

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```
0001 0 MODULE
0002 0 UTILITY (IDENT = 'V04-000') =
0003 1 BEGIN
0004 1
0005 1
0006 1 *****
0007 1 *
0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0010 1 * ALL RIGHTS RESERVED.
0011 1 *
0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0017 1 * TRANSFERRED.
0018 1 *
0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 1 * CORPORATION.
0022 1 *
0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0025 1 *
0026 1 *
0027 1 *****
0028 1
0029 1 ++
0030 1 FACILITY: ACC, Account file dumper
0031 1
0032 1 ABSTRACT:
0033 1
0034 1 This module contains utility routines used by the ACC facility.
0035 1
0036 1 ENVIRONMENT:
0037 1
0038 1 VAX/VMS operating system. unprivileged user mode.
0039 1
0040 1 AUTHOR: Greg Robert and Steve Forgey, January 1982
0041 1
0042 1 Modified by:
0043 1
0044 1 V03-003 MHB0159 Mark Bramhall 11-May-1984
0045 1 Remove special -2 and -4 status code checks.
0046 1
0047 1 V03-002 DAS0001 David Solomon 12-Jan-1984
0048 1 Get ACCDEF.REQ from SRC$, not MSRC$.
0049 1
0050 1 V03-001 SPF0079 Steve Forgey 05-Feb-1982
0051 1 Don't signal "end of file" errors.
0052 1
0053 1 --
0054 1
0055 1 -----
0056 1
0057 1 INCLUDE FILES
```

```

: 58      0058 1  !
: 59      0059 1  !-----!
: 60      0060 1
: 61      0061 1 SWITCHES LIST (REQUIRE);
: 62      0062 1
: 63      0063 1 REQUIRE 'SRCS:ACCDEF';           ! Command ACC definitions
```

R0064 1
R0065 1
R0066 1
R0067 1
R0068 1
R0069 1
R0070 1
R0071 1
R0072 1
R0073 1
R0074 1
R0075 1
R0076 1
R0077 1
R0078 1
R0079 1
R0080 1
R0081 1
R0082 1
R0083 1
R0084 1
R0085 1
R0086 1
R0087 1
R0088 1
R0089 1
R0090 1
R0091 1
R0092 1
R0093 1
R0094 1
R0095 1
R0096 1
R0097 1
R0098 1
R0099 1
R0100 1
R0101 1
R0102 1
R0103 1
R0104 1
R0105 1
R0106 1
R0107 1
R0108 1
R0109 1
R0110 1
R0111 1
R0112 1
R0113 1
R0114 1
R0115 1
R0116 1
R0117 1
R0118 1
R0119 1
R0120 1

!MODULE ACCDEF(IDENT = 'V04-000') =

```
*****  
*  
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
* ALL RIGHTS RESERVED.  
*  
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
* TRANSFERRED.  
*  
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
* CORPORATION.  
*  
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
*  
*****
```

++

FACILITY: ACC, Account file dumper

ABSTRACT:

This file contains definitions of general applicability to
the accounting facility

ENVIRONMENT:

VAX/VMS operating system. unprivileged user mode,

AUTHOR: Greg Robert and Steve Forgey, January 1982

Modified by:

V03-002 TMH0002 Tim Halvorsen 14-Apr-1984
Remove MOVE_QUAD macro, which now exists in UTILDEF.

V03-001 DAS0001 David Solomon 03-Jan-1984
Remove two unused shared message definitions (CREATED and
EXISTS). Change declaration of message codes from ACC_ to ACCS_.
Add literal for summation table index. Add ACCS_INVACCREC.
Allow error code LIBS_INPSTRTRU on call to SCR\$GET_SCREEN in
GET_REPLY macro.

--

PROGRAM CONTROL

: R0121 1
: R0122 1
: R0123 1
: R0124 1
: R0125 1
: R0126 1
: R0127 1
: R0128 1
: R0129 1
: R0130 1
: R0131 1
: R0132 1
: R0133 1
: R0134 1
: R0135 1
: R0136 1
: R0137 1
: R0138 1
: R0139 1
: R0140 1
: R0141 1
: R0142 1
: R0143 1
: R0144 1

!----->

SWITCHES
ADDRESSING_MODE(
EXTERNAL=GENERAL,
NONEXTERNAL=WORD_RELATIVE);

PSECT
CODE= CODE,
PLIT= CODE,
OWN= DATA(ADDRESSING_MODE(LONG_RELATIVE)),
GLOBAL= DATA;

----->
INCLUDE FILES
----->

LIBRARY 'SYSS\$LIBRARY:STARLET'; ! VAX/VMS common definitions
REQUIRE 'SHRLIBS:UTILDEF'; ! Common VMS/DEVELOPMENT macros

R0145 1
R0146 1
R0147 1
R0148 1
R0149 1
R0150 1
R0151 1
R0152 1
R0153 1
R0154 1
R0155 1
R0156 1
R0157 1
R0158 1
R0159 1
R0160 1
R0161 1
R0162 1
R0163 1
R0164 1
R0165 1
R0166 1
R0167 1
R0168 1
R0169 1
R0170 1
R0171 1
R0172 1
R0173 1
R0174 1
R0175 1
R0176 1
R0177 1
R0178 1
R0179 1
R0180 1
R0181 1
R0182 1
R0183 1
R0184 1
R0185 1
R0186 1
R0187 1
R0188 1
R0189 1
R0190 1

Commonly used definitions for VMS modules written in BLISS
Version: 'V04-000'

*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
* ALL RIGHTS RESERVED. *
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
* TRANSFERRED. *
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
* CORPORATION. *
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
*

++
ABSTRACT:
This is the common require file for any module written
in BLISS
ENVIRONMENT:
VAX/VMS operating system.
AUTHOR: Tim Halvorsen, Feb 1980
MODIFIED BY:
V03-001 MHB0127 Mark Bramhall 5-Apr-1984
Added the MOVE_QUAD macro.

```

: R0191 1 |
: R0192 1 |
: R0193 1 |
: R0194 1 |
: R0195 1 |
: R0196 1 |
: R0197 1 |
: R0198 1 |
: R0199 1 |
: R0200 1 |
: R0201 1 |
: R0202 1 |
: R0203 1 |
: R0204 1 |
: R0205 1 |
: R0206 1 |
: R0207 1 |
: R0208 1 |
: R0209 1 |
: R0210 1 |
: R0211 1 |
: R0212 1 |
: R0213 1 |
: R0214 1 |
: R0215 1 |
: R0216 1 |
: R0217 1 |
: R0218 1 |
: R0219 1 |
: R0220 1 |
: R0221 1 |
: R0222 1 |
: R0223 1 |
: R0224 1 |
: R0225 1 |
: R0226 1 |
: R0227 1 |
: R0228 1 |
: R0229 1 |
: R0230 1 |
: R0231 1 |
: R0232 1 |
: R0233 1 |
: R0234 1 |
: R0235 1 |
: R0236 1 |
: R0237 1 |
: R0238 1 |
: R0239 1 |
: R0240 1 |
: R0241 1 |
: R0242 1 |
: R0243 1 |
: R0244 1 |
: R0245 1 |
: R0246 1 |
: R0247 1 |

|
| Equated symbols
|
| LITERAL
| true = 1, | boolean true
| false = 0, | boolean false
| ok = 1, | success return code
| error = 2, | error return code
| quad = 8; | quadword allocation definition
|
| Define structure type for VMS structures
|
| STRUCTURE
| bblock [o, p, s, e; n] =
| [n]
| (bblock+o)<p,s,e>;
|
| MACRO
| move_quad (src, dst) = ! Move a quadword
| BEGIN
| (dst)+0 = .(src)<0, 32>;
| (dst)+4 = .(src)<32, 32>;
| ENDX;
|
| MACRO
| descriptor [] = ! Generate a static string descriptor
| UPLIT (%CHARCOUNT (%STRING (%REMAINING))),
| UPLIT BYTE (%STRING (%REMAINING))) %;
|
| MACRO
| own_descriptor [] = ! Generate the actual static string descriptor
| %BBLOCK [8] INITIAL(%CHARCOUNT(%STRING(%REMAINING))),
| UPLIT BYTE (%STRING(%REMAINING))) %;
|
| MACRO
| return_if_error(command) =
| BEGIN
| LOCAL
| status;
|
| status = command;
| IF NOT .status
| THEN
| RETURN .status;
| ENDX;
|
| MACRO
| signal_if_error(command) =
| BEGIN
| LOCAL
| status;
|
| status = command;
| IF NOT .status
```

```
MR0248 1      THEN
MR0249 1          BEGIN
MR0250 1          SIGNAL(.status);
MR0251 1          RETURN .status;
MR0252 1          END;
R0253 1      ENDX;
R0254 1
R0255 1
R0256 1      !
R0257 1      ! Macro to implement a function (f) of the message severity level that
R0258 1      ! maps the various severity levels such that arithmetic comparisons of the
R0259 1      ! mapped values ( f(severity) ) yield a order of precedence that is
R0260 1      ! intuitivtely acceptable:
R0261 1
R0262 1          ERROR NAME      OLDVAL      NEWVAL
R0263 1
R0264 1          F(SUCCESS)        1      --> 0
R0265 1          F(INFORMATIONAL)    3      --> 0
R0266 1          F(WARNING)          0      --> 0
R0267 1          F(ERROR)            2      --> 5
R0268 1          F(SEVERE_ERROR)     4      --> 7
R0269 1
R0270 1
R0271 1      MACRO
MR0272 1          severity_level (status) =
MR0273 1          BEGIN
MR0274 1          LOCAL code: BBLOCK [LONG];
MR0275 1          code = status;
MR0276 1          .code [sts$severity] - (4 * .code [sts$success]) + 3
R0277 1          ENDX;
R0278 1
R0279 1      MACRO
MR0280 1          cli$external(prefix) =
MR0281 1          %IF %DECLARED(%QUOTE %QUOTE cli$prefix)
MR0282 1          %THEN UNDECLARE %QUOTE %QUOTE cli$prefix; %FI
MR0283 1          MACRO cli$prefix = prefix %QUOTE %;
MR0284 1          EXTERNAL LITERAL
R0285 1          cli$external_loop(%REMAINING)%,
R0286 1
MR0287 1          cli$external_loop[name] =
R0288 1          %NAME(cli$prefix,name): UNSIGNED(8)%;
R0289 1
R0290 1      MACRO
MR0291 1          Sexternal_literal(symbol) =
MR0292 1          BEGIN
MR0293 1          %IF NOT %DECLARED(symbol) %THEN EXTERNAL LITERAL symbol
MR0294 1          %IF %LENGTH GTR 1 %THEN : %REMAINING %FI; %FI
MR0295 1          symbol
R0296 1          ENDX;
R0297 1
R0298 1      MACRO
MR0299 1          $fab_dev(dev_bit) =          ! Access FAB$L_DEV bits of FAB block
MR0300 1          %BYTEOFFSET(fab$L dev),
R0301 1          %BITPOSITION(%NAME('dev$dev_',dev_bit)),1,0%;
R0302 1
R0303 1      !
R0304 1      ! SSHR_MESSAGES - a macro which defines facility-specific message codes
```

```
: R0305 1
: R0306 1
: R0307 1
: R0308 1
: R0309 1
: R0310 1
: R0311 1
: R0312 1
: R0313 1
: R0314 1
: R0315 1
: R0316 1
: MR0317 1
: MR0318 1
: R0319 1
: R0320 1
: MR0321 1
: R0322 1
: R0323 1
: MR0324 1
: MR0325 1
: MR0326 1
: MR0327 1
: R0328 1
```

```
which are based on the system-wide shared message codes.
```

```
SSHR_MESSAGES( name, code, (msg,severity), ... )
```

```
where:
```

```
  "name" is the name of the facility (e.g., COPY)
  "code" is the corresponding facility code (e.g., 103)
  "msg" is the name of the shared message (e.g., BEGIN)
  "severity" is the desired message severity (e.g., 1, 0, 2)
```

```
MACRO
```

```
SSHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
```

```
  [ LITERAL
```

```
    SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,
```

```
  SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE ) [ VALUE ] =
```

```
    SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,
```

```
  SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG ID, SEVERITY ) =
```

```
    %NAME(FACILITY_NAME, 'S', MSG ID) = %NAME('SHR$', MSG_ID) + FACILITY_CODE*65536 +
```

```
    %IF %DECLARED(%NAME('STSSK', SEVERITY))
```

```
      %THEN %NAME('STSSK_', SEVERITY)
```

```
    %ELSE SEVERITY %FI %;
```

UTILITY
V04-000

9
15-Sep-1984 23:51:05
15-Sep-1984 22:40:34

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[ACC.SRC]ACCDEF.REQ;1

Page 9
(1)

: R0329 1
: R0330 1

LIBRARY 'SYSSLIBRARY:TPAMAC'; ! TPARSE macros

DEFINE EXTERNAL REFERENCES

EXTERNAL LITERAL

RO339	ACCS_INVACCREC,	Invalid accounting record format
RO340	ACCS_TOTAL,	Totals selected/rejected message
RO341	ACCS_MERGE,	'Merge phase beginning' message
RO342	ACCS_INPUT,	File name/number message
RO343	ACCS_TITLETRUNC;	Title truncation warning

EXTERNAL ROUTINE

RO348	ADD_SYMBOL,	Add a symbol to a symbol table
RO349	ALLOCATE,	Gets virtual memory
RO350	FIND_WATERMARK,	Determine high watermarks
RO351	HANDLER,	Local signal processor
RO352	LIBSADDX,	Extended binary addition
RO353	LIBSCVT_DTB,	Converts decimal to binary
RO354	LIBSCVT-HTB,	Converts hex to binary
RO355	LIBSCVT-TIME,	Converts ascii to binary time
RO356	LIBSDAY,	Gets days since epoch
RO357	LIB\$FILE_SCAN,	Does wildcarding and stickiness
RO358	LIB\$CHAR,	Converts character to integer
RO359	LIB\$LOOKUP_KEY,	Searches keyword lists
RO360	LIB\$SUBX,	Extended precision subtract
RO361	LIB\$SYS_ASCTIM,	Converts binary time to ascii
RO362	LIB\$SYS-FAO,	Formatted ascii output
RO363	LIB\$SYS-FAOL,	Library FAO routine
RO364	LIB\$PARSE,	Library parse routine
RO365	LOG_FILENAME,	Signals filenames and error messages
RO366	LOOKUP_SYMBOL,	Lookup a symbol in a symbol table
RO367	MAP_QUALIFIERS,	Establish qualifer maps
RO368	PARSE_OUTPUT_FILES,	Sets up output fabs, rabs
RO369	RELEASE_TO_SORT,	Build keys and release to sort
RO370	SCAN_SYMBOLS,	Call action routine for every symbol
RO371	SHOW-RECORD,	Dispatch to output routines
RO372	SORSEND SORT,	Clean up files etc.
RO373	SORSINIT SORT,	Initialize sort routines
RO374	SOR\$RELEASE REC,	SORT32 record interface routine
RO375	SOR\$RETURN REC,	Fetch a sorted record
RO376	SOR\$SORT MERGE,	Initiate merge processing
RO377	STR\$APPEND,	Appends strings
RO378	STR\$COMPARE,	Compares two strings
RO379	STR\$DUPL_CHAR,	Generates a string
RO380	STR\$LEFT,	Extract left justified substring
RO381	STR\$PREFIX,	Prefix strings
RO382	STR\$REPLACE,	Replaces substrings
RO383	STR\$RIGHT,	Strips leading strings
RO384	STRIP-NEGATOR,	Check/strip leading negator
RO385	STRIP-TRAIL,	Strip trailing garbage
RO386	SUMMARIZE,	Summation main control loop
RO387	SYSSNUMTIM,	Converts times

UTILITY
V04-000

0 9
15-Sep-1984 23:51:05
15-Sep-1984 22:40:34

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[ACC.SRC]ACCDEF.REQ;1

Page 11
(2)

: R0388 1
: R0389 1
: R0390 1
: R0391 1
: R0392 1

TRANSLATE_STATUS,
WRITE_BAR_GRAPH,
WRITE_BINARY,
WRITE_SUMMARY,
WRITE_TOTALS;

! Looks up status messages
! Output bar graph
! Output a binary record
! Output summary at end of file
! Output totals at end of program

DEFINE INTERNAL MACROS

MACRO

COMPARE QUADWORD VALUES--

Macro to compare two quadword values using the user supplied operator. This macro is somewhat inefficient if the supplied operator is EQL or NEQL since the expansion becomes (in part):

IF Q1 EQL Q2 THEN IF Q1 EQL Q2 THEN TRUE ELSE FALSE

However the compiler may remove this inefficiency through optimization. This macro works better for values that are close to one another as it does the equality check of the high order words first. This macro could be improved by inspecting the supplied operator and generating a tailored macro.

COMPARE_QUAD (Q1, OPER, Q2) =

BEGIN

BIND A = Q1: VECTOR [2, LONG];

BIND B = Q2: VECTOR [2, LONG];

IF .A[1] EQL .B[1] THEN .A[0] OPER .B[0] ELSE .A[1] OPER .B[1]
ENDX,

- A) Macro to describe a string
- B) Macro to generate a quadword string descriptor
- C) Macro to generate the address of a string descriptor
- D) Macro to abbreviate last macro

PRIMDESC [] = %CHARCOUNT (%STRING (%REMAINING)),
 UPLIT (%STRING (%REMAINING))%,
INITDESC [] = %BLOCK [DSC% S %BLN]
 INITIAL (PRIMDESC (%REMAINING))%,
ADDRDESC [] = UPLIT (PRIMDESC (%REMAINING))%,
AD [] = ADDRDESC (%REMAINING)%,

FIELD REFERENCE MACROS --

Define macros to reference fields.

KEY_W_TYPE	= 0,0,16,0%,	! Sort key type
KEY_W_ORDER	= 2,0,16,0%,	! Sort order
KEY_W_POS	= 4,0,16,0%,	! Item position
KEY_W_LENGTH	= 6,0,16,0%,	! Key length
SORT_TYPE	= 0,0,16,0%,	! Sort key type (binary/char etc)

```

: R0450 1 SORT_DESC = 1,0,32,0%, ! Address of item descriptor
R0451 1 SORT_LENGTH = 2,0,16,0%, ! Max. length of item in bytes
R0452 1
R0453 1
R0454 1
R0455 1
R0456 1 ! SIGNAL_RETURN --
R0457 1 ! Signal the given arguments and then return the first parameter.
R0458 1
R0459 1
MRO460 1 Signal_return (status) [] =
MRO461 1 BEGIN
MRO462 1 signal (status, %remaining);
MRO463 1 return (status);
R0464 1 END%,
R0465 1
R0466 1
R0467 1
R0468 1
R0469 1
R0470 1 ! PERFORM MACRO --
R0471 1 ! This renames the RETURN_IF_ERROR macro to be the PERFORM macro.
R0472 1
R0473 1
R0474 1 PERFORM (COMMAND) = RETURN_IF_ERROR (COMMAND)%, ! *** HACK
R0475 1
R0476 1
R0477 1
R0478 1
R0479 1
R0480 1
R0481 1 ! EXPAND FAO STRING
R0482 1 ! Expand an FAO directive and yeild the address of a descriptor
R0483 1 ! of the resultant buffer.
R0484 1
R0485 1
MRO486 1 XFAO (DESC) =
MRO487 1 BEGIN
MRO488 1 EXTERNAL ROUTINE LIB$SYS_FAO: ADDRESSING_MODE (GENERAL);
MRO489 1 LOCAL $$buffer: vector [512, byte];
MRO490 1 LOCAL $$buffdesc: vector [2, long];
MRO491 1
MRO492 1
MRO493 1 $$buffdesc [0] = 512; ! Initialize descriptor length
MRO494 1 $$buffdesc [1] = $$buffer; ! Initialize descriptor address
MRO495 1
MRO496 1 signal_if_error (lib$sys_fao (
MRO497 1 desc, ! Control string address
MRO498 1 $$buffdesc [0], ! Resultant length
MRO499 1 $$buffdesc ! Buffer descriptor
MRO500 1 %IF %LENGTH GTR 1 %THEN , %REMAINING %FI
MRO501 1 ));
MRO502 1 $$buffdesc
R0503 1 END%,
R0504 1
R0505 1
R0506 1

```

```

: R0507 1
: MR0508 1      XFAOL (DESC, LIST) =
: MR0509 1
: MR0510 1      BEGIN
: MR0511 1      EXTERNAL ROUTINE LIB$SYS_FAOL: ADDRESSING_MODE (GENERAL);
: MR0512 1      LOCAL $$buffer:      vector [512, byte];
: MR0513 1      LOCAL $$buffdesc:    vector [2, long];
: MR0514 1
: MR0515 1      $$buffdesc [0] = 512;      ! Initialize descriptor length
: MR0516 1      $$buffdesc [1] = $$buffer;  ! Initialize descriptor address
: MR0517 1
: MR0518 1      signal_if_error (lib$sys_faol (
: MR0519 1          desc,                  ! Control string address
: MR0520 1          $$buffdesc [0],      ! Resultant length
: MR0521 1          $$buffdesc,        ! Buffer descriptor
: MR0522 1          list                ! Arguement list
: MR0523 1          ));
: MR0524 1      $$buffdesc
: R0525 1      ENDX,
: R0526 1
: R0527 1
: R0528 1
: R0529 1
: R0530 1      ! CLI PARSING MACROS --
: R0531 1      ! Determine if a command line entity is present or get its value
: R0532 1      !
: R0533 1
: MR0534 1      GET_PRESENT (DESC) =
: MR0535 1      BEGIN
: MR0536 1      EXTERNAL ROUTINE CLISP$PRESENT: ADDRESSING_MODE (GENERAL);
: MR0537 1      CLISP$PRESENT (DESC)
: R0538 1      ENDX,
: R0539 1
: MR0540 1      PRESENT (QUAL_NUMB) =
: MR0541 1      BEGIN
: MR0542 1      EXTERNAL QUALIFIERS: BITVECTOR [64];
: MR0543 1      .QUALIFIERS [QUAL_NUMB]
: R0544 1      ENDX,
: R0545 1
: MR0546 1      GET_VALUE (STRING, DESC) =
: MR0547 1      BEGIN
: MR0548 1      EXTERNAL ROUTINE CLIS$GET_VALUE: ADDRESSING_MODE (GENERAL);
: MR0549 1      CLIS$GET_VALUE (AD (STRING), DESC)
: R0550 1      ENDX,
: R0551 1
: R0552 1      !
: R0553 1      ! ATTRIBUTE DEFINITIONS
: R0554 1      !
: R0555 1
: R0556 1      BOLD                = SCRSM_BOLDX,
: R0557 1      REVERSE            = SCRSM_REVERSEX,
: R0558 1      BLINK              = SCRSM_BLINKX,
: R0559 1      UNDERLINE         = SCRSM_UNDERLINEX,
: R0560 1
: R0561 1
: R0562 1      !
: R0563 1      ! SET OUTPUT MACRO --

```

```

: R0564 1  | This macro establishes an output stream through the screen package.
: R0565 1  |
: R0566 1  |
: MR0567 1  SET_OUTPUT (STREAM, FILENAME, USERSUB, ARGUMENT, PREVIOUS) =
: MR0568 1  BEGIN
: MR0569 1  EXTERNAL ROUTINE SCR$SET_OUTPUT: ADDRESSING_MODE (GENERAL);
: MR0570 1  SIGNAL_IF_ERROR (SCR$SET_OUTPUT (
: MR0571 1  %IF %NULL (STREAM) %THEN 1 %ELSE STREAM %FI,
: MR0572 1  %IF %NULL (FILENAME) %THEN 0 %ELSE FILENAME %FI,
: MR0573 1  %IF %NULL (USERSUB) %THEN 0 %ELSE USERSUB %FI,
: MR0574 1  %IF %NULL (ARGUMENT) %THEN 0 %ELSE ARGUMENT %FI,
: MR0575 1  %IF %NULL (PREVIOUS) %THEN 0 %ELSE PREVIOUS %FI
: MR0576 1  ));
: R0577 1  END%,
: R0578 1  |
: R0579 1  |
: R0580 1  |
: R0581 1  | SCREEN MACRO --
: R0582 1  | This macro invokes the screen package to determine output
: R0583 1  | characteristics.
: R0584 1  |
: R0585 1  |
: MR0586 1  SCREEN_INFO (BUFFER) =
: MR0587 1  BEGIN
: MR0588 1  EXTERNAL ROUTINE SCR$SCREEN_INFO: ADDRESSING_MODE (GENERAL);
: MR0589 1  SIGNAL_IF_ERROR (SCR$SCREEN_INFO (BUFFER));
: R0590 1  END%,
: R0591 1  |
: R0592 1  |
: MR0593 1  SCREEN (ARG) =
: MR0594 1  BEGIN
: MR0595 1  EXTERNAL SCREEN_CHAR: BBLOCK [SCR$K_LENGTH];
: MR0596 1  .SCREEN_CHAR [
: MR0597 1  %IF %IDENTICAL (ARG, FLAGS) %THEN SCR$L_FLAGS %FI
: MR0598 1  %IF %IDENTICAL (ARG, WIDTH) %THEN SCR$W_WIDTH %FI
: MR0599 1  %IF %IDENTICAL (ARG, LENGTH) %THEN SCR$W_PAGESIZE %FI
: MR0600 1  %IF %IDENTICAL (ARG, TYPE) %THEN SCR$B_DEVTYPE %FI
: R0601 1  ] END%,
: R0602 1  |
: R0603 1  |
: R0604 1  |
: R0605 1  | SET CURSOR --
: R0606 1  |
: R0607 1  |
: MR0608 1  SET_CURSOR (LINE, COLUMN) =
: MR0609 1  BEGIN
: MR0610 1  EXTERNAL ROUTINE SCR$SET_CURSOR: ADDRESSING_MODE (GENERAL);
: MR0611 1  SIGNAL_IF_ERROR (SCR$SET_CURSOR (LINE, COLUMN))
: R0612 1  END%,
: R0613 1  |
: R0614 1  |
: R0615 1  |
: R0616 1  | SET SCROLL --
: R0617 1  | Establish a scrolling region
: R0618 1  |
: R0619 1  |
: MR0620 1  SET_SCROLL (TOP, BOTTOM) =

```

```
MR0621 1 BEGIN
MR0622 1 EXTERNAL ROUTINE SCR$SET_SCROLL: ADDRESSING_MODE (GENERAL);
MR0623 1 SIGNAL_IF_ERROR (SCR$SET_SCROLL (
MR0624 1 TOP
MR0625 1 %IF %LENGTH GTR 1 %THEN , %FI
MR0626 1 BOTTOM))
R0627 1 END%,
R0628 1
R0629 1
R0630 1
R0631 1
R0632 1 ! ERASE SCREEN --
R0633 1 !
R0634 1
MR0635 1 ERASE_PAGE (LINE, COLUMN) =
MR0636 1 BEGIN
MR0637 1 EXTERNAL ROUTINE SCR$ERASE_PAGE: ADDRESSING_MODE (GENERAL);
MR0638 1 SIGNAL_IF_ERROR (SCR$ERASE_PAGE (
MR0639 1 %IF %NULL (LINE) %THEN 1 %ELSE LINE %FI,
MR0640 1 %IF %NULL (COLUMN) %THEN 1 %ELSE COLUMN %FI
MR0641 1 ))
R0642 1 END%,
R0643 1
R0644 1
R0645 1
R0646 1
R0647 1 ! ERASE LINE--
R0648 1 !
R0649 1
MR0650 1 ERASE_LINE (LINE, COLUMN) =
MR0651 1 BEGIN
MR0652 1 EXTERNAL ROUTINE SCR$ERASE_LINE: ADDRESSING_MODE (GENERAL);
MR0653 1 SIGNAL_IF_ERROR (SCR$ERASE_LINE (LINE, COLUMN))
R0654 1 END%,
R0655 1
R0656 1
R0657 1
R0658 1
R0659 1 ! WRITE TO SCREEN --
R0660 1 ! Output a string to the screen with associated attributes at
R0661 1 ! indicated cursor position. Do not append carriage control.
R0662 1 !
R0663 1 !
MR0664 1 WRITE_AT (DESC, LINE, COLUMN, ATTR) =
MR0665 1 BEGIN
MR0666 1 EXTERNAL ROUTINE SCR$PUT_SCREEN: ADDRESSING_MODE (GENERAL);
MR0667 1 SIGNAL_IF_ERROR (SCR$PUT_SCREEN (
MR0668 1 DESC,
MR0669 1 LINE,
MR0670 1 COLUMN,
MR0671 1 %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI
MR0672 1 ))
R0673 1 END%,
R0674 1
R0675 1
R0676 1 ! WRITE LINE TO SCREEN --
R0677 1 ! Output a string to the screen with associated attributes,
```

```

: R0678 1  ! scroll the indicated number of LINES, and append a carriage return.
: R0679 1  !
: R0680 1  !
: MR0681 1  WRITE_LINE (DESC, LINES, ATTR) =
: MR0682 1  BEGIN
: MR0683 1  EXTERNAL ROUTINE SCR$PUT_LINE: ADDRESSING_MODE (GENERAL);
: MR0684 1  SIGNAL_IF_ERROR (SCR$PUT_LINE (
: MR0685 1  DESC
: MR0686 1  %IF %NULL (LINES) %THEN 1 %ELSE LINES %FI,
: MR0687 1  %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI
: MR0688 1  ))
: R0689 1  END%,
: R0690 1
: R0691 1
: R0692 1
: R0693 1
: R0694 1
: R0695 1  ! WRITE FAO STRING TO SCREEN --
: R0696 1  ! Write a STRING to the screen at the given line and
: R0697 1  ! column with no attributers after first processing it
: R0698 1  ! through FAO.
: R0699 1  !
: R0700 1  !
: MR0701 1  WRITE_FAO_AT (DESC,LINE,COLUMN) =
: MR0702 1  WRITE_AT (
: MR0703 1  XFAO (DESC
: MR0704 1  %IF %LENGTH GTR 3 %THEN , %REMAINING %FI
: MR0705 1  ),
: MR0706 1  LINE,
: MR0707 1  COLUMN,
: MR0708 1  0)% ,
: R0709 1
: R0710 1
: R0711 1  ! GET INPUT
: R0712 1  !
: R0713 1  !
: R0714 1  !
: MR0715 1  GET_REPLY (REPLY, PROMPT, LENGTH) =
: MR0716 1  BEGIN
: MR0717 1  EXTERNAL ROUTINE SCR$GET_SCREEN: ADDRESSING_MODE (GENERAL);
: MR0718 1  EXTERNAL LITERAL
: MR0719 1  LIB$INPSTRTRU;
: MR0720 1
: MR0721 1  LOCAL
: MR0722 1  STATUS;
: MR0723 1
: MR0724 1  %IF %NULL (REPLY) %THEN
: MR0725 1  LOCAL $$TEMP,
: MR0726 1  $$TEMPDESC: VECTOR [2];
: MR0727 1  $$TEMPDESC [0] = 4;
: MR0728 1  $$TEMPDESC [1] = $$TEMP;
: MR0729 1  %FI
: MR0730 1  STATUS = SCR$GET_SCREEN (
: MR0731 1  %IF %NULL (REPLY) %THEN $$TEMPDESC %ELSE REPLY %FI,
: MR0732 1  PROMPT,
: MR0733 1  LENGTH);
: MR0734 1  IF NOT .STATUS AND ( .STATUS NEQU LIB$INPSTRTRU )
```

```

: MR0735 1      THEN
: MR0736 1      BEGIN
: MR0737 1      SIGNAL( .STATUS );
: MR0738 1      RETURN .STATUS;
: MR0739 1      END;
: R0740 1      ENDX,
: R0741 1
: R0742 1
: R0743 1
: R0744 1      ! SET AND FLUSH BUFFER
: R0745 1
: R0746 1
: MR0747 1      SET_BUFFER (BUFFER) =
: MR0748 1      BEGIN
: MR0749 1      EXTERNAL ROUTINE SCR$SET_BUFFER: ADDRESSING_MODE (GENERAL);
: MR0750 1      SIGNAL_IF_ERROR (SCR$SET_BUFFER (BUFFER))
: R0751 1      ENDX,
: R0752 1
: MR0753 1      PUT_BUFFER (BUFFER) =
: MR0754 1      BEGIN
: MR0755 1      EXTERNAL ROUTINE SCR$PUT_BUFFER: ADDRESSING_MODE (GENERAL);
: MR0756 1      SIGNAL_IF_ERROR (SCR$PUT_BUFFER (BUFFER))
: R0757 1      ENDX,
: R0758 1
: R0759 1
: R0760 1
: R0761 1
: R0762 1
: R0763 1      ! LOOKUP MACRO --
: R0764 1      ! This macro invokes lib$lookup_key and, if the lookup fails,
: R0765 1      ! signals the user with the status and the failed value and
: R0766 1      ! returns to the caller.
: R0767 1
: R0768 1
: MR0769 1      LOOKUP (KEY, TABLE, RESULT) =
: MR0770 1      BEGIN
: MR0771 1      EXTERNAL ROUTINE LIB$LOOKUP_KEY: ADDRESSING_MODE (GENERAL);
: MR0772 1      LOCAL STATUS;
: MR0773 1      IF NOT (STATUS = LIB$LOOKUP_KEY (KEY, TABLE, RESULT))
: MR0774 1      THEN SIGNAL_RETURN (MSG$SYNTAX, 1, KEY);
: R0775 1      ENDX,
: R0776 1
: R0777 1
: R0778 1      ! THIS RENAMES THE $BYTEOFFSET MACRO TO BE $BOFF FOR BREVITY
: R0779 1
: R0780 1
: R0781 1      $BOFF (arg) = $BYTEOFFSET (arg)X;      ! Make a shorter name
: R0782 1
: R0783 1
: R0784 1
: R0785 1
: R0786 1
: R0787 1
: R0788 1
: R0789 1

```

DEFINE INTERNAL STRUCTURES

DEFINE MESSAGE CODES

SSHR_MESSAGES (MSG,159,

! COMMON I/O AND MISC. MESSAGES

* NAME *	* SEVERITY *	* DESCRIPTION *
(SEARCHFAIL,	ERROR),	-Error while searching for file
(OPENIN,	ERROR),	-Unable to open or connect to input
(READERR,	ERROR),	-Error while reading input
(CLOSEIN,	ERROR),	-Unable to close output
(OPENOUT,	ERROR),	-Unable to create, open, or connect to output
(WRITEERR,	ERROR),	-Error while writing output
(CLOSEOUT,	ERROR),	-Unable to close output
(SYNTAX,	SEVERE),	-Parse failure

R0790 1
R0791 1
R0792 1
R0793 1
R0794 1
R0795 1
PR0796 1
PR0797 1
PR0798 1
PR0799 1
PR0800 1
PR0801 1
PR0802 1
PR0803 1
PR0804 1
PR0805 1
PR0806 1
PR0807 1
PR0808 1
PR0809 1
PR0810 1
R0811 1
R0812 1
R0813 1

EQUATED SYMBOLS

LITERAL

FLAGS AND MISCELLANEOUS VALUES --

COLUMNS_PER_GROUP = 15,	! Bar graph column grouping factor
REC_PREFIX = 8,	! Size of data prefixed to record
ACCSK_UNKNOWN = 0,	! Reserve the value 0 for unknown types
NEGATOR = '-',	! Define list negator character

QUALIFIER NUMBERS

Define local bitnumbers for qualifers (also used as symbol table index numbers for summation).

ACCOUNT	= 00.
BAR_GRAPH	= 01.
BEFORE	= 02.
BINARY	= 03.
USER	= 04.
ENTRY	= 05.
FULL	= 06.
IDENT	= 07.
IMAGE	= 08.
JOB	= 09.
LOG	= 10.
ADDRESS	= 11.
NODE	= 12.
OWNER	= 13.
OUTPUT	= 14.
PRIORITY	= 15.
PROCESS	= 16.
QUEUE	= 17.
REJECTED	= 18.
REMOTE_ID	= 19.
REPORT	= 20.
SINCE	= 21.
STATUS	= 22.
SORT	= 23.
SUMMARY	= 24.
TERMINAL	= 25.
TITLE	= 26.
TYPE	= 27.
UIC	= 28.
QUAL_COUNT	= 29.
EXPAND_DATE	= 31.
UIC_GROUP	= 32.
UIC_MEMBER	= 33.

! See /REPORT parsing

SUMMATION_TABLE = 63.

! Must be different than above numbers.

: R0871 1
: R0872 1
: R0873 1
: R0874 1
: R0875 1
: R0876 1
: R0877 1
: R0878 1
: R0879 1
: R0880 1
: R0881 1
: R0882 1
: R0883 1
: R0884 1
: R0885 1
: R0886 1
: R0887 1
: R0888 1
: R0889 1
: R0890 1
: R0891 1
: R0892 1
: R0893 1
: R0894 1
: R0895 1
: R0896 1
: R0897 1
: R0898 1
: R0899 1
: R0900 1
: R0901 1

! SUMMATION TYPES --

! These numbers describes the various summation rules for fields
! in accounting records when /SUMMARY is invoked

SUM_TYPE_ADD	= 0.	! Simple longword addition
SUM_TYPE_ADDX	= 1.	! Extended longword addition
SUM_TYPE_PEAK	= 2.	! Longword peak value recording
SUM_TYPE_INCR	= 3.	! Increment per occurrence
SUM_TYPE_TYPE	= 4.	! Increment if type matches
SUM_TYPE_ELAP	= 5.	! Summarize connect time
SUM_ENT_TYPE	= 0.	! Type of summation entry
SUM_ENT_ADR	= 1.	! Address or other value
SUM_ENT_BSIZE	= 2.	! Accumulation bucket size in longwords
SUM_ENT_FAO	= 3.	! Address of FAO directive
SUM_ENT_HDR1	= 4.	! Address of 1st column header
SUM_ENT_HDR2	= 5.	! Address of 2nd column header

! MAXIMUM TABLE SIZES --

! Specify the maximum number of entries the user can make in qualifier
! value lists for /SORT and /SUMMARIZE.

MAX_DEFAULT	= 30.	! Default maximum
MAX_SUM	= MAX_DEFAULT.	! Maximum number of /SUMMARY values
MAX_REPORT	= MAX_DEFAULT.	! Maximum number of /REPORT values
MAX_SORT	= 10;	! Sort keys -- sort package limit

: 64
: 65
: 66
: 67
: 68
: 69

0902 1
0903 1
0904 1
0905 1
0906 1
0907 1

TABLE OF CONTENTS	
-------------------	--

```

71 0908 1 UNDECLARE WRITE_BINARY;
72 0909 1
73 0910 1 GLOBAL ROUTINE WRITE_BINARY (BUFFER, RAB) =
74 0911 1
75 0912 1 |----
76 0913 1 |
77 0914 1 | Functional description
78 0915 1 |
79 0916 1 |     This routine accepts a pointer to a buffer and writes
80 0917 1 |     the buffer to an output stream in binary format.
81 0918 1 |
82 0919 1 | Input parameters
83 0920 1 |
84 0921 1 |     BUFFER = address of an input record buffer
85 0922 1 |             the size of the record is stored in the second
86 0923 1 |             word of the buffer
87 0924 1 |
88 0925 1 |     RAB =   address of output rab
89 0926 1 |
90 0927 1 |----
91 0928 1
92 0929 1 BEGIN
93 0930 1
94 0931 1 MAP
95 0932 1     rab:      ref bblock,      ! Pointer to rab
96 0933 1     buffer:  ref bblock;      ! Describe the input buffer
97 0934 1
98 0935 1 LOCAL
99 0936 1     desc:    vector [2, long];  ! Temporary string descriptor
100 0937 1
101 0938 1
102 0939 1 If .rab eql 0 then return true;  ! Exit immediately if no output
103 0940 1
104 0941 1 |
105 0942 1 | INITIALIZE THE RAB
106 0943 1 |     Store the buffer address and length in the RAB.
107 0944 1 |
108 0945 1 |
109 0946 1 | rab [rab$l_rbf] = .buffer;      ! Store buffer address in RAB
110 0947 1 | rab [rab$w_rsz] = .buffer [acc$w_msgsiz]; ! Store buffer size in RAB
111 0948 1 |
112 0949 1 |
113 0950 1 |
114 0951 1 | WRITE TO FILE ---
115 0952 1 |     Output the buffer via RMS.
116 0953 1 |
117 0954 1 |
118 0955 1 |
119 P 0956 1 Perform ($put (      ! Call RMS with
120 P 0957 1     rab = .rab,      ! -record stream identifier
121 0958 1     err = log_filename)); ! -error action routine
122 0959 1
123 0960 1 return true;
124 0961 1 END;

```

.TITLE UTILITY

.IDENT \V04-000\

.EXTRN ACCS_INVACCRC, ACCS_TOTAL
.EXTRN ACCS_MERGE, ACCS_INPOT
.EXTRN ACCS_TITLETRUNC
.EXTRN ADD_SYMBOL, ALLOCATE
.EXTRN FIND_WATERMARK, HANDLER
.EXTRN LIB\$ADDX, LIB\$CVT DTB
.EXTRN LIB\$CVT_HTB, LIB\$CVT TIME
.EXTRN LIB\$DAY, LIB\$FILE_SCAN
.EXTRN LIB\$ICHAR, LIB\$LOOKUP_KEY
.EXTRN LIB\$SUBX, LIB\$SYS_ASCTIM
.EXTRN LIB\$SYS_FAO, LIB\$SYS_FAO_L
.EXTRN LIB\$TPARSE, LOG_FILENAME
.EXTRN LOOKUP_SYMBOL, MAP_QUALIFIERS
.EXTRN PARSE_OUTPUT_FILES
.EXTRN RELEASE_TO_SORT
.EXTRN SCAN_SYMBOLS, SHOW_RECORD
.EXTRN SOR\$END_SORT, SOR\$INIT_SORT
.EXTRN SOR\$RELEASE_REC
.EXTRN SOR\$RETURN_REC, SOR\$SORT_MERGE
.EXTRN STR\$APPEND, STR\$COMPARE
.EXTRN STR\$DUPL_CHAR, STR\$LEFT
.EXTRN STR\$PREFIX, STR\$REPLACE
.EXTRN STR\$RIGHT, STRIP_NEGATOR
.EXTRN STRIP_TRAIL, SUMMARIZE
.EXTRN SYSS\$NOMTIM, TRANSLATE_STATUS
.EXTRN WRITE_BAR_GRAPH
.EXTRN WRITE_BINARY, WRITE_SUMMARY
.EXTRN WRITE_TOTALS, SYSS\$PUT

.PSECT CODE, NOWRT, 2

			0000	00000
	5E		08	C2 00002
	51	08	AC	D0 00005
			1F	13 00009
	50	04	AC	D0 0000B
28	A1		50	D0 0000F
22	A1	02	A0	B0 00013
		00000000G	00	9F 00018
			51	DD 0001E
00000000G	00		02	FB 00020
	03		50	E9 00027
	50		01	D0 0002A 1\$:
			04	0002D 2\$:

.ENTRY	WRITE_BINARY, Save nothing	: 0910
SUBL2	#8, SP	: 0939
MOVL	RAB, R1	: 0946
BEQL	1\$: 0947
MOVL	BUFFER, R0	: 0958
MOVL	R0, 40(R1)	: 0960
MOVW	2(R0), 34(R1)	: 0961
PUSHAB	LOG_FILENAME	
PUSHL	R1	
CALLS	#2, SYSS\$PUT	
BLBC	STATUS, 2\$	
MOVL	#1, R0	
RET		

; Routine Size: 46 bytes, Routine Base: CODE + 0000

```

126 0962 1 UNDECLARE STRIP_NEGATOR;
127 0963 1
128 0964 1 GLOBAL ROUTINE STRIP_NEGATOR (TARGET) =
129 0965 1
130 0966 1 |-----
131 0967 1 |
132 0968 1 | Functional description
133 0969 1 |
134 0970 1 | This routine is called to strip list negators. If the first
135 0971 1 | character of the string pointed to by the descriptor is the
136 0972 1 | negator character then strip that character from the string and
137 0973 1 | return TRUE else return FALSE.
138 0974 1 |
139 0975 1 | Input parameters
140 0976 1 |
141 0977 1 | TARGET = Address of a descriptor
142 0978 1 |
143 0979 1 | Output parameters
144 0980 1 |
145 0981 1 | If the item is negated return FALSE
146 0982 1 | Else return TRUE
147 0983 1 | Any errors encountered are RETURNed immediately.
148 0984 1 |
149 0985 1 |-----
150 0986 1
151 0987 2 BEGIN
152 0988 2
153 0989 2
154 0990 2 MAP
155 0991 2 target: ref bblock [dsc$k_d_bln];! Describe te input parameter
156 0992 2
157 0993 2
158 0994 2 |
159 0995 2 | TEST FIRST CHARACTER --
160 0996 2 | If its te negator character then strip the character and
161 0997 2 | return false.
162 0998 2 |
163 0999 2 |
164 1000 2 If .target [dsc$w_length] neq 0 ! If string is non-null
165 1001 2 and lib$ichar (.target) eql negator ! - and char[1] = negator
166 1002 2 then BEGIN
167 1003 2 str$right (.target, .target, %ref(2)); ! - then strip it off
168 1004 2 return false; ! - and tell the user
169 1005 2 END;
170 1006 2
171 1007 2 return true;
172 1008 1 END;

```

```

SE          0004 00000      .ENTRY STRIP_NEGATOR, Save R2      : 0964
52          04  C2 00002     SUBL2 #4, SP                          :
AC          62  D0 00005     MOVL TARGET, R2                          : 1000
62          20  B5 00009     TSTW (R2)
20          13  0000B     BEQL 1$

```

00000000G	00	52	DD	0000D	PUSHL	R2		1001
	2D	01	FB	0000F	CALLS	#1,	LIB\$ICHR	
		50	D1	00016	CMPL	R0,	#45	
		12	12	00019	BNEQ	1\$		
	6E	02	D0	0001B	MOVL	#2,	(SP)	1003
		8F	BB	0001E	PUSHR	#2,	<R2,SP>	
		52	DD	00022	PUSHL	R2		
00000000G	00	03	FB	00024	CALLS	#3,	STR\$RIGHT	
		04	11	0002B	BRB	2\$		1004
	50	01	D0	0002D	MOVL	#1,	R0	1007
			04	00030	RET			
		50	D4	00031	CLRL	R0		1008
			04	00033	RET			

: Routine Size: 52 bytes, Routine Base: CODE + 002E

```

174 1009 1 UNDECLARE STRIP_TRAIL;
175 1010 1
176 1011 1 GLOBAL ROUTINE STRIP_TRAIL (DESC) =
177 1012 1
178 1013 1 |-----
179 1014 1 |
180 1015 1 | Functional description
181 1016 1 |
182 1017 1 |     This routine is called to strip garbage from the end of a string.
183 1018 1 |
184 1019 1 | Input parameters
185 1020 1 |
186 1021 1 |     DESC     = address of a descriptor of the string
187 1022 1 |
188 1023 1 | Output parameters
189 1024 1 |
190 1025 1 |     The byte count in the descriptor is decremented by the
191 1026 1 |     number of bytes of trailing garbage.  Garbage is defined
192 1027 1 |     as NULLS, TABS, and SPACES.
193 1028 1 |-----
194 1029 1 |
195 1030 1
196 1031 1 BEGIN
197 1032 1
198 1033 1 MAP
199 1034 1     desc:  ref bblock [dsc$k_d_bln];      ! Address of a descriptor
200 1035 1
201 1036 1     decr ptr from (.desc [dsc$a_pointer] + .desc [dsc$w_length] - 1)
202 1037 1         to  (.desc [dsc$a_pointer])
203 1038 1     do
204 1039 1         if  (.ptr)<0,8> eql 0              ! NULLS
205 1040 1         or  (.ptr)<0,8> eql ' '          ! TABS
206 1041 1         or  (.ptr)<0,8> eql ' '          ! SPACES
207 1042 1         then desc [dsc$w_length] = .desc [dsc$w_length] - 1
208 1043 1         else exitloop;
209 1044 1
210 1045 1
211 1046 1     return true;
212 1047 1
213 1048 1 END;

```

			0004 00000	.ENTRY	STRIP_TRAIL, Save R2	: 1011
51	04	AC	D0 00002	MOVL	DESC, R1	: 1036
52	04	A1	D0 00006	MOVL	4(R1), R2	
50		61	3C 0000A	MOVZWL	(R1), R0	
50		52	C0 0000D	ADDL2	R2, R0	
		10	11 00010	BRB	3\$	
		60	95 00012	1\$: TSTB	(PTR)	: 1039
		0A	13 00014	BEQL	2\$	
09		60	91 00016	CMPB	(PTR), #9	: 1040
		05	13 00019	BEQL	2\$	
20		60	91 0001B	CMPB	(PTR), #32	: 1041
		09	12 0001E	BNEQ	4\$	

UTILITY
V04-000

H 10
12-Sep-1984 23:51:05
14-Sep-1984 11:52:08

VAX-11 BLISS-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]UTILITY.B32;1 Page 28
(4)

	61	B7	00020	2\$:	DECW	(R1)	:	1042
	50	D7	00022	3\$:	DECL	PTR	:	1039
52	50	D1	00024		CPL	PTR, R2	:	
	E9	18	00027		BGEQ	1\$:	
50	01	D0	00029	4\$:	MOVL	#1, R0	:	1046
	04	0002C			RET		:	1048

; Routine Size: 45 bytes, Routine Base: CODE + 0062

```

215 1049 1 UNDECLARE TRANSLATE_STATUS;
216 1050 1
217 1051 1 GLOBAL ROUTINE TRANSLATE_STATUS (EXIT_STATUS,DESCRIP) =
218 1052 1
219 1053 1 !++
220 1054 1
221 1055 1 Functional description:
222 1056 1
223 1057 1 This routine will return the text associated with a given exit status.
224 1058 1 If the status code has no text associated with it, return a routine
225 1059 1 status value of SSS_MSGNOTFND, and the text "<no text>" in DESCRIP.
226 1060 1
227 1061 1 Input:
228 1062 1
229 1063 1 EXIT_STATUS - a longword containing the exit status
230 1064 1
231 1065 1 Output:
232 1066 1
233 1067 1 DESCRIP - address of a quadword descriptor that points to the
234 1068 1 buffer to receive the message text. Note that the
235 1069 1 size field of the descriptor will reflect the actual
236 1070 1 message length.
237 1071 1
238 1072 1 !--
239 1073 1
240 1074 2 BEGIN
241 1075 2
242 1076 2 MAP
243 1077 2 exit_status : REF bblock [LONG],
244 1078 2 descrip : REF BBLOCK [dsc$c_s_bln];
245 1079 2
246 1080 2 OWN
247 1081 2 no_text_msg : initdesc ('<no text>');
248 1082 2
249 1083 2 LOCAL
250 1084 2 status : bblock [LONG];
251 1085 2
252 1086 2
253 1087 2 ! TRANSLATE THE EXIT STATUS ---
254 1088 2
255 1089 2
256 1090 2 P P P status= ($getmsg ( ! Call message translator with
257 1091 2 msgid = .exit_status, ! -message code
258 1092 2 msglen = descrip [dsc$w_length], ! -$GETMSG returns length here
259 1093 2 bufadr = .descrip)); ! -address of work buffer desc
260 1094 2
261 1095 2
262 1096 2 ! If the status code returned by $GETMSG
263 1097 2 was SSS_MSGNOTFND, then return the 'no text' message.
264 1098 2
265 1099 2
266 1100 2 IF .status EQL ss$msgnotfnd
267 1101 2 THEN
268 1102 2 BEGIN
269 1103 2 CHSMOVE ( .no_text_msg [dsc$w_length], ! Length
270 1104 2 .no_text_msg [dsc$a_pointer], ! Source address
271 1105 2 .descrip [dsc$a_pointer] ! Destination address

```

```

: 272      1106      3
: 273      1107      3      )
: 274      1108      3      )
: 275      1109      3      )
: 276      1110      3      )
: 277      1111      3      )
      1      RETURN .status
      1      END;

```

```

                                .PSECT DATA,NOEXE,2
                                00000009 00000 NO_TEXT_MSG:
                                00000000' 00004 .LONG 9
                                .ADDRESS P.AAA
                                .PSECT CODE,NOWRT,2
00 00 00 3E 74 78 65 74 20 6F 6E 3C 0008F .BLKB 1
                                00090 P.AAA: .ASCII \<no text>\<0><0><0>
                                .EXTRN SYSSGETMSG
                                .ENTRY TRANSLATE_STATUS, Save R2,R3,R4,R5,R6,R7,R8 : 1051
58 00000000' EF 9E 00002 MOVAB NO_TEXT_MSG, R8 : 1093
7E 0F 7D 00009 MOVQ #15, -(SP)
56 08 AC D0 0000C MOVL DESCRIP, R6
                                56 DD 00010 PUSHL R6
                                56 DD 00012 PUSHL R6
                                04 AC DD 00014 PUSHL EXIT_STATUS
00000000G 00 05 FB 00017 CALLS #5, SYSSGETMSG
00000621 8F 50 D0 0001E MOVL R0, STATUS
                                09 12 00028 CMPL STATUS, #1569 : 1100
04 B6 04 B8 68 28 0002A BNEQ 1$
                                66 68 B0 00030 MOVC3 NO_TEXT_MSG, @NO_TEXT_MSG+4, @4(R6) : 1105
                                50 57 D0 00033 1$: MOVW NO_TEXT_MSG, (R6) : 1107
                                04 00036 RET MOVL STATUS, -R0 : 1110
                                : 1111

```

: Routine Size: 55 bytes, Routine Base: CODE + 009C

```

279 1112 1 UNDECLARE LOG_FILENAME;
280 1113 1
281 1114 1 GLOBAL ROUTINE LOG_FILENAME (RMS) =
282 1115 1
283 1116 1 |-----
284 1117 1 |
285 1118 1 | Functional description
286 1119 1 |
287 1120 1 |     This routine is called to signal a message to
288 1121 1 |     the user based on an error code and file name
289 1122 1 |     that are imbedded in the passed parameter.
290 1123 1 |
291 1124 1 | Input parameters
292 1125 1 |
293 1126 1 |     RMS = Either a FAB or a RAB
294 1127 1 |     RAB$FAB = pointer to fab block           (If input was a RAB)
295 1128 1 |     FAB$NAM = pointer to name block
296 1129 1 |     RAB$CTX = error message to be used     (If input was a RAB)
297 1130 1 |     FAB$CTX = error message to be used     (If input was a FAB)
298 1131 1 |
299 1132 1 |
300 1133 1 | Output parameters
301 1134 1 |
302 1135 1 |     Expanded error messages to user
303 1136 1 |     Status is RETURNed
304 1137 1 |
305 1138 1 |-----
306 1139 1 |
307 1140 2 BEGIN
308 1141 2
309 1142 2 MAP
310 1143 2     rms:  ref bblock;                ! Define block format
311 1144 2
312 1145 2
313 1146 2 LOCAL
314 1147 2     fab: ref bblock,                ! Pointer to FAB block
315 1148 2     nam: ref bblock,                ! Pointer to NAM block
316 1149 2     rms_sts,                       ! Temporary primary status holder
317 1150 2     rms_stv,                       ! Temporary secondary status holder
318 1151 2     rms_ctx,                       ! Temporary user context holder
319 1152 2     status: bblock [long],         ! Local "catch all" status return
320 1153 2     desc:  vector [2, long];       ! Temporary string descriptor
321 1154 2
322 1155 2
323 1156 2 |
324 1157 2 | SET UP VALUES --
325 1158 2 |     Fetch the primary and secondary status values and the user
326 1159 2 |     context field from the RMS structure.  If a RAB was passed
327 1160 2 |     then fetch the address of the associated FAB.
328 1161 2 |
329 1162 2 |
330 1163 2 | If .rms [rab$b_bid] eql rab$c_bid then ! If this is a rab
331 1164 2 | BEGIN
332 1165 2 |     fab = .rms [rab$f_fab];
333 1166 2 |     rms_sts = .rms [rab$f_sts];
334 1167 2 |     rms_stv = .rms [rab$f_stv];
335 1168 2 |     rms_ctx = .rms [rab$f_ctx];

```

```

336 1169 END
337 1170 else BEGIN
338 1171     fab = .rms;
339 1172     rms_sts = .rms [fab$_sts];
340 1173     rms_stv = .rms [fab$_stv];
341 1174     rms_ctx = .rms [fab$_ctx];
342 1175 END;
343 1176
344 1177 nam = .fab [fab$_nam];           ! Fetch address of NAM block
345 1178
346 1179
347 1180
348 1181
349 1182
350 1183
351 1184
352 1185
353 1186
354 1187 If .rms [rab$_bid] eql rab$_cid   ! If this is a rab
355 1188 and .rms_sts eql rms$_eof        ! - and error is end of file
356 1189 and .rms_ctx eql msg$_readerr  ! - and this was a read call
357 1190 then return rms$_eof;           ! don't bother to report it
358 1191
359 1192
360 1193
361 1194
362 1195
363 1196
364 1197
365 1198
366 1199
367 1200
368 1201 If .nam[nam$_rsl] neq 0 then      ! IF result string nonblank,
369 1202 BEGIN                             ! then display it
370 1203     desc[0] = .nam[nam$_rsl];
371 1204     desc[1] = .nam[nam$_rsa];
372 1205 END
373 1206
374 1207 else if .nam[nam$_esl] neq 0 then ! Or if expanded name nonblank
375 1208 BEGIN                             ! then display it
376 1209     desc[0] = .nam[nam$_esl];
377 1210     desc[1] = .nam[nam$_esa];
378 1211 END
379 1212
380 1213
381 1214 else BEGIN
382 1215     desc[0] = .fab[fab$_fns];      ! Otherwise, use original
383 1216     desc[1] = .fab[fab$_fna];      ! name string in FAB
384 1217 END;
385 1218
386 1219
387 1220
388 1221
389 1222
390 1223
391 1224
392 1225

```

```

: 393      1226 2
: 394      1227 2
: 395      1228 2
: 396      1229 2
: 397      1230 2
: 398      1231 2
: 399      1232 2
: 400      1233 2
: 401      1234 1

```

```

signal (.rms_ctx, 1 ,desc,
      .rms_sts,
      .rms_stv);

return .rms_sts;

END;

```

```

! Output an error message
! with RMS error code
! and secondary code

! Pass on the status

```

			003C 00000	.ENTRY	LOG_FILENAME, Save R2,R3,R4,R5	1114
	5E		08 C2 00002	SUBL2	#8, SP	
	50	04	AC D0 00005	MOVL	RMS, R0	1163
			52 D4 00009	CLRL	R2	
	01		60 91 0000B	CMPB	(R0), #1	
			08 12 0000E	BNEQ	1\$	
			52 D6 00010	INCL	R2	
	51	3C	A0 D0 00012	MOVL	60(R0), FAB	1165
			03 11 00016	BRB	2\$	1166
	51		50 D0 00018	MOVL	R0, FAB	1171
	53	08	A0 D0 0001B	MOVL	8(R0), RMS_STS	1172
	55	0C	A0 D0 0001F	MOVL	12(R0), RMS_STV	1173
	54	18	A0 D0 00023	MOVL	24(R0), RMS_CTX	1174
	50	28	A1 D0 00027	MOVL	40(FAB), NAM	1177
	1A		52 E9 0002B	BLBC	R2, 3\$	1186
0001827A	8F		53 D1 0002E	CMPB	RMS_STS, #98938	1187
			11 12 00035	BNEQ	3\$	
009F10B2	8F		54 D1 00037	CMPB	RMS_CTX, #10424498	1188
			08 12 0003E	BNEQ	3\$	
	50	0001827A	8F D0 00040	MOVL	#98938, R0	1189
			04 00047	RET		
		03	A0 95 00048	TSTB	3(NAM)	1201
			0B 13 0004B	BEQL	4\$	
	6E	03	A0 9A 0004D	MOVZBL	3(NAM), DESC	1203
04	AE	04	A0 D0 00051	MOVL	4(NAM), DESC+4	1204
			19 11 00056	BRB	6\$	1201
		0B	A0 95 00058	TSTB	11(NAM)	1207
			0B 13 0005B	BEQL	5\$	
	6E	0B	A0 9A 0005D	MOVZBL	11(NAM), DESC	1209
04	AE	0C	A0 D0 00061	MOVL	12(NAM), DESC+4	1210
			09 11 00066	BRB	6\$	1207
	6E	34	A1 9A 00068	MOVZBL	52(FAB), DESC	1214
04	AE	2C	A1 D0 0006C	MOVL	44(FAB), DESC+4	1215
			28 BB 00071	PUSHR	#^M<R3,R5>	1228
		0B	AE 9F 00073	PUSHAB	DESC	1227
			01 DD 00076	PUSHL	#1	
			54 DD 00078	PUSHL	RMS_CTX	
00000000G	00		05 FB 0007A	CALLS	#5, _LIB\$SIGNAL	
	50		53 D0 00081	MOVL	RMS_STS, R0	1232
			04 00084	RET		1234

; Routine Size: 133 bytes, Routine Base: CODE + 00D3

UTILITY
V04-000

N 10
15-Sep-1984 23:51:05
14-Sep-1984 11:52:08

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[ACC.SRC]UTILITY.B32;1 Page 34
(6)

```

: 403      1235 1 UNDECLARE MAP_QUALIFIERS;
: 404      1236 1
: 405      1237 1 GLOBAL ROUTINE MAP_QUALIFIERS =
: 406      1238 1
: 407      1239 1 -----
: 408      1240 1
: 409      1241 1     Functional description
: 410      1242 1
: 411      1243 1         This routine searches all possible qualifiers and makes
: 412      1244 1         a bit mask of those that are present
: 413      1245 1
: 414      1246 1     Input parameters
: 415      1247 1
: 416      1248 1         None
: 417      1249 1
: 418      1250 1     Output parameters
: 419      1251 1
: 420      1252 1         The mask is established
: 421      1253 1         Always returns success
: 422      1254 1
: 423      1255 1 -----
: 424      1256 1
: 425      1257 1 BEGIN
: 426      1258 1
: 427      1259 1 GLOBAL
: 428      1260 1     qualifiers: bitvector [64];           ! global storage for mask
: 429      1261 1
: 430      1262 1 OWN
: 431      1263 1     qual_array: vector [qual_count * 2] initial (
: 432      1264 1         primdesc (ACCOUNT),
: 433      1265 1         primdesc (ACCOUNT),
: 434      1266 1         !
: 435      1267 1         primdesc (BAR_GRAPH),
: 436      1268 1         primdesc (BEFORE),
: 437      1269 1         primdesc (BINARY),
: 438      1270 1         primdesc (USER),
: 439      1271 1         primdesc (ENTRY),
: 440      1272 1         primdesc (FULL),
: 441      1273 1         primdesc (IDENT),
: 442      1274 1         primdesc (IMAGE),
: 443      1275 1         primdesc (JOB),
: 444      1276 1         primdesc (LOG),
: 445      1277 1         primdesc (ADDRESS),
: 446      1278 1         primdesc (NODE),
: 447      1279 1         primdesc (OWNER),
: 448      1280 1         primdesc (OUTPUT),
: 449      1281 1         primdesc (PRIORITY),
: 450      1282 1         primdesc (PROCESS),
: 451      1283 1         primdesc (QUEUE),
: 452      1284 1         primdesc (REJECTED),
: 453      1285 1         primdesc (REMOTE_ID),
: 454      1286 1         primdesc (REPORT),
: 455      1287 1         primdesc (SINCE),
: 456      1288 1         primdesc (STATUS),
: 457      1289 1         primdesc (SORT),
: 458      1290 1         primdesc (SUMMARY),
: 459      1291 1         primdesc (TERMINAL),
:         1291 1         primdesc (TITLE),

```

```

: 460      1292      Z          primdesc (TYPE),
: 461      1293      Z          primdesc (UIC)
: 462      1294      Z          );
: 463      1295      Z
: 464      1296      Z
: 465      1297      Z          Establish a bit mask of qualifiers that are present
: 466      1298      Z
: 467      1299      Z
: 468      1300      Z          Inca i to qual_count - 1 do qualifiers [.i] = get_present (qual_array [.i*2]);
: 469      1301      Z          qualifiers [bar_graph] = 0;
: 470      1302      Z          return true;
: 471      1303      Z          END;

```

```

                                .PSECT DATA,NOEXE,2
                                00008 QUALIFIERS::
                                .BLKB 8
00000007 00010 QUAL_ARRAY:
                                .LONG 7
00000000' 00014 .ADDRESS P.AAB
00000007 00018 .LONG 7
00000000' 0001C .ADDRESS P.AAC
00000006 00020 .LONG 6
00000000' 00024 .ADDRESS P.AAD
00000006 00028 .LONG 6
00000000' 0002C .ADDRESS P.AAE
00000004 00030 .LONG 4
00000000' 00034 .ADDRESS P.AAF
00000005 00038 .LONG 5
00000000' 0003C .ADDRESS P.AAG
00000004 00040 .LONG 4
00000000' 00044 .ADDRESS P.AAH
00000005 00048 .LONG 5
00000000' 0004C .ADDRESS P.AAI
00000005 00050 .LONG 5
00000000' 00054 .ADDRESS P.AAJ
00000003 00058 .LONG 3
00000000' 0005C .ADDRESS P.AAK
00000003 00060 .LONG 3
00000000' 00064 .ADDRESS P.AAL
00000007 00068 .LONG 7
00000000' 0006C .ADDRESS P.AAM
00000004 00070 .LONG 4
00000000' 00074 .ADDRESS P.AAN
00000005 00078 .LONG 5
00000000' 0007C .ADDRESS P.AAO
00000006 00080 .LONG 6
00000000' 00084 .ADDRESS P.AAP
00000008 00088 .LONG 8
00000000' 0008C .ADDRESS P.AAQ
00000007 00090 .LONG 7
00000000' 00094 .ADDRESS P.AAR
00000005 00098 .LONG 5
00000000' 0009C .ADDRESS P.AAS
00000008 000A0 .LONG 8

```

.....

```

00000000' 000A4 .ADDRESS P.AAT
00000009' 000A8 .LONG 9
00000000' 000AC .ADDRESS P.AAU
00000006' 000B0 .LONG 6
00000000' 000B4 .ADDRESS P.AAV
00000005' 000B8 .LONG 5
00000000' 000BC .ADDRESS P.AAW
00000006' 000C0 .LONG 6
00000000' 000C4 .ADDRESS P.AAX
00000004' 000C8 .LONG 4
00000000' 000CC .ADDRESS P.AAY
00000007' 000D0 .LONG 7
00000000' 000D4 .ADDRESS P.AAZ
00000008' 000D8 .LONG 8
00000000' 000DC .ADDRESS P.ABA
00000005' 000E0 .LONG 5
00000000' 000E4 .ADDRESS P.ABB
00000004' 000E8 .LONG 4
00000000' 000EC .ADDRESS P.ABC
00000003' 000F0 .LONG 3
00000000' 000F4 .ADDRESS P.ABD

```

.PSECT CODE,NOWRT,2

```

00 54 4E 55 4F 43 43 41 00158 P.AAB: .ASCII \ACCOUNT\<0>
00 54 4E 55 4F 43 43 41 00160 P.AAC: .ASCII \ACCOUNT\<0>
00 00 45 52 4F 46 45 42 00168 P.AAD: .ASCII \BEFORE\<0><0>
00 00 59 52 41 4E 49 42 00170 P.AAE: .ASCII \BINARY\<0><0>
00 00 00 59 52 45 53 55 00178 P.AAF: .ASCII \USER\
00 00 00 59 52 54 4E 45 0017C P.AAG: .ASCII \ENTRY\<0><0><0>
00 00 00 54 4E 4C 4C 46 00184 P.AAH: .ASCII \FULL\
00 00 00 54 4E 45 44 49 00188 P.AAI: .ASCII \IDENT\<0><0><0>
00 00 00 45 47 41 4D 49 00190 P.AAJ: .ASCII \IMAGE\<0><0><0>
00 00 00 42 4F 4A 00198 P.AAK: .ASCII \JOB\<0>
00 00 00 47 4F 4C 0019C P.AAL: .ASCII \LOG\<0>
00 53 53 45 52 44 44 41 001A0 P.AAM: .ASCII \ADDRESS\<0>
00 00 00 52 45 44 4F 4E 001A8 P.AAN: .ASCII \NODE\
00 00 00 54 55 50 54 55 4F 001AC P.AAO: .ASCII \OWNER\<0><0><0>
59 54 49 52 4F 49 52 50 001B4 P.AAP: .ASCII \OUTPUT\<0><0>
00 53 53 45 43 4F 52 50 001BC P.AAQ: .ASCII \PRIORITY\
00 00 00 45 55 45 55 51 001C4 P.AAR: .ASCII \PROCESS\<0>
44 45 54 43 45 4A 45 52 001CC P.AAS: .ASCII \QUEUE\<0><0><0>
00 00 00 44 45 54 4F 4D 45 52 001D4 P.AAT: .ASCII \REJECTED\
00 00 00 54 52 4F 50 45 52 001DC P.AAU: .ASCII \REMOTE_ID\<0><0><0>
00 00 00 45 43 4E 49 53 001E8 P.AAV: .ASCII \REPORT\<0><0>
00 00 53 55 54 41 54 53 001F0 P.AAW: .ASCII \SINCE\<0><0><0>
00 00 53 55 54 41 54 53 001F8 P.AAX: .ASCII \STATUS\<0><0>
00 59 52 41 4D 4D 55 53 00200 P.AAY: .ASCII \SORT\
4C 41 4E 49 4D 52 45 54 00204 P.AAZ: .ASCII \SUMMARY\<0>
00 00 00 45 4C 54 49 54 0020C P.ABA: .ASCII \TERMINAL\
45 50 59 54 00214 P.ABB: .ASCII \TITLE\<0><0><0>
00 43 49 55 0021C P.ABC: .ASCII \TYPE\
00 43 49 55 00220 P.ABD: .ASCII \UIC\<0>

```

.EXTRN CLISPRESNT

000C 00000

.ENTRY MAP_QUALIFIERS, Save R2,R3

		53	00000000'	EF	9E	00002	MOVAB	QUALIFIERS, R3		
				52	D4	00009	CLRL	I	:	1300
	50	52		01	78	0000B	ASHL	#1, I, R0	:	
			08 A340	DF		0000F	PUSHAL	QUAL_ARRAY[R0]	:	
63		00		01	FB	00013	CALLS	#1, CLISPRESNT	:	
	01	52	00000000G	50	F0	0001A	INSV	R0, I, #1, QUALIFIERS	:	
				52	D6	0001F	INCL	I	:	
		1C		52	D1	00021	CMPL	I, #28	:	
				E5	1B	00024	BLEQU	1\$:	
		63		02	8A	00026	BICB2	#2, QUALIFIERS	:	1301
		50		01	D0	00029	MOVL	#1, R0	:	1302
				04	0002C		RET		:	1303

: Routine Size: 45 bytes, Routine Base: CODE + 0224

```

473 1304 1 UNDECLARE HANDLER;
474 1305
475 1306 GLOBAL ROUTINE HANDLER (SIGNAL_ARGS, MECHANISM_ARGS) =
476 1307
477 1308 |---
478 1309 |
479 1310 |       This condition handler gets control on any signalled
480 1311 |       condition in order to save the highest severity error
481 1312 |       to be returned by exit from the image.
482 1313 |
483 1314 |       Inputs:
484 1315 |
485 1316 |       signal_args = Address of signal argument list
486 1317 |       mechanism_args = Address of mechanism argument list
487 1318 |
488 1319 |       Outputs:
489 1320 |
490 1321 |       WORST_ERROR is updated with highest severity error.
491 1322 |---
492 1323
493 1324
494 1325 BEGIN
495 1326
496 1327 GLOBAL worst_error: initial (1);          ! Holds worst error encountered
497 1328
498 1329 MAP
499 1330     signal_args:      ref bblock,          ! Address of signal argument list
500 1331     mechanism_args:  ref bblock;         ! Address of mechanism argument list
501 1332
502 1333 LOCAL
503 1334     cond_code:        long;              ! Condition code (longword)
504 1335
505 1336     cond_code = .signal_args [chf$l_sig_name];    ! Get condition code
506 1337     If .cond_code eql rms$eof then return true;
507 1338     If severity_level (.cond_code) gtr
508 1339         severity_level (.worst_error)          ! If higher than watermark
509 1340     then worst_error = .cond_code or sts$m_inhib_msg; ! -then set new worst error
510 1341
511 1342     ss$_resignal          ! Continue signalling
512 1343
513 1344 1 END;

```

```

.PSECT DATA,NOEXE,2
00000001 000F8 WORST_ERROR::
.LONG 1 ;

.PSECT CODE,NOWRT,2
54 00000000' 001C 0000 .ENTRY HANDLER, Save R2,R3,R4 ; 1306
50 04 AC D0 00002 MOVAB WORST_ERROR, R4 ;
53 04 A0 D0 00009 MOVL SIGNAL_ARGS, R0 ; 1336
MOVL 4(R0), COND_CODE ;

```


: 515 1345 1 END
: 516 1346 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
CODE	687	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
DATA	252	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	57	0	581	00:01.0
_\$255\$DUA28:[SYSLIB]TPAMAC.L32;1	42	0	0	14	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:UTILITY/OBJ=OBJ\$:UTILITY MSRC\$:UTILITY/UPDATE=(ENHS:UTILITY)

: Size: 470 code + 469 data bytes
 : Run Time: 00:19.5
 : Elapsed Time: 00:46.6
 : Lines/CPU Min: 4135
 : Lexemes/CPU-Min: 22525
 : Memory Used: 134 pages
 : Compilation Complete

