occurs if the site specific longword
0000 78 : contains a zero.
0000 79 :
0000 80 : TBIT__BADHANDLR The data block pointed to by EXE$GL_SITESPEC
0000 81 : contains an inconsistency. This is either the
0000 82 : wrong value in the type field, or a
0000 83 : handler address that does not agree with the
0000 84 : contents of the SCB vector.
0000 85 :
0000 86 :--
0000 87
00000000 88      .psect nonshared_data pic,noexe,long
0000 89
00000029* 0000 90 range: .address lock_begin
0000007F* 0004 91      .address lock_end
0000 92
00000000 93      .psect tbit_code pic,shr,noerr
0000 94
0000 0000 95      .entry remove_tbit,0
0002 0002 96      %cmkrnl_s -
0002 0002 97      routin=10$
04 0011 98      ret
0012 99
003C 0012 100 10$: .word ^m<r2,r3,r4,r5>      ;save volatile registers
0014 101      %lkuset_s inadr=range      ;lock pages in working set
01 50 E8 0025 102      bibs r0,lock_begin      ;continue on success
04 0028 103      ret
0029 104 lock_begin:
0029 105      setipl %ipl$_synch      ;prevent process deletion in the middle
002C 106      ;synchronize access to EXE$GL_SITESPEC
52 00000000*GF D0 002C 107      movl g^exesgl_sitespec,r2      ;get address of tbit block.
34 13 0033 108      beql nohandler

```

USER-WRITTEN SYSTEM SERVICES

REMOVTBIT.MAR File (Page 3 of 3)

```

REMOVE_TBIT
/02
Remove special TBIT exception handler      14-MAY-1982 18:22:50 VAX-11 Macro V03-00
Remove special TBIT dispatcher             27-MAR-1982 17:10:13 WORK:CMUIZINIEKS.SYSPRG.USS

    7F 8F 0A A2 91 0035 109      cmnb   tbit_b_type(r2),#dyn_k_tbit ;Correct type?
    38 12 003A 110      bnequ  inconsistent
51 00000000*GF 00 003C 111      movl   g^exe$gl_scb,r1      ;get SCB address
    62 28 A1 01 0043 112      cmpl   scb_l_tbit(r1),tbit_l_handler(r2) ;are handler addresses
    2E 12 0047 113      bnequ  inconsistent
    0049 114
    0049 115 : Consistency checks have succeeded. Now remove the handler from the sys
    0049 116
28 A1 00000000*GF DE 0049 117      movl   g^exe$tbit,scb_l_tbit(r1) ;Single instruction requires
    50 52 00 0051 118      ; no synchronization.
    0054 120      ;get tbit block address for
    00000000*GF 16 0054 121      jsb   g^exe$deanonpaged      ; deallocation routine
    00000000*GF 04 005A 122      clrl  g^exe$gl_sitespec      ;clear site specific vector
    50 0000*8F 3C 0063 124      movzwl $sss_normal,r0      ;OK to be deleted now
    04 0068 125      ret
    0069 126
50 00000000*8F 00 0069 127 nohandler: movl   $TBIT__NOHANDLER,r0
    0070 128      setipl #0
    04 0073 129      ret
    0074 130
50 00000000*8F 00 0074 131 inconsistent: movl   $TBIT__BADHANDLR,r0
    0078 132      setipl #0
    04 007E 133      ret
    007F 134 lock_end:
    007F 135
    007F 136      .end   remove_tbit

```

USER-WRITTEN SYSTEM SERVICES

PART 4 - Test Programs (TESTTBIT.MAR and TESTTBIT2.MAR)

Write a program that tests the USSDCLTBIT system service as follows:

1. Assign a channel to the terminal.
2. Display the contents of a location initially set blank.
3. Call the USSDCLTBIT system service, specifying a TBIT handler.
 - The handler should place some new data in the previously printed (blank) locations before exiting
 - Give some thought to which instruction should be used to exit handler
4. Set the T-Bit (bit 4) in the PSW (Hint: BISPSW instruction)
5. Issue a NOP instruction to cause the TBIT exception
 - The handler established above should be entered
6. Display the contents of the modified location to verify that the handler was entered.

The TESTTBIT2 program should be the same as the TESTTBIT program, except the call to the USSDCLTBIT system service should be removed. Enough spare bytes should be inserted so that the TBIT handler previously declared by the TESTTBIT program is still at the same virtual address (even though it should never be called).

If the RUNDOWN routine of the USSDCLTBIT system service has executed properly, when you run TESTTBIT2 you should get a TBIT error message. However, if you have not correctly cleared the CTL\$GL_SITESPEC location as part of the RUNDOWN routine, the TBIT exception will be handled properly by the TBIT handler, and no error message will be displayed.

USER-WRITTEN SYSTEM SERVICES

TESTTBIT.MAR File

.MAIN.

14-MAY-1982 18:22:37 VAX-11 Macro V03-00
27-MAR-1982 14:07:08 WORK:CMUIZNIKES.SYSPRG.US

```

0000 1 : TESTTBIT.MAR
0000 2
0000 3 : This program tests the fast TBIT dispatcher using the
0000 4 : USSDCLTBIT user-written system service.
0000 5
0000 6 : NOTE -- The LOADTBIT program must be run before this
0000 7 : program will work
0000 8
00000000 9 .psect nonshared_data pic, noexe, long
0000 10
6F 63 24 73 79 73 00000008*010E0000* 0000 11 term: .ascid /sys$command/ ; to communicate with terminal
64 6E 61 6D 6D 000E
0000 0013 12 ttchan: .word 0
0015 13
20 66 6F 20 73 74 6E 65 74 6E 6F 43 0015 14 msg: .ascii /Contents of dummy location is /
69 74 61 63 6F 6C 20 79 6C 6D 75 64 0021
20 73 69 20 6E 6F 002D
20 20 20 20 0033 15 dummy: .ascii / / / ; four blank bytes initially
2E 0037 16 .ascii ./
00000023 0038 17 $hlen = . - msg
0038 18
43 44 53 53 55 20 66 6F 20 74 75 4F 0038 19 out: .ascii /Out of USSDCLTBIT call/
6C 6C 61 63 20 54 49 42 54 4C 0044
00000016 004E 20 outlen = . - out
004E 21
00000000 004E 22 prev: .long 0 ; receive previous handler address
0052 23 arglist: ; argument list for USSDCLTBIT call
00000002 0052 24 .long 2 ; 2 parameters
000000A7* 0056 25 .address newhand ; address of new handler
0000004E* 005A 26 .address prev ; address of old handler returned
005E 27
00000000 28 .psect code pic, shr, noerr
0000 0000 29 start: .word 0 ; save no registers
01 50 E8 0017 30 $assign_s chan=ttchan, devnam=term
06 001A 31 blbs r0, 5$
0018 32 ret
01 50 E8 0042 33 5$: $qioe_s chan=ttchan,func=$iof_writevblk,p1=msg,p2=$hlen,p4=$32
04 0045 34 blbs r0,10$
00000000*GF 00000052*EF FA 0046 35 ret
01 50 E8 0051 36 10$: callg arglist, g^ussdcltbit ; establish TBIT handler
04 0054 37 blbs r0, 20$
0055 38 RET
10 88 007C 39 20$: $qioe_s chan=ttchan,func=$iof_writevblk,p1=out,p2=$outlen,p4=$32
01 007E 40 bispse #*X10 ; set T-bit in mask
007F 41 nop ; causes t-bit exception to happen
04 00A6 42 $qioe_s chan=ttchan,func=$iof_writevblk,p1=msg,p2=$hlen,p4=$32
00A7 43 RET
00A7 44
00A7 45 ; TBIT handler --- indicates being invoked by updating
00A7 46 ; location that is part of msg buffer
00A7 47
00000033*EF 00000038*EF D0 00A7 48 NEWHAND: ; entered via dispatch through SCB
02 00B2 49 movl out, dummy ; update value in field
00B3 50 rei ; dismiss T-BIT exception
00B3 51
00B3 52 .end start

```


USER-WRITTEN SYSTEM SERVICES

TESTTBIT2.MAR File (Page 1 of 2)

14-MAY-1982 18:22:40 VAX-11 Macro V03-00 Page 1
 7-APR-1982 17:18:25 WORK:CMUIZMIEKS.SYSPRG.USSJTESTTBI(1)

```

0000 1 : TESTTBIT2.MAR
0000 2
0000 3 : This program tests to make sure that the TBIT handler
0000 4 : has been removed by the RUNDOWN routine
0000 5
0000 6 : NOTE -- The TESTTBIT program must be run before this
0000 7 : program will work
0000 8
0000 9 : To work properly, this program should generate a TBIT
0000 10 : exception that is not handled by the TBIT handler
0000 11
00000007 12 .psect nonshared_data pic, noexe, long
0000 13
73 79 73 00000008 010E0000 0000 14 term: .ascid /syscommand/ : to communicate with terminal
64 6E 61 6D 6D 000E 0013
0000 0015
20 73 74 6E 65 74 6E 6F 63 0015 15 ttchan: .word 0
63 6F 6C 20 79 6D 6D 75 64 0021 16
20 73 69 20 6E 6F 002D 17 msg: .ascii /Contents of dummy location is /
20 20 20 20 0033 18 dummy: .ascii / / : four blank bytes initially
2E 0037 19 .ascii ./
00000023 0038 20 mslen = . - msg
0038 21
53 55 20 66 6F 20 74 75 4F 0038 22 out: .ascii /Out of USSDCLTBIT call/
6C 61 63 20 54 49 42 54 4C 0044 23 outlen = . - out
00000016 004E 24
004E 004E 25 prev: .long 0 ; receive previous handler address
00000000 004E 26 arglist: ; argument list for USSDCLTBIT call
0052 0052 27 .long 2 ; 2 parameters
000000A7 0056 28 .address newhand ; address of new handler
0000004E 005A 29 .address prev ; address of old handler returned
005E 30
00000000 31 .psect code pic, shr, nowrt
0000 0000 32 start: .word 0 ; save no registers
0002 33 $assign_s chan=ttchan, devnam=term
01 50 E8 0017 34 blbs r0, 5$
04 001A 35 ret
01 50 E8 0019 36 5$: $qio_w_s chan=ttchan,func=$iof_writevblk,p1=msg,p2=$mslen,p4=$32
04 0042 37 blbs r0,10$
04 0045 38 ret
0046 39
0046 40 : Make changes from TESTTBIT so that no handler is declared
0046 41 :10$: callg arglist, g$ussdcltbit : establish TBIT handler
0046 42 : blbs r0, 20$
0046 43 : RET
0046 44
10 0046 45 10$: $qio_w_s chan=ttchan,func=$iof_writevblk,p1=out,p2=$outlen,p4=$32
01 0060 46 blbpsw $^X10 ; set T-bit in mask
006F 47 noo ; causes t-bit exception to happen
0070 48 $qio_w_s chan=ttchan,func=$iof_writevblk,p1=msg,p2=$mslen,p4=$32
04 0097 49 RET
0098 50
0098 51 : TBIT handler --- indicates being invoked by updating
0098 52 : location that is part of msg buffer
0098 53
    
```

USER-WRITTEN SYSTEM SERVICES

TESTTBIT2.MAR File (Page 2 of 2)

MAIN.

14-MAY-1982 18:22:40 VAX-11 Macro V03-00
7-APR-1982 17:18:25 WORK:CMUIZNIKES.SYSPRG.USS:

```
000000A7 0098 54 FILLER: .BLKB 15 : make sure handler at same address as in
00A7 55 : TESTTBIT case
00A7 56
00A7 57 : SHOULD NOT BE ENTERED IF RUNDOWN WORKS PROPERLY
00A7 58
00A7 59 NEWHANDS : entered via dispatch through SCB
00000033*EF 00000039*EF D0 00A7 60 movl out, dummy : update value in field
02 00E2 61 rei : dismiss T-BIT exception
00B3 62
00B3 63 .end start
```

USER-WRITTEN SYSTEM SERVICES

PART 5 - Assembling and Linking Programs (TBIT.COM)

You will want to write a command procedure that assembles and links all the programs used by this example. In particular you need to:

1. Create a macro library from the TBITDEF.MAR file for use by the other programs.
2. Assemble all the macro programs, including the debugger, the macro library from step 1 (where appropriate), and the system macro library (where appropriate). Also, generate listing files to aid in debugging.
3. Compile the TBIT-specific error message file (TBITMSG) using the MESSAGE compiler.
4. Link all the programs
 - Taking special actions with the system service dispatcher (/PROTECT qualifier)
 - Including the debugger and system symbol table file, where appropriate
 - Generating map files to help with debugging
5. Enabling the necessary privileges to run the programs (e.g., CMKRNL)
6. Defining DELTA as the debugger to use
7. Installing the user-written system service file (/SHARE/PROTECT).

USER-WRITTEN SYSTEM SERVICES

TBIT.COM File

```

$!                                     TBIT.COM
$!
$! This command procedure builds all components for the modified TBIT
$! exception dispatcher, system service, and sample test programs.
$!
$! The following components are included:
$!
$! 1. A program that loads the new dispatcher into the system.
$!
$! 2. A procedure that illustrates its use.
$!
$! 3. A program that removes the modified handler from the system.
$
$ Create the macro library used by the various programs
$
$ SET VERIFY
$
$ LIBRARY /CREATE=(BLOCKS:10,MODULES:10) /MACRO -
$     TBITLIB.MLB TBITDEF.MAR
$
$ Assemble all components (include debugger support)
$ MACRO/ENABLE=DBG/LIST TESTTBIT
$ MACRO/ENABLE=DBG/LIST TESTTBIT2
$ MACRO/ENABLE=DBG/LIST LOADTBIT+TBITLIB/LIBRARY+SYS$LIBRARY:LIB/LIB
$ MACRO/ENABLE=DBG/LIST REMOVTBIT+TBITLIB/LIBRARY+SYS$LIBRARY:LIB/LIB
$ MACRO/LIST TBITDISP+TBITLIB/LIBRARY+SYS$LIBRARY:LIB/LIB
$
$ Compile image specific message codes
$ MESSAGE/LIST TBITMSG
$
$ LINK ALL PROGRAMS
$ LINK/PROTECT/NOSYSSHR/SHARE=TBITDISP/MAP=TBITDISP/FULL SYS$INPUT/OPTIONS
$
$ Options file for the link of User System Service example.
$
$ SYS$SYSTEM:SYS.STB/SELECTIVE
$
$ Create a separate cluster for the transfer vector.
$
$ CLUSTER=TRANSFER_VECTOR,,,SYS$DISK:C:TBITDISP
$
$ SMATCH=LEQUAL,1,1
$ LINK/DEBUG/MAP/FULL TESTTBIT,SYS$INPUT/OPTIONS
$
$ Options file for TBITTEST
$ TBITDISP.EXE/SHARE
$ LINK/DEBUG/MAP/FULL TESTTBIT2
$ LINK/DEBUG/MAP/FULL LOADTBIT+TBITMSG+SYS$SYSTEM:SYS.STB/SEL
$ LINK/DEBUG/MAP/FULL REMOVTBIT+TBITMSG+SYS$SYSTEM:SYS.STB/SEL
$
$ Prepare to test programs
$ DEFINE LIB$DEBUG DELTA
$ SET PROCESS/PRIV=(CHKRNL,SYS$PRV)
$ COPY TBITDISP.EXE SYS$SHARE:*.
$ PURGE *.OBJ, *.MAP, *.LIS, *.EXE, *.MLB, SYS$SHARE:TBITDISP.EXE
$ RUN SYS$SYSTEM:INSTALL
$ SYS$SHARE:TBITDISP.EXE/SHARE/PROTECT
$ SET NOVERIFY

```

USER-WRITTEN SYSTEM SERVICES

Sample Run

```

$ RUN/NODEBUG TESTTBIT
Contents of dummy location is .
Out of USSDCLTBIT call
XSYSTEM-F-TBIT, T-bit pending trap at PC=0000047F, PSL=03C00010
XTRACE-F-TRACEBACK, symbolic stack dump follows
module name      routine name      line      rel PC      abs PC
.MAIN.           CODE                        0000007F  0000047F
$
$ RUN/NODEBUG TESTTBIT2
Contents of dummy location is .
Out of USSDCLTBIT call
XSYSTEM-F-TBIT, T-bit pending trap at PC=00000470, PSL=03C00010
XTRACE-F-TRACEBACK, symbolic stack dump follows
module name      routine name      line      rel PC      abs PC
.MAIN.           CODE                        00000070  00000470
$
$ RUN/NODEBUG LOADTBIT
$
$ RUN/NODEBUG TESTTBIT
Contents of dummy location is .
Out of USSDCLTBIT call
Contents of dummy location is Out .
$
$ RUN/NODEBUG TESTTBIT2
Contents of dummy location is .
Out of USSDCLTBIT call
XSYSTEM-F-TBIT, T-bit pending trap at PC=00000470, PSL=03C00010
XTRACE-F-TRACEBACK, symbolic stack dump follows
module name      routine name      line      rel PC      abs PC
.MAIN.           CODE                        00000070  00000470
$
$ RUN/NODEBUG REMOVTBIT
$
$ RUN/NODEBUG TESTTBIT
Contents of dummy location is .
Out of USSDCLTBIT call
XSYSTEM-F-TBIT, T-bit pending trap at PC=0000047F, PSL=03C00010
XTRACE-F-TRACEBACK, symbolic stack dump follows
module name      routine name      line      rel PC      abs PC
.MAIN.           CODE                        0000007F  0000047F
$
$ RUN/NODEBUG TESTTBIT2
Contents of dummy location is .
Out of USSDCLTBIT call
XSYSTEM-F-TBIT, T-bit pending trap at PC=00000470, PSL=03C00010
XTRACE-F-TRACEBACK, symbolic stack dump follows
module name      routine name      line      rel PC      abs PC
.MAIN.           CODE                        00000070  00000470
$

```

WRITING COMMAND LANGUAGE INTERPRETERS

TEST

PART 1

1. Write your own program that outputs a message to the terminal. Add a verb to your own DCL tables that activates the program.
2. Rewrite the program to test for one parameter and one qualifier. Modify the CLD file such that the parameter is required but the qualifier has a default value if not given.
3. Remove the verbs MOUNT, ASSIGN and DELETE from your DCL tables.
4. Alter the DCL commands in your DCL tables in the manner described below. To accomplish this, select CLD files from the directory [COURSE.SYSPRG.CLI].
 - a. force the /CONFIRM qualifier on the DELETE command to be used unless turned off.
 - b. alter the "normal" number of columns on a DIRECTORY command to be 1 (one).
5. Add the verb TOPCPU such that it has the same effect as the command:

```
$ MONITOR PROCESS/TOPCPU
```

6. Alter the PRINT command such that it automatically places files in the queue LPAØ unless instructed otherwise.

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 1

```

;
;
; This program will output a message
; a command line and process it.
  .TITLE message
  .PSECT NONSHARED DATA PIC, NOEXE, LONG
MESSAGE: .ASCID /HI!!!/
  .PSECT CODE PIC, SHR, NOWRT, LONG
  .ENTRY START, ^M<>

; WRITE MESSAGE
  PUSHAQ MESSAGE
  CALLS #1,G^LIB$PUT_OUTPUT
  RET
  .END START
```

VERB.CLD

```
DEFINE VERB TELLME
      IMAGE your_disk:[your_directory]VERB.EXE
```

To set up the verb in your own DCL tables, execute the command:

```
$ SET COMMAND VERB
```

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 2

```

;                                     NAME.MAR
; This program will obtain the information from
; a command line and process it.
;
; .TITLE NAME
; $DSCDEF
; $CLIDEF
;
; .PSECT NONSHARED_DATA PIC, NOEXE, LONG
FIRST_NAME:
; .BLKW 1
; .BYTE DSC$K_DTYPE_T, DSC$K_CLASS_D
; .BLKL 1
LAST_NAME:
; .BLKW 1
; .BYTE DSC$K_DTYPE_T, DSC$K_CLASS_D
; .BLKL 1
FIRST: .ASCID /FIRST/
LAST: .ASCID /LAST/
; .PSECT CODE PIC, SHR, NOWRT, LONG
; .ENTRY BEGIN, ^M<>
;
; Get values for parameters and qualifiers
; PUSHAQ FIRST_NAME
; PUSHAQ FIRST_
; CALLS #2,G^CLISGET_VALUE
; PUSHAQ LAST_NAME
; PUSHAQ LAST_
; CALLS #2,G^CLISGET_VALUE
;
; Process paramters and qualifiers
; PUSHAQ FIRST_NAME
; CALLS #1,G^LIB$PUT_OUTPUT
; PUSHAQ LAST_NAME
; CALLS #1,G^LIB$PUT_OUTPUT
; RET
; .END BEGIN

```

NOD.CLD Command Descriptor file

```

DEFINE VERB Nod
IMAGE your_disk::[your_directory]exer2.exe
parameter P1, label=last,prompt="Last Name"
value(required)
qualifier first, default,value(default="Mr")

```

To place this verb in your own DCL tables, execute the command:

```
$ SET COMMAND NOD.CLD
```


WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 3

To remove DCL verbs from your tables, execute the command:

```
$ SET COMMAND/DELETE=(MOUNT,ASSIGN,DELETE)
```

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 4a Altered DELETE.CLD File

```
define syntax delete_entry_or_queue
  image queman
  parameter pl,prompt="Queue",value(required)

define syntax delete_all_symbols
  noparameters

define syntax delete_symbol,mcrignore
  routine delsym
  prefix cli$kdlsy_
  parameter pl,prompt="Symbol",value(required,type=$insym)
  qualifier all,          syntax=delete_all_symbols
  qualifier global
  qualifier local

define verb delete
  image delete
  prefix cli$kdle_
  parameter pl,prompt="File",
    value(required,list,impcat,type=$infile)
  qualifier created
  qualifier entry,
    value(required,list),
    syntax=delete_entry_or_queue
  qualifier modified
  qualifier queue,mcrignore,syntax=delete_entry_or_queue
  qualifier confirm, default ← Added
  qualifier log
  qualifier since,      value(type=$datetime)
  qualifier before,    value(type=$datetime)
  qualifier expired
  qualifier symbol,mcrignore,syntax=delete_symbol
  qualifier erase

define verb purge
  image delete
  prefix cli$kd_purg_
  parameter pl,prompt="File",value(list,impcat,type=$infile)
  qualifier keep,      value(required)
  qualifier log
  qualifier erase
```

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 4b

```

define type date_options
  keyword all
  keyword created
  keyword expired
  keyword modified

define type size_options
  keyword all
  keyword allocation
  keyword used

define verb directory
  image directory
  prefix cli$K dire
  parameter pl,prompt="File",value(list,impcat,type=$infile)
  qualifier zzzz,          label=dummy1
  qualifier before,       value(default=today,type=$datetime)
  qualifier brief,        default
  qualifier column,       default,value{(default=1)} ← Changed
  qualifier creation,     value(type=date_options)      from
  qualifier date,         value(required,list)          default = 1
  qualifier exclude,
  qualifier expiration
  qualifier full
  qualifier heading,      default
  qualifier modification
  qualifier output,       value(required,type=$outfile)
  qualifier owner
  qualifier printer
  qualifier protection
  qualifier since,        value(default=today,type=$datetime)
  qualifier size,         value(type=size_options)
  qualifier total
  qualifier trailing,     default
  qualifier versions,     value
  outputs(output)

```

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 5

```

define      verb      topcpu ← Changed
  image     MONITOR
  parameter P1,      label=CLASS NAME, prompt="Class(es)",
                    value (list, default=process) ← Added
  qualifier BEGINNING, nonnegatable,
                    value (required)
  qualifier ENDING,   nonnegatable,
                    value (required)
  qualifier INTERVAL, nonnegatable,
                    value (required)
  qualifier VIEWING_TIME, nonnegatable,
                    value (required)
  qualifier INPUT,    value
  qualifier DISPLAY, value, default
  qualifier RECORD,   value
  qualifier SUMMARY,  value
  qualifier COMMENT,  value (required)

  qualifier ALL,      nonnegatable, placement=local
  qualifier CURRENT, nonnegatable, placement=local
  qualifier AVERAGE, nonnegatable, placement=local
  qualifier MINIMUM, nonnegatable, placement=local
  qualifier MAXIMUM, nonnegatable, placement=local

  qualifier TOPCPU,   default ← Added, nonnegatable, placement=local
  qualifier TOPDIO,  nonnegatable, placement=local
  qualifier TOPBIO,  nonnegatable, placement=local
  qualifier TOPFAULT, nonnegatable, placement=local

  qualifier CPU,     placement=local

  qualifier PERCENT, placement=local

```

WRITING COMMAND LANGUAGE INTERPRETERS

Solution to Exercise 6

```

define verb print
  image submit
  prefix cli$K_prin_
  parameter pl,prompt="File",
    value(required,list,impcat,type=$infile)
  qualifier after,      value(required,type=$datetime)
  qualifier burst,     default,placement=positional
  qualifier copies,    value(required),placement=positional
  qualifier delete,    placement=positional
  qualifier device,    value(required,type=$device)
  qualifier flag,      default,placement=positional
  qualifier forms_type,value(required)
  qualifier header,    default,placement=positional
  qualifier hold
  qualifier identify,  default
  qualifier job_count,value(required)
  qualifier name,      value(required)
  qualifier lowercase
  qualifier page_count,value(required),placement=local
  qualifier parameters,value(required,list)
  qualifier priority,  value(required)
  qualifier queue,     default,value(default=1pa0) ← Altered
  qualifier space,     value(default=2),placement=positional
  qualifier feed,      default,placement=positional
  qualifier remote
  qualifier wsquota,   value(required)
  qualifier wsdefault, value(required)
  qualifier cputime,   value(required)
  qualifier characteristics,value(required,list)
  qualifier log_file,  value
  qualifier printer,   value(default=sys$print)
  qualifier keep,
  qualifier notify
  qualifier wsextent,  value(required)

```

WRITING COMMAND LANGUAGE INTERPRETERS

PART 2

1. The following problem has been found with the CLI discussed in class. It appears that after some images are run, the BYE command (and control-Z) no longer log the user off (but running the LOGINOUT image does).

Sample Run Illustrating Problem With CLI

```
Username: MUIZNIEKS/CLI=MYCLI
Password: _____
          Welcome to VAX/VMS version V3.0 on node HARDY

          *** EXAMPLE CLI ***

ENTER IMAGE NAME SYS$SYSTEM:PIP 2.EXE/FU

Directory DRO:[COURSE.SYSPRG.CLI]
31-MAY-82 21:30

MYCLI.EXE:1          (2416,4)          6./6.          31-MAY-82 21:27 [11,250] [RWED,RWED,RE,]
TODO.EXE:20         (2410,2)          18./18.        31-MAY-82 21:25 [11,250] [RWED,RWED,RE,]

Total of 24./24. blocks in 2. files

ENTER IMAGE NAME BYE
ZRMS-E-FNF, file not found
ENTER IMAGE NAME -Z
ZRMS-E-FNF, file not found
ENTER IMAGE NAME SYS$SYSTEM:LOGINOUT
MUIZNIEKS          logged out at 31-MAY-1982 21:31:01.36
```

WRITING COMMAND LANGUAGE INTERPRETERS

Your assignment will be to alter the CLI so that the DELTA debugger can be used to locate the error, and then identify and solve the problem. In order to do this, you will have to link the CLI with the file containing the DELTA debugger (SY\$LIBRARY:DELTA.OBJ). This file is distributed as a standard part of the VMS system.

You should try to imitate the scheme used by DCL to invoke the debugger. The scheme used by DCL is shown in the following partial listings of DCL modules DCXSTART and INITIAL. Note that calling the routine XDT\$START invokes the DELTA debugger from a program. (You will probably find it convenient to have R11 contain the starting address of the CLI, as done by DCL.)

You may find that you have some problems receiving control at the breakpoints you set. If so, you will need to examine the listings for DELTA (see your instructor), and take appropriate actions to solve that problem in your CLI.

WRITING COMMAND LANGUAGE INTERPRETERS

Partial Listing of DCXSTART Module

DCXSTART
/03-000

- DCL DEBUG VERSION START MODULE 26-APR-1982 22:50:42 VAX-11 MacPo V03-00
DEBUGGER START UP 10-MAR-1982 20:15:04 _DBB0:DCCL.SRCJDCXSTART.MAI

```

0000 57      .SBTTL  DEBUGGER START UP
0000 58      ;+
0000 59      ; START-UP WITH DEBUGGER
0000 60      ;
0000 61      ; THIS ENTRY POINT IS JUMPED TO AT THE CONCLUSION OF LOGGING A USER ONTO
0000 62      ; THE SYSTEM. ALL INPUT AND OUTPUT FILES ARE OPEN AND THE PROCESS PERPANEI
0000 63      ; DATA AREA (PPD) HAS BEEN INITIALIZED.
0000 64      :-
0000 65
00000000 66      .PSECT  DCL$BASE, BYTE, RD, NOWRT
0000 67
0000 68      BASE_OF_CLI:
03 00000002 GF 01 E1 0000 69      SBC      @PPDSV_MODE, G^CTL$AG_CLI$DATA+PPDSW_FLAGS, 10$ ;BR IF NOT B/
      FFF7 31 0000 70      DCL$STARTUP+2      ; IF BATCH, SKIP DEBUGGER ENTRY PT
5E 00000009 GF 03 0000 71 10$ :  MOVL      G^CTL$AL_STACK+8, SP      ; RESET SUPERVISOR MODE STACK POI)
      00 0D 0012 72      PUSHL     @0      ; ALLOCATE SPACE FOR XFER VECTOR I
      18 AF 01 F3 0014 73      CALLS     @1, B^20$      ; MAKE DUMMY CALL FRAME (HANDLER=(
      00C0 0019 74 20$ :  .WORD     0
50 000000C0 EF 9E 001A 75      MOVAB     XFER_ARRAY, R0      ; GET ADDRESS OF TRANSFER VECTOR
06 A0 0000 CF 9E 0021 76      MOVAB     W^DCL$STARTUP, 4(R0)      ; SET SECOND TRNFER ADDR TO HERE
      04 AC 50 D0 0027 77      MOVL     R0, 4(AP)      ; SET TRANSFER ARRAY FOR DEBUGGER
      58 D2 AF 9E 0029 78      MOVAB     BASE_OF_CLI, R11      ; R11 = BASE OF CLI FOR DEBUGGING
00000000 GF 6C FA 002F 79      CALLG     (AP), G^XDT$START      ; CALL DEBUGGER INITIALIZATION
      0035 80      SEXIT_S      ; EXIT IF ANY DEBUG INIT PROBLEMS
      003F 81
00000000 82      .PSECT  DCL$DEBUG, WRT
0000 83      XFER_ARRAY:
00000000 0000 84      .LONG     0      ; PRIMARY TRANSFER (NOT USED)
00000000 0004 85      .LONG     0      ; SECONDARY = DCL INITIALIZATION
0000 86
0000 87      .END

```


WRITING COMMAND LANGUAGE INTERPRETERS

Partial Listing of INITIAL Module

- COMMAND INTERPRETER INITIALIZATION 26-APR-1982 22:32:41 VAX-11 Macro V03-00 Page 4
 COMMAND INTERPRETER START UP 24-APR-1982 17:07:14 _DB80:COCL.SRC\INITIAL.MAR:1 (3)

```

0021 134      .SBTTL  COMMAND INTERPRETER START UP
0021 135 :+
0021 136 : OCL$STARTUP - COMMAND INTERPRETER START UP
0021 137 :
0021 138 : THIS ENTRY POINT IS JUMPED TO AT THE CONCLUSION OF LOGGING A USER ONTO
0021 139 : THE SYSTEM. ALL INPUT AND OUTPUT FILES ARE OPEN AND THE COMMAND LANGUAGE
0021 140 : INDEPENDENT DATA AREA HAS BEEN INITIALIZED.
0021 141 :-
0021 142
00000000 143      .PSECT  OCL$BASE,BYTE,RD,NOVRT
0000 144
SE 00000008*GF 00 0000 145      MOVL  G^CTLSAL_STACK+8,SP      ;RELOAD SUPERVISOR STACK POINTER
50 04 0007 146      CLRL  FP              ;INDICATE NO PREVIOUS FRAME
0021*CF 6E FA 0009 147      CALLG (SP),W^OCL$STARTUP      ;SETUP INITIAL CALL FRAME
000E 148      $EXIT_5
0017 149
00000021 150      .PSECT  OCL$ZCODE,BYTE,RD,NOVRT
0021 151 OCL$STARTUP::      ;COMMAND INTERPRETER START UP
0000 0021 152      .WORD  ^M<>          ;ENTRY MASK
7E 04 0023 153      CLRL  -(SP)          ;SETUP DUMMY PSL (@PRC_L_SAVAP=PRVPSL)
2A*AF 6E FA 0025 154      CALLG (SP),B^10$      ;CREATE DUMMY FP AFTER DUMMY AP
04 0029 155      RET
0000 002A 156 10$:      .WORD  0
6D 0000*CF 9E 002C 157      MOVAB  W^OCL$CONDHAND,(CFP)      ;ESTABLISH CONDITION HANDLER
00000000*GF 0002*CF 9E 0031 158      MOVAB  W^OCL$UTLSERV+2,G^CTLSAL_CLICALBK ; SET CALL BACK VECTOR
5A 00000000*GF 9E 003A 159      MOVAB  G^CTLSAG_CLIDATA,R10      ;GET ADDRESS OF PPO
0041 160 :
0041 161 : INITIALIZE CLI PROCESS WORK AREA
0041 162 :
04 AA 5B 08 AA C0 0041 163      MOVL  PPD$Q_CLIREG+4(R10),R11 ;GET ADDRESS OF CLI PRIVATE STORAGE
00 6E 00 2C 0043 164      MOVCS  #0,(SP),#0,PPD$Q_CLIREG(R10),(R11) ;ZERO ALL STORAGE
6B 5C 7D 004C 165      MOVQ  AP,PRC_L_SAVAP(R11) ;SAVE INITIAL ARGUMENT AND FRAME POINTERS
50 AB 1E AA B0 004F 166      MOVW  PPD$W_INPCHAN(R10),PRC_W_INPCHAN(R11) ;COPY INPUT CHANNEL
06 02 AA 01 E1 0054 167      SBC   #PPD$V_MODE,PPD$W_FLAGS(R10),20$ ;COPY JOB MODE
54 AB 00C0 8F A9 0059 168      B1SW  #PRC_W_MODE|PRC_W_VERIFY,PRC_W_FLAGS(R11) ;AND TURN VERIFY ON
0098 C8 02000000 8F C3 005F 169 20$: B1SL  #PRC_W_CTRLY,PRC_L_OUTOFBAND(R11) ;ENABLE CTRL/Y
05 02 AA 00 E1 0069 170      SBC   #PPD$V_NOCTLY,PPD$W_FLAGS(R10),25$ ;COPY NOCONTROL MODE
006D 171      CLR$IT PRC_V_CTRLY,PRC_L_OUTOFBAND(R11)
6C AB 00000000*GF 9E 0073 172 25$: MOVAB  G^OCL$AL_TAB_VEC,PRC_L_TAB_VEC(R11) ;ADDRESS OF DATABASE
0090 C3 04 B9 0079 173      MOVW  #4,PRC_B_EXHDEPID(R11) ;SET EXAMINE MODE TO HEX,WIDTH TO 4
0080 174 :
0080 175 : FOR BATCH JOBS, SETUP TO EXIT ON ERRORS. FOR INTERACTIVE JOBS,
0080 176 : DO AN IMPLIED "SET NOON".
0080 177 :
56 AB 02 80 0080 178      MOVW  #2,PRC_W_ONLEVEL(R11) ;ASSUME "ON ERROR THEN EXIT"
06 54 A3 06 E0 0084 179      BBS   #PRC_V_MODE,PRC_W_FLAGS(R11),30$ ;IF BATCH JOB, THIS IS OK
56 AB 0208 8F 80 0089 180      MOVW  #20818,PRC_W_ONLEVEL(R11) ;IF INTERACTIVE, "SET NOON"
008F 181 :
008F 182 : INITIALIZE CLI SYMBOL TABLE
008F 183 :
50 29 AB 9E 008F 184 30$: MOVAB  PRC_Q_GLOBAL(R11),R0 ;GET ADDRESS OF GLOBAL TABLE LISTHEAD
60 50 C0 C093 185      MOVL  R0,(R0) ;INIT GLOBAL SYMBOL TABLE EMPTY
80 80 C0 C096 186      MOVL  (R0)+,(R0)+
0099 187      ASSUME PRC_Q_LABEL EQ PRC_Q_GLOBAL+8
60 50 D0 C099 188      MOVL  R0,(R0) ;INIT LABEL TABLE EMPTY
80 90 D0 C09C 189      MOVL  (R0)+,(R0)+
009F 190      ASSUME PRC_Q_LOCAL EQ PRC_Q_LABEL+8
    
```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 1 of 7)

.MAIN.

31-MAY-1982 21:09:05 VAX-11 Macro V03-00
31-MAY-1982 21:08:58 DRAG:[COURSE,SYSPRG,CLI]MYI

```

0000 1 ; MYCLIDBG.MAR
0000 2
0000 3 .TITLE - MYCLIDBG - EXAMPLE CLI (COMMAND LANGUAGE INTERPRETER)
0000 4 ;
0000 5 ; Original author John Wolf - Training - Reading.
0000 6 ; Significantly rewritten/alterd for V3 interfaces by Vik Mutznicks
0000 7 ;
0000 8 ; To use this CLI the file MYCLIDBG.EXE must be installed in SYSSYSTEM,
0000 9 ; using the /CLI= option (or else set up the relevant default CLI with UA
0000 10 ; USERNAME: NAME/CLI=MYCLIDBG
0000 11 ;
0000 12 ; The CLI prompts for an image file name and runs the specified image (of
0000 13 ; native mode or compatibility mode). Command lines can be passed to util
0000 14 ; such as PIP as follows (DCL utilities CANNOT be used):
0000 15 ; $ SYSSYSTEM:PIP *,*/FU (to get a full directory listing)
0000 16 ;
0000 17 ; CONTROL-Y aborts the current image.
0000 18 ;
0000 19 ; To logout, type BYE, use a CTRL-Z, or execute SYSSYSTEM:LOGINDUT
0000 20 ;
0000 21 ; MACRO library calls
0000 22 ;
0000 23 .SPSDEF ; access mode symbols
0000 24 .SINDEF ; image header symbols
0000 25 .SCLIDEF ; command interpreter flags
0000 26 .SCLMSGDEF ; CLI message codes
0000 27 .SPPDEF ; from own macro library
0000 28 .PRCDEF ; from own macro library
0000 29 .PRDDEF ; from own macro library
0000 30 ;
0000 31 ; CLI private work area - this will be created on the stack
0000 32 ;
0000 33 .PSECT SABSS,ABS
0000 34
00000050 01F0 35 MSGBUF$IZ=60
FFFFFF9C 01F0 36 WRK_K_LENGTH=100
FFFFFF9C 01F0 37 .WRK_K_LENGTH
FF9C 38 WRK_L_CMDLEN ; user command length
FF9C 39 .BLKL 1
FFA0 40 WRK_L_CMDADR ; address of user command
FFA0 41 .BLKL 1
FFA4 42 WRK_Q_MSGBUPDSC ; descriptor for user input
FFA4 43 .BLKG 1
FFAC 44 WRK_T_MSGBUP ; storage for user input
FFAC 45 WRK_L_SAVESP ; saved stack pointer
FFFC 46 WRK_L_CONHAND ; address of condition handler
00000000 0000 47 .BLKL 1
00000004 0000 48 WRK_L_CONHAND ; address of condition handler
0000 49 .BLKL 1
0000 50 ;
0000 51 .PSECT
0000 52 ;
0000 53 ; No entry mask - must start at first location of image
0000 54 ;
0000 55 SUP$START:
SE 00000008*CF 00 0000 56 MOVL 0*CTLSAL_STACK+6,SP ; reload supervisor stack pointer
0007 57

```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 2 of 7)

31-MAY-1982 21:09:05 VAX-11 Macro V03=00 Page 2
 31-MAY-1982 21:08:58 DRAB:[COURSE,SYSPRG,CLI]MYCLIDBG.M(1)

```

00000010*EF 00 DD 0007 58 ;**** New stuff for debugging ****
00000010*EF 01 FB 0009 59 PUSHL 00 ; all, space for transfer vector addr
00000010*EF 02 0000 0010 60 CALLS #1, 58 ; make dummy call frame (handler=0)
00000010*EF 03 0000 0011 61 SSI .WORD 0
00000010*EF 04 0000 0012 62 MOVAB XFER_ARRAY,R0 ; get address of transfer vector
00000010*EF 05 0000 0013 63 MOVAB GO,(R0) ; set second transfer address
00000010*EF 06 0000 0014 64 MOVL R0,(AP) ; set transfer array for debugger
00000010*EF 07 0000 0015 65 MOVAB SUPSSTART,R11 ; R11 = base of CLI for debugging
00000010*EF 08 0000 0016 66 CALLG (AP),G^XDTSTART ; call debugger initialization
00000010*EF 09 0000 0017 67 SEEXIT,S ; exit if any debug init problems
00000010*EF 10 0000 0018 68
00000010*EF 11 0000 0019 69 .PSECT MAKE_WRT,WRT ; needed since writable area
00000010*EF 12 0000 0020 70 XFER_ARRAY:
00000010*EF 13 0000 0021 71 .LONG 0 ; primary transfer (not used)
00000010*EF 14 0000 0022 72 .LONG 0 ; secondary = CLI initialization
00000010*EF 15 0000 0023 73 BPTHAND:
00000010*EF 16 0000 0024 74 .LONG 0 ; address of DELTA primary exc. vec. handler
00000010*EF 17 0000 0025 75
00000010*EF 18 0000 0026 76 .PSECT
00000010*EF 19 0000 0027 77 GO: .WORD 2^M<> ; entry from debugger
00000010*EF 20 0000 0028 78 SSETEXV,S ACHODE=#PSLSC_USER,PRVHND=BPTHAND ; save DELTA handler
00000010*EF 21 0000 0029 79
00000010*EF 22 0000 0030 80 ;**** End of stuff for debugging
00000010*EF 23 0000 0031 81
00000010*EF 24 0000 0032 82 MOVAL -4(SP),FP ; set a good FP
00000010*EF 25 0000 0033 83 SRUNDN,S #PSLSC_USER ; run down LOGINOUT image
00000010*EF 26 0000 0034 84
00000010*EF 27 0000 0035 85 ;**** Here stuff for debugging
00000010*EF 28 0000 0036 86 SSETEXV,S ADDRESS=BPTHAND,ACHODE=#PSLSC_USER ; establish primary
00000010*EF 29 0000 0037 87 ; exc. vector for user mode to
00000010*EF 30 0000 0038 88 ; transfer control to DELTA handler
00000010*EF 31 0000 0039 89 SSETEXV,S ADDRESS=BPTHAND,ACHODE=#PSLSC_SUPER ; same for super mode
00000010*EF 32 0000 0040 90 ;**** End of debugging additions
00000010*EF 33 0000 0041 91
00000010*EF 34 0000 0042 92 SDELLOG,S #PSLSC_SUPER ; delete all supervisor mode process
00000010*EF 35 0000 0043 93 ; logical names. Specifically gets rid
00000010*EF 36 0000 0044 94 ; of initial SYSINPUT, leaving the
00000010*EF 37 0000 0045 95 ; correct exec. mode logical name,
00000010*EF 38 0000 0046 96 CLRL -(SP) ; set up dummy zero arg. block
00000010*EF 39 0000 0047 97 CALLG (SP),B^105 ; create initial call frame
00000010*EF 40 0000 0048 98 1031 .WORD 0 ; entry mask
00000010*EF 41 0000 0049 99 MOVAB G^CTLSAG_CLIDATA,R10 ; R10 = Address of PPD area from LOGIN
00000010*EF 42 0000 0050 100 MOVL PPD$Q_CLIREG+4(R10),R11 ; R11 = Address of CLI private storage
00000010*EF 43 0000 0051 101 MOVCS #0,(SP),#0,PPD$Q_CLIREG(R10),(R11) ; zero all storage
00000010*EF 44 0000 0052 102 MOVW PPD$W_INPCHAN(R10),PRC_W_INPCHAN(R11) ; save TTY channel number
00000010*EF 45 0000 0053 103 MOVQ AP,PRC_L_SAVAP(R11) ; save initial arg and frame pointers
00000010*EF 46 0000 0054 104 MOVAB W^SUPSUTILSERV+2,G^CTLSAL_CLICALBK ; CLI callback routine
00000010*EF 47 0000 0055 105 SQIOW,S #1,PRC_W_INPCHAN(R11),#IOS_SETMODE:IOSH_CTRLYABT-
00000010*EF 48 0000 0056 106 P1=#SUPSCTRLY-
00000010*EF 49 0000 0057 107 P3=#PSLSC_SUPER ; set up CONTROL-Y ABT
00000010*EF 50 0000 0058 108 BSBW ERROR ; set up CHMS handler
00000010*EF 51 0000 0059 109 SDCLCHM,S HAND
00000010*EF 52 0000 0060 110 BSBW ERROR
00000010*EF 53 0000 0061 111 MOVL #SSS_NORMAL,PPD$L_LSTSTATUS(R10) ; establish normal success
00000010*EF 54 0000 0062 112 ; for exit to LOGINOUT
00000010*EF 55 0000 0063 113 ;
00000010*EF 56 0000 0064 114 MOVAB PRC_C_LENGTH(R11),R0 ; set address of RMS structures
    
```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 3 of 7)

.MAIN,

31-MAY-1982 21:09:05 VAX-11 Macro V83-08
31-MAY-1982 21:08:58 DRAB1[COURSE,SYSPRG,CLI]

1C AB 68	9E 0101	115	NOVAB	PRD_G_FAB(R8),PRC_L_INOFAB(R11)	; addr. of gen. purp. FI
59 0000 C8	9E 0105	116	NOVAB	PRD_G_INPRAB(R8),R9	; set address of input RAB
57 017C C8	9E 010A	117	NOVAB	PRD_G_OUTRAB(R8),R7	; set address of output RAB
68 0000 8F	80 010F	118	MOVW	#FABSC_BID+<FABSC_BLN#8>,PRD_G_FAB(R8)	; set FAB ID/Length
0000 C8 50 A8	9E 0114	119	NOVAB	PRD_G_NAM(R8),RABSL_NAM(R8)	; set address of NAM block
58 A8 0000 8F	80 011A	120	MOVW	#NAMSC_BID+<NAMSC_BLN#8>,PRD_G_NAM(R8)	; set NAM ID/Length
0000 C9 0000 8F	80 0120	121	MOVW	#RABSC_BID+<RABSC_BLN#8>,RABSS_BID(R9)	; set RAB ID/Length
0000 C9 22 AA	80 0127	122	MOVW	PPDSW_INPISI(R18),RABSW_ISI(R9)	; set input ISI
0000 C7 0000 C9	80 012D	123	MOVW	RABSS_BID(R9),RABSS_BID(R7)	; set RAB ID/Length
0000 C7 26 AA	80 0134	124	MOVW	PPDSW_OUTISI(R18),RABSW_ISI(R7)	; set output ISI
0000 C9 58	00 013A	125	MOVL	R8,RABSL_FAB(R9)	; set address of FAB
0000 C7 58	00 013F	126	MOVL	R8,RABSL_FAB(R7)	; set address of FAB
0000 C9 0000 8F	AA 0144	127	BICW	#RABSW_PPF_IND,RABSW_ISI(R9)	; set PPF direct access
0000 C7 0000 8F	AA 0148	128	BICW	#RABSW_PPF_IND,RABSW_ISI(R7)	; disable user-mode EOP
00F4 C8 0000 C9	00 0152	129	MOVL	RABSS_BID(R8),PRD_G_ALYINPRAB(R8)	; set RAB ID/Length/ISI
0138 C8 0000 C7	00 0159	130	MOVL	RABSS_BID(R7),PRD_G_ALYOUTRAB(R8)	; set RAB ID/Length/ISI
0000 C9 44 AA	00 0160	131	MOVL	PPDSL_INPDEV(R18),RABSL_CTX(R9)	; store input device ch
0000 C7 64 AA	00 0164	132	MOVL	PPDSL_OUTDEV(R18),RABSL_CTX(R7)	; store output device ch
0000 C9 81	90 016C	133	MOV8	#1,RABSS_MBC(R9)	; allocate 1 block/buffer
0000 C9 FF 8F	90 0171	134	MOV8	#-1,RABSS_MBF(R9)	; allocate 1 buffer/stride
0000 C7 0000 C9	80 0177	135	MOVW	RABSS_MBF(R9),RABSS_MBF(R7)	; set same MBC/MBF for 8
0000 C9 00000000 8F	C8 017E	136	BISL	#RABSW_PMT,RABSL_RDP(R9)	; allow read with prompt
0C AB 57	00 0187	137	MOVL	R7,PRC_L_OUTRAB(R11)	; set address of output
88 AB 59	00 0188	138	MOVL	R9,PRC_L_INPRAB(R11)	; set address of input R
	018F	139			
	018F	140		Display CLI running message to user	
5A 0C AB	00 018F	141	MOVL	PRC_L_OUTRAB(R11),R18	; get address of output
0000 CA 00000462 8F	9E 0193	142	NOVAB	#MSG8,RABSL_RBF(R18)	; set message address
0000 CA 0019 8F	80 019C	143	MOVW	#LEN8,RABSW_RSZ(R18)	; set record size
	01A3	144	SPUT	RABW(R18)	; display CLI running me
028D	30 01AC	145	BSSW	ERROR	; check for errors
	01AF	146			; and drop through to me
	01AF	147			
	01AF	148		; Main CLI loop entered once from the initialize routine, then	
	01AF	149		; subsequently from the exit handler to process next command after	
	01AF	150		; image exit.	
	01AF	151			
	01AF	152		SUPPRESTART:	
6D 0000034E 8F	9E 01AF	153	NOVAB	SUPSEXCEPT,(FP)	; set condition handler address
	01B6	154			
	01B6	155		*** Here stuff for debugging	
	01B6	156		SSETEXV_3 ADDRESS=08PTHAND,ACHODE=#PSLSC_USER	; establish primary
	01C9	157			; exc. vector for user mode to
	01C9	158			; transfer control to DELTA hand
	01C9	159		SSETEXV_3 ADDRESS=08PTHAND,ACHODE=#PSLSC_SUPER	; same for super m
	01DC	160		*** End of debugging additions	
	01DC	161			
5E 0C AD	9E 01DC	162	NOVAB	WRK_K_LENGTH=16(FP),SP	; reserve CLI work area
AA AD 00000050 8F	00 01E9	163	MOVL	#MSGBUF3IZ,WRK_Q_MSGBUF08C(FP)	; size of RMS MSG buffer
AB AD AC AD	9E 01E8	164	NOVAB	WRK_T_MSGBUF(FP),WRK_Q_MSGBUF08C+4(FP)	; address of buffi
5A 88 AB	00 01ED	165	MOVL	PRC_L_INPRAB(R11),R18	; address of input RAB
0000 CA 00000496 8F	9E 01F1	166	NOVAB	#MSG2,RABSL_PBF(R18)	; set prompt address
0000 CA 13 8F	90 01FA	167	MOV8	#LEN2,RABSS_RSZ(R18)	; set prompt size
0000 CA AC AD	9E 0200	168	NOVAB	WRK_T_MSGBUF(FP),RABSL_UBF(R18)	; input buffer address
0000 CA 8850 8F	80 0206	169	MOVW	#MSGBUF3IZ,RABSW_USZ(R18)	; input buffer size
	020D	170	SGET	RABW(R18)	; get next record
52 0000 CA	3C 0216	171	NOVZWL	RABSW_RSZ(R18),R2	; size of input line

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 4 of 7)

31-MAY-1982 21:09:85 VAX-11 Macro V03-00 Page 4
 31-MAY-1982 21:08:58 DRAG:[COURSE,SYSPRG,CLI]MYCLID8G,M(1)

```

53 0000*CA D0 0218 172      MOVL  RAB3L,R0F(R10),R3      ; address of input line
      0F 00 0220 173      PUSHR  s*M<R0,R1,R2,R3>      ; save registers across CMPC3
1*EF AC AD 0003*0F 29 0222 174      CMPC3  #BYE_LEN,WRK_T_MSGBUF(FP),BYE ; check for BYE command
      11 13 0220 175      BEQL  193                      ; if so, log out
      0F 0A 022F 176      POPR  s*M<R0,R1,R2,R3>      ; restore registers
50 00000000*0F D1 0231 177      CMPL  #RMS3_EOF,R0          ; was it end-of-file
      06 13 0238 178      BEQL  193                      ; if so, log out
      13 50 E6 023A 179      BLBS  R0,203                ; valid GET?
      FF6F 31 023D 180      BRW  SUPSRESTART          ; if not, try again
      52 13*0F 9A 0240 181 193:  MOVZBL #LOGOSZ,R2         ; if EOF = logout
53 000004A9*0F 9E 0244 182      MOVAB  LOGO,R3             ; set up for LOGINOUT
      0248 183
      0248 184 ;***      Fix problem with BYE and control-Z commands not working
54 52 D0 0248 185      MOVL  R2,R4                ; store command line length
      1E 11 024E 186      BRB  233                  ; join common code
      0250 187 ;***      End of fix--- also add label 233: below
      0250 188
      52 D5 0250 189 203:  TSTL  R2                    ; blank line?
      03 12 0252 190      BNEQ  183                  ; if not, continue
      FF58 31 0254 191      BRW  SUPSRESTART          ; if blank - get another
55 54 52 D0 0257 192 183:  MOVL  R2,R0                ; length of command
      AC AD DE 025A 193      MOVL  WRK_T_MSGBUF(FP),R5   ; address of command
000004C0*0F 03 05 3A 025E 194 213: LOCC  (R5)+,#3,SEP         ; is this char. in separator list
      03 12 0266 195      BNEQ  223                  ; if NEQ = yes
      F3 54 F5 0268 196      SOBGR  R4,213             ; else try next char.
52 54 C2 0268 197 223:  SUBL2  R4,R2                ; Reset image name length
      026E 198
      026E 199 ;***      Added label 233 to this statement to fix bug from above
9C AD 54 D0 026E 200 233:  MOVL  R4,WRK_L_CMDLEN(FP) ; save user command length
      0272 201 ;***      End of bug fix
      0272 202
A0 AD 6342 9E 0272 203      MOVAB  (R3)(R2),WRK_L_CMDADR(FP) ; save user command address
00000286*0F 16 0277 204      JSB  SUPSIMGACT           ; do run the selected image
      03 50 E8 027D 205      BLBS  R0,243             ; skip if ok
      01C0 30 0280 206      BSBW  ERRPRY            ; output error message
      FF29 31 0283 207 243:  BRW  SUPSRESTART
      0286 208 ;
      0286 209 ; Image activation
      0286 210 ;
      0286 211 SUPSIMGACT:
74 AB 00000397*0F E2 0286 212      BBS  #PRC_V_EXIT,PRC_W_FLAGS(R11),403 ; skip if handler active
      78 AB 01 D0 0293 214      MOVL  SUPSEXIT,PRC_L_EXTEND(R11) ; set exit handler address
7C AB 0000 CB DE 0297 215      MOVL  #1,PRC_L_EXTARG(R11)      ; set count of arguments
      0192 30 02A7 217      MOVL  PRC_L_EXTCOD(R11),PRC_L_EXTPRM(R11) ; address of parameter
55 00000000*0F DC 02AA 218 403:  SOCLEX  #3,PRC_L_EXTBLK(R11) ; set up exit handler
      65 52 7D 02B1 219      BSBW  ERROR
80 A5 00000004*0F DA 02B4 220      MOVL  G^MMSGIMGHDRBUF,R5     ; R5 = Address of image header buffer
8C A5 0000048C*0F 9E 02BC 221      MOVQ  R2,(R5)                ; pointers to image file desc.
      02C4 222      MOVL  #DEFLEN,8(R5)         ; pointers to default image file desc.
      02C4 223      MOVAB  DEF,12(R5)          ; activate image
      02CA 224      SIMGACT_3=
      02C4 225      NAME=(R5)=
      0F 50 E8 02DC 226      DFLNAM=8(R5)=
      50 DD 02DF 227      HDRBUF=(R5)
      02E1 228      BLBS  R0,443             ; image header buffer
      ; if set = activation ok
      ; save error code
      ; run down bad image
      SRUNOWN_3 #PSLSC_USER
    
```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 5 of 7)

MAIN.

31-MAY-1982 21:09:05 VAX-11 Macro V03-00
 31-MAY-1982 21:08:58 DRAB:[COURSE,SYSPRG,CLI]MYC

```

        50 0E00 02EA 229      POPL   R0                ; restore error code
        05 02ED 230      RSB                ; return to main loop
    FC AD 5E 09 02EE 231 443:  MOVL   SP,WRK_L_SAVESP(FP) ; save current SP
    7E 0F 16 78 02F2 232      A8HL   #PSL3V_PVMOD,#PSL3C_USER; #PSL3C_USER,=(SP)
        02F6 233      ; set up user P8L on stack
    FA AF 0F 02F6 234      PUSHAB B*503           ; set up user PC on stack
        02 02F9 235      REI                ; switch to user mode
        5C 7C 02FA 236 503:  CLRQ   AP                ; clear AP,FP
    00 AF 00 00 02FB 237      CALLS  #0,0*603         ; set top level call frame
        0000 0300 238 603:  .WORD  0
    6D 00000000 GF 9E 0302 239      MOVAB  C*EXE3CATCH_ALL,(FP) ; set exception to catch all
        0309 240      SSETEXV_S #2,C*EXE3CATCH_ALL ; last chance vector
        031C 241      SINGPIX_S ; perform address relocation
        22 50 0323 242      BLBC   R0,053          ; exit if error
    54 00000000 GF 7D 0326 243      MOVG   C*MHG3IMGHDRBUF,R4 ; addresses of image header & file
        032D 244 ; CLI argument list
        7E 04 032D 245      CLRQ   =(SP)           ; CLI flags
        28 A4 00 032F 246      PUSHL  [M03L_LNKFLAGS(R4) ; link flags from image header
    7E 00000370 EF 9E 0335 248      MOVAB  R4,=(SP)         ; image file name & image header
        7E 54 7D 0332 247      MOVAB  SUPSUTILSERV,=(SP) ; CLI callback address
    50 02 A4 3C 033C 249      MOVZWL [M03M_ACTIVOFF(R4),R0 ; offset to transfer vectors
        50 54 C9 0340 250      ADDL   R4,R0           ; address of transfer vector array
        60 0F 0343 251      PUSHAL (R0)          ; save address of transfer vector
        90 06 F8 0345 252      CALLS  #6,0(R0)+      ; call image entry
    00000000 GF 17 0348 253 653:  JMP    C*EXE3EXIT_IMAGE ; go do SEXIT_3
        034E 254 ;
        034E 255 ; Condition handler for CLI errors. Not called for user errors as these
        034E 256 ; are caught by EXE3CATCH_ALL which prints error dump and does an exit.
        034E 257 ; CLI errors are special - reset exit handler and jump to EXE3CATCH_ALL.
        034E 258 ;
    0000 034E 259      .ENTRY  SUPSEXCEPT,"M<R11>
    50 00000000 GF 9E 0350 260      MOVAB  C*CTLSAG_CLIDATA,R11 ; get address of PPD
        50 08 A8 00 0357 261      MOVL   PPD30_CLIREG+4(R11),R11 ; get address of process work area
    74 A8 66 AF 9E 035B 262      MOVAB  B*103,PRC_L_EXTMND(R11) ; reset exit handler address
    00000002 GF 17 0360 263      JMP    C*EXE3CATCH_ALL+2 ; take special error exit path
    50 00000000 GF 00 0366 264 103:  .WORD  0                ; entry mask for special error han
        00 00 0368 265      MOVL   #SS3_NORMAL,R0 ; set success
        04 036F 266      RET
        0370 267 ;
        0370 268 ; CLI service routine to pass command line to utilities
        0370 269 ; as for RSX type GRCRS. Note that will NOT handle requests from DCL
        0370 270 ; utilities like DIRECTORY. Assumes callback request of proper type.
        0370 271 ;
    0000 0370 272      .ENTRY  SUPSUTILSERV,"M<R10,R11>
    50 00000000 GF 9E 0372 273      MOVAB  C*CTLSAG_CLIDATA,R11 ; address of PPD
        50 08 A8 00 0379 274      MOVL   PPD30_CLIREG+4(R11),R11 ; address of process work area
        50 04 A8 00 037D 275      MOVL   PRC_L_SAVFP(R11),R11 ; address of saved FP
        5A 04 AC 00 0381 276      MOVL   4(AP),R10 ; address of CLI request block
    00 AA 9C A8 00 0385 277      MOVL   WRK_L_CMDOLEN(R11),CLISW_RQSIZE(R10) ; return line length
    0C AA A8 A8 00 038A 278      MOVL   WRK_L_CMADDR(R11),CLISA_RQADDR(R10) ; return line address
    50 00030001 GF 00 038F 279      MOVL   #CLIS_NORMAL,R0 ; set success return
        04 0396 280      RET
        0397 281 ;
        0397 282 ; Exit handler, called after image exit to rundown image. Handler does
        0397 283 ; not return (as this would delete process), but resets the stack so that
        0397 284 ; the RSB returns to the main loop to get the next command for processing.
        0397 285 ;
    
```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 6 of 7)

31-MAY-1982 21:09:05 VAX-11 Macro V83-00 Page 6
 31-MAY-1982 21:08:58 DRAB:[COURSE,SYSPRG,CLI]MYCLID80,M(1)

```

0804 2397 286 ,ENTRY SUPSEXIT,"M<R2,R11>
5B 00000000*GF 9E 2399 287 MOVAB G*CTLSAG_CLIDATA,R11 ; R11 = address of PPD
5B 08 AB 00 03A0 288 MOVL PPD$Q_CLIREG+4(R11),R11 ; address of process work area
54 AB 08 00 AA 03A4 289 BICW #PRC_M_EXIT,PRC_M_FLAGS(R11) ; exit handler no longer active
5D 04 AB 00 03A8 290 MOVL PRC_L_SAVFP(R11),FP ; FP = address of CLI work area
52 AA AD 9E 03AC 291 MOVAB WRK_Q_M$GBUF$DSC(FP),R2 ; R2 = RMS msg buffer desc.
62 5F 8F 9A 03B0 292 10%1 MOVZBL #M$GBUF$IZ,(R2) ; reset size of message buffer
00 00 DD 03B4 293 PUSHAB 00 ; run down only image files
00 00 9F 73B6 294 PUSHAB (R2) ; address of message buffer desc.
00000000*GF 02 98 03B8 295 CALLB #2,G*SYSSRMSBRUNDWN ; run down RMS-32 files
EE 50 E9 03BF 296 BLSC R0,10$ ; if error = try next file
5B 04 BC 00 03C2 297 SRUNDWN_8 ; run down image
5E FC AD 00 03C8 298 MOVL #4(AP),R0 ; retrieve reason for exit
05 03D3 300 MOVL WRK_L_SAVESP(FP),SP ; get saved SP
03D4 301 RSB ; return to original caller
03D4 302 ; i.e. JSB SUP$INGACT in main loop
03D4 303 ;
03D4 304 ; CONTROL-Y AST routine. Prints out a message and forces image to exit.
03D4 305 ; It does not get involved in command processing as DCL does.
03D4 306 ;
2C00 03D4 306 ,ENTRY SUP$CTRL_Y,"M<R10,R11>
5B 00000000*GF 9E 03D6 307 MOVAB G*CTLSAG_CLIDATA,R11 ; R11 = address of PPD
5B 08 AB 00 03D0 308 MOVL PPD$Q_CLIREG+4(R11),R11 ; address of process work area
03E1 309 $QIOW_3 #1,PRC_M_INPCHAN(R11),#10$,SETMODELIOSM_CTRL_YAST=
03E1 310 #1,SUP$CTRL_Y=
03E1 311 #3=#PSL3C_SUPER ; re-activate CTRL-Y AST
0037 30 0402 312 BSBW ERROR ;
5A 0C AB 00 0405 313 MOVL PRC_L_OUTTAB(R11),R10 ; address of output TAB
0000*CA 0000047B*EF 9E 0409 314 MOVAB MSG1,RAB$L_RBF(R10) ; set message address
0000*CA 0010*0F 00 0412 315 MOVW #LEN1,RAB$W_RSZ(R10) ; set record size
0017 30 0419 316 SPUT RAB=(R10) ; output in CTRL/Y AST message
11 54 AB 03 E1 0422 317 BSBW ERROR ; check for errors
042A 318 BSC #PRC_V_EXIT,PRC_M_FLAGS(R11),10$ ; any image active?
04 0438 320 10%1 $FORCEX_3 CODE=#SSS_NORMAL ; force user to exit with success code
043C 321 ; RET ; return from AST
043C 322 ;
043C 323 ; Error test routine
01 50 E9 043C 324 ERROR: BLSC R0,10$ ; skip on error
05 043F 325 RSB ; return ok
01 10 0440 326 10%1 BSBW ERRPRT ; do print the error message
00 0442 327 HALT
0443 328 ;
0443 329 ; Error message output routine
0443 330 ;
5B 00 0443 331 ERRPRT: PUSHL R0 ; status code
01 00 0445 332 PUSHL #1 ; argument count
51 5E 00 0447 333 MOVL SP,R1
044A 334 SPUTMSG_3 (R1) ; output error message
0E 05 0459 335 TSTL (SP)+ ; pop arg count off stack
50 0ED0 045B 336 POPL R0 ; restore error code
05 045E 337 RSB
045F 338 ;
045F 339 ; CHMS handler - for this CLI is a no-op
045F 340 ;
0E 05 045F 341 HANO: TSTL (SP)+ ; remove change mode code from stack
02 0461 342 REI ; return after CHMS call
    
```

WRITING COMMAND LANGUAGE INTERPRETERS

Corrected CLI With Debugger Support (Page 7 of 7)

.MAIN,

31-MAY-1982 21:09:05 VAX-11 Macro V03-00
31-MAY-1982 21:08:58 ORAB:[COURSE,SYSPRG,CLI]M1

```

00000020 0462 343 ;
00000000 0462 344 ;      Messages displayed at terminal
00000004 0462 345 ;
00000009 0462 346 ;      SPACE = 32
0000000A 0462 347 ;      CR = 13
0000000B 0462 348 ;      LF = 10
0000000C 0462 349 ;      TAB = 9
40 41 50 45 20 2A 2A 2A 09 09 0A 3D 0462 350 MSG0: .ASCII <CR><LF><TAB><TAB>/** EXAMPLE CLI **/<CR><LF>
80 2A 2A 2A 2A 20 49 8C 43 20 45 4C 50 0462
      0A 047A
00000019 047B 351      LEN0=-MSG0
54 4E 4F 43 20 2A 2A 2A 09 09 0A 0D 047B 352 MSG1: .ASCII <CR><LF><TAB><TAB>/** CONTROL-Y AST **/<CR><LF>
2A 2A 20 54 53 41 20 59 2D 4C 4F 52 0487
      0A 0D 2A 0493
0000001B 0496 353      LEN1=-MSG1
47 01 4D 49 20 52 45 54 4E 45 0A 0D 0496 354 MSG2: .ASCII <CR><LF>/ENTER IMAGE NAME /
      20 45 4D 41 4E 20 45 04A2
00000013 04A9 355      LEN2=-MSG2
      20A9 356 ;
4C 3A 4D 45 54 53 59 53 24 53 59 53 04A9 357 LOGO: .ASCII /SYSSSYSTEMLOGINOUT/ ; image name for LOGOUT
      54 55 4F 4E 49 47 4F 04B5
00000013 04BC 358      LOGOSZ=-LOGO
45 50 45 2E 04BC 359 DEF: .ASCII /,EXE/ ; defaults for image file name
00000004 04C0 360      DEFLN=-DEF
      20C0 361 ;
      2F 09 20 04C0 362 SEP: .ASCII <SPACE><TAB>"/" ; separators
      04C3 363 ;
      45 59 42 04C3 364 BYE: .ASCII /BYE/ ; command to leave system
00000003 04C6 365      BYE_LEN=-BYE ; length of command
      04C6 366 ;
      04C6 367      .END SUPSTART

```


WRITING COMMAND LANGUAGE INTERPRETERS

MYCLIDBG.COM Procedure

```

$!                                     MYCLIDBG.COM
$!
$ SET VERIFY
$ LIBRARY/CREATE=(BLOCKS:18,MODULES:13)/MACRO DEFS,HLB PPODEF,MAP,OCLEDF,MAP
$ MAC/LIST MYCLIDBG+DEFS/LIB+SYSSLIBRARY:LIB/LIB
$ LINK/NOTRACE/NOSYSSHR/MAP MYCLIDBG,SYSSLIBRARY;DELTA,SYSSSYSTEM;SYS,STB/SEL
$ SET PROCESS/PRIV=(SYSPRV,CMKRN,PRMGBL)
$ COPY MYCLIDBG,EXE SYSSSYSTEM:*,*
$ RUN SYSSSYSTEM;INSTALL
SYSSSYSTEM;MYCLIDBG,EXE/SHARE
$ SET NOVERIFY
```

WRITING COMMAND LANGUAGE INTERPRETERS

Sample Run

Username: HUIZNIKES/CLI=MYCLIDBG
Password:
Welcome to VAX/VMS version V3.0 on node HARDY
DELTA Version X2.2

7FFD1E3B/PUSHAL 7FFD22CE RB/7FFD1E00

7FFD1E00:1IX
7FFD1E00 X1+21B:MOVL 0028(R10),R3

X1+21B:BP

*** EXAMPLE CLI ***

ENTER IMAGE NAME SYS\$SYSTEM:PIP 1.EXE/FU
1 BRK AT 7FFD201B
X1+21B/MOVL 0028(R10),R3 R2/00000017

BP

Directory DR0:[COURSE.SYSPRG.CLI]
31-MAY-82 21:33

MYCLI.EXE:1	(2416,4)	6./6.	31-MAY-82 21:27 [11,250] [RWED,RWED,RE,]
MYCLIDBG.EXE:1	(2436,5)	26./26.	31-MAY-82 21:32 [11,250] [RWED,RWED,RE,]
TODO.EXE:28	(2410,2)	18./18.	31-MAY-82 21:25 [11,250] [RWED,RWED,RE,]

Total of 50./50. blocks in 3. files

ENTER IMAGE NAME BYE
1 BRK AT 7FFD201B
X1+21B/MOVL 0028(R10),R3 BP
HUIZNIKES logged out at 31-MAY-1982 21:33:48.48

WRITING SYMBIONTS

TEST

In writing a symbiont, it will be necessary to eventually test it out under the control of the Job Controller. At the same time it would be very helpful if that subprocess was also under the control of the DELTA debugger. That way it would be possible to step through the program under actual conditions. Your assignment is to alter the standard VMS print symbiont such that when it starts up, it will be under the control of the DELTA debugger and you control the DELTA debugger input.

To accomplish this, you will need the following information:

1. The source files for the print symbiont are in the directory [COURSE.SYSPROG.SYMB] along with a command procedure (listed below) for creating the PRTSMB.EXE file.

```
$SET VERIFY
$ASSIGN SYSS$LIBRARY:LIB.MLB EXECMLS
$MACRO/ENABLE=DBG/OBJ=SMBCHRGEN SMBPRE+SMBCHRGEN
$MACRO/ENABLE=DBG/OBJ=SMBFLGPAG SMBPRE+SMBFLGPAG
$MACRO/ENABLE=DBG/OBJ=SMBGET SMBPRE+SMBGET
$MACRO/ENABLE=DBG/OBJ=SMBINIT SMBPRE+SMBINIT
$MACRO/ENABLE=DBG/OBJ=SMBMAIN SMBPRE+SMBMAIN
$MACRO/ENABLE=DBG/OBJ=SMBMBAST SMBPRE+SMBMBAST
$MACRO/ENABLE=DBG/OBJ=SMBHANDLE SMBPRE+SMBHANDLE
$MACRO/ENABLE=DBG/OBJ=SMBSUBR SMBPRE+SMBSUBR
$ LINK/NOSYSSHR/NOTRACE/EXE=PRTSMB/MAP=PRTSMB/FULL/CROSS -
    SMBINIT, SMBFLGPAG, FIDTONAME, SMBCHRGEN, SMBMAIN, -
    SMBGET, SMBSUBR, SMBMBAST, SMBHANDLE, -
    SYSS$SYSTEM:SYS.STB/SELECT
$SET NOVERIFY
```

Listing 1 MAKESYMB.COM

2. The DELTA debugger directs its I/O through the logical name DBG\$DELTA
3. To invoke the DELTA debugger rather than the normal Symbolic Debugger, you must use the logical name LIB\$DEBUG

WRITING SYMBIONTS

Solution

1.

The necessary change is to force the new symbiont to generate an SSS DEBUG signal. The listing at the end of this section shows the symbiont initialization section generating the signal.

2.

The command procedure [COURSE.SYSPROG.SYMB.SOLUTION]MYSYMB.COM (listed below) is used to create the executable file but with the name MYSYMB.EXE

```
$SET VERIFY
$! This logical name is needed by the MACRO assembler
$ASSIGN SY$$LIBRARY:LIB.MLB EXECMLS
$MACRO/ENABLE=DBG/OBJ=SMBCHRGEN SMBPRE+SMBCHRGEN
$MACRO/ENABLE=DBG/OBJ=SMBFLGPAG SMBPRE+SMBFLGPAG
$MACRO/ENABLE=DBG/OBJ=SMBGET SMBPRE+SMBGET
$!
$! Note I used a different filename for the INIT routine
$! that includes the LIB$$SIGNAL
$!
$MACRO/ENABLE=DBG/OBJ=MYINIT SMBPRE+MYINIT
$MACRO/ENABLE=DBG/OBJ=SMBMAIN SMBPRE+SMBMAIN
$MACRO/ENABLE=DBG/OBJ=SMBMBAST SMBPRE+SMBMBAST
$MACRO/ENABLE=DBG/OBJ=SMBHANDLE SMBPRE+SMBHANDLE
$MACRO/ENABLE=DBG/OBJ=SMBSUBR SMBPRE+SMBSUBR
$!
$! I altered the name of the final executable file
$!
$ LINK/NOSYSSHR/DEBUG/EXE=MYSYMB/MAP=PRTSMB/FULL/CROSS -
MYINIT, SMBFLGPAG, FIDTONAME, SMBCHRGEN, SMBMAIN, -
SMBGET, SMBSUBR, SMBMBAST, SMBHANDLE, -
SY$$SYSTEM:SYS.STB/SELECT
$SET NOVERIFY
```

Listing 2 MYSYMB.COM

WRITING SYMBIONTS

3. The final step is to set up the logical names and 'load' in the symbiont. This is done by the command procedure [COURSE.SYSPROG.SYMB.SOLUTION]LOADSYMB.COM which is listed below.

```
$SET VERIFY
$!
$! Make sure you have the necessary privileges
$SET PROCESS/PRIV=(OPER,CMKRNL,GRPNAM)
$!
$! Reset the protection on the terminal
$! So it can be allocated for I/O
$SET PROTECTION=(W:RW)/DEV terminal
$!
$! Copy the symbiont file to SYS$SYSTEM
$COPY [MARSH.SYMB]MYSYMB.EXE sys$system
$!
$! Alter the UIC and set up the logical
$! names for the symbiont to use
$SET UIC [1,4]
$DEFINE/GROUP DBG$DELTA terminal
$DEFINE/GROUP LIB$DEBUG DELTA
$!
$! Create the proper queue forcing the
$! Job Controller to use your file
$! for the symbiont
$INIT/QUEUE/PROCESS=[MARSH.SYMB]MYSYMB.EXE terminal
$!
$! Start the queue. This should cause the
$! symbiont to startup with the DELTA debugger
$START/QUEUE terminal
$SET NOVERIFY
```

Listing 3 LOADSYMB.COM

WRITING SYMBIONTS

Listing 4 of MYINIT.MAR (page 2 of 4)

LED SYMBOLS:

 \$JBCMSGDEF :JOB CONTROLLER MESSAGES

STORAGE:

.BOX: :NAME FOR JOB CONTROLLER MAILBOX
 .LONG 209-108
 .LONG 108
 .ASCII /_/
 .LONG SYS8C_JOBCTLMB
 .ASCII /://

.PAGE
.SBYTL SYMBIONT INITIALIZATION ROUTINE

IGNAL DESCRIPTION:

 THIS ROUTINE PERFORMS ALL ONE TIME FUNCTIONS FOR THE
 PRINT SYMBIONT.

NG SEQUENCE:

 MAIN ENTRY POINT OF SYMBIONT

PARAMETERS:

 NONE

IT INPUTS:

 NONE

T PARAMETERS:

 R11 CONTAINS THE ADDRESS OF THE IMPURE DATA BLOCK
 INIT DONE MESSAGE SENT TO SYMBIONT MANAGER

IT OUTPUTS:

 CHANNEL ASSIGNED TO SYMBIONT MANAGER MAILBOX
 MAILBOX CREATED FOR RECEIPT OF MANAGER MESSAGES

TION CODES:

 NONE

FFECTS:

 THIS ROUTINE DISPATCHES DIRECTLY TO THE SYMBIONT IDLE LOOP

WRITING SYMBIONTS

Listing 4 of MYINIT.MAR (page 3 of 4)

```

SMB_START:                                :SYMBIONT INITIAL ENTRY
        .WORD 0                            :ENTRY MASK
:*****
        pushl  %eax,debug
        call  %1,%lib$signal
:*****
        MOVAL  W^SMB$HANDLER,(FP)          :SET CONDITION HANDLER ADDRESS
:
: DISABLE ASTS UNTIL MESSAGE IS SENT
:
        %SETAST_S      #0                  :DISABLE ASTS
        MOVAL  W^SMB$G_DATA,R11           :SET ADDR OF IMPURE DATA BLOCK
:
: RUN-TIME INITIALIZATION OF DATA FIELDS
:
        MOVW   %SMB$K_SIZE,SD_W_MBREADLEN(R11) :SET INITIAL MB READ LENGTH
        MOVW   %SMB$K_TBUFSIZ,SD_W_TBUFCNT(R11) :SET LENGTH OF TEMP BUFFER
        MOVW   %SMB$K_TBUFSIZ,SD_W_TBUFSIZ(R11) :SET LENGTH OF TEMP BUFFER
        MOVAL  SD_T_TBUF(R11),SD_A_TBUPADR(R11) :SET ADDRESS OF TEMP BUFFER
:
        MOVAL  SD_G_FAB(R11),R6            :GET ADDRESS OF FAB
        MOVAL  SD_G_RAB(R11),R7            :GET ADDRESS OF RAB
        MOVAL  SD_G_NAM(R11),R8            :GET ADDRESS OF NAM BLK
:
        ASSUME  FAB$B_BID+1 EQ FAB$B_BLN
        MOVW   %FAB$C_BID+<FAB$C_BLN%>,FAB$B_BID(R6) :CREATE FAB
        ASSUME  RAB$B_BID+1 EQ RAB$B_BLN
        MOVW   %RAB$C_BID+<RAB$C_BLN%>,RAB$B_BID(R7) :CREATE RAB
        ASSUME  NAM$B_BID+1 EQ NAM$B_BLN
        MOVW   %NAM$C_BID+<NAM$C_BLN%>,NAM$B_BID(R8) :CREATE NAM BLK
:
        MOVL   R8,FAB$L_NAM(R6)            :SET NAME BLOCK ADDRESS IN FAB
        MOVL   R6,RAB$L_FAB(R7)           :SET FAB ADDRESS IN RAB
        MOVW   %SMB$K_LBUFSIZ,RAB$W_USZ(R7) :SET RECORD BUFFER SIZE
        SETBIT  RAB$V_RAM,RAB$L_ROP(R7)    :USE READ-AHEAD
:
        MOVZBL %QIO$_NARGS,SD_G_QIOBLK(R11) :SET QIO BLOCK LENGTH
        MOVZBL %QIO$_WRITEBLK,SD_G_QIOBLK+QIO$_FUNC(R11) :SET I/O FUNCTION
        CLRW   SD_B_ERR_FLAGS(R11)        :CLEAR BOTH SETS OF FLAGS
        MOVAL  SD_T_LBUF(R11),SD_Q_BUFPNT(R11) : SET FIRST ADDRESS
        MOVAL  SD_T_LBUF(R11),SD_Q_BUFPNT+4(R11) : SET SECOND ADDRESS
:
        ASSUME  STATE$_IDLE EQ 0
        CLRB  SD_B_STATE(R11)             :SET INITIAL STATE TO IDLE
:
: ASSIGN THE SYMBIONT MANAGERS MAILBOX
:
        %ASSIGN_S      JBCMAILBOX,-       :ASSIGN CHANNEL TO THE JOB CONTROLLER'S
        SD_W_JBCCHAN(R11)                  :MAILBOX-CHANNEL NUMBER STORED HERE
        BLBS  R0,%0                        :BR IF NO ERROR
        SIGNAL  JBC$_MBASGN,%0,R0         :SIGNAL THE ERROR
        BRB   20%                          :EXIT

```


WRITING SYMBIONTS

Listing 4 of MYINIT.MAR (page 4 of 4)

```

:
: CREATE SYMBIONTS MAILBOX
:
100:
    $CREMBX_S      -           :CREATE A MAIL BOX FOR COMMANDS
                PROMSK = #^XOFF0F,- : PROTECTION
                MAXMSG = #SIMSK_SIZE,- : MAXIMUM MESSAGE SIZE
                BUFQUD = #2*SIMSK_SIZE,- : 2 MESSAGES MAX
                CHAN = SD_W_MBCHAN(R11) : CHANNEL OF CREATED MAILBOX GOES HERE
    BLS          R0,309        :BR IF NO ERROR
    SIGNAL       JBC$_SYMBCRE,#0,R0 :SIGNAL THE ERROR
200:
    $EXIT_S       :FORCE IMAGE EXIT

:
: GET MAILBOX CHANNEL INFO
:
300:
    MOVAL       SD_W_MBCHAN(R11),R0 :SET ADDR OF CHANNEL
    BSBW        SMB$GETCHAN         :GET CHANNEL INFO

:
: SET UNSOLICITED AST FOR MY MAILBOX
:
    BSBW        SMB$SETMBAST        :SET THE MAILBOX AST
    MOVW        SD_T_TBUF+12(R11),R0 :SET MAILBOX UNIT NUMBER FOR INIT MSG
    MOVW        R0,SD_W_MBUNIT(R11)  :SAVE UNIT FOR $DELMBX
    BSBW        SMB$INIT_DONE        :SEND MGR THE INIT DONE MESSAGE

:
: ENABLE ASTS NOW
:
    $SETAST_S    #1                 :ENABLE ASTS
    BRV         SMB$MAIN             :GOTO MAIN LOOP
    .END        SMB_START

```

WRITING AN AME

TEST

Your instructor will supply you with a file containing the following MACRO-11 program (called RSXPROG.DAT):

Address (octal)	Opcodes (octal)	Instruction
000000	012700 000004	START: MOV #4,R0
000004	012701 000005	MOV #5,R1
000010	000004	IOT
000012	012701 000024	MOV #20.,R1
000016	012700 000012	MOV #10.,R0
000022	000004	IOT
	000000	.END START

The file will contain only the opcodes (no image header information normally placed in an EXE file by a task-builder or linker). The file was created running the FORTRAN program MAKEFILE, and entering the octal opcodes as listed above. Note that an opcode of 0 was inserted as the last instruction in the file. Executing this opcode will generate a reserved instruction compatibility mode exception.

Your assignment is to write an AME that will allow the above program to execute to completion. In particular, you should:

1. Choose whether you want to write an AME using the standard VMS condition handler method, or the special compatibility mode handler method. (If you have time, try to write both types of handlers.)
2. If you choose to write a standard VMS condition handler then:
 1. Be sure to link your AME at a base address of 64K.
 2. Use \$CRMPSC to establish the virtual address space for the compatibility mode program (starting at address 0).
 3. When a condition occurs, display the contents of the signal and mechanism arrays on your terminal (including the contents of registers R0 and R1, and the exception code causing the handler to be entered).
 4. Check for the IOT compatibility mode exception. If that was the condition causing the handler to be entered, update the PC (by adding 2 to the value in the signal array), and return a success code in R0.

WRITING AN AME

5. For any other exception, exit the program.
3. If you choose to write a customized compatibility mode handler then:
 1. Be sure to link your AME at a base address of 64K.
 2. Use \$CRMPSC to establish the virtual address space for the compatibility mode program (starting at address 0).
 3. Use \$DCLCMH to establish the customized handler.
 4. Display the contents of R0 and R1 from the P1 space area.
 5. Check for the IOT compatibility mode exception. If that was the condition causing the handler to be entered, update the PC (by adding 2 to the value in the P1 space area), place the appropriate information on the stack, restore registers R0-R6 from the P1 space area, and return control to the compatibility mode image via an REI instruction.
 6. If any other type of exception is encountered, exit the program.

Check the values of R0 and R1 after each exception, to verify that the AME is working properly. Remember that only the low-order word of those registers is valid on PDP-11 systems, for which registers are only 16-bits big.

Listing of MAKEFILE.FOR Program

```
C                                     MAKEFILE.FOR
C
C   This program allows you to enter the octal opcodes for
C   a program, and create a binary file with just those
C   opcodes stored. You enter a word at a time.
C
C   The file is defined by the logical name RSXPROG, or
C   defaults to RSXPROG.DAT when no translation.
C
C   INTEGERS2      NUMBER
C   OPEN (UNIT=1, FILE="RSXPROG", STATUS = "NEW",
C   1             RECORDTYPE="FIXED", RECORDSIZE=2)
C   READ (5,10,END=30) NUMBER
C   10            FORMAT (O6)
C   WRITE (1,20) NUMBER
C   20            FORMAT (A2)
C   GOTO 1
C   30            CLOSE (UNIT=1)
C   END
```

WRITING AN AME

Listing of MYAME.MAR Program (Page 1 of 2)

```

.MAIN.
14-MAY-1982 13:35:18 VAX-11 Macro V03-00
14-MAY-1982 13:34:31 WORK:CHUIZNIEKS.SYSPRG.

0000 1 : MYAME.MAR
0000 2 :
0000 3 : This program runs the compatibility mode program
0000 4 : RSXPROG.DAT, and responds to IOT instructions by
0000 5 : updating the PC and continuing execution.
0000 6 :
0000 7 : Written by Vik Huiznieks 02-APR-1982
0000 8 :
0000 9 : $CMPODEF ; Condition handling codes
0000 10 : $SECODEF ; Flags for $CRMPSC
0000 11 :
0000 12 : Local symbols
0000 13 PSL = ^X83C00000 ; PSL to enter Comp. Mode
00000020 0000 14 CR_LF = 32 ; normal P4 for $QIO
0000 15 :
00000000 0000 16 .PSECT NONSHARED_DATA PIC, NOEXE, LONG
0000 17 :
0000 18 : File control block; need channel to file for $CRMPSC
0000 19 SECFAB: $FAB FNM=<RSXPROG.DAT>, FOP=<UPD>, FAC=<GET, PUT>
0050 20 :
0050 21 : Arguments to $CRMPSC
00000000 0050 22 RANGE: .LONG 0 ; start at address 0
000001FF 0054 23 .LONG 511 ; only need one page
0058 24 :
0058 25 : Describe terminal for I/O to user
4F 43 24 53 59 53 00000060*010E0000* 0058 26 TT: .ASCII /SYS$COMMAND/ ; identify terminal
46 4E 41 4D 4D 0066
0000 0068
0060 27 TTCHAN: .WORD 0
0060 28 :
0060 29 : Messages generated by compatibility mode handler
62 69 74 61 70 6D 6F 63 2D 6E 6F 4E 0060 30 NONCOMP: .ASCII /Non-compatibility mode exception/
65 20 65 64 6F 6D 20 79 74 69 6C 69 0079
6E 6F 69 74 70 65 63 78 0085
00000020 0080
74 6C 69 62 69 74 61 70 6D 6F 43 0080 31 NONCOMPLEN = . - NONCOMP
7 63 78 65 20 55 64 6F 6D 20 79 0099
6E 6F 69 74 00A5
0000001C 00A9 32 COMP: .ASCII /Compatibility mode exception/
00A9 33 COMPLEN = . - COMP
00000000 00A9 34 :
0000 0000 35 .PSECT CODE PIC, SHR, NOWRT, LONG
0000 0000 36 .ENTRY START, ^M<> ; no registers saved
0002 37 :
0002 38 : Establish condition handler
6D 00000073*EF 0E 0002 39 NOVAL MYHAND, (FP)
0009 40 :
0009 41 : Assign channel to terminal to communicate
0009 42 $ASSIGN_S CHAN=TTCHAN, DEVNAM=TT
03 50 E8 001E 43 BLBS R0, 5$
0046 31 0021 44 BRW ERROR
0024 45 :
0024 46 : Open file containing CM image as private section
0024 47 5$: $OPEN FAB=SECFAB
03 50 E8 0031 48 BLBS R0, 10$
0033 31 0034 49 BRW ERROR
0037 50 :
0037 51 : Create virtual address space and map CM image
0037 52 10$: $CRMPSC_S INADR=RANGE, FLAGS=#SECS$WRT, -

```

WRITING AN AME

Listing of MYAME.MAR Program (Page 2 of 2)

14-MAY-1982 13:35:18 VAX-11 Macro V03-00 Page 2
 14-MAY-1982 13:34:31 WORK:CMUIZNIIEKS.SYSPRG.AME\MYAME.M(1)

```

03 50 E8 0037 53          CHAN=SECFAB+FAB$$_STV
0009 31 0058 54          BLBS   R0, 20$
0061 56          BRW   ERROR
0061 57 ;          Establish PSL and PC for compatibility mode image
83C00000 8F 00 0061 58 20$: PUSHL  #PSL          ; PSL predefined
TE 04 0067 59          CLR   -(SP)         ; PC = 0
02 0069 60          REI          ; transfer control to image
006A 61
006A 62 ;          Error occurred in main part of AME
006A 63 ERROR: $EXIT_S CODE=R0
0073 64
0073 65
0073 66 ;          Condition handler
0073 67
0004 0073 68          .ENTRY MYHAND, ^M(R2) ; save one register
0075 69
0075 70 ;          Get offset to signal array
52 04 AC D0 0075 71          MOVL  CHF$$_SIGARGLST(AP), R2
0079 72
0079 73 ;          Test for compatibility mode exception
00000000 8F 04 A2 D1 0079 74          CMPL  CHF$$_SIG_NAME(R2), #SS$$_COMPAT
32 13 0081 75          BEQLU  30$
0083 76
0083 77 ;          Non-compatibility mode exception - resignal error
0083 78          $QIOW_S CHAN=YTCHAN, FUNC=#IOS_WRITEVBLK, P1=NONCOMP, -
0083 79          P2=#NONCOMPLEN, P4=#CR_LF
58 50 E9 00AA 80          BLBC  R0, ERR
50 00000000 8F 00 00AD 81          MOVL  #SS$$_RESIGNAL, R0
04 0084 82          RET
0085 83
0085 84 ;          Compatibility mode exception
0085 85 30$: $QIOW_S CHAN=YTCHAN, FUNC=#IOS_WRITEVBLK, P1=COMP, -
0085 86          P2=#COMPLEN, P4=#CR_LF
26 50 E9 00DC 87          BLBC  R0, ERR
00DF 88
00DF 89 ;          Print contents of signal and mechanism arrays
04 AC D0 00DF 90          PUSHL CHF$$_SIGARGLST(AP)
00000000 EF 01 F8 00E2 91          CALLS #1, OUTSIG
08 AC D0 00E9 92          PUSHL CHF$$_MCHARGLST(AP)
00000000 EF 01 F8 00EC 93          CALLS #1, OUTMEC
00F3 94
00F3 95 ;          Check for IOT (CM code 2) in signal array (offset 8)
08 A2 02 D1 00F3 96          CMPL  #2, 8(R2)
0C 00F7 97          BNEQU ERR          ; if not, exit
00F9 98
00F9 99 ;          Update PC so don't loop (IOT instruction uses 2 bytes)
0C A2 02 C0 00F9 100         ADDL2 #2, 12(R2)      ; PC in signal array
00FD 101
00FD 102 ;          Return control to compatibility mode image
50 00000000 8F D0 00FD 103         MOVL  #SS$$_CONTINUE, R0
04 0104 104          RET
0105 105
0105 106 ;          Error in Handler or Non-IOT Compatibility Mode Exception
0105 107 ERR: $EXIT_S CODE=R0
010E 108
010E 109          .END   START
    
```

WRITING AN AME

Listing of OUTSIG.FOR Program

```
C                                     OUTSIG.FOR
C
C   This program outputs the contents of the signal array.
C   Of particular interest are the exception code, PC, PSL.
C
C   SUBROUTINE OUTSIG(ARRAY)
C   INTEGER ARRAY(10)
C   CHARACTER*15 LABEL(4)
C   DATA LABEL/'NUM. OF ARGS.:', 'CONDITION:', 'PC:', 'PSL:/'
C   N = ARRAY(1)
C
C   Display heading for signal array
C   CALL HEADING ("SIGNAL ARRAY")
C
C   Display elements in array
C   Always have number of args, exception code, PC, and PSL
C
C   WRITE (6,1000) LABEL(1), (ARRAY(1), I=1,3)
C   WRITE (6,1000) LABEL(2), (ARRAY(2), I=1,3)
1000  FORMAT (1X,A,I16,Z16,O16)
C
C   Variable number of arguments based on exception code
C   IF (N .GT. 3) THEN
C       DO 100 I=3,N-1
C           WRITE (6,2000) "ARGUMENT", I-2, (ARRAY(I), J=1,3)
C           FORMAT (1X,A,I2,5X,I16,Z16,O16)
2000  CONTINUE
100  END IF
C
C   Display PC and PSL
C   WRITE (6,1000) LABEL(3), (ARRAY(N), I=1,3)
C   WRITE (6,1000) LABEL(4), (ARRAY(N+1), I=1,3)
C
C   RETURN
C   END
```

WRITING AN AME

Listing of OUTMEC.FOR Program

```
C                                     OUTMEC.FOR
C      This subroutine displays the contents of the mechanism
C      array in decimal, hex, and octal.
C      The input parameter is the address of the mech. array
      SUBROUTINE OUTMEC(ARRAY)
      INTEGER ARRAY(5)
      CHARACTER*16 LABEL(5)
      DATA LABEL/'M:', 'ESTAB. FRAME:', 'DEPTH:', 'R0:', 'R1:' /
C      Print heading identifying mechanism array
      CALL HEADING('MECHANISM ARRAY:')
C      Write elements of array (fixed arguments)
      WRITE (6,1000) LABEL(1), (ARRAY(1), I=1,3)
      WRITE (6,1000) LABEL(2), (ARRAY(2), I=1,3)
      WRITE (6,1000) LABEL(3), (ARRAY(3), I=1,3)
      WRITE (6,1000) LABEL(4), (ARRAY(4), I=1,3)
      WRITE (6,1000) LABEL(5), (ARRAY(5), I=1,3)
1000   FORMAT(1X,A16,I16,Z16,O16)
      RETURN
      END
C      This subroutine prints a heading for the signal or
C      mechanism array
      SUBROUTINE HEADING(HEAD)
      CHARACTER*(*) HEAD, LOC*16
      LOC = 'LOCATION'
      WRITE (6,3000) HEAD, LOC, 'DECIMAL', 'HEXADECIMAL', 'OCTAL'
3000   FORMAT(//1X,A,//1X,3A16,A16/)
      RETURN
      END
```

Listing of MYAME.COM Procedure

```
$$                                     MYAME.COM
$$
$$      This command procedure assembles and links ↵
$$      (at a base address of 64K), a user-written AME
$$
$ MAC/LIST MYAME
$ FOR OUTSIG, OUTMEC
$ LINK/MAP/FULL MYAME,OUTSIG,OUTMEC,SYS$INPUT/OPTION
BASE=$X10000
```

WRITING AN AME

Sample Run

% RUN MYAME
Compatibility mode exception

SIGNAL ARRAY

LOCATION	DECIMAL	HEXADECIMAL	OCTAL	
NUM. OF ARGS.:	4	4	4	
CONDITION:	1068	42C	2054	<-- SSS_COMPAT
ARGUMENT 1	2	2	2	<-- IOT code
PC:	8	8	10	<-- Matches program
PSL:	-2084569088	83C00000	20360000000	<-- Matches supplied PSL

MECHANISM ARRAY:

LOCATION	DECIMAL	HEXADECIMAL	OCTAL	
N:	4	4	4	
ESTAB. FRAME:	2147179908	7FFB5D84	17776656604	
DEPTH:	0	0	0	
RO:	4	4	4	<-- Correct R0
R1:	-65531	FFFF0005	37777600005	<-- Correct R1 (low order word)

Compatibility mode exception

SIGNAL ARRAY

LOCATION	DECIMAL	HEXADECIMAL	OCTAL
NUM. OF ARGS.:	4	4	4
CONDITION:	1068	42C	2054
ARGUMENT 1	2	2	2
PC:	18	12	22
PSL:	-2084569088	83C00000	20360000000

MECHANISM ARRAY:

LOCATION	DECIMAL	HEXADECIMAL	OCTAL
N:	4	4	4
ESTAB. FRAME:	2147179908	7FFB5D84	17776656604
DEPTH:	0	0	0
RO:	10	A	12
R1:	-65516	FFFF0014	37777600024

Compatibility mode exception

SIGNAL ARRAY

LOCATION	DECIMAL	HEXADECIMAL	OCTAL	
NUM. OF ARGS.:	4	4	4	
CONDITION:	1068	42C	2054	
ARGUMENT 1	0	0	0	<-- Reserved instruction CM code (#0)
PC:	20	14	24	
PSL:	-2084569088	83C00000	20360000000	

MECHANISM ARRAY:

LOCATION	DECIMAL	HEXADECIMAL	OCTAL
N:	4	4	4
ESTAB. FRAME:	2147179908	7FFB5D84	17776656604
DEPTH:	0	0	0
RO:	10	A	12
R1:	-65516	FFFF0014	37777600024

INONAME-W-NOMSG, Message number 00000000
%

WRITING AN AME

Listing of MYCOMPAME.MAR Program (Page 1 of 3)

6-JUN-1982 15:13:03 VAX-11 Macro V03-00 Page 1
 6-JUN-1982 15:12:56 DRA0:CCOURSE.SYSPRG.AMEJMYCOMPAME.(1)

```

0000      1 :                                     MYCOMPAME.MAR
0000      2
0000      3 :      This program runs the compatibility mode program
0000      4 :      RSXPROG.DAT, and responds to IDT instructions by
0000      5 :      updating the PC and continuing execution.
0000      6
0000      7 :      Written by Vik Muiznieks      02-APR-1982
0000      8
0000      9      $SECTDEF                               : Flags for $CRMPSC
0000     10
0000     11 :      Local symbols
83C00000 0000 12 PSL = ^X83C00000                       : PSL to enter Comp. Mode
0000C020 0000 13 CR_LF = 32                             : normal P4 for $QIO
0000     14
0000000C 15      .PSECT NONSHARED_DATA PIC, NOEXE, LONG
0000     16
0000     17 :      File control block: need channel to file for $CRMPSC
0000     18 SECDEF: $FAB      FNM=<RSXPROG.DAT>, FOP=<UFO>, FAC=<GET, PUT>
0000     19
00000000 0050 20 :      Arguments to $CRMPSC
000001FF 0054 21 RANGE:  .LONG  0                          : start at address 0
000001FF 0054 22         .LONG 511                        : only need one page
0000     23
0000     24 :      Describe terminal for I/O to user
: 53 59 53 0000006C 010E0000 0058 25 TT:      .ASCII /SYSCOMMAND/           : identify terminal
      44 4E 41 40 42 0065
      0000 0068
      006D 006D 26 TTCHAN: .WORD  0
0000     27
: 69 62 69 74 61 70 6C 6F 43 006E 28 :      Messages generated by compatibility mode handler
: 78 65 20 65 64 5F 6D 20 79 0079 29 CCMP:  .ASCII /Compatibility mode exception/
      5E 6F 59 74 0085
      0000001C 0089 30 COMPLEN = . - COMP
      00000017 0089 31 CTR:      .LONG  STR_END - STR
      00000091 008D 32         .ADDRESS STR
: 20 6E 69 20 65 75 6C 61 56 0091 33 STR:      .ASCII /Value in R0 = IXW (hex)/
: 65 68 29 20 57 58 21 20 3D 009D
      00A8
      00000017 00A9 34 STR2_END:
      00000030 00AC 35 CTR2:     .LONG  STR2_END - STR2
: 20 6E 69 20 65 75 6C 61 56 00B0 36         .ADDRESS STR2
: 65 68 23 20 57 58 21 20 3D 00B0 37 STR2:     .ASCII /Value in R1 = IXW (hex)/
      00C7
      00000000 00C7 38 STR2_END:
      00000050 00C9 39 PADLEN:  .LONG  0
      000000C3 00CF 40 JUT:      .LONG  BUFF_END - BUFF
      00000123 00D3 41         .ADDRESS BUFF
      00000123 00D3 42 $UFF:     .BLKS  80
      0123 43 $UFF_END:
      0123 44
      00000000 0123 45 :      Temporary storage of updated PC
      00000000 0123 46 NEWPC:   .LONG  0
      0002 47
      00000000 0000 48      .PSECT CCDE      PIC, SHR, NOWRT, LONG
      0002 49      .ENTRY  START, ^M<>           : no registers saved
      0002 50
      0002 51 :      Establish condition handler
      0002 52      $OCLMNH_S ADDRES=MYHAND, TYPE=41
    
```

WRITING AN AME

Listing of MYCOMPAME.MAR Program (Page 2 of 3)

MAIN.

6-JUN-1982 15:13:03 VAX-11 Macro V03-00
6-JUN-1982 15:12:56 DRA0:CCOURSE.SYSPRG.AMEJ1

```

03 50 E8 0013 53          BLBS   R0, 38
0041 31 0016 54          BRW   ERROR
          0019 55
          0019 56 :      Assign channel to terminal to communicate
          0019 57 38:    $ASSIGN_S CHAN=TTCHAN, DEVNAM=TT
03 50 E8 002E 59          BLBS   R0, 58
0046 31 0031 59          BRW   ERROR
          0034 60
          0034 61 :      Open file containing CM image as private section
          0034 62 58:    $OPEN   FAB=SECFAB
03 50 E8 0041 63          BLBS   R0, 108
0033 31 0044 64          BRW   ERROR
          0047 65
          0047 66 :      Create virtual address space and map CM image
          0047 67 108:   $CRMPSC_S INADR=RANGE, FLAGS=#SEC$M_WRT, -
          0047 68          CHAN=SECFAB+FAB$$_STV
03 50 E8 0068 69          BLBS   R0, 208
0009 31 006E 70          BRW   ERROR
          0071 71
          0071 72 :      Establish PSL and PC for compatibility mode image
83C00900 9F 00 0071 73 208:   PUSHL  #PSL          ; PSL predefined
          7E 04 0077 74          CLRL  -(SP)          ; PC = 0
          02 0079 75          REI    ; transfer control to image
          007A 76
          007A 77 :      Error occurred in main part of AME
          007A 78 ERROR:   $EXIT_S CODE=R0
          0083 79
          0083 80
          0083 81 :      Condition handler
          0083 82
          0083 83 :      Compatibility mode exception had to occur to get here
          0083 84 MYHAND::
          0083 85          $QIOW_S CHAN=TTCHAN, FUNC=#IOS_WRITEVBLK, P1=COMP, -
          0083 86          P2=#COMPLEN, P4=#CR_LF
03 50 E8 00AA 87          BLBS   R0, 108
0000 31 00AD 88          BRW   ERR
          00B0 89
          00B0 90 :      Display contents of P1 space area (registers R0 and R1)
5A 00000003 GF 00 00B0 91 108:   MOVL  G*SYS$GL_CMCTX, R10 ; Base of CM context area
          50 5A 3C 00B7 92          MOVZWL (R10), R0 ; pick up R0
          00BA 93          $FAO_S CTRSTR=CTR, OUTLEN=FAOLEN, OUTBUF=OUT, P1=R0
03 50 E8 00D5 94          BLBS   R0, 208
0045 31 00DB 95          BRW   ERR
          00DB 96 208:   $OUTPUT CHAN=TTCHAN, LENGTH=FAOLEN, BUFFER=BUFF
79 50 E8 0104 97          BLBS   R0, ERR
50 04 AA 3C 0107 98          MOVZWL 4(R10), R0 ; pick up R1
          C108 99          $FAO_S CTRSTR=CTR2, OUTLEN=FAOLEN, OUTBUF=OUT, P1=R0
57 50 E8 0124 100         BLBS   R0, ERR
          0129 101         $OUTPUT CHAN=TTCHAN, LENGTH=FAOLEN, BUFFER=BUFF
23 50 E8 0152 102         BLBS   R0, ERR
          0155 103
          0155 104 :      Check for IOT (CM code 2) in CM context area
1C AA 02 C1 0155 105         CMPL  #2, 28(R10)
          25 12 0159 106         SNEQU ERR ; if not, exit
          0159 107
          0159 108 :      Update PC so don't loop (IOT instruction uses 2 bytes)
20 AA 02 C0 0159 109         ADDL2 #2, 32(R10)

```

WRITING AN AME

Listing of MYCOMPAME.MAR Program (Page 3 of 3)

6-JUN-1982 15:13:03 VAX-11 Macro V03-00 Page 3
6-JUN-1982 15:12:56 DRA0:[COURSE.SYSPRG.AME]MYCOMPAME.(1)

```
00000123'EF 20 4A 00 015F 110      MOVL    32(R10), NEWPC      : save to push on stack
                                0167 111
                                0167 112 :      Return control to compatibility mode image
                                0167 113 :      after restoring registers R0-R6
          50 38 70 0167 114      MOVQ    (R10)+,R0
          52 3A 70 016A 115      MOVQ    (R10)+,R2
          54 3A 70 016D 116      MOVQ    (R10)+,R4
          56 3A 00 0170 117      MOVL    (R10),R6
03C0C100 3F 00 0173 118      PUSHL   #PSL
00000123'EF 00 0179 119      PUSHL   NEWPC
          02 017E 120      REI
                                0180 121
                                0180 122 :      Error in Handler or Non-IOT Compatibility Mode Exception
                                0180 123 EPR:  SEXIT_S CODE=R0
                                0189 124
                                0189 125      .END    START
```

Listing of MYCOMPAME.COM Procedure

```
$!                                     MYCOMPAME.COM
$!
$!   This command procedure assembles and links a user
$!   written AME (at a base address of 64K)
$!
$ MAC/LIST MYCOMPAME
$ LINK/HAP/FULL MYCOMPAME.SYS$INPUT/OPTION
BASE=2X10000
```

Sample Run

```
$ RUN MYCOMPAME
Compatibility mode exception
Value in R0 = 0004 (hex)
Value in R1 = 0005 (hex)
Compatibility mode exception
Value in R0 = 000A (hex)
Value in R1 = 0014 (hex)
Compatibility mode exception
Value in R0 = 000A (hex)
Value in R1 = 0014 (hex)
$
```

