


```

000000 88888888 JJ EEEEEEEEEE XX XX EEEEEEEEEE 000000 UU UU TTTTTTTTTT
000000 88888888 JJ EEEEEEEEEE XX XX EEEEEEEEEE 000000 UU UU TTTTTTTTTT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88888888 JJ EEEEEEEEEE XX XX EEEEEEEEEE 00 00 UU UU TT
00 00 88888888 JJ EEEEEEEEEE XX XX EEEEEEEEEE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
00 00 88 88 JJ EE XX XX EE 00 00 UU UU TT
000000 88888888 JJJJJJ EEEEEEEEEE XX XX EEEEEEEEEE 000000 UUUUUUUUUU TT
000000 88888888 JJJJJJ EEEEEEEEEE XX XX EEEEEEEEEE 000000 UUUUUUUUUU TT

```

```

LL 111111 SSSSSSSS
LL 111111 SSSSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SSSSSS
LL 11 SSSSSS
LL 11 SS
LL 11 SS
LL 11 SS
LL 11 SS
LLLLLLLLLLLL 111111 SSSSSSSS
LLLLLLLLLLLL 111111 SSSSSSSS

```

```

1 0001 0 %title 'OBJEXEOUT - Handle Report Output'
2 0002 0     module objexeout(
3 0003 1         ident='V04-000') = begin
4 0004 1
5 0005 1
6 0006 1 .....
7 0007 1 *
8 0008 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 *  ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 *  TRANSFERRED.
18 0018 1 *
19 0019 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 *  CORPORATION.
22 0022 1 *
23 0023 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 .....
28 0028 1
29 0029 1
30 0030 1 **
31 0031 1 Facility:      VAX/VMS Analyze Facility, Handle Report Output
32 0032 1
33 0033 1 Abstract:      This module is responsible for generating report output
34 0034 1               for ANALYZE/OBJECT and ANALYZE/IMAGE. It provides the
35 0035 1               capability to create report files and fill them with
36 0036 1               output lines.
37 0037 1
38 0038 1
39 0039 1 Environment:
40 0040 1
41 0041 1 Author: Paul C. Anagnostopoulos, Creation Date: 8 January 1981
42 0042 1
43 0043 1 Modified By:
44 0044 1
45 0045 1     V03-005 DGB0067      Donald G. Blair      03-Jul-1984
46 0046 1               Support the /NOOUTPUT qualifier.
47 0047 1
48 0048 1     V03-004 DGB0053      Donald G. Blair      10-May-1984
49 0049 1               When an error occurs, save the error status so
50 0050 1               we can return it correctly at image exit.
51 0051 1
52 0052 1     V03-003 PCA1011      Paul C. Anagnostopoulos 1-Apr-1983
53 0053 1               Change the message prefix to ANLOBJ$ to ensure that
54 0054 1               message symbols are unique across all ANALYZEs. This
55 0055 1               is necessitated by the new merged message files.
56 0056 1
57 0057 1     V03-002 PCA0021      Paul Anagnostopoulos  24-Mar-1982

```

OBJEXECOUT
V04-000

OBJEXECOUT - Handle Report Output

1 8
15-Sep-1984 23:36:57
14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXECOUT.B32;1

Page 2
(1)

: 58
: 59
: 60
: 61
: 62

0058 1 :
0059 1 :
0060 1 :
0061 1 :
0062 1 :-

Signal errors using the correct SIV values.

V03-001 PCA0015 Paul Anagnostopoulos 22-Mar-1982
Don't constrain report file lines to 132 characters.

```

: 64      0063 1 %sbttl 'Module Declarations'
: 65      0064 1
: 66      0065 1   Libraries and Requires:
: 67      0066 1
: 68      0067 1
: 69      0068 1   library 'starlet';
: 70      0069 1   require 'objexereq';
: 71      0505 1
: 72      0506 1
: 73      0507 1   Table of Contents:
: 74      0508 1
: 75      0509 1
: 76      0510 1   forward routine
: 77      0511 1       anl$prepare_report_file: novalue,
: 78      0512 1       anl$report_page: novalue,
: 79      0513 1       anl$report_line: novalue,
: 80      0514 1       anl$exit_with_status: novalue,
: 81      0515 1       anl$format_line: novalue,
: 82      0516 1       anl$format_error: novalue,
: 83      0517 1       anl$error_count: novalue,
: 84      0518 1       anl$format_hex: novalue,
: 85      0519 1       anl$format_flags: novalue,
: 86      0520 1       anl$format_data_type: novalue,
: 87      0521 1       anl$format_mask: novalue,
: 88      0522 1       anl$format_protection: novalue,
: 89      0523 1       anl$format_severity: novalue,
: 90      0524 1       anl$interact;
: 91      0525 1
: 92      0526 1   External References:
: 93      0527 1
: 94      0528 1
: 95      0529 1
: 96      0530 1   external routine
: 97      0531 1       cli$get_value: addressing_mode(general),
: 98      0532 1       lib$get_input: addressing_mode(general),
: 99      0533 1       lib$p_lines: addressing_mode(general),
100     0534 1       cli$present: addressing_mode(general),
101     0535 1       str$trim: addressing_mode(general);
102     0536 1
103     0537 1   external
104     0538 1       anl$gb_interactive: byte;
105     0539 1
106     0540 1   Global Variables
107     0541 1
108     0542 1
109     0543 1
110     0544 1   global
111     0545 1       anl$worst_error: ! This contains either success status, or
112     0546 1       initial(anobj$ok); ! if errors occur it contains the first error
113     0547 1                       ! of the worst severity that occurred.
114     0548 1                       ! This status is returned at image exit.
115     0549 1
116     0550 1
117     0551 1   Own Variables:
118     0552 1
119     0553 1   The following data structures are needed to create and print to the
120     0554 1   report file. They include the FAB and RAB, and a buffer for the report

```

```
: 121      0555 1  ! spec.
: 122      0556 1
: 123      0557 1  own
: 124      0558 1      own_described_buffer(report_spec,namSc_maxrss),
: 125      0559 1
: 126      P 0560 1      report_fab: $fab(dnm='ANALYZE.ANL',
: 127      P 0561 1          fac=put,
: 128      P 0562 1          fna=report_spec+8,
: 129      P 0563 1          fns=namSc_maxrss,
: 130      P 0564 1          fop=sqo,
: 131      P 0565 1          org=seq,
: 132      P 0566 1          rat=cr,
: 133      0567 1          rfm=var),
: 134      0568 1
: 135      P 0569 1      report_rab: $rab(fab=report_fab,
: 136      0570 1          rac=seq);
: 137      0571 1
: 138      0572 1  ! The following variables are needed to format the report.
: 139      0573 1
: 140      0574 1  own
: 141      0575 1      generating_report,
: 142      0576 1      own_described_buffer(input_file_spec,namSc_maxrss),
: 143      0577 1      report_heading_msy: long,
: 144      0578 1      page_number: long,
: 145      0579 1      line_counter: signed long;
: 146      0580 1
: 147      0581 1  ! We also need to keep track of how many errors were reported.
: 148      0582 1
: 149      0583 1  own
: 150      0584 1      error_count: long initial(0);
```

```

152 0585 1 %sbttl 'ANLSPREPARE_REPORT_FILE - Prepare Report File'
153 0586 1 '*
154 0587 1 Functional Description:
155 0588 1 This routine is called whenever we begin the analysis of a new
156 0589 1 file. It sets up a report file to receive the analysis.
157 0590 1
158 0591 1 Formal Parameters:
159 0592 1 output_spec The report file spec as specified by the user.
160 0593 1 This is used on the first call to create the file.
161 0594 1 input_spec The spec of the input file we are analyzing.
162 0595 1 heading_msg An optional message code specifying the report
163 0596 1 page heading.
164 0597 1
165 0598 1 Implicit Inputs:
166 0599 1 global data
167 0600 1
168 0601 1 Implicit Outputs:
169 0602 1 global data
170 0603 1
171 0604 1 Returned Value:
172 0605 1 none
173 0606 1
174 0607 1 Side Effects:
175 0608 1
176 0609 1 --
177 0610 1
178 0611 1
179 0612 2 global routine anl$prepare_report_file(output_spec,input_spec,heading_msg): novalue = begin
180 0613 2
181 0614 2 bind
182 0615 2 output_spec_dsc = .output_spec: descriptor,
183 0616 2 input_spec_dsc = .input_spec: descriptor;
184 0617 2
185 0618 2 local
186 0619 2 status: long;
187 0620 2
188 0621 2 builtin
189 0622 2 nullparameter;
190 0623 2
191 0624 2
192 0625 2 ! Are we generating a report?
193 0626 2
194 0627 2 generating_report = cli$present(describe('OUTPUT'));
195 0628 2
196 0629 2 ! If the report file is not open, then we want to create it and prepare
197 0630 2 ! for the report.
198 0631 2
199 0632 2 if (.report_rab[rab$w_isi] eqlu 0) and .generating_report then (
200 0633 2
201 0634 2 ! Save the output file spec as the principal name of the report file.
202 0635 2
203 0636 2 ch$copy(.output_spec_dsc[len],.output_spec_dsc[ptr],
204 0637 2 .report_spec[len],.report_spec[ptr]);
205 0638 2 str$trim(report_spec,report_spec,report_spec);
206 0639 2
207 0640 2 ! Now let's create the report file and connect it.
208 0641 2

```

```

: 209 0642 3 status = $create(fab=report_fab);
: 210 0643 3 check (.status, anlob)$_openout, f, report_spec, .status, .report_fab[fab$l_stv]);
: 211 0644 3 status = $connect(rab=report_rab);
: 212 0645 3 check (.status, .status);
: 213 0646 2 );
: 214 0647 2
: 215 0648 2 ! Now let's save the report heading message and the input file spec for
: 216 0649 2 ! a subheading.
: 217 0650 2
: 218 0651 2 report_heading_msg = .heading_msg;
: 219 0652 2 input_file_spec[len] = .input_spec_dsc[len];
: 220 0653 2 ch$copy(.input_spec_dsc[len], .input_spec_dsc[ptr],
: 221 0654 2 ; ..input_file_spec[len], .input_file_spec[ptr]);
: 222 0655 2
: 223 0656 2 ! Now reset the page counter and start a new page.
: 224 0657 2
: 225 0658 2 page_number = 0;
: 226 0659 2 anl$report_page();
: 227 0660 2
: 228 0661 2 return;
: 229 0662 2
: 230 0663 1 end;

```

										.TITLE	OBJEXEOUT OBJEXEOUT - Handle Report Output				
										.IDENT	\V04-000\				
										.PSECT	\$PLITS, NOWRT, NOEXE, 2				
4C	4E	41	2E	45	5A	59	4C	41	4E	41	00000	P.AAA:	.ASCII	\ANALYZE.ANL\	:
					54	55	50	54	55	4F	0000B	P.AAC:	.ASCII	\OUTPUT\	:
											00011		.BLKB	3	:
											00000006	P.AAB:	.LONG	6	:
											00C00000		.ADDRESS	P.AAC	:
													.PSECT	\$OWNS, NOEXE, 2	
											000000FF	00000	REPORT_SPEC:		:
													.LONG	255	:
											00000000	00004	.ADDRESS	REPORT_SPEC+8	:
												00008	.BLKB	255	:
												00107	.BLKB	1	:
											03	00108	REPORT_FAB:		:
													.BYTE	3	:
												50	.BYTE	80	:
												0000	.WORD	0	:
												00000040	.LONG	64	:
												00000000	.LONG	C	:
												00000000	.LONG	0	:
												000C0000	.LONG	0	:
												0000	.WORD	0	:
												01	.BYTE	1	:
												00	.BYTE	0	:
												00000000	.LONG	0	:
												00	.BYTE	0	:
												00	.BYTE	0	:
												02	.BYTE	2	:


```

02 00127 .BYTE 2
00000000 00128 .LONG 0
00000000 0012C .LONG 0
00000000 00130 .LONG 0
00000000 00134 .ADDRESS REPORT_SPEC+8
00000000 00138 .ADDRESS P.AAA
FF 0013C .BYTE -1
0B 0013D .BYTE 11
0000 0013E .WORD 0
00000000 00140 .LONG 0
0000 00144 .WORD 0
00 00146 .BYTE 0
00 00147 .BYTE 0
00000000 00148 .LONG 0
00000000 0014C .LONG 0
0000 00150 .WORD 0
00 00152 .BYTF 0
00 00153 .BYT 0
00000000 00154 .LON 0
01 00158 REPORT_RAB:
. BYTE 1
44 00159 .BYTE 68
0000 0015A .WORD 0
00000000 0015C .LONG 0
00000000 00160 .LONG 0
00000000 00164 .LONG 0
0000# 00168 .WORD 0[3]
0000 0016E .WORD 0
00000000 00170 .LONG 0
0000 00174 .WORD 0
00 00176 .BYTE 0
00 00177 .BYTE 0
0000 00178 .WORD 0
0000 0017A .WORD 0
00000000 0017C .LONG 0
00000000 00180 .LONG 0
00000000 00184 .LONG 0
00000000 00188 .LONG 0
00 0018C .BYTE 0
00 0018D .BYTE 0
00 0018E .BYTE 0
00 0018F .BYTE 0
00000000 00190 .LONG 0
00000000 00194 .ADDRESS REPORT_FAB
00000000 00198 .LONG 0
0019C GENERATING REPORT:
.B[KB 4
000000FF 001A0 INPUT_FILE_SPEC:
.LONG 255
00000000 001A4 .ADDRESS INPUT_FILE_SPEC+8
001A8 .BLKB 255
002A7 .BLKB 1
002A8 REPORT_HEADING_MSG:
.BLKB 4
002AC PAGE_NUMBER:
.BLKB 4
002B0 LINE_COUNTER:

```

.....

00000000 002B4 ERROR_COUNT:
.BLKB 4
.LONG 0

.PSECT \$GLOBAL\$,NOEXE,2

00000000G 00000 ANL\$WORST_ERROR:
.LONG ANLOBS\$ _OK

- .EXTRN ANLOBS\$ _OK, ANLOBS\$ _ANYTHING
- .EXTRN ANLOBS\$ _DATATYPE
- .EXTRN ANLOBS\$ _ERRORCOUNT
- .EXTRN ANLOBS\$ _ERRORNONE
- .EXTRN ANLOBS\$ _ERRORS, ANLOBS\$ _EXEFIXA
- .EXTRN ANLOBS\$ _EXEFIXAIMAGE
- .EXTRN ANLOBS\$ _EXEFIXALINE
- .EXTRN ANLOBS\$ _EXEFIXCOUNT
- .EXTRN ANLOBS\$ _EXEFIXEXTRA
- .EXTRN ANLOBS\$ _EXEFIXFIXED
- .EXTRN ANLOBS\$ _EXEFIXFLAGS
- .EXTRN ANLOBS\$ _EXEFIXG
- .EXTRN ANLOBS\$ _EXEFIXGIMAGE
- .EXTRN ANLOBS\$ _EXEFIXGLINE
- .EXTRN ANLOBS\$ _EXEFIXLIST
- .EXTRN ANLOBS\$ _EXEFIXNAME
- .EXTRN ANLOBS\$ _EXEFIXNAME0
- .EXTRN ANLOBS\$ _EXEFIXP
- .EXTRN ANLOBS\$ _EXEFIXPSECT
- .EXTRN ANLOBS\$ _EXEFIXUP
- .EXTRN ANLOBS\$ _EXEFIXUPNONE
- .EXTRN ANLOBS\$ _EXEGST, ANLOBS\$ _EXEHDR
- .EXTRN ANLOBS\$ _EXEHDRACTIVE
- .EXTRN ANLOBS\$ _EXEHDRBLKCOUNT
- .EXTRN ANLOBS\$ _EXEHDRCHANCOUNT
- .EXTRN ANLOBS\$ _EXEHDRCHANDEF
- .EXTRN ANLOBS\$ _EXEHDRDECECO
- .EXTRN ANLOBS\$ _EXEHDRDMT
- .EXTRN ANLOBS\$ _EXEHDRDST
- .EXTRN ANLOBS\$ _EXEHDRFILEID
- .EXTRN ANLOBS\$ _EXEHDRFIXED
- .EXTRN ANLOBS\$ _EXEHDRFLAGS
- .EXTRN ANLOBS\$ _EXEHDRGBLIDENT
- .EXTRN ANLOBS\$ _EXEHDRGST
- .EXTRN ANLOBS\$ _EXEHDRIDENT
- .EXTRN ANLOBS\$ _EXEHDRIMAGEID
- .EXTRN ANLOBS\$ _EXEHDRISD
- .EXTRN ANLOBS\$ _EXEHDRISDBASE
- .EXTRN ANLOBS\$ _EXEHDRISDCOUNT
- .EXTRN ANLOBS\$ _EXEHDRISDFLAGS
- .EXTRN ANLOBS\$ _EXEHDRISDGBLNAM
- .EXTRN ANLOBS\$ _EXEHDRISDNUM
- .EXTRN ANLOBS\$ _EXEHDRISDPFCDEF
- .EXTRN ANLOBS\$ _EXEHDRISDPFCISZ
- .EXTRN ANLOBS\$ _EXEHDRISDTYPE
- .EXTRN ANLOBS\$ _EXEHDRISDVBN
- .EXTRN ANLOBS\$ _EXEHDRLINKID
- .EXTRN ANLOBS\$ _EXEHDRMATCH

.EXTRN ANLOBS\$_EXEHDRNAME
.EXTRN ANLOBS\$_EXEHDRNOPATCH
.EXTRN ANLOBS\$_EXEHDRPAGECOUNT
.EXTRN ANLOBS\$_EXEHDRPAGEDEF
.EXTRN ANLOBS\$_EXEHDRPATCH
.EXTRN ANLOBS\$_EXEHDRPATCHDATE
.EXTRN ANLOBS\$_EXEHDRPRIV
.EXTRN ANLOBS\$_EXEHDRROPATCH
.EXTRN ANLOBS\$_EXEHDRRWPATCH
.EXTRN ANLOBS\$_EXEHDRSYMDBG
.EXTRN ANLOBS\$_EXEHDRSYSVER
.EXTRN ANLOBS\$_EXEHDRTXTVBN
.EXTRN ANLOBS\$_EXEHDRTIME
.EXTRN ANLOBS\$_EXEHDRTYPEEXE
.EXTRN ANLOBS\$_EXEHDRTYPELIM
.EXTRN ANLOBS\$_EXEHDRUSERECO
.EXTRN ANLOBS\$_EXEHDRXFER1
.EXTRN ANLOBS\$_EXEHDRXFER2
.EXTRN ANLOBS\$_EXEHDRXFER3
.EXTRN ANLOBS\$_EXEHEADING
.EXTRN ANLOBS\$_EXEPATCH
.EXTRN ANLOBS\$_FLAG, ANLOBS\$_HEXDATA
.EXTRN ANLOBS\$_HEXHEADING1
.EXTRN ANLOBS\$_HEXHEADING2
.EXTRN ANLOBS\$_INDMSGSEC
.EXTRN ANLOBS\$_INTERACT
.EXTRN ANLOBS\$_MASK, ANLOBS\$_OBJCPRREC
.EXTRN ANLOBS\$_OBJDBGREC
.EXTRN ANLOBS\$_OBJENV, ANLOBS\$_OBJEOMFLAGS
.EXTRN ANLOBS\$_OBJEOMREC
.EXTRN ANLOBS\$_OBJEOMSEVABT
.EXTRN ANLOBS\$_OBJEOMSEVERR
.EXTRN ANLOBS\$_OBJEOMSEVIGN
.EXTRN ANLOBS\$_OBJEOMSEVRES
.EXTRN ANLOBS\$_OBJEOMSEVSUC
.EXTRN ANLOBS\$_OBJEOMSEVWRN
.EXTRN ANLOBS\$_OBJEOMWREC
.EXTRN ANLOBS\$_OBJFADPASSMECH
.EXTRN ANLOBS\$_OBJGSDENV
.EXTRN ANLOBS\$_OBJGSDENVFLAGS
.EXTRN ANLOBS\$_OBJGSDENVPAR
.EXTRN ANLOBS\$_OBJGSDENVPM
.EXTRN ANLOBS\$_OBJGSDENVPMW
.EXTRN ANLOBS\$_OBJGSDIDC
.EXTRN ANLOBS\$_OBJGSDIDCENT
.EXTRN ANLOBS\$_OBJGSDIDCFLAGS
.EXTRN ANLOBS\$_OBJGSDIDCMATCH
.EXTRN ANLOBS\$_OBJGSDIDCOBJ
.EXTRN ANLOBS\$_OBJGSDIDCVALA
.EXTRN ANLOBS\$_OBJGSDIDCVALB
.EXTRN ANLOBS\$_OBJGSDLEPM
.EXTRN ANLOBS\$_OBJGSDLPRO
.EXTRN ANLOBS\$_OBJGSDLSY
.EXTRN ANLOBS\$_OBJGSDPRO
.EXTRN ANLOBS\$_OBJGSDPROW
.EXTRN ANLOBS\$_OBJGSDPSC
.EXTRN ANLOBS\$_OBJGSDPSCALIGN

.EXTRN ANLOBS\$_OBJGSDPSCALLOC
.EXTRN ANLOBS\$_OBJGSDPSCBASE
.EXTRN ANLOBS\$_OBJGSDPSCFLAGS
.EXTRN ANLOBS\$_OBJGSDPREC
.EXTRN ANLOBS\$_OBJGSDSFSC
.EXTRN ANLOBS\$_OBJGSDSYM
.EXTRN ANLOBS\$_OBJGSDSYMW
.EXTRN ANLOBS\$_OBJGTXREC
.EXTRN ANLOBS\$_OBJHDRIGNREC
.EXTRN ANLOBS\$_OBJHEADING
.EXTRN ANLOBS\$_OBJLITINDEX
.EXTRN ANLOBS\$_OBJLNKREC
.EXTRN ANLOBS\$_OBJLNMREC
.EXTRN ANLOBS\$_OBJMHDCREATE
.EXTRN ANLOBS\$_OBJMHDNAME
.EXTRN ANLOBS\$_OBJMHDPATCH
.EXTRN ANLOBS\$_OBJMHDREC
.EXTRN ANLOBS\$_OBJMHDRECSIZ
.EXTRN ANLOBS\$_OBJMHDSTRLVL
.EXTRN ANLOBS\$_OBJMHDVERSION
.EXTRN ANLOBS\$_OBJMTCORRECT
.EXTRN ANLOBS\$_OBJMTCINPUT
.EXTRN ANLOBS\$_OBJMTCNAME
.EXTRN ANLOBS\$_OBJMTCREC
.EXTRN ANLOBS\$_OBJMTCSEQNUM
.EXTRN ANLOBS\$_OBJMTCUIC
.EXTRN ANLOBS\$_OBJMTCVERSION
.EXTRN ANLOBS\$_OBJMTCWHEN
.EXTRN ANLOBS\$_OBJPROARGCOUNT
.EXTRN ANLOBS\$_OBJPROARGNUM
.EXTRN ANLOBS\$_OBJPSECT
.EXTRN ANLOBS\$_OBJSRCREC
.EXTRN ANLOBS\$_OBJSTATHEADING1
.EXTRN ANLOBS\$_OBJSTATHEADING?
.EXTRN ANLOBS\$_OBJSTATLINE
.EXTRN ANLOBS\$_OBJSTATTOTAL
.EXTRN ANLOBS\$_OBJSYMBOL
.EXTRN ANLOBS\$_OBJSYMFLAGS
.EXTRN ANLOBS\$_OBJTIRARGINDEX
.EXTRN ANLOBS\$_OBJTIRCMD
.EXTRN ANLOBS\$_OBJTIRCMDSTK
.EXTRN ANLOBS\$_OBJTBTRC
.EXTRN ANLOBS\$_OBJTIRREC
.EXTRN ANLOBS\$_OBJTIRSTOIM
.EXTRN ANLOBS\$_OBJTIRVIELD
.EXTRN ANLOBS\$_OBJTTLREC
.EXTRN ANLOBS\$_OBJVALUE
.EXTRN ANLOBS\$_OBJUVALUE
.EXTRN ANLOBS\$_PROTECTION
.EXTRN ANLOBS\$_SEVERITY
.EXTRN ANLOBS\$_TEXT, ANLOBS\$_TEXTHDR
.EXTRN ANLOBS\$_NOSUCHMOD
.EXTRN ANLOBS\$_BADDATE
.EXTRN ANLOBS\$_BADHDRBLKCOUNT
.EXTRN ANLOBS\$_BADSEVERITY
.EXTRN ANLOBS\$_BADSYM1ST
.EXTRN ANLOBS\$_BADSYMCHAR

```
.EXTRN ANLOBS$_BADSYMLEN  
.EXTRN ANLOBS$_EXEBADFIXUPEND  
.EXTRN ANLOBS$_EXEBADFIXUPISD  
.EXTRN ANLOBS$_EXEBADFIXIIPVBN  
.EXTRN ANLOBS$_EXEBADISDS1  
.EXTRN ANLOBS$_EXEBADISDTYPE  
.EXTRN ANLOBS$_EXEBADMATCH  
.EXTRN ANLOBS$_EXEBADPATCHLEN  
.EXTRN ANLOBS$_EXEBADOBJ  
.EXTRN ANLOBS$_EXEBADTYPE  
.EXTRN ANLOBS$_EXEBADXFERO  
.EXTRN ANLOBS$_EXEHDRISDLONG  
.EXTRN ANLOBS$_EXEHDRLONG  
.EXTRN ANLOBS$_EXEISDLENDZRO  
.EXTRN ANLOBS$_EXEISDLENGBL  
.EXTRN ANLOBS$_EXEISDLENPRIV  
.EXTRN ANLOBS$_EXENOTNATIVE  
.EXTRN ANLOBS$_EXTRABYTES  
.EXTRN ANLOBS$_FIELDFIT  
.EXTRN ANLOBS$_FLAGERROR  
.EXTRN ANLOBS$_NOTOK, ANLOBS$_OBJBADIDCMATCH  
.EXTRN ANLOBS$_OBJBADNUM  
.EXTRN ANLOBS$_OBJBADPOP  
.EXTRN ANLOBS$_OBJBADPUSH  
.EXTRN ANLOBS$_OBJBADTYPE  
.EXTRN ANLOBS$_OBJBADVIELD  
.EXTRN ANLOBS$_OBJEOMBADSEV  
.EXTRN ANLOBS$_OBJEOMMISSING  
.EXTRN ANLOBS$_OBJFADBADAVC  
.EXTRN ANLOBS$_OBJFADBADRBC  
.EXTRN ANLOBS$_OBJGSDBADALIGN  
.EXTRN ANLOBS$_OBJGSDBADSUBTYP  
.EXTRN ANLOBS$_OBJHDRRES  
.EXTRN ANLOBS$_OBJMHDBADRECSIZ  
.EXTRN ANLOBS$_OBJMHDBADSTRLVL  
.EXTRN ANLOBS$_OBJMHDMISSING  
.EXTRN ANLOBS$_OBJNONTIRCMD  
.EXTRN ANLOBS$_OBJNOPSC  
.EXTRN ANLOBS$_OBJNULLREC  
.EXTRN ANLOBS$_OBJPOSPACE  
.EXTRN ANLOBS$_OBJPROMINMAX  
.EXTRN ANLOBS$_OBJPSCABSLEN  
.EXTRN ANLOBS$_OBJRECTOOBIG  
.EXTRN ANLOBS$_OBJTIRRES  
.EXTRN ANLOBS$_OBJUNDEFENV  
.EXTRN ANLOBS$_OBJUNDEFLIT  
.EXTRN ANLOBS$_OBJUNDEFPSC  
.EXTRN ANALYZE$ FACILITY  
.EXTRN CLISGET VALUE, LIB$GET INPUT  
.EXTRN LIB$LP LINES, CLISPRESENT  
.EXTRN STR$TRM, ANL$GB INTERACTIVE  
.EXTRN SYSSCREATE, SYSSCONNECT  
  
.PSECT $CODE$,NOWRT,2  
  
.ENTRY ANL$PREPARE_REPORT_FILE, Save R2,R3,R4,R5,- : 0612  
R6,R7,R8
```

01FC 00000

			58	00000000G	00	9E	00002	MOVAB	LIB\$SIGNAL, R8			
			57	0000'	CF	9E	00009	MOVAB	REPORT_SPEC, R7			
			52	04	AC	D0	0000E	MOVL	OUTPUT_SPEC, R2	0615		
			56	08	AC	D0	00012	MOVL	INPUT_SPEC, R6	0616		
				0000'	CF	9F	00016	PUSHAB	P.AAB	0627		
				00000000G	00	01	FB	0001A	CALLS	#1, CLIS\$PRESENT		
				019C	C7	50	D0	00021	MOVL	R0, GENERATING_REPORT		
						015A	C7	B5	00026	TSTW	REPORT_RAB+2	0632
							54	12	0002A	BNEQ	2\$	
			67		4F	019C	C7	E9	0002C	BLBC	GENERATING_REPORT, 2\$	
	20	04			B2	62	2C	00031	MOVCS	(R2), @4(R2), #32, REPORT_SPEC, -	0637	
					04	B7		00037		@REPORT_SPEC+4		
						57	DD	00039	PUSHL	R7	0638	
						57	DD	0003B	PUSHL	R7		
						57	DD	0003D	PUSHL	R7		
				00000000G	00	C3	FB	0003F	CALLS	#3, STR\$TRIM		
						0108	C7	9F	00046	PUSHAB	REPORT_FAB	0642
				00000000G	00	01	FB	0004A	CALLS	#1, SY\$CREATE		
					52	50	D0	00051	MOVL	R0, STA^US		
					13	52	E8	00054	BLBS	STATUS, 1\$	0643	
						0114	C7	DD	00057	PUSHL	REPORT_FAB+12	
							52	DD	0005B	PUSHL	STATUS	
							57	DD	0005D	PUSHL	R7	
							01	DD	0005F	PUSHL	#1	
						00B110A4	8F	DD	00061	PUSHL	#11604132	
				68		05	FB	00067	CALLS	#5, LIB\$SIGNAL		
						0158	C7	9F	0006A	1\$: PUSHAB	REPORT_RAB	0644
				00000000G	00	01	FB	0006E	CALLS	#1, SY\$CONNECT		
					52	50	D0	00075	MOVL	R0, STATUS		
					05	52	E8	00078	BLBS	STATUS, 2\$	0645	
							52	DD	0007B	PUSHL	STATUS	
					68	01	FB	0007D	CALLS	#1, LIB\$SIGNAL		
				02A8	C7	0C	AC	D0	00080	2\$: MOVL	HEADING_MSG, REPORT_HEADING_MSG	0651
				01A0	C7	66	B0	00086	MOVW	(R6), INPUT_FILE_SPEC	0652	
01A0	C7			04	B6	66	2C	0008B	MOVCS	(R6), @4(R6), #32, INPUT_FILE_SPEC, -	0654	
						01A4	D7		00093		@INPUT_FILE_SPEC+4	
						02AC	C7	D4	00096	CLRL	PAGE_NUMBER	0658
				0000V	CF	00	FB	0009A	CALLS	#0, ANL\$REPORT_PAGE	0659	
							04	0009F	RET		0663	

; Routine Size: 160 bytes, Routine Base: \$CODE\$ + 0000

```

: 232 0664 1 %sbttl 'ANL$REPORT_PAGE - Eject Page in Report'
: 233 0665 1 : **
: 234 0666 1 : Functional Description:
: 235 0667 1 :     This routine is called to eject the page in a report and print
: 236 0668 1 :     the heading on the new page.
: 237 0669 1 :
: 238 0670 1 : Formal Parameters:
: 239 0671 1 :     none
: 240 0672 1 :
: 241 0673 1 : Implicit Inputs:
: 242 0674 1 :     global data
: 243 0675 1 :
: 244 0676 1 : Implicit Outputs:
: 245 0677 1 :     global data
: 246 0678 1 :
: 247 0679 1 : Returned Value:
: 248 0680 1 :     none
: 249 0681 1 :
: 250 0682 1 : Side Effects:
: 251 0683 1 :
: 252 0684 1 : --
: 253 0685 1 :
: 254 0686 1 :
: 255 0687 2 global routine anl$report_page: novalue = begin
: 256 0688 2
: 257 0689 2
: 258 0690 2 ! Since we are starting a new page, reset the line counter.
: 259 0691 2
: 260 0692 2 line_counter = lib$lp_lines() - 7;
: 261 0693 2
: 262 0694 2 ! If this is an interactive session, don't print any page headings.
: 263 0695 2 ! They will really annoy the poor guy.
: 264 0696 2
: 265 0697 2 if .anl$gb interactive then
: 266 0698 2     return;
: 267 0699 2
: 268 0700 2 ! Eject the page.
: 269 0701 2
: 270 0702 2 anl$report_line(-1,describe(%char(formfeed)));
: 271 0703 2
: 272 0704 2 ! Increment the page number for the new page and print the heading lines.
: 273 0705 2
: 274 0706 2 increment (page number);
: 275 0707 2 anl$format_line(-1,0,.report_heading_msg,0,.page_number);
: 276 0708 2 anl$format_line(-1,0,anlobj$_anything,input_file_spec);
: 277 0709 2 anl$report_line(-1);
: 278 0710 2 anl$report_line(-1);
: 279 0711 2
: 280 0712 2 return;
: 281 0713 2
: 282 0714 1 end;

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

OC 0001C P.AAE: .ASCII <12>

00000001 0001D
00000000 00020 P.AAD: .BLKB 3
00000000 00024 .LONG 1
.ADDRESS P.AAE

.PSECT \$CODE\$,NOWRT,2

			000C	00000	.ENTRY	ANLSREPORT_PAGE, Save R2,R3	:	0687
	53	0000V	CF	9E	C0002	MOVAB	ANLSREPORT_LINE, R3	:
	52	0000'	CF	9E	00007	MOVAB	PAGE_NUMBER, R2	:
00000000G	00		00	FB	0000C	CALLS	#0, [IB\$LP_LINES	:
04	A2	F9	A0	9E	00013	MOVAB	-7(R0), LINE_COUNTER	:
	3D	0000G	CF	EB	00018	BLBS	ANLSGB_INTERACTIVE, 1\$:
		0000'	CF	9F	0001D	PUSHAB	P.AAD	:
	7E		01	CE	00021	MNEGL	#1, -(SP)	:
	63		02	FB	00024	CALLS	#2, ANLSREPORT_LINE	:
			62	D6	00027	INCL	PAGE_NUMBER	:
			62	DD	00029	PUSHL	PAGE_NUMBER	:
			7E	D4	0002B	CLRL	-(SP)	:
		FC	A2	DD	0002D	PUSHL	REPORT_HEADING_MSG	:
			7E	D4	00030	CLRL	-(SP)	:
	7E		01	CE	00032	MNEGL	#1, -(SP)	:
0000V	CF		05	FB	00035	CALLS	#5, ANLSFORMAT_LINE	:
		FEF4	C2	9F	0003A	PUSHAB	INPUT_FILE_SPEC	:
		00000000G	8F	DD	0003E	PUSHL	#ANLOBJ\$_ANYTHING	:
			7E	D4	00044	CLRL	-(SP)	:
	7E		01	CE	00046	MNEGL	#1, -(SP)	:
0000V	CF		04	FB	00049	CALLS	#4, ANLSFORMAT_LINE	:
	7E		01	CE	0004E	MNEGL	#1, -(SP)	:
	63		01	FB	00051	CALLS	#1, ANLSREPORT_LINE	:
	7E		01	CE	00054	MNEGL	#1, -(SP)	:
	63		01	FB	00057	CALLS	#1, ANLSREPORT_LINE	:
			04	0005A	1\$:	RET		:

; Routine Size: 91 bytes, Routine Base: \$CODE\$ + 00A0


```

: 284 0715 1 %sbttl 'ANLSREPORT_LINE - Print a Line in Report'
: 285 0716 1 **
: 286 0717 1 : Functional Description:
: 287 0718 1 : This routine is called to print a line into the report file.
: 288 0719 1 :
: 289 0720 1 : Formal Parameters:
: 290 0721 1 :     widow_control    Controls widowning as follows:
: 291 0722 1 :         positive     specifies number of lines that
: 292 0723 1 :                       must remain on the page.
: 293 0724 1 :         zero         doesn't matter how many lines.
: 294 0725 1 :         negative     force line onto current page.
: 295 0726 1 :     line             Address of descriptor of line. Optional.
: 296 0727 1 :
: 297 0728 1 : Implicit Inputs:
: 298 0729 1 :     global data
: 299 0730 1 :
: 300 0731 1 : Implicit Outputs:
: 301 0732 1 :     global data
: 302 0733 1 :
: 303 0734 1 : Returned Value:
: 304 0735 1 :     none
: 305 0736 1 :
: 306 0737 1 : Side Effects:
: 307 0738 1 :
: 308 0739 1 : --
: 309 0740 1 :
: 310 0741 1 :
: 311 0742 2 global routine anl$report_line(widow_control,line): novalue = begin
: 312 0743 2
: 313 0744 2 bind
: 314 0745 2     line_dsc = .line: descriptor;
: 315 0746 2
: 316 0747 2 local
: 317 0748 2     status: long;
: 318 0749 2
: 319 0750 2 builtin
: 320 0751 2     nullparameter;
: 321 0752 2
: 322 0753 2
: 323 0754 2 ! Don't do anything if we're not generating a report.
: 324 0755 2
: 325 0756 2 if not .generating_report then
: 326 0757 2     return;
: 327 0758 2
: 328 0759 2 ! If the caller isn't forcing this line onto the page, and there are not
: 329 0760 2 ! enough lines left for prevention of widowning, then eject the page.
: 330 0761 2
: 331 0762 2 if (.widow_control geq 0) and
: 332 0763 2     (.line_counter lss .widow_control) then
: 333 0764 2     anl$report_page();
: 334 0765 2
: 335 0766 2 ! Print the line if there is one. Otherwise put out a blank line.
: 336 0767 2
: 337 0768 2 if nullparameter(2) then
: 338 0769 2     report_rab[rab$w_rsz] = 0
: 339 0770 3 else (
: 340 0771 3     report_rab[rab$w_rsz] = .line_dsc[len];

```

```

: 341      0772 3      report_rab[rab$l_rbf] = .line_dsc[ptr];
: 342      0773 2      );
: 343      0774 2      status = $put(rab=report_rab);
: 344      0775 2      check (.status, anobj$_writeerr, 1, report_spec, .status, .report_rab[rab$_stv]);
: 345      0776 2
: 346      0777 2      ! Account for the line on the page.
: 347      0778 2
: 348      0779 2      decrement (line_counter);
: 349      0780 2
: 350      0781 2      return;
: 351      0782 2
: 352      0783 1      end;

```

					.EXTRN	SYSSPUT	
			0004	00000	.ENTRY	ANLSREPORT_LINE, Save R2	: 0742
	52	08	AC	D0 00002	MOVL	LINE, R2	: 0745
	57	0000*	CF	E9 00006	BLBC	GENERATING REPORT, 6\$: 0756
		04	AC	D5 0000B	TSTL	WIDOW_CONTROL	: 0762
			0C	19 0000E	BLSS	1\$	
	04	AC	0000*	CF D1 00010	CMPL	LINE_COUNTER, WIDOW_CONTROL	: 0763
				04 18 00016	BGEQ	1\$	
	89	AF		00 FB 00018	CALLS	#0, ANLSREPORT_PAGE	: 0764
		02		6C 91 C001C 1\$:	CMPB	(AP), #2	: 0768
				05 1F 0001F	BLSSU	2\$	
			08	AC D5 00021	TSTL	8(AP)	
				06 12 00024	BNEQ	3\$	
			0000*	CF B4 00026 2\$:	CLRW	REPORT_RAB+34	: 0769
				0B 11 0002A	BRB	4\$	
	0000*	CF		62 B0 0002C 3\$:	MOVW	(R2), REPORT_RAB+34	: 0771
	0000*	CF	04	A2 DC 00031	MOVL	4(R2), REPORT_RAB+40	: 0772
			0000*	CF 9F 00037 4\$:	PUSHAB	REPORT_RAB	: 0774
00000000G	00			01 FB 0003B	CALLS	#1, SYSSPUT	
	19			50 EB 00042	BLBS	STATUS, 5\$: 0775
			0000*	CF DD 00045	PUSHL	REPORT_RAB+12	
				50 DD 00049	PUSHL	STATUS	
			0000*	CF 9F 0004B	PUSHAB	REPORT_SPEC	
				01 DD 0004F	PUSHL	#1	
	00000000G	00	00B110D4	8F DD 00051	PUSHL	#11604180	
				05 FB 00057	CALLS	#5, LIBSSIGNAL	
			0000*	CF D7 0005E 5\$:	DECL	LINE_COUNTER	: 0779
				04 00062 6\$:	RET		: 0783

; Routine Size: 99 bytes, Routine Base: \$CODE\$ + 00FB

```

354 0784 1 %sbttl 'ANLS$FORMAT_LINE - Format Line for Report'
355 0785 1 **
356 0786 1 : Functional Description:
357 0787 1 : This routine is called to format a line and print it in the
358 0788 1 : report file.
359 0789 1 :
360 0790 1 : Formal Parameters:
361 0791 1 : widow_control The number of lines that must be remaining on the
362 0792 1 : current page.
363 0793 1 : indent_level The number of tab stops to indent the line.
364 0794 1 : template_msg The status code of the message defining the line
365 0795 1 : template.
366 0796 1 : faol... $FAO arguments to fill in the template.
367 0797 1 :
368 0798 1 : Implicit Inputs:
369 0799 1 : global data
370 0800 1 :
371 0801 1 : Implicit Outputs:
372 0802 1 : global data
373 0803 1 :
374 0804 1 : Returned Value:
375 0805 1 : none
376 0806 1 :
377 0807 1 : Side Effects:
378 0808 1 :
379 0809 1 : --
380 0810 1 :
381 0811 1 :
382 0812 2 global routine anl$format_line(widow_control,indent_level,template_msg,faol): novalue = begin
383 0813 2
384 0814 2 local
385 0815 2 status: long;
386 0816 2
387 0817 2
388 0818 2 : First we obtain the text of the template message.
389 0819 2
390 0820 3 begin
391 0821 3 local
392 0822 3 local_described_buffer(template_buf,132);
393 0823 3
394 P 0824 3 status = $getmsg(msgid=.template_msg,
395 P 0825 3 msglen=template_buf,
396 P 0826 3 bufadr=template_buf,
397 0827 3 flags=%b'0001');
398 0828 3 check (.status,.status);
399 0829 3
400 0830 3 : Now we can plug the $FAO arguments into the message template.
401 0831 3
402 0832 4 begin
403 0833 4 local
404 0834 4 local_described_buffer(result_buf,132);
405 0835 4
406 P 0836 4 status = $faol(ctrstr=template_buf,
407 P 0837 4 outlen=result_buf,
408 P 0838 4 outbuf=result_buf,
409 0839 4 prmlst=faol);
410 0840 4 check (.status,.status);

```

```

: 411 0841 4
: 412 0842 4 ' Prefix the resulting text with enough tabs to effect the indentation.
: 413 0843 4
: 414 0844 4 ch$move(.result_buf[.len],.result_buf[ptr],.result_buf[ptr]+.indent_level);
: 415 0845 4 result_buf[.len]=.result_buf[.len]+.indent_level;
: 416 0846 4 ch$fill(%char(tab),.indent_level,.result_buf[ptr]);
: 417 0847 4
: 418 0848 4 . Print the line, passing along the widow control number.
: 419 0849 4
: 420 0850 4 anl$report_line(.widow_control,result_buf);
: 421 0851 4
: 422 0852 3 end;
: 423 0853 2 end;
: 424 0854 2
: 425 0855 2 return;
: 426 0856 2
: 427 0857 1 end;

```

.EXTRN SYSSGETMSG, SYSSFAOL

				007C 00000	.ENTRY ANL\$FORMAT_LINE, Save R2,R3,R4,R5,R6	: 0812
	56	00000000G	00	9E 00002	MOVAB LIB\$SIGNAL, R6	
	5E	FEEB	CE	9E 00009	MOVAB -280(SP), SP	
	FF74	CD	84	8F 9A 0000E	MOVZBL #132, TEMPLATE_BUF	: 0822
	FF78	CD	FF7C	CD 9E 00014	MOVAB TEMPLATE_BUF+8, TEMPLATE_BUF+4	
			7E	01 7D 0001B	MOVQ #1, -(SP)	: 0827
			FF74	CD 9F 0001E	PUSHAB TEMPLATE_BUF	
			FF74	CD 9F 00022	PUSHAB TEMPLATE_BUF	
			0C	AC DD 00026	PUSHL TEMPLATE_MSG	
	00000000G	00	05	FB 00029	CALLS #5, SYSSGETMSG	
		52	50	DD 00030	MOVL R0, STATUS	
		05	52	E8 00033	BLBS STATUS, 1\$: 0828
			52	DD 00036	PUSHL STATUS	
		66	01	FB 00038	CALLS #1, LIB\$SIGNAL	
		6E	84	8F 9A 0003B	MOVZBL #132, RESULT_BUF	: 0834
	04	AE	08	AE 9E 0003F	MOVAB RESULT_BUF+8, RESULT_BUF+4	
			10	AC 9F 00044	PUSHAB FA01	: 0839
			04	AE 9F 00047	PUSHAB RESULT_BUF	
			08	AE 9F 0004A	PUSHAB RESULT_BUF	
			FF74	CD 9F 0004D	PUSHAB TEMPLATE_BUF	
	00000000G	00	04	FB 00051	CALLS #4, SYSSFAOL	
		52	50	DD 00058	MOVL R0, STATUS	
		05	52	E8 0005B	BLBS STATUS, 2\$: 0840
			52	DD 0005E	PUSHL STATUS	
		66	01	FB 00060	CALLS #1, LIB\$SIGNAL	
	50	04	AE	08 AC C1 00063	ADDL3 INDENT_LEVEL, RESULT_BUF+4, R0	: 0844
	60	04	BE	6E 28 00069	MOVCS RESULT_BUF, @RESULT_BUF+4, (R0)	
			6E	08 AC A0 0006E	ADDW2 INDENT_LEVEL, RESULT_BUF	: 0845
08	AC	09	6E	00 2C 00072	MOVCS #0, (SP), #9, INDENT_LEVEL, @RESULT_BUF+4	: 0846
			04	BE 00078		
			5E	DD 0007A	PUSHL SP	: 0850
			04	AC DD U007C	PUSHL WIDOW_CONTROL	
	FF19	CF	02	FB 0007F	CALLS #2, ANL\$REPORT_LINE	: 0857
			04	00084	RET	

OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANLS\$FORMAT_LINE - Format Line for Report

; Routine Size: 133 bytes, Routine Base: \$CODE\$ + 015E

M 9
15-Sep-1984 23:36:57
14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXEOUT.B32:1

Page 19
(6)

7
)

OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANL\$FORMAT_ERROR - Put Error Message in Report

N 9
15-Sep-1984 23:36:57
14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXEOUT.B32:1

Page 20
(7)

```

: 429 0858 1 %sbttl 'ANL$FORMAT_ERROR - Put Error Message in Report'
: 430 0859 1 **
: 431 0860 1 Functional Description:
: 432 0861 1 This routine is called to format an error message into the report
: 433 0862 1 file.
: 434 0863 1
: 435 0864 1 Formal Parameters:
: 436 0865 1 error_msg Status code for the error message.
: 437 0866 1 fao1... $FAO substitution parameters for the message.
: 438 0867 1
: 439 0868 1 Implicit Inputs:
: 440 0869 1 global data
: 441 0870 1
: 442 0871 1 Implicit Outputs:
: 443 0872 1 global data
: 444 0873 1
: 445 0874 1 Returned Value:
: 446 0875 1 none
: 447 0876 1
: 448 0877 1 Side Effects:
: 449 0878 1
: 450 0879 1 --
: 451 0880 1
: 452 0881 1
: 453 0882 2 global routine anl$format_error(error_msg,fao1,fao2,fao3,fao4): novalue - begin
: 454 0883 2
: 455 0884 2 bind
: 456 0885 2 flag_string = describe('*** ');
: 457 0886 2
: 458 0887 2 builtin
: 459 0888 2 actualcount;
: 460 0889 2
: 461 0890 2
: 462 0891 2 ! We case on the number of $FAO parameters and call ANL$FORMAT_LINE to
: 463 0892 2 ! do the work. In all cases, however, we add our own first parameter,
: 464 0893 2 ! which is the error message flag string.
: 465 0894 2
: 466 0895 2 case actualcount() from 1 to 5 of set
: 467 0896 2 [1]: anl$format_line(-1,0,.error_msg,flag_string);
: 468 0897 2 [2]: anl$format_line(-1,0,.error_msg,flag_string,.fao1);
: 469 0898 2 [3]: anl$format_line(-1,0,.error_msg,flag_string,.fao1,.fao2);
: 470 0899 2 [4]: anl$format_line(-1,0,.error_msg,flag_string,.fao1,.fao2,.fao3);
: 471 0900 2 [5]: anl$format_line(-1,0,.error_msg,flag_string,.fao1,.fao2,.fao3,.fao4);
: 472 0901 2 tes;
: 473 0902 2
: 474 0903 2 ! Keep track of the number of errors reported. Also keep track of
: 475 0904 2 ! most severe error which has occurred.
: 476 0905 2
: 477 0906 2 increment (error_count);
: 478 0907 2 if severity_level (.error_msg) gtr
: 479 0908 3 severity_level (.anl$worst_error) ! If higher than watermark
: 480 0909 2 then anl$worst_error = .error_msg; ! -then set new worst error
: 481 0910 2
: 482 0911 2 return;
: 483 0912 2
: 484 0913 1 end;

```

```
.PSECT $SPLITS,NOWRT,NOEXE,2
20 20 2A 2A 2A 00028 P.AAG: .ASCII \*** \
                                .BLKB 3
                                00000005 00030 P.AAF: .LONG 5
                                00000000 00034 .ADDRESS P.AAG
                                FLAG_STRING= P.AAF

.PSECT $CODES,NOWRT,2
.ENTRY ANLS$FORMAT_ERROR, Save R2,R3,R4,R5
55 0000 003C 00000 .ENTRY ANLS$FORMAT_ERROR, Save R2,R3,R4,R5 : 0882
54 FF70 CF 9E 00002 MOVAB FLAG_STRING, R5
52 04 AC DO 00007 MOVAB ANLS$FORMAT_LINE, R4
01 6C 8F C0010 MOVL ERROR_MSG, R2
0035 0025 0016 000A 00014 1$: CASEB (AP) #1, #4 : 0896
                                0048 0001C .WORD : 0895
                                2$: PUSHR #^M<R2,R5> : 0896
                                7E 7E D4 00020 CLRL -(SP)
                                64 01 CE 00022 MNEGL #1, -(SP)
                                04 FB 00025 CALLS #4, ANLS$FORMAT_LINE
                                44 11 00028 BRB 7$ : 0897
                                08 AC DD 0002A 3$: PUSHL FA01
                                24 BB 0002D PUSHR #^M<R2,R5>
                                7E 7E D4 0002F CLRL -(SP)
                                7E 01 CE 00031 MNEGL #1, -(SP)
                                64 05 FB 00034 CALLS #5, ANLS$FORMAT_LINE
                                35 11 00037 BRB 7$ : 0898
                                7E 08 AC 7D 00039 4$: MOVQ FA01, -(SP)
                                24 BB 0003D PUSHR #^M<R2,R5>
                                7E 7E D4 0003F CLRL -(SP)
                                7E 01 CE 00041 MNEGL #1, -(SP)
                                64 06 FB 00044 CALLS #6, ANLS$FORMAT_LINE
                                25 11 00047 BRB 7$ : 0899
                                7E 0C AC 7D 00049 5$: MOVQ FA02, -(SP)
                                08 AC DD 0004D PUSHL FA01
                                24 BB 00050 PUSHR #^M<R2,R5>
                                7E 7E D4 00052 CLRL -(SP)
                                7E 01 CE 00054 MNEGL #1, -(SP)
                                64 07 FB 00057 CALLS #7, ANLS$FORMAT_LINE
                                12 11 0005A BRB 7$ : 0900
                                7E 10 AC 7D 0005C 6$: MOVQ FA03, -(SP)
                                7E 08 AC 7D 00060 MOVQ FA01, -(SP)
                                24 BB 00064 PUSHR #^M<R2,R5>
                                7E 7E D4 00066 CLRL -(SP)
                                7E 01 CE 00068 MNEGL #1, -(SP)
                                64 08 FB 0006B CALLS #8, ANLS$FORMAT_LINE
                                0000 CF D6 0006E 7$: INCL ERROR_COUNT : 0906
                                50 52 DO 00072 MOVL R2, TMP_CODE : 0907
                                03 00 EF 00075 EXTZV #0, #3, TMP_CODE, R1
```

OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANL\$FORMAT_ERROR - Put Error Message in Report

C 10
15-Sep-1984 23:36:57
14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXEOUT.B32;1

Page 22
(7)

50	50	01	00	EF	0007A	EXTZV	#0, #1, TMP_CODE, R0	
		50	04	C4	0007F	MULL2	#4, R0	
		51	50	C2	00082	SUBL2	R0, R1	
		51	03	C0	00085	ADDL2	#3, R1	
		50	0000'	CF	D0	00088	MOVL	ANL\$WORST_ERROR, TMP_CODE
53	50	03	00	EF	0008D	EXTZV	#0, #3, TMP_CODE, R3	0908
50	50	01	00	EF	00092	EXTZV	#0, #1, TMP_CODE, R0	
		50	04	C4	00097	MULL2	#4, R0	
		53	50	C2	0009A	SUBL2	R0, R3	
		50	03	A3	9E	0009D	MOVAB	3(R3), R0
		50	51	D1	000A1	CMPL	R1, R0	
			05	15	000A4	BLEQ	8\$	
		0000'	CF	52	D0	000A6	MOVL	R2, ANL\$WORST_ERROR
				04	000AB	8\$:	RET	0909
								0913

; Routine Size: 172 bytes, Routine Base: \$CODE\$ + 01E3


```

: 486 0914 1 %sbttl 'ANL$ERROR_COUNT - Report Count of Errors'
: 487 0915 1 **
: 488 0916 1 Functional Description:
: 489 0917 1 This routine is called to print a line telling how many errors
: 490 0918 1 were discovered during the analysis.
: 491 0919 1
: 492 0920 1 Formal Parameters:
: 493 0921 1 none
: 494 0922 1
: 495 0923 1 Implicit Inputs:
: 496 0924 1 global data
: 497 0925 1
: 498 0926 1 Implicit Outputs:
: 499 0927 1 global data
: 500 0928 1
: 501 0929 1 Returned Value:
: 502 0930 1 none
: 503 0931 1
: 504 0932 1 Side Effects:
: 505 0933 1
: 506 0934 1 --
: 507 0935 1
: 508 0936 1
: 509 0937 2 global routine anl$error_count: novalue = begin
: 510 0938 2
: 511 0939 2
: 512 0940 2 ! first we print the error count in the report.
: 513 0941 2
: 514 0942 2 if .error_count eqv 0 then
: 515 0943 2     anl$format_line(0,0,anlobj$_errornone)
: 516 0944 2 else
: 517 0945 2     anl$format_line(0,0,anlobj$_errorcount,.error_count);
: 518 0946 2 anl$report_line(0);
: 519 0947 2 anl$report_line(0);
: 520 0948 2
: 521 0949 2 ! If the report is not going to SYSS$OUTPUT, we also want to display one line
: 522 0950 2 ! for the user at the terminal. This contains the report heading text and
: 523 0951 2 ! the error count.
: 524 0952 2
: 525 0953 2 if ch$neq(.report_spec[ptr],.report_spec[ptr], 10,uplit byte('SYSS$OUTPUT'),' ') then
: 526 0954 2     signal(anlobj$_errors,2,input_file_spec,.error_count);
: 527 0955 2
: 528 0956 2 ! Now we can reset the error counter for the next file.
: 529 0957 2
: 530 0958 2 error_count = 0;
: 531 0959 2
: 532 0960 2 return;
: 533 0961 2
: 534 0962 1 end;

```

.PSECT SPLITS,NOWRT,NOEXE,2

54 55 50 54 55 4F 24 53 59 53 00038 P.AAH: .ASCII \SYSS\$OUTPUT\ ;

				.PSECT	\$CODE\$,NOWRT,2	
			003C 00000	.ENTRY	ANL\$ERROR_COUNT, Save R2,R3,R4,R5	: 0937
55	FEC9	CF	9E 00002	MOVAB	ANL\$FORMAT_LINE, R5	: 0938
54	0000*	CF	9E 00007	MOVAB	ERROR_COUNT, R4	: 0939
50		64	DD 0000C	MOVL	ERROR_COUNT, R0	: 0942
		0D	12 0000F	BNEQ	1\$: 0943
	00000000G	8F	DD 00011	PUSHL	#ANL\$OBJ\$_ERRORNONE	: 0944
		7E	7C 00017	CLRQ	-(SP)	: 0945
65		03	FB 00019	CALLS	#3, ANL\$FORMAT_LINE	: 0946
		0D	11 0001C	BRB	2\$: 0947
	00000000G	50	DD 0001E 1\$:	PUSHL	R0	: 0948
		8F	DD 00020	PUSHL	#ANL\$OBJ\$_ERRORCOUNT	: 0949
		7E	7C 00026	CLRQ	-(SP)	: 0950
65		04	FB 00028 2\$:	CALLS	#4, ANL\$FORMAT_LINE	: 0951
		7E	D4 0002B	CLRL	-(SP)	: 0952
	9D A5	01	FB 0002D	CALLS	#1, ANL\$REPORT_LINE	: 0953
		7E	D4 00031	CLRL	-(SP)	: 0954
	9D A5	01	FB 00033	CALLS	#1, ANL\$REPORT_LINE	: 0955
0A	20	FD50 D4	FD4C 0000*	CMPC5	REPORT_SPEC, @REPORT_SPEC+4, #32, #10, -	: 0956
			CF 00040	P.AAH		: 0957
		15	13 00043	BEQL	3\$: 0958
		64	DD 00045	PUSHL	ERROR_COUNT	: 0959
		C4	9F 00047	PUSHAB	INPUT_FILE_SPEC	: 0960
		02	DD 0004B	PUSHL	#2	: 0961
	00000000G C0	8F	DD 0004D	PUSHL	#ANL\$OBJ\$_ERRORS	: 0962
		04	FB 00053	CALLS	#4, LIB\$SIGNAL	: 0963
		64	D4 0005A 3\$:	CLRL	ERROR_COUNT	: 0964
		04	0005C	RET		: 0965

; Routine Size: 93 bytes, Routine Base: \$CODE\$ + 028F

```

: 536 0963 1 %sbttl 'ANL$EXIT_WITH_STATUS - Exit to VMS with a Status'
: 537 0964 1 **
: 538 0965 1 Functional Description:
: 539 0966 1 This routine is called when it's time to exit back to VMS. We
: 540 0967 1 exit with the status in anl$worst_error. (This contains
: 541 0968 1 success status if no errors have occurred.)
: 542 0969 1
: 543 0970 1 Formal Parameters:
: 544 0971 1 none
: 545 0972 1
: 546 0973 1 Implicit Inputs:
: 547 0974 1 global data
: 548 0975 1
: 549 0976 1 Implicit Outputs:
: 550 0977 1 global data
: 551 0978 1
: 552 0979 1 Returned Value:
: 553 0980 1 does not return
: 554 0981 1
: 555 0982 1 Side Effects:
: 556 0983 1
: 557 0984 1 --
: 558 0985 1
: 559 0986 1
: 560 0987 2 global routine anl$exit_with_status: novalue = begin
: 561 0988 2
: 562 0989 2 ! if it was an interactive session, always return success. otherwise
: 563 0990 2 ! return worst error
: 564 0991 2
: 565 0992 2 if .anl$gb_interactive then
: 566 0993 3     $exit(code=anlobj$_ok)
: 567 0994 2 else
: 568 0995 2     $exit(code=.anl$worst_error or sts$m_inhib_msg);
: 569 0996 2
: 570 0997 1 end;

```

```

                                .EXTRN  SYS$EXIT
                                .ENTRY  ANL$EXIT WITH STATUS, Save nothing
                                BLBC    ANL$GB INTERACTIVE, i$
                                PUSHL   #ANLOBJ$_OK
                                BRB     2$
                                BISL3   #268435456, ANL$WORST_ERROR, -(SP)
                                CALLS   #1, SYS$EXIT
                                RET
                                : 0987
                                : 0992
                                : 0993
                                :
                                : 0995
                                :
                                : 0997

```

; Routine Size: 33 bytes, Routine Base: \$CODE\$ + 02EC

```

: 572 0998 1 %sbttl 'ANLSFORMAT_HEX - Format Hex Dump of Data'
: 573 0999 1 **
: 574 1000 1 : Functional Description:
: 575 1001 1 :   This routine is called to format a hex dump of some bytes.
: 576 1002 1 :   It includes the character representation of the bytes also.
: 577 1003 1
: 578 1004 1 : Formal Parameters:
: 579 1005 1 :   indent_level   The indentation level at which to place the dump.
: 580 1006 1 :   data           Address of descriptor of data to be dumped.
: 581 1007 1
: 582 1008 1 : Implicit Inputs:
: 583 1009 1 :   global data
: 584 1010 1
: 585 1011 1 : Implicit Outputs:
: 586 1012 1 :   global data
: 587 1013 1
: 588 1014 1 : Returned Value:
: 589 1015 1 :   none
: 590 1016 1
: 591 1017 1 : Side Effects:
: 592 1018 1
: 593 1019 1 :--
: 594 1020 1
: 595 1021 1
: 596 1022 2 global routine anl$format_hex(indent_level,data): novalue = begin
: 597 1023 2
: 598 1024 2 bind
: 599 1025 2     data_dsc = .data: descriptor,
: 600 1026 2     data_vector = .data_dsc[ptr]: vector[,byte];
: 601 1027 2
: 602 1028 2 local
: 603 1029 2     i: long,
: 604 1030 2     arg_list: vector[20,long],
: 605 1031 2     count: long;
: 606 1032 2
: 607 1033 2 builtin
: 608 1034 2     callg;
: 609 1035 2
: 610 1036 2
: 611 1037 2 : If the data is null, just quit.
: 612 1038 2
: 613 1039 2 if .data_dsc[len] eqlu 0 then
: 614 1040 2     return;
: 615 1041 2
: 616 1042 2 : We begin by printing two heading lines. The first shows the offsets
: 617 1043 2 : of the bytes and the second is a line of dashes.
: 618 1044 2
: 619 1045 2 anl$format_line(3,.indent_level,anlobj$_hexheading1);
: 620 1046 2 anl$format_line(0,.indent_level,anlobj$_hexheading2);
: 621 1047 2
: 622 1048 2 : We will be building argument lists to ANLSFORMAT LINE. It will always
: 623 1049 2 : include widow control, indentation level, and the message code.
: 624 1050 2
: 625 1051 2 arg_list[1] = 0;
: 626 1052 2 arg_list[2] = .indent_level;
: 627 1053 2 arg_list[3] = anlobj$_hexdata;
: 628 1054 2

```

```

: 629      1055 2 ! Now we go into a loop, once through for each 8 bytes to be formatted.
: 630      1056 2
: 631      1057 2 i = 0;
: 632      1058 2 while .i lssu .data_dsc[len] do (
: 633      1059 2
: 634      1060 2     ! Calculate the number of bytes that will go on this line.
: 635      1061 2
: 636      1062 2     count = minu(.data_dsc[len]-.i,8);
: 637      1063 2
: 638      1064 2     ! Next in the argument list we need a count of the spaces to skip
: 639      1065 2     ! so the bytes will be lined up from right to left.
: 640      1066 2
: 641      1067 2     arg_list[4] = (8 - .count) * 3;
: 642      1068 2
: 643      1069 2     ! Now we need the count itself.
: 644      1070 2
: 645      1071 2     arg_list[5] = .count;
: 646      1072 2
: 647      1073 2     ! Now we loop through 8 (or less) bytes and put them in the
: 648      1074 2     ! argument list (backwards, of course).
: 649      1075 2
: 650      1076 2     decr j from .count-1 to 0 do (
: 651      1077 2         arg_list[6+.j] = .data_vector[.i];
: 652      1078 2         increment (i);
: 653      1079 2     );
: 654      1080 2
: 655      1081 2     ! Next we have the byte offset.
: 656      1082 2
: 657      1083 2     arg_list[6+.count] = .i - .count;
: 658      1084 2
: 659      1085 2     ! Now we have to add to the argument list the byte count and a
: 660      1086 2     ! pointer to the byte string.
: 661      1087 2
: 662      1088 2     arg_list[7+.count] = .count;
: 663      1089 2     arg_list[8+.count] = data_vector[.i - .count];
: 664      1090 2
: 665      1091 2     ! Finally, fill in the argument count.
: 666      1092 2
: 667      1093 2     arg_list[0] = 8 + .count;
: 668      1094 2
: 669      1095 2     ! Now we can print the hex data.
: 670      1096 2
: 671      1097 2     callg(arg_list,anl$format_line);
: 672      1098 2 );
: 673      1099 2
: 674      1100 2 return;
: 675      1101 2
: 676      1102 1 end;

```

```

55      FE4B      CF 9E 00002
5E      B0       AE 9E 00007
54      08       AC D0 0000B

```

```

.ENTRY ANL$FORMAT_HEX, Save R2,R3,R4,R5
MOVAB ANL$FORMAT_LINE, R5
MOVAB -80(SP), SP
MOVL DATA, R4

```

```

: 1022
:
: 1025

```

			64	B5	0000F	TSTW	(R4)		1039	
			33	13	00011	BEQL	2\$			
		00000000G	8F	DD	00013	PUSHL	#ANLOBS_HEXHEADING1		1045	
		04	AC	DD	00019	PUSHL	INDENT_LEVEL			
			03	DD	0001C	PUSHL	#3			
		65	03	FB	0001E	CALLS	#3, ANLSFORMAT_LINE			
		00000000G	8F	DD	00021	PUSHL	#ANLOBS_HEXHEADING2		1046	
		04	AC	DD	00027	PUSHL	INDENT_LEVEL			
			7E	D4	0002A	CLRL	-(SP)			
		65	03	FB	0002C	CALLS	#3, ANLSFORMAT_LINE			
			04	AE	D4	0002F	CLRL	ARG_LIST+4	1051	
08	AE	04	AC	D0	00032	MOVL	INDENT_LEVEL, ARG_LIST+8		1052	
0C	AE	00000000G	8F	D0	00037	MOVL	#ANLOBS_HEXDATA, ARG_LIST+12		1053	
			53	D4	0003F	CLRL	I		1057	
53		64	10	00	ED	00041	1\$:	CMPZV	#0, #16, (R4), I	1058
				4F	1B	00046	2\$:	BLEQU	6\$	
			50	64	3C	00048		MOVZWL	(R4), R0	1062
			50	53	C2	0004B		SUBL2	I, R0	
			08	50	D1	0004E		CMP	R0, #8	
				03	1B	00051		BLEQU	3\$	
			50	08	D0	00053		MOVL	#8, R0	
			52	50	D0	00056	3\$:	MOVL	R0, COUNT	
			50	F8	A2	9E	00059	MOVAB	-8(R2), R0	1067
			50	03	C4	0005D		MULL2	#3, R0	
10	AE		50	CE	00060	MNEGL	R0, ARG_LIST+16			
14	AE		52	D0	00064	MOVL	COUNT, ARG_LIST+20		1071	
			50	52	D0	00068	MOVL	COUNT, J	1076	
				09	11	0006B		BRB	5\$	
18	AE40	04	B443	9A	0006D	4\$:	MOVZBL	@4(R4)[I], ARG_LIST+24[J]	1077	
			53	D6	00074		INCL	I	1078	
			F4	50	F4	00076	5\$:	SOBGEQ	J, 4\$	1076
		50	53	52	C3	00079		SUBL3	COUNT, I, R0	1083
18	AE42		50	D0	0007D	MOVL	R0, ARG_LIST+24[COUNT]			
1C	AE42		52	D0	00082	MOVL	COUNT, ARG_LIST+28[COUNT]		1088	
20	AE42	04	B440	9E	00087	MOVAB	@4(R4)[R0], ARG_LIST+32[COUNT]		1089	
			08	A2	9E	0008E	MOVAB	8(R2), ARG_LIST	1093	
			6E	FA	00092	CALLG	ARG_LIST, ANLSFORMAT_LINE		1097	
			65	AA	11	00095	BRB	1\$	1058	
				04	00097	6\$:	RET		1102	

; Routine Size: 152 bytes, Routine Base: \$CODE\$ + 030D

```

: 678      1103 1 %sbttl 'ANL$FORMAT_FLAGS - Format Flag Bits'
: 679      1104 1 **
: 680      1105 1 Functional Description:
: 681      1106 1     This routine is called to format the flags in a byte/word/longword
: 682      1107 1     of flags.
: 683      1108 1
: 684      1109 1 Formal Parameters:
: 685      1110 1     indent_level    The level at which the introductory message is to
: 686      1111 1     intro_msg        be indented. The flags are indented one more level.
: 687      1112 1     flags            The introductory message.
: 688      1113 1     flag_def         The flag bits.
: 689      1114 1     flag_def         A longword vector defining the flags. The zeroth
: 690      1115 1     flag_def         entry specifies the highest-numbered flag. The
: 691      1116 1     flag_def         remaining longwords contain the address of a counted
: 692      1117 1     flag_def         string giving the name of the flag. If the flag is
: 693      1118 1     flag_def         undefined, the longword contains zero.
: 694      1119 1
: 695      1120 1 Implicit Inputs:
: 696      1121 1     global data
: 697      1122 1
: 698      1123 1 Implicit Outputs:
: 699      1124 1     global data
: 700      1125 1
: 701      1126 1 Returned Value:
: 702      1127 1     none
: 703      1128 1
: 704      1129 1 Side Effects:
: 705      1130 1
: 706      1131 1 --
: 707      1132 1
: 708      1133 1
: 709      1134 2 global routine anl$format_flags(indent_level,intro_msg,flags,flag_def): novalue - begin
: 710      1135 2
: 711      1136 2 bind
: 712      1137 2     flags_vector = flags: bitvector[],
: 713      1138 2     flag_def_vector = .flag_def: vector[,long];
: 714      1139 2
: 715      1140 2 local
: 716      1141 2     i: long;
: 717      1142 2
: 718      1143 2
: 719      1144 2 ! Begin by printing the introductory message.
: 720      1145 2
: 721      1146 2 anl$format_line(2,..indent_level,..intro_msg);
: 722      1147 2
: 723      1148 2 ! Now we loop through the flags and process each one that is defined.
: 724      1149 2 ! We print the flag name, bit number, and current setting.
: 725      1150 2
: 726      1151 3 incru i from 0 to .flag_def_vector[0] do (
: 727      1152 3     if .flag_def_vector[i+1] nequ 0 then
: 728      1153 3         anl$format_line(0,..indent_level+1,anlobj$flag,
: 729      1154 3             .i,..flag_def_vector[i+1],.flags_vector[i]);
: 730      1155 2 );
: 731      1156 2
: 732      1157 2 return;
: 733      1158 2
: 734      1159 1 end;

```

				0004 00000	.ENTRY	ANLS\$FORMAT_FLAGS, Save R2	:	1134
	7E	04	AC	7D 00002	MOVQ	INDENT_LEVEL, -(SP)	:	1146
			02	DD 00006	PUSHL	#2	:	
	FDAC	CF		03 FB 00008	CALLS	#3, ANLS\$FORMAT_LINE	:	
				52 D4 0000D	CLRL	I	:	1151
				29 11 00C7F	BRB	3\$:	
	50		10 BC	42 DE 00011	MOVAL	@FLAG_DEF[I], R0	:	1152
			04	A0 D5 00016	TSTL	4(R0)	:	
				1D 13 00019	BEQL	2\$:	
7F	OC	AC		52 EF 0001B	EXTZV	I, #1, FLAGS_VECTOR, -(SP)	:	1154
				04 A0 DD 00021	PUSHL	4(R0)	:	
				52 DD 00024	PUSHL	I	:	
			00000000G	8F DD 00026	PUSHL	#ANLOBJ\$ FLAG	:	1153
	7E	04	AC	01 C1 0002C	ADDL3	#1, INDENT_LEVEL, -(SP)	:	
				7E D4 00031	CLRL	-(SP)	:	
	FDB1	CF		06 FB 00033	CALLS	#6, ANLS\$FORMAT_LINE	:	
				52 D6 00038	INCL	I	:	1151
	10	BC		52 D1 0003A	CMP	I, @FLAG_DEF	:	
				D1 1B 0003E	BLEQU	1\$:	1159
				04 00040	RET		:	

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 03A5


```

: 736 1160 1 %sbttl 'ANLS$FORMAT_DATA_TYPE - Format a Data Type'
: 737 1161 1 : **
: 738 1162 1 . Functional Description:
: 739 1163 1   This routine is called to format a nice line for a data type,
: 740 1164 1   as defined in the VAX architecture manual.
: 741 1165 1
: 742 1166 1 . Formal Parameters:
: 743 1167 1   indent_level   The level of indentation for the line.
: 744 1168 1   data_type      The data type byte.
: 745 1169 1
: 746 1170 1 . Implicit Inputs:
: 747 1171 1   global data
: 748 1172 1
: 749 1173 1 . Implicit Outputs:
: 750 1174 1   global data
: 751 1175 1
: 752 1176 1 . Returned Value:
: 753 1177 1   none
: 754 1178 1
: 755 1179 1 . Side Effects:
: 756 1180 1
: 757 1181 1 --
: 758 1182 1
: 759 1183 1
: 760 1184 2 global routine anl$format_data_type(indent_level,data_type): novalue = begin
: 761 1185 2
: 762 1186 2
: 763 1187 2 own
: 764 1188 2   data_type_table: vector[33,long] initial(
: 765 1189 2       uplit byte(%ascic 'Z'),
: 766 1190 2       uplit byte(%ascic 'V'),
: 767 1191 2       uplit byte(%ascic 'BU'),
: 768 1192 2       uplit byte(%ascic 'WU'),
: 769 1193 2       uplit byte(%ascic 'LU'),
: 770 1194 2       uplit byte(%ascic 'QU'),
: 771 1195 2       uplit byte(%ascic 'B'),
: 772 1196 2       uplit byte(%ascic 'W'),
: 773 1197 2       uplit byte(%ascic 'L'),
: 774 1198 2       uplit byte(%ascic 'O'),
: 775 1199 2       uplit byte(%ascic 'F'),
: 776 1200 2       uplit byte(%ascic 'D'),
: 777 1201 2       uplit byte(%ascic 'FC'),
: 778 1202 2       uplit byte(%ascic 'DC'),
: 779 1203 2       uplit byte(%ascic 'T'),
: 780 1204 2       uplit byte(%ascic 'NU'),
: 781 1205 2       uplit byte(%ascic 'NL'),
: 782 1206 2       uplit byte(%ascic 'NLO'),
: 783 1207 2       uplit byte(%ascic 'NR'),
: 784 1208 2       uplit byte(%ascic 'NRO'),
: 785 1209 2       uplit byte(%ascic 'NZ'),
: 786 1210 2       uplit byte(%ascic 'P'),
: 787 1211 2       uplit byte(%ascic 'ZI'),
: 788 1212 2       uplit byte(%ascic 'ZEM'),
: 789 1213 2       uplit byte(%ascic 'DSC'),
: 790 1214 2       uplit byte(%ascic 'OU'),
: 791 1215 2       uplit byte(%ascic 'O'),
: 792 1216 2       uplit byte(%ascic 'G'),

```

```

: 793      1217 2          uplit byte(%ascic 'M'),
: 794      1218 2          uplit byte(%ascic 'GC'),
: 795      1219 2          uplit byte(%ascic 'HC'),
: 796      1220 2          uplit byte(%ascic 'CIT'),
: 797      1221 2          uplit byte(%ascic 'BPV'));
: 798      1222 2
: 799      1223 2
800      1224 2          ! If it is a standard data type, print it's name and number.  Otherwise just
801      1225 2          ! use the number.
802      1226 2
803      1227 2          anl$format_line(0,indent_level,anlobj$ datatype,
804      1228 2          (if .data_type (ssu %allocation(data_type_table)/4 then .data_type_table[.data_type]
805      1229 2          .data_type);
806      1230 2
807      1231 2          return;
808      1232 2
809      1233 2
: 810      1234 1          end;

```

				.PSECT	\$SPLITS	NOVRT	NOEXE	2
	5A	01	00042	P.AAI:	.ASCII	<1>	\Z\	
	56	01	00044	P.AAJ:	.ASCII	<1>	\V\	
55	42	02	00046	P.AAK:	.ASCII	<2>	\BU\	
55	57	02	00049	P.AAL:	.ASCII	<2>	\WU\	
55	4C	02	0004C	P.AAM:	.ASCII	<2>	\LU\	
55	51	02	0004F	P.AAN:	.ASCII	<2>	\QU\	
	42	01	00052	P.AAO:	.ASCII	<1>	\B\	
	57	01	00054	P.AAP:	.ASCII	<1>	\W\	
	4C	01	00056	P.AAQ:	.ASCII	<1>	\L\	
	51	01	00058	P.AAR:	.ASCII	<1>	\Q\	
	46	01	0005A	P.AAS:	.ASCII	<1>	\F\	
	44	01	0005C	P.AAT:	.ASCII	<1>	\D\	
43	46	02	0005E	P.AAU:	.ASCII	<2>	\FC\	
43	44	02	00061	P.AAV:	.ASCII	<2>	\DC\	
	54	01	00064	P.AAW:	.ASCII	<1>	\T\	
55	4E	02	00066	P.AAX:	.ASCII	<2>	\NU\	
4C	4E	02	00069	P.AAY:	.ASCII	<2>	\NL\	
4F	4C	03	0006C	P.AAZ:	.ASCII	<3>	\NLO\	
	52	02	00070	P.ABA:	.ASCII	<2>	\NR\	
4F	52	03	00073	P.ABB:	.ASCII	<3>	\NRO\	
	5A	02	00077	P.ABC:	.ASCII	<2>	\NZ\	
	50	01	0007A	P.ABD:	.ASCII	<1>	\P\	
49	5A	02	0007C	P.ABE:	.ASCII	<2>	\ZI\	
4D	45	03	0007F	P.ABF:	.ASCII	<3>	\ZEM\	
43	53	03	00083	P.ABG:	.ASCII	<3>	\DSC\	
	55	02	00087	P.ABH:	.ASCII	<2>	\OU\	
	4F	01	0008A	P.ABI:	.ASCII	<1>	\O\	
	47	01	0008C	P.ABJ:	.ASCII	<1>	\G\	
	48	01	0008E	P.ABK:	.ASCII	<1>	\H\	
43	47	02	00090	P.ABL:	.ASCII	<2>	\GC\	
43	48	02	00093	P.ABM:	.ASCII	<2>	\HC\	
54	49	03	00096	P.ABN:	.ASCII	<3>	\CIT\	
56	50	02	0009A	P.ABO:	.ASCII	<2>	\BPV\	
3F	3F	03	0009E	P.ABP:	.ASCII	<3>	\???\	

.....

.PSECT \$OWNS,NOEXE,2

00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	002B8	DATA_TYPE TABLE:	
00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	002D0	.ADDRESS	P.AAI, P.AAJ, P.AAK, P.AAL, P.AAM, -
00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	002E8		P.AAN, P.AAO, P.AAP, P.AAQ, P.AAR, P.AAS, -
00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	00300		P.AAT, P.AAU, P.AAV, P.AAW, P.AAX, P.AAY, -
00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	00318		P.AAZ, P.ABA, P.ABB, P.ABC, P.ABD, P.ABE, -
00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	00330		P.ABF, P.ABG, P.ABH, P.ABI, P.ABJ, P.ABK, -
								P.ABL, P.ABM, P.ABN, P.ABO

.PSECT \$CODE\$,NOWRT,2

			0000	00000		.ENTRY	ANLS\$FORMAT_DATA_TYPE, Save nothing	: 1184
50	08	AC	D0	00002		MOVL	DATA_TYPE, R0	: 1230
		50	DD	00006		PUSHL	R0	
21		50	D1	00008		CML	R0, #33	: 1228
		07	1E	0000B		BGEQU	1\$	
		0000*	CF40	DD 0000D		PUSHL	DATA_TYPE_TABLE[R0]	
		07	11	00012		BRB	2\$	
50	0000*	CF	9E	00014	1\$:	MOVAB	P.ABP, R0	: 1229
		50	DD	00019		PUSHL	R0	
		00000000G	8F	DD 0001B	2\$:	PUSHL	#ANL\$OBJ\$ DATATYPE	: 1227
		04	AC	DD 00021		PUSHL	INDENT_LEVEL	
		7E	D4	00024		CLRL	-(SP)	
FD4D	CF	05	FB	00026		CALLS	#5, ANLS\$FORMAT_LINE	
		04	04	0002B		RET		: 1234

; Routine Size: 44 bytes, Routine Base: \$CODE\$ + 03E6

```

: 812 1235 1 %sbttl 'ANL$FORMAT_MASK - Format an Entry Mask'
: 813 1236 1 ++
: 814 1237 1 Functional Description:
: 815 1238 1 This routine is called to format an entry mask word.
: 816 1239 1
: 817 1240 1 Formal Parameters:
: 818 1241 1 indent_level The level of indentation for the mask.
: 819 1242 1 mask The mask itself.
: 820 1243 1
: 821 1244 1 Implicit Inputs:
: 822 1245 1 global data
: 823 1246 1
: 824 1247 1 Implicit Outputs:
: 825 1248 1 global data
: 826 1249 1
: 827 1250 1 Returned Value:
: 828 1251 1 none
: 829 1252 1
: 830 1253 1 Side Effects:
: 831 1254 1
: 832 1255 1 --
: 833 1256 1
: 834 1257 1
: 835 1258 2 global routine anl$format_mask(indent_level,mask): novalue = begin
: 836 1259 2
: 837 1260 2 bind
: 838 1261 2 mask_vector = mask: bitvector[16];
: 839 1262 2
: 840 1263 2 own
: 841 1264 2 bit_name: vector[16,long] initial(
: 842 1265 2 'R0', 'R1', 'R2', 'R3',
: 843 1266 2 'R4', 'R5', 'R6', 'R7',
: 844 1267 2 'R8', 'R9', 'R10', 'R11',
: 845 1268 2 '--', '--', 'IV', 'DV');
: 846 1269 2
: 847 1270 2 local
: 848 1271 2 i: long,
: 849 1272 2 bit_name_len: long;
: 850 1273 2 local
: 851 1274 2 local_described_buffer(mask_buf,64);
: 852 1275 2
: 853 1276 2
: 854 1277 2 ! We are going to scan the entry mask and concatenate together the names
: 855 1278 2 ! of the bits that are on.
: 856 1279 2
: 857 1280 2 mask_buf[len] = 0;
: 858 1281 3 incru i from 0 to 15 do (
: 859 1282 4 if .mask_vector[.i] then (
: 860 1283 4 bit_name_len = (if .i eglu 10 or .i eglu 11 then 4 else 3);
: 861 1284 4 ch$move(.bit_name_len,bit_name[.i], .mask_buf[ptr]+.mask_buf[len]);
: 862 1285 4 mask_buf[len] = .mask_buf[len] + .bit_name_len;
: 863 1286 3 );
: 864 1287 2 );
: 865 1288 2
: 866 1289 2 ! If any bits were set in the mask, we will have a spurious trailing comma.
: 867 1290 2 ! Get rid of it.
: 868 1291 2

```

```

: 869      1292 2 if .mask_buf[len] gtru 0 then
: 870      1293 2     decrement (mask_buf[len]);
: 871      1294 2
: 872      1295 2 ! Now we can print the mask.
: 873      1296 2
: 874      1297 2 anl$format_line(0,.indent_level,anlobj$mask,mask_buf);
: 875      1298 2
: 876      1299 2 return;
: 877      1300 2
: 878      1301 1 end;

```

```

.PSECT $OWNS,NOEXE,2
00 2C 30 52 0033C BIT_NAME:
00 2C 31 52 00340 .ASCII \R0,\<0>
00 2C 32 52 00344 .ASCII \R1,\<0>
00 2C 33 52 00348 .ASCII \R2,\<0>
00 2C 34 52 0034C .ASCII \R3,\<0>
00 2C 35 52 00350 .ASCII \R4,\<0>
00 2C 36 52 00354 .ASCII \R5,\<0>
00 2C 37 52 00358 .ASCII \R6,\<0>
00 2C 38 52 0035C .ASCII \R7,\<0>
00 2C 39 52 00360 .ASCII \R8,\<0>
2C 30 31 52 00364 .ASCII \R9,\<0>
2C 31 31 52 00368 .ASCII \R10,\
00 2C 2D 2D 0036C .ASCII \--, \<0>
00 2C 2D 2D 00370 .ASCII \--, \<0>
00 2C 56 49 00374 .ASCII \IV,\<0>
00 2C 56 44 00378 .ASCII \DV,\<0>

```

```

.PSECT $CODE$,NOWRT,2
00FC 00000 .ENTRY ANL$FORMAT_MASK, Save R2,R3,R4,R5,R6,R7 : 1258
5E BC AE 9E 00002 MOVAB -68(SP), SP
7E 40 8F 9A 00006 MOVZBL #64, MASK_BUF : 1274
G4 AE 08 AE 9E 0000A MOVAB MASK_BUF+8, MASK_BUF+4
6E B4 0000F CLRW MASK_BUF : 1280
56 D4 00011 CLRL I : 1281
25 08 AC 56 E1 00013 1$: BBC I, MASK_VECTOR, 5$ : 1282
0A 56 D1 00018 CMPL I, #10 : 1283
05 13 0001B BEQL 2$
0B 56 D1 0001D CMPL I, #11
05 12 00020 BNEQ 3$
57 04 D0 00022 2$: MOVL #4, BIT_NAME_LEN
03 11 00025 BRB 4$
57 03 D0 00027 3$: MOVL #3, BIT_NAME_LEN
50 6E 3C 0002A 4$: MOVZWL MASK_BUF, R0 : 1284
50 04 AE C0 0002D ADDL2 MASK_BUF+4, R0
0000'CF 46 DF 00031 PUSHAL BIT_NAME[I]
60 9E 57 28 00036 MOVCL3 BIT_NAME_LEN, @ (SP)+, (R0)
6E 57 A0 0003A ADDW2 BIT_NAME_LEN, MASK_BUF : 1285
56 D6 0003D 5$: INCL I : 1281

```

OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANL\$FORMAT_MASK - format an Entry Mask

D 11
15-Sep-1984 23:36:57
14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXEOUT.B32;1

Page 36
(13)

OF	56	D1	0003F	CMP	I, #15	:
	CF	1B	00042	BLEQU	1\$:
	6E	B5	00044	TSTW	MASK_BUF	: 1292
	02	13	00046	BEQL	6\$:
	6E	B7	00048	DECW	MASK_BUF	: 1293
	5E	DD	0004A	PUSHL	SP	: 1297
00000000G	8F	DD	0004C	PUSHL	#ANLOBJ\$ MASK	:
04	AC	DD	00052	PUSHL	INDENT_LEVEL	:
	7E	D4	00055	CLRL	-(SP)	:
FCFO	CF	04	FB	CALLS	#4, ANL\$FORMAT_LINE	:
		04	0005C	RET		: 1301

; Routine Size: 93 bytes, Routine Base: \$CODE\$ + 0412

```

880 1302 1 %sbttl 'ANL$FORMAT_PROTECTION - Format Memory Protection Code'
881 1303 1 **
882 1304 1 Functional Description:
883 1305 1 This routine is responsible for formatting a 4-bit memory
884 1306 1 protection code in a nice way.
885 1307 1
886 1308 1 Formal Parameters:
887 1309 1 indent_level The level of indentation for the line.
888 1310 1 prot_code The 4-bit protection code.
889 1311 1
890 1312 1 Implicit Inputs:
891 1313 1 global data
892 1314 1
893 1315 1 Implicit Outputs:
894 1316 1 global data
895 1317 1
896 1318 1 Returned Value:
897 1319 1 none
898 1320 1
899 1321 1 Side Effects:
900 1322 1
901 1323 1 --
902 1324 1
903 1325 1
904 1326 2 global routine anl$format_protection(indent_level,prot_code): novalue = begin
905 1327 2
906 1328 2 own
907 1329 2 prot_code_table: vector[16,long] initial(
908 1330 2 uplit byte (%ascic 'NA'),
909 1331 2 uplit byte (%ascic '???'),
910 1332 2 uplit byte (%ascic 'KW'),
911 1333 2 uplit byte (%ascic 'KR'),
912 1334 2 uplit byte (%ascic 'UW'),
913 1335 2 uplit byte (%ascic 'EW'),
914 1336 2 uplit byte (%ascic 'ERKW'),
915 1337 2 uplit byte (%ascic 'ER'),
916 1338 2 uplit byte (%ascic 'SW'),
917 1339 2 uplit byte (%ascic 'SREW'),
918 1340 2 uplit byte (%ascic 'SRKW'),
919 1341 2 uplit byte (%ascic 'SR'),
920 1342 2 uplit byte (%ascic 'URSW'),
921 1343 2 uplit byte (%ascic 'UREW'),
922 1344 2 uplit byte (%ascic 'URKW'),
923 1345 2 uplit byte (%ascic 'UR'));
924 1346 2
925 1347 2
926 1348 2 ! Simply print a line with the protection code.
927 1349 2
928 1350 2 anl$format_line(0,..indent_level,anlobj$protection,..prot_code_table[.prot_code]);
929 1351 2
930 1352 2 return;
931 1353 2
932 1354 1 end;

```

.PSECT \$SPLITS,NOWR',NOEXE,2

```

      41 4E 02 000A2 P.ABQ: .ASCII <2>\NA\
3F 3F 3F 03 000A5 P.ABR: .ASCII <3>\???\
      57 4B 02 000A9 P.ABS: .ASCII <2>\KW\
      52 4B 02 000AC P.ABT: .ASCII <2>\KR\
      57 55 02 000AF P.ABU: .ASCII <2>\UW\
      57 45 02 000B2 P.ABV: .ASCII <2>\EW\
57 4B 52 45 04 000B5 P.ABW: .ASCII <4>\ERKW\
      52 45 02 000BA P.ABX: .ASCII <2>\ER\
      57 53 02 000BD P.ABY: .ASCII <2>\SW\
57 45 52 53 04 000C0 P.ABZ: .ASCII <4>\SREW\
57 4B 52 53 04 000C5 P.ACA: .ASCII <4>\SRKW\
      52 53 02 000CA P.ACB: .ASCII <2>\SR\
57 53 52 55 04 000CD P.ACC: .ASCII <4>\URSW\
57 45 52 55 04 000D2 P.ACD: .ASCII <4>\UREW\
57 4B 52 55 04 000D7 P.ACE: .ASCII <4>\URKW\
      52 55 02 000DC P.ACF: .ASCII <2>\UR\

```

.PSECT \$OWNS,NOEXE,2

```

00000000* 00000000* 00000000* 00000000* 00000000* 00000000* 0037C PROT_CODE TABLE:
00000000* 00000000* 000000C0* 00000000* 00000000* 00000000* 00394 .ADDRESS P.ABQ, P.ABR, P.ABS, P.ABT, P.ABU, -
00000000* 00000000* 00000000* 00000000* 00000000* 00000000* 003AC P.ABV, P.ABW, P.ABX, P.ABY, P.ABZ, P.ACA, -
P.ACB, P.ACC, P.ACD, P.ACE, P.ACF

```

.PSECT \$CODE\$,NOWRT,2

```

      0000 00000
50      08 AC D0 00002 .ENTRY ANLS$FORMAT_PROTECTION, Save nothing : 1326
      0000*CF40 DD 00006 MOVL PROT_CODE, R0 : 1350
      00000000G 8F DD 0000B PUSHL PROT_CODE_TABLE[R0]
      04 AC DD 00011 PUSHL #ANL$OBJ$ PROTECTION
      7E D4 00014 PUSHL INDENT_LEVEL
FCD4 CF 04 FB 00016 CLRL -(SP)
      04 0001B CALLS #4, ANLS$FORMAT_LINE
RET : 1354

```

; Routine Size: 28 bytes, Routine Base: \$CODE\$ + 046F


```

: 934 1355 1 %sbttl 'ANLS$FORMAT_SEVERITY - Format Error Severity Code'
: 935 1356 1 **
: 936 1357 1 Functional Description:
: 937 1358 1 This routine is called to format a standard VMS error severity
: 938 1359 1 code. It also checks to make sure the code is valid.
: 939 1360 1
: 940 1361 1 Formal Parameters:
: 941 1362 1 indent_level Level of indentation for report.
: 942 1363 1 severity The severity code.
: 943 1364 1
: 944 1365 1 Implicit Inputs:
: 945 1366 1 global data
: 946 1367 1
: 947 1368 1 Implicit Outputs:
: 948 1369 1 global data
: 949 1370 1
: 950 1371 1 Returned Value:
: 951 1372 1 none
: 952 1373 1
: 953 1374 1 Side Effects:
: 954 1375 1
: 955 1376 1 --
: 956 1377 1
: 957 1378 1
: 958 1379 2 global routine anl$format_severity(indent_level,severity): novalue = begin
: 959 1380 2
: 960 1381 2 own
: 961 1382 2 severity_code_table: vector[8,long] initial(
: 962 1383 2 uplit byte(%ascic 'WARNING'),
: 963 1384 2 uplit byte(%ascic 'SUCCESS'),
: 964 1385 2 uplit byte(%ascic 'ERROR'),
: 965 1386 2 uplit byte(%ascic 'INFO'),
: 966 1387 2 uplit byte(%ascic 'SEVERE'),
: 967 1388 2 uplit byte(%ascic 'reserved'),
: 968 1389 2 uplit byte(%ascic 'reserved'),
: 969 1390 2 uplit byte(%ascic 'reserved'));
: 970 1391 2
: 971 1392 2
: 972 1393 2 ! Format a line with the severity code on it.
: 973 1394 2
: 974 1395 2 anl$format_line(0,.indent_level,anlobj$severity,.severity_code_table[.severity]);
: 975 1396 2
: 976 1397 2 ! Check for a reserved severity.
: 977 1398 2
: 978 1399 2 if .severity gequ 5 then
: 979 1400 2 anl$format_error(anlobj$badseverity,.severity);
: 980 1401 2
: 981 1402 2 return;
: 982 1403 2
: 983 1404 1 end;

```

.PSECT \$PLITS,NOWRT,NOEXE,2

```

47 4E 49 4E 52 41 57 07 000DF P.ACG: .ASCII <7>\WARNING\
53 53 45 43 43 55 53 07 000E7 P.ACH: .ASCII <7>\SUCCESS\

```

:

OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANLSFORMAT_SEVERITY - Format Error Severity

H 11
15-Sep-1984 23:36:57
Cod 14-Sep-1984 11:52:52

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]OBJEXEOUT.B32:1

			52	4F	52	52	45	05	000EF	P.ACI:	.ASCII	<5>\ERROR\	:
				4F	46	4E	49	04	000F5	P.ACJ:	.ASCII	<4>\INFO\	:
		45	52	45	56	45	53	06	000FA	P.ACK:	.ASCII	<6>\SEVERE\	:
64	65	76	72	65	73	65	72	08	00101	P.ACL:	.ASCII	<8>\reserved\	:
64	65	76	72	65	73	65	72	08	0010A	P.ACM:	.ASCII	<8>\reserved\	:
64	65	76	72	65	73	65	72	08	00113	P.ACN:	.ASCII	<8>\reserved\	:

.PSECT \$OWNS,NOEXE,2

00000000*	00000000*	00000000*	00000000*	00000000*	00000000*	003BC	SEVERITY_CODE_TABLE:	:
							ADDRESS P.ACG, P.ACH, P.ACI, P.ACJ, P.ACK, -	:
						00000000*	P.ACL, P.ACM, P.ACN	:

.PSECT \$CODE\$,NOWRT,2

						0004	00000	.ENTRY	ANLSFORMAT SEVERITY, Save R2	:	1379
	52		08	AC	D0	00002		MOVL	SEVERITY, R2	:	1395
			0000*	CF	42	DD	00006	PUSHL	SEVERITY_CODE_TABLE[R2]	:	
			00000000G		8F	DD	0000B	PUSHL	#ANLOBS\$ SEVERITY	:	
				04	AC	DD	00011	PUSHL	INDENT_LEVEL	:	
					7E	D4	00014	CLRL	-(SP)	:	
	FCBB	CF			04	FB	00016	CALLS	#4, ANLSFORMAT_LINE	:	
		05			52	D1	0001B	CMPL	R2, #5	:	1399
					0D	1F	0001E	BLSSU	1\$:	
					52	DD	00020	PUSHL	R2	:	1400
					8F	DD	00022	PUSHL	#ANLOBS\$ BADSEVERITY	:	
	FD2B	CF			02	FB	00028	CALLS	#2, ANLSFORMAT_ERROR	:	
					04	0002D	1\$:	RET		:	1404

; Routine Size: 46 bytes, Routine Base: \$CODE\$ + 048B


```

00000000G 00      5E DD 0001C      PUSHL SP
0001827A 8F      01 FB 0001E      CALLS #1, LIB$GET_INPUT
FE00 CF      50 D1 00025      CMLP STATUS, #98938
                05 12 0002C      BNEQ 1$
                FE00 CF      00 FB 0002E      CALLS #0, ANL$EXIT_WITH_STATUS
                2E      04 50 D4 00033 1$: CLRL R0
                BE 91 00035      CMPB @ANSWER_BUF+4, #46
                02 13 00039      BEQL 2$
                50 D6 0003B      INCL R0
                04 0003D 2$: RET
    
```

; Routine Size: 62 bytes, Routine Base: \$CODE\$ + 04B9

```

: 1029      1449 1
: 1030      1450 0 end eludom
    
```

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	988	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	284	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1271	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_S255\$DUA28:[SYSLIB]STARLET.L32;1	9776	50 0	581	00:01.0

COMMAND QUALIFIERS

; BLISS/(CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:OBJEXEOUT/OBJ=OBJ\$:OBJEXEOUT MSRC\$:OBJEXEOUT/UPDATE=(ENH\$:OBJEXEOUT)

```

: Size:      1271 code + 1276 data bytes
: Run Time:  00:28.7
: Elapsed Time: 00:54.1
: Lines/CPU Min: 3028
: Lexemes/CPU-Min: 20190
: Memory Used: 187 pages
    
```

0
1
2
3
4
5
6
7
8
9
OBJEXEOUT
V04-000

OBJEXEOUT - Handle Report Output
ANLSINTERACT - See if User wants to Continue

K 11
15-Sep-1984 23:36:57

VAX-11 Bliss-32 V4.0-742

Page 43

; Compilation Complete

0006 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY