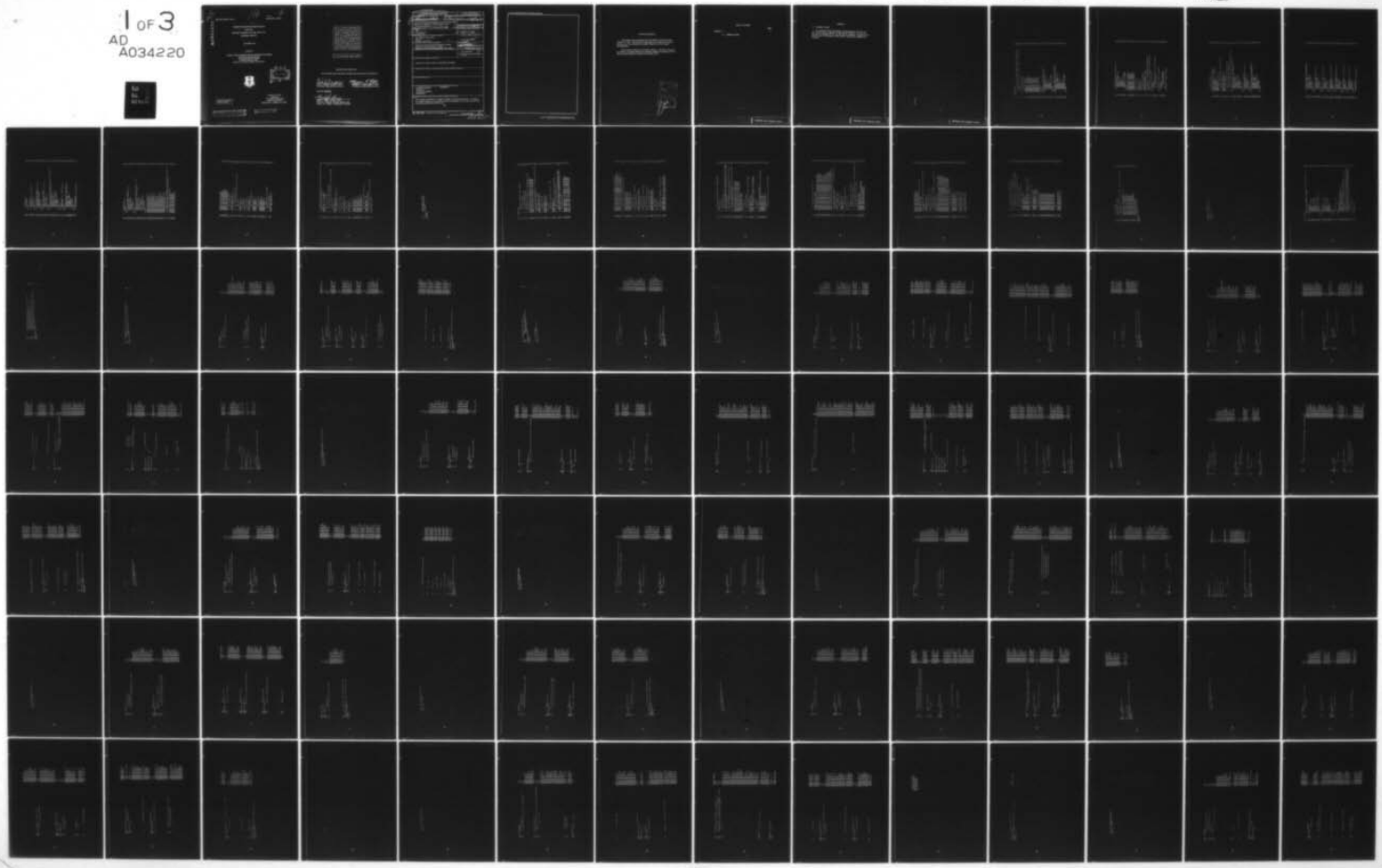
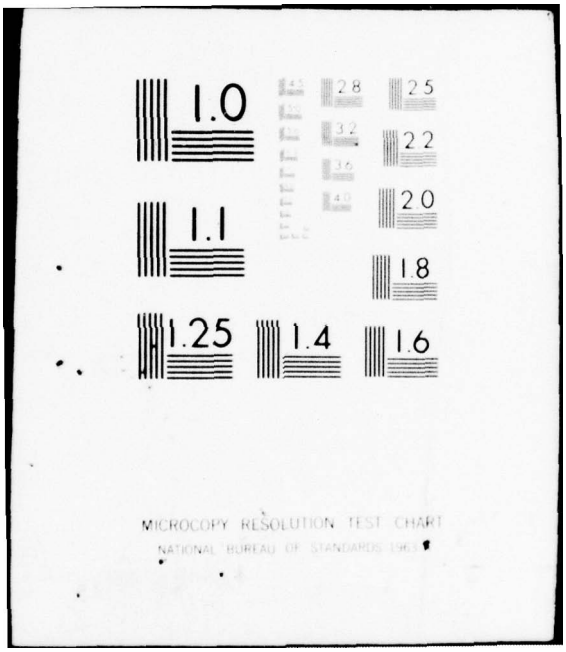
AD-A034 220 MITRE CORP BEDFORD MASS F/G 9/2
COMPUTER PROGRAM SPECIFICATION FOR THE SECURITY KERNEL FOR THE --ETC(U)
OCT 76 S R HARPER F19628-77-C-0001
UNCLASSIFIED MTR-3178-VOL-2 ESD-TR-76-288-VOL-2 NL

1 OF 3
AD A034220




1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963

ADA034220

ESD-TR-76-288, Vol. II

MTR-3178, Vol. II

COMPUTER PROGRAM SPECIFICATION
FOR THE
SECURITY KERNEL FOR THE PDP-11/45
PROGRAM LISTING

OCTOBER 1976

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

DDC
RECEIVED
JAN 11 1977
C

Approved for public release;
distribution unlimited.

Project No. 7070
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-77-C-0001

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

Copy available to DDC does not
permit fully legible reproduction

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

Paul A. Karger

PAUL A. KARGER, Captain, USAF
Techniques Engineering Division

Lawrence A. Noble

LAWRENCE A. NOBLE, Major, USAF
Techniques Engineering Division

FOR THE COMMANDER

Frank J. Emma

FRANK J. EMMA, Colonel, USAF
Director, Computer Systems Engineering
Deputy for Command & Management Systems

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: (18) ESD-TR-76-288, Vol II-2	2. GOVT ACCESSION NO: (9) Technical report
4. TITLE (and Subtitle): (6) COMPUTER PROGRAM SPECIFICATION FOR THE SECURITY KERNEL FOR THE PDP-11/45, PROGRAM LISTING, Volume IIe	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER: (14) MTR-3178, Vol II-2
7. AUTHOR(s): (10) S.R. Harper	8. CONTRACT OR GRANT NUMBER(s): (15) F19628-77-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, Box 208, Bedford, MA 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 7070
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Command and Management Systems, Electronic Systems Division, AFSC, Hanscom Air Force Base, Bedford, MA 01731	12. REPORT DATE: (11) OCTOBER 1976
	13. NUMBER OF PAGES: 227 (12) 228 pa
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ACCESS CONTROL
OBJECTS
SECURITY KERNEL
SEGMENTS
SUBJECTS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume is Section 10, Program Listing, of the basic document. It contains a complete listing of all computer program components (CPC's) that comprise the Security Kernel for the PDP-11/45.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

235050

ACKNOWLEDGMENT

This report has been prepared by The MITRE Corporation under Project No. 7070. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

This volume is Section 10, Program Listing, of the basic document. It contains a complete listing of all computer program components (CPC's) that comprise the Security Kernel for the PDP-11/45.

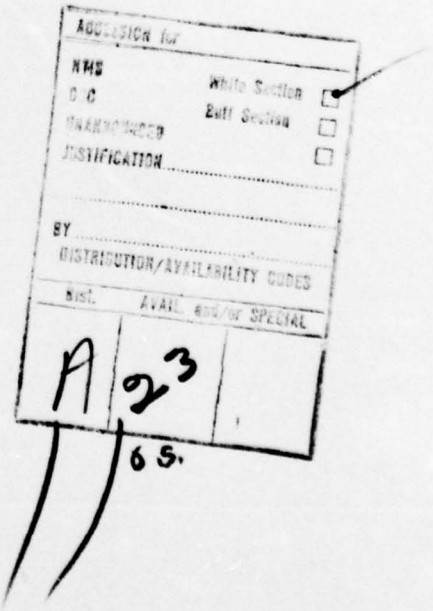


TABLE OF CONTENTS

	Page
APPENDIX I	5
10. PROGRAM LISTING	5

PRECEDING PAGE BLANK-NOT FILMED

APPENDIX I

10. PROGRAM LISTING

The program listing contained in this appendix includes the two context blocks relevant to the Security Kernel, the Project SUE System Language CPC's, and the PAL-11 assembler language CPC's. Emitted code is provided in the Project SUE System Language CPC listings.

PRECEDING PAGE BLANK-NOT FILMED

DATA NOFORN

PRECEDING PAGE BLANK-NOT FILMED

```
1   CONTEXT NOFORN;                                                                                    !0
                                                                                                       !
2       /*    THIS CONTEXT BLOCK CONTAINS DEFINITIONS RELEVANT TO ALL KERNEL USERS                     */!0
                                                                                                       !
3       /*    WARNING - WATCH OUT FOR BYTE VARIABLES - MOVB TO A REGISTER DOES A SIGN                  * !0
4        * EXTEND                                                                                     */!0
                                                                                                       !
5       CONSTANT BYTE_MASK = "00FF";                                                                   !0
                                                                                                       !
6       /*    HARDWARE INSTRUCTIONS                                                                    */!0
                                                                                                       !
7       CONSTANT ADD = "(3)0006";                                                                      !0
8       CONSTANT ASL = "(3)0063";                                                                      !0
9       CONSTANT ASR = "(3)0062";                                                                      !0
10      CONSTANT BCC = "(3)103(1)0";                                                                   !0
11      CONSTANT BVS = "(3)102(1)1";                                                                   !0
12      CONSTANT CLR = "(3)0050";                                                                      !0
13      CONSTANT DEC = "(3)0053";                                                                      !0
14      CONSTANT INC = "(3)0052";                                                                      !0
15      CONSTANT JMP = "(3)000001";                                                                    !0
16      CONSTANT MOV = "(3)0001";                                                                      !0
17      CONSTANT MOVB = "(3)0011";                                                                     !0
18      CONSTANT NEG = "(3)0054";                                                                      !0
19      CONSTANT SUB = "(3)0016";                                                                      !0
20      CONSTANT SWAB = "(3)0003";                                                                     !0
21      CONSTANT TRAP = "(3)104400";                                                                   !0
22      MACRO RTI;                                                                                     !0
23          0, 0, 0, 0, 2                                                                              !0
24      END MACRO;                                                                                     !0
                                                                                                       !
25      CONSTANT MUL = "(3)070100"; /* R1 := R0 * R1 */                                                !0
26      MACRO MULTIPLY(OP1, OP2, PRODUCT, FLAG);                                                       !0
27          INLINE(MOV, OP1, 0, 0);                                                                    !0
28          INLINE(MOV, OP2, 0, 1);                                                                    !0
29          INLINE(CLR, 0, 3);                                                                         !0
30          INLINE(MUL);                                                                               !0
31          INLINE(BCC, 1);                                                                            !0
32          INLINE(INC, 0, 3);                                                                         !0
33          INLINE(MOVB, 0, 3, FLAG);                                                                  !0
34          INLINE(MOV, 0, 1, PRODUCT)                                                                 !0
35      END MACRO;                                                                                     !0
                                                                                                       !
36      CONSTANT DIV = "(3)071002"; /* R0 := R0R1/R2 */                                                !0
37      MACRO DIVIDE(DIVIDEND, DIVISOR, QUOTIENT, FLAG);                                               !0
38          INLINE(CLR, 0, 0);                                                                         !0
39          INLINE(MOV, DIVIDEND, 0, 1);                                                               !0
40          INLINE(MOV, DIVISOR, 0, 2);                                                                !0
41          INLINE(CLR, 0, 3);                                                                         !0
42          INLINE(DIV);                                                                               !0
43          INLINE(BVS, 1);                                                                            !0
44          INLINE(BCC, 1);                                                                            !0
45          INLINE(INC, 0, 3);                                                                         !0
46          INLINE(MOVB, 0, 3, FLAG);                                                                  !0
47          INLINE(MOV, 0, 0, QUOTIENT)                                                                !0
48      END MACRO;                                                                                     !
```

```
49        MACRO MODULO(DIVIDEND, DIVISOR, REMAINDER, FLAG);
50             INLINE(CLR, 0, 0);
51             INLINE(MOV, DIVIDEND, 0, 1);
52             INLINE(MOV, DIVISOR, 0, 2);
53             INLINE(CLR, 0, 3);
54             INLINE(DIV);
55             INLINE(BVS, 1);
56             INLINE(BCC, 1);
57             INLINE(INC, 0, 3);
58             INLINE(MOVB, 0, 3, FLAG);
59             INLINE(MOV, 0, 1, REMAINDER)
60        END MACRO;

61        CONSTANT ASHR1 = "(3)072127";
62        CONSTANT ASHR3R3 = "(3)072003";
63        CONSTANT ASHR1R3 = "(3)072103";
64        CONSTANT XOR10 = "(3)074100";
65        CONSTANT MFPIR6 = "(3)006506";
66        CONSTANT MTPIR6 = "(3)006606";
67        CONSTANT SPLHIGH = "(3)000237";
68        CONSTANT SPLLOW = "(3)000230";

69        TYPE BYTE = BIT(8);

70        TYPE WORD = BIT(16);

71        CONSTANT TRUE = 1;
72        CONSTANT FALSE = 0;

73        TYPE BOOLEAN = (FALSE TO TRUE);

74        CONSTANT MAXIMUM_INTEGER = 32767;
75        CONSTANT MAX_NEG_INTEGER = - 32768;

76        TYPE INTEGER_TYPE = (- MAXIMUM_INTEGER TO MAXIMUM_INTEGER);

77        CONSTANT CV_MAX_LEN = 72; /* CHARACTER VARYING MAXIMUM LENGTH */

78        TYPE CV_LENGTH = (0 TO CV_MAX_LEN);

79        TYPE CHAR_VARYING =
80             RECORD
81                  ARRAY ((1 TO CV_MAX_LEN)) OF CHARACTER(1) (STRING),
82                  CV_LENGTH (LENGTH)
83             END;

84        TYPE CV_POINTER = POINTER TO CHAR_VARYING;

85        TYPE TOKEN_BLOCK =
86             RECORD
87                  CV_LENGTH (TOKEN_START),
88                  CV_LENGTH (TOKEN_LENGTH)
89             END;

90        TYPE TB_POINTER = POINTER TO TOKEN_BLOCK;
```

```
91        /*    TERMINAL IO DEFINITIONS                                                    */
92        CONSTANT CARRIAGE_RETURN = ' '; /* X'D' */
93        CONSTANT LINE_FEED = ' '; /* X'25' */
94        CONSTANT LOW_CHARACTER = ' '; /* X'80' */
95        CONSTANT HIGH_CHARACTER = ' '; /* X'07' */
96        CONSTANT NEW_LINE = ' ';
97        CONSTANT END_OF_FILE = ' ';
98        CONSTANT BS_CHAR = '8';
99        CONSTANT CANCEL_CHAR = LINE_FEED;

100       /*    TERMINAL IO CONTROL REGISTERS ARE ACCESSED VIA SEG REG 1                   */

101       ABSOLUTE ("2070") WORD TKS;
102       ABSOLUTE ("2072") CHARACTER(1) TKB;
103       ABSOLUTE ("2074") WORD TPS;
104       ABSOLUTE ("2076") CHARACTER(1) TPB;

105       /*    MACRO DEFINITIONS OF CALLS TO KERNEL FUNCTIONS                             */

106       /*    ASSUME STACK ACCESSABLE THRU SEGMENTATION REG 0                            */

107       /*    COMMENTS WITHIN MACROS ARE AVOIDED DUE TO LIMITED MACRO TABLE SIZE         */

108       MACRO KCREATE (SEG#, OFFSET, CLASS, CAT, SEG_TYPE, SIZE, RC);
109            FUNCTION_CODE_KPARM := CREATE_FUNCTION_CODE;
110            SEG#_KPARM := SEG#;
111            OFFSET_KPARM := OFFSET;
112            CLASS_KPARM := CLASS;
113            CAT_KPARM := CAT;
114            SEG_TYPE_KPARM := SEG_TYPE;
115            SIZE_KPARM := SIZE;
116            INLINE (TRAP);
117            RC := KERNEL_RC
118       END MACRO;

119       MACRO KDELETE (SEG#, OFFSET, RC);
120            FUNCTION_CODE_KPARM := DELETE_FUNCTION_CODE;
121            SEG#_KPARM := SEG#;
122            OFFSET_KPARM := OFFSET;
123            INLINE (TRAP);
124            RC := KERNEL_RC
125       END MACRO;

126       MACRO KGIVE (SEG#, OFFSET, MODE, USER, PROJECT, RC);
127            FUNCTION_CODE_KPARM := GIVE_FUNCTION_CODE;
128            SEG#_KPARM := SEG#;
129            OFFSET_KPARM := OFFSET;
130            MODE_KPARM := MODE;
131            USER_KPARM := USER;
132            PROJECT_KPARM := PROJECT;
133            INLINE (TRAP);
134            RC := KERNEL_RC
135       END MACRO;
```

```
136     MACRO KRESCIND (SEG#, OFFSET, USER, PROJECT, RC);                    10
137          FUNCTION_CODE_KPARM := RESCIND_FUNCTION_CODE;                   10
138          SEG#_KPARM := SEG#;                                             10
139          OFFSET_KPARM := OFFSET;                                         10
140          USER_KPARM := USER;                                             10
141          PROJECT_KPARM := PROJECT;                                       10
142          INLINE (TRAP);                                                  10
143          RC := KERNEL_RC                                                 10
144     END MACRO;                                                           10
                                                                             |
145     MACRO KGETW (SEG#, OFFSET, RC);                                      10
146          FUNCTION_CODE_KPARM := GETW_FUNCTION_CODE;                      10
147          SEG#_KPARM := SEG#;                                             10
148          OFFSET_KPARM := OFFSET;                                         10
149          INLINE (TRAP);                                                  10
150          RC := KERNEL_RC                                                 10
151     END MACRO;                                                           10
                                                                             |
152     MACRO KGETR (SEG#, OFFSET, RC);                                      10
153          FUNCTION_CODE_KPARM := GETR_FUNCTION_CODE;                      10
154          SEG#_KPARM := SEG#;                                             10
155          OFFSET_KPARM := OFFSET;                                         10
156          INLINE (TRAP);                                                  10
157          RC := KERNEL_RC                                                 10
158     END MACRO;                                                           10
                                                                             |
159     MACRO KRELEASE (SEG#);                                               10
160          FUNCTION_CODE_KPARM := RELEASE_FUNCTION_CODE;                   10
161          SEG#_KPARM := SEG#;                                             10
162          INLINE (TRAP)                                                   10
163     END MACRO;                                                           10
                                                                             |
164     MACRO KENABLE (SEG#, REG#, RC);                                      10
165          FUNCTION_CODE_KPARM := ENABLE_FUNCTION_CODE;                    10
166          SEG#_KPARM := SEG#;                                             10
167          REG#_KPARM := REG#;                                             10
168          INLINE (TRAP);                                                  10
169          RC := KERNEL_RC                                                 10
170     END MACRO;                                                           10
                                                                             |
171     MACRO KDISABLE (REG#);                                               10
172          FUNCTION_CODE_KPARM := DISABLE_FUNCTION_CODE;                   10
173          REG#_KPARM := REG#;                                             10
174          INLINE (TRAP)                                                   10
175     END MACRO;                                                           10
                                                                             |
176     MACRO KP (SEG#, RC);                                                 10
177          FUNCTION_CODE_KPARM := P_FUNCTION_CODE;                         10
178          SEG#_KPARM := SEG#;                                             10
179          INLINE (TRAP);                                                  10
180          RC := KERNEL_RC                                                 10
181     END MACRO;                                                           10
                                                                             |
182     MACRO KV (SEG#, RC);                                                 10
183          FUNCTION_CODE_KPARM := V_FUNCTION_CODE;                         10
```

```
184             SEG#_KPARM := SEG#;                                                    !0
185             INLINE (TRAP);                                                         !0
186             RC := KERNEL_RC                                                        !0
187         END MACRO;                                                                 !0

188         MACRO KT (SEG#, PC);                                                       !
189             FUNCTION_CODE_KPARM := T_FUNCTION_CODE;                                !0
190             SEG#_KPARM := SEG#;                                                    !0
191             INLINE (TRAP);                                                         !0
192             RC := KERNEL_RC                                                        !0
193         END MACRO;                                                                 !0

194         MACRO KIPCSEND (PROCESS#, MESSAGE);                                        !
195             FUNCTION_CODE_KPARM := IPCSEND_FUNCTION_CODE;                          !0
196             PROCESS#_KPARM := PROCESS#;                                            !0
197             MESSAGE_KPARM := MESSAGE;                                              !0
198             INLINE (TRAP);                                                         !0
199         END MACRO;                                                                 !0

200         MACRO KIPCRCV (PROCESS#, MESSAGE);                                         !
201             FUNCTION_CODE_KPARM := IPCRCV_FUNCTION_CODE;                           !0
202             INLINE (TRAP);                                                         !0
203             PROCESS# := KERNEL_RC;                                                 !0
204             MESSAGE := KERNEL_RC2                                                  !0
205         END MACRO;                                                                 !0

206         MACRO KSTARTP (USER_ID, PROJECT_ID, CLASS, CAT, PROCESS#, PROC_OFFSET, RC); !
207             FUNCTION_CODE_KPARM := STARTP_FUNCTION_CODE;                           !0
208             USER_KPARM := USER_ID;                                                 !0
209             PROJECT_KPARM := PROJECT_ID;                                           !0
210             CLASS_KPARM := CLASS;                                                  !0
211             CAT_KPARM := CAT;                                                      !0
212             PROCESS#_KPARM := PROCESS#;                                            !0
213             OFFSET_KPARM := PROC_OFFSET;                                           !0
214             INLINE (TRAP);                                                         !0
215             PC := KERNEL_RC                                                        !0
216         END MACRO;                                                                 !0

217         MACRO KSTOPP;                                                              !
218             FUNCTION_CODE_KPARM := STOPP_FUNCTION_CODE;                            !0
219             INLINE (TRAP)                                                          !0
220         END MACRO;                                                                 !0

221         MACRO KCHANGEO (SEG#, OFFSET, CLASS, CAT, RC);                             !
222             FUNCTION_CODE_KPARM := CHANGEO_FUNCTION_CODE;                          !0
223             SEG#_KPARM := SEG#;                                                    !0
224             OFFSET_KPARM := OFFSET;                                                !0
225             CLASS_KPARM := CLASS;                                                  !0
226             CAT_KPARM := CAT;                                                      !0
227             INLINE (TRAP);                                                         !0
228             RC := KERNEL_RC                                                        !0
229         END MACRO;                                                                 !0

230         MACRO KPROCID (PROCESS#);                                                  !
231             FUNCTION_CODE_KPARM := PROCID_FUNCTION_CODE;                           !0
```

```
232             INLINE (TRAP);                                                  !0
233             PROCESS0 := KERNEL_RC                                           !0
234         END MACRO;                                                          !
235         MACRO KINITH (SEG0, OFFSET, ASTE0, RC);                             !0
236             FUNCTION_CODE_KPARM := INITH_FUNCTION_CODE;                     !0
237             SEG0_KPARM := SEG0;                                             !0
238             OFFSET_KPARM := OFFSET;                                         !0
239             CAT_KPARM := ASTE0;                                             !0
240             INLINE (TRAP);                                                  !0
241             RC := KERNEL_RC                                                 !0
242         END MACRO;                                                          !
243         MACRO KREADIR (SEG0, OFFSET, CLASS, CAT, SEG_TYPE, SIZE, RC);      !0
244             FUNCTION_CODE_KPARM := READIR_FUNCTION_CODE;                    !0
245             SEG0_KPARM := SEG0;                                             !0
246             OFFSET_KPARM := OFFSET;                                         !0
247             INLINE (TRAP);                                                  !0
248             CLASS := CLASS_KPARM;                                           !0
249             CAT := CAT_KPARM;                                               !0
250             SEG_TYPE := SEG_TYPE_KPARM;                                     !0
251             SIZE := SIZE_KPARM;                                             !0
252             RC := KERNEL_RC                                                 !0
253         END MACRO;                                                          !
254         CONSTANT CREATE_FUNCTION_CODE = 1;                                  !0
255         CONSTANT DELETE_FUNCTION_CODE = 2;                                  !0
256         CONSTANT GIVE_FUNCTION_CODE = 3;                                    !0
257         CONSTANT RESCIND_FUNCTION_CODE = 4;                                 !0
258         CONSTANT GETW_FUNCTION_CODE = 5;                                    !0
259         CONSTANT GPTP_FUNCTION_CODE = 6;                                    !0
260         CONSTANT RELPASP_FUNCTION_CODE = 7;                                 !0
261         CONSTANT ENABLP_FUNCTION_CODE = 8;                                  !0
262         CONSTANT DISABLE_FUNCTION_CODE = 9;                                 !0
263         CONSTANT P_FUNCTION_CODE = 10;                                      !0
264         CONSTANT V_FUNCTION_CODE = 11;                                      !0
265         CONSTANT T_FUNCTION_CODE = 12;                                      !0
266         CONSTANT IPCSEND_FUNCTION_CODE = 13;                                !0
267         CONSTANT IPCRCV_FUNCTION_CODE = 14;                                 !0
268         CONSTANT STARTP_FUNCTION_CODE = 15;                                 !0
269         CONSTANT STOPP_FUNCTION_CODE = 16;                                  !0
270         CONSTANT CHANGEO_FUNCTION_CODE = 17;                                !0
271         CONSTANT PROCID_FUNCTION_CODE = 18;                                 !0
272         CONSTANT INITH_FUNCTION_CODE = 19;                                  !0
273         CONSTANT READIR_FUNCTION_CODE = 20;                                 !0
274         /*    PARAMETERS ARE PASSED TO THE KERNEL BY PLACING THEM IN FIXED LOCATIONS IN    * !0
275          * THE STACK ACCESSED THROUGH SUPERVISOR SEGMENTATION REGISTER 0                   */!0
276         ABSOLUTE ("3FE") WORD FUNCTION_CODE_KPARM;                          !0
277         ABSOLUTE ("3FC") WORD SEG0_KPARM;                                   !0
278         ABSOLUTE ("3FA") WORD OFFSET_KPARM;                                 !0
279         ABSOLUTE ("3F8") WORD CLASS_KPARM;                                  !0
280         ABSOLUTE ("3F6") WORD CAT_KPARM;                                    !0
281         ABSOLUTE ("3F4") WORD SEG_TYPE_KPARM;                               !0
```

```
292     ABSOLUTE ("3F2") WORD SIZE_KPARM;                                       !0
293     ABSOLUTE ("3F0") WORD MODE_KPARM;                                       !0
294     ABSOLUTE ("3EE") WORD USER_KPARM;                                       !0
295     ABSOLUTE ("3EC") WORD PROJECT_KPARM;                                    !0
296     ABSOLUTE ("3EA") WORD SEG0_KPARM;                                       !0
297     ABSOLUTE ("3E8") WORD PROCESS0_KPARM;                                   !0
298     ABSOLUTE ("3F6") WORD MESSAGE_KPARM;                                    !0
299     ABSOLUTE ("3E4") WORD KERNEL_RC;                                        !0
300     ABSOLUTE ("3F2") WORD KERNEL_RC2;                                       !0

301     /*    RANGE OF FUNCTION CODE                                          */!0

302     CONSTANT FUNCTION_CODE_MIN = 1;                                         !0
303     CONSTANT FUNCTION_CODE_MAX = 20;                                        !0

304     /*    SEG0 IDENTIFIES AN ACTIVE SEGMENT                               */!0

305     /*    (SEG0, OFFSET) IDENTIFIES A SEGMENT WHOSE PARENT IS ACTIVE      */!0

306     /*    RANGE OF SEG0                                                   */!0

307     CONSTANT SEG0_MIN = 1;                                                  !0
308     CONSTANT SEG0_MAX = 31;                                                 !0

309     /*    SEG0 CONSTANTS                                                  */!0

310     CONSTANT ROOT_SEG0 = 1;                                                 !0

311     /*    RANGE OF OFFSETS                                                */!0

312     CONSTANT OFFSET_MIN = 1;                                                !0
313     CONSTANT OFFSET_MAX = 63;                                               !0

314     /*    OFFSET CONSTANTS (OFF ROOT)                                     */!0

315     CONSTANT PDD_OFFSET = 1;                                                !0
316     CONSTANT IOD_OFFSET = 2;                                                !0
317     CONSTANT CD_OFFSET = 3;                                                 !0
318     CONSTANT FMS_OFFSET = 4;                                                !0

319     /*    DEFINITION OF CLASS                                             */!0

320     CONSTANT CLASS_MIN = 1;                                                 !0
321     CONSTANT UNCLASSIFIED = 1;                                              !0
322     CONSTANT CONFIDENTIAL = 2;                                              !0
323     CONSTANT SECRET = 3;                                                    !0
324     CONSTANT TOP_SECRET = 4;                                                !0
325     CONSTANT CLASS_MAX = 4;                                                 !0

326     /*    CAT (CATEGORY SET) IS NOT YET DEFINED                           */!0

327     /*    DEFINITION OF SEG_TYPE                                          */!0

328     CONSTANT SEG_TYPE_DIRECTORY = "80";                                     !0
329     CONSTANT SEG_TYPE_DATA = "00";                                          !0
```

```
320         /*    DEFINITION OF SIZE - THE SIZE OF A SEGMENT IS FIXED AT CREATE TIME, SIZE IS  * |0
321          * SPECIFIED AS NUMBER OF 256 BYTE BLOCKS                                        */|0
                                                                                              |
322         CONSTANT SIZE1 = 1; /* 256 BYTES */                                               |0
323         CONSTANT SIZE2 = 4; /* 1K BYTES */                                               |0
324         CONSTANT SIZE3 = 16; /* 4K BYTES */                                              |0
                                                                                              |
325         /*    DIRECTORIES MUST BE SIZE2                                                  */|0
                                                                                              |
326         /*    NOTE - ONLY SIZE2 IS CURRENTLY IMPLEMENTED                                 */|0
                                                                                              |
327         /*    AN ACCESS CONTROL LIST (ACL) ELEMENT CONSISTS OF (MODE, USER, PROJECT)     */|0
                                                                                              |
328         /*    DEFINITION OF MODE                                                         */|0
                                                                                              |
329         CONSTANT NO_ACCESS = 0;                                                           |0
330         CONSTANT READ$EXECUTE_ACCESS = "4000";                                            |0
331         CONSTANT WRITE$READ$EXECUTE_ACCESS = "C000";                                      |0
                                                                                              |
332         /*    DEFINITION OF USER                                                         */|0
                                                                                              |
333         CONSTANT ALL_USERS = "3FFF";                                                      |0
                                                                                              |
334         /*    DEFINITION OF PROJECT                                                      */|0
                                                                                              |
335         CONSTANT ALL_PROJECTS = "7F";                                                     |0
336         CONSTANT SYSTEM_PROJECT = 1;                                                      |0
                                                                                              |
337         /*    RANGE OF REG#                                                              */|0
                                                                                              |
338         CONSTANT REG#_MIN = 0;                                                            |0
339         CONSTANT REG#_MAX = 15;                                                           |0
                                                                                              |
340         /*    PROCESS# CONSTANTS                                                         */|0
                                                                                              |
341         CONSTANT PROCESS#_MIN = 1;                                                        |0
342         CONSTANT PROCESS#_MAX = 7;                                                        |0
343         CONSTANT PROCESS#_2MAX = 14;                                                      |0
344         CONSTANT EXEC_PROCESS# = 1;                                                       |0
345         CONSTANT DECW_PROCESS# = 3;                                                       |0
346         CONSTANT TTY_PROCESS# = 2;                                                        |0
347         CONSTANT SCOPE1_PROCESS# = 4;                                                     |0
348         CONSTANT SCOPE2_PROCESS# = 5;                                                     |0
349         CONSTANT USER_PROCESS#_MIN = 2;                                                   |0
350         CONSTANT USER_PROCESS#_MAX = 5;                                                   |0
                                                                                              |
351         /*    IPCRCV PROCESS# IS REALLY (PROCESS#, DOMAIN)                               */|0
                                                                                              |
352         CONSTANT PROCESS#_MASK = "7F";                                                    |0
353         CONSTANT DOMAIN_MASK = "80";                                                      |0
354         CONSTANT KERNEL_DOMAIN = DOMAIN_MASK;                                             |0
                                                                                              |
355         /*    DEFINITION OF KERNEL RC (RC2 IS USED ONLY FOR IPCRCV)                      */|0
                                                                                              |
356         CONSTANT CK_FLAG = "FFFF";                                                        |0
```

```
                                                                        10
                                                                        10
 357        CONSTANT ERR_FLAG = "FFFF";
 358        CONSTANT SEVERR_FLAG = "FFFD"
                358 LINES WERE COMPILED.                                10
NO ERRORS WERE DETECTED.
DATA KERNEL
 359  _!_
```

```
360   CONTEXT KERNEL;                                                                    10
                                                                                         |
361         /*    DEFINITION OF THE KERNEL'S VIRTUAL ADDRESS SPACE                     */10
                                                                                         |
362         /*    SEGMENTATION REGISTER 0                                              */10
                                                                                         |
363         /*    0 - PE CONTAIN INTERRUPT VECTORS                                     */10
                                                                                         |
364         /*    DECLARATION OF MEMORY BLOCK TABLE                                    */10
                                                                                         |
365         /*    ONE ENTRY FOR EACH 256 BYTE BLOCK - THE MBT REQUIRES1. 5K BYTES      */10
                                                                                         |
366         CONSTANT MEM_SIZE = 511; /* 128K BYTES = 512 256 BYTE BLOCKS */              10
367         ABSOLUTE ("0100") ARRAY ((0 TO MEM_SIZE)) OF WORD MBT_FLAGS;                 10
368         ABSOLUTE ("0100") ARRAY ((0 TO MEM_SIZE)) OF WORD MBT_CHAIN;                 10
369         ABSOLUTE ("0100") ARRAY ((0 TO MEM_SIZE)) OF WORD MBT_ASTE;                  10
370         ABSOLUTE ("0500") ARRAY ((0 TO MEM_SIZE)) OF BYTE MBT_SIZE;                  10
                                                                                         |
371         /*    FLAGS, CHAIN, AND ASTE ALL SHARE A WORD - IT IS IMPORTANT THAT ASTE IS ONLY * 10
372          * MEANINGFUL WHEN FLAGS = ALLOCATED = 0                                   */10
                                                                                         |
373         CONSTANT MBT_FLAGS_MASK = "C000";                                            10
374         CONSTANT MBT_CHAIN_MASK = "3FFF";                                            10
375         CONSTANT END_BLOCK# = "3E0";                                                 10
                                                                                         |
376         /*    SETTINGS OF FLAGS FIELD                                              */10
                                                                                         |
377         CONSTANT ALLOCATED = 0;                                                      10
378         CONSTANT CONCATENATED = "4000";                                              10
379         CONSTANT FREE_MEM = "8000";                                                  10
380         CONSTANT RESERVED_MEM = "C000";                                              10
                                                                                         |
381         /*    END OF MBT                                                           */10
                                                                                         |
382         /*    DEFINITION OF THE HASH TABLE - REQUIRES0. 5 BYTES                    */10
                                                                                         |
383         CONSTANT ASTE0_MIN = 1;                                                      10
384         CONSTANT ASTE0_MAX = 255;                                                    10
385         ABSOLUTE ("0700") ARRAY ((0 TO ASTE0_MAX)) OF WORD HASH_TABLE;               10
                                                                                         |
386         /*    DECLARATION OF THE ACTIVE SEGMENT TABLE (AST)                        */10
                                                                                         |
387         /*    THERE IS AN ENTRY (ASTE) IN THE AST FOR EACH SEGMENT THAT HAS BEEN   * 10
388          * ACTIVATED - THE ENTRY CONTAINS ALL THE INFORMATION NECESSARY TO PERMIT A * 10
389          * PROCESS CONNECTED TO THE SEGMENT TO ENABLE AND DISABLE ACCESS TO IT.    */10
                                                                                         |
390         /*    THE AST REQUIRES 4K BYTES                                            */10
                                                                                         |
391         ABSOLUTE ("0900") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_TYPE;                 10
392         ABSOLUTE ("0900") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_STATUS;               10
393         ABSOLUTE ("0900") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_CHANGE;               10
394         ABSOLUTE ("0900") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_UNLOCK;               10
395         ABSOLUTE ("0900") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_CLASS;                10
396         ABSOLUTE ("0A00") ARRAY ((0 TO ASTE0_MAX)) OF BYTE AST_SIZE;                 10
397         ABSOLUTE ("0B00") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_DISK;                 10
```

```
398         ABSOLUTE ("0D00") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_CHAIN;                  !0
399         ABSOLUTE ("0F00") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_ADR;                    !0
400         ABSOLUTE ("1100") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_DES_COUNT;              !0
401         ABSOLUTE ("1100") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_SWAP_CHAIN;             !0
402         ABSOLUTE ("1300") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_CPL;                    !0
403         ABSOLUTE ("1500") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_WAL;                    !0
404         ABSOLUTE ("1500") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_AGP_CHAIN;              !0
405         ABSOLUTE ("1700") ARRAY ((0 TO ASTE0_MAX)) OF WORD AST_CAT;                    !0
                                                                                           !
406         /*    TYPE, STATUS, CHANGE, UNLOCK, AND CLASS SHARE A BYTE                     */!0
                                                                                           !
407         CONSTANT AST_TYPE_MASK = "80";                                                 !0
408         CONSTANT AST_STATUS_MASK = "40";                                               !0
409         CONSTANT AST_STATUS_NOTMASK = "BF";                                            !0
410         CONSTANT AST_CHANGE_MASK = "20";                                               !0
411         CONSTANT AST_UNLOCK_MASK = "10";                                               !0
412         CONSTANT AST_LOCK_MASK = "EF";                                                 !0
413         CONSTANT AST_CLASS_MASK = "0F";                                                !
                                                                                           !
414         /*    DEFINITION OF TYPE                                                       */!0
                                                                                           !
415         CONSTANT AST_TYPE_DIRECTORY = AST_TYPE_MASK;                                   !0
                                                                                           !
416         /*    DEFINITION OF CHANGE BIT                                                 */!0
                                                                                           !
417         CONSTANT AST_CHANGED = AST_CHANGE_MASK;                                        !0
418         CONSTANT AST_UNCHANGED_MASK = "DF";                                            !0
                                                                                           !
419         /*    DEFINITION OF STATUS                                                     */!0
                                                                                           !
420         CONSTANT AST_UNINITIALIZED = AST_STATUS_MASK;                                  !0
                                                                                           !
421         /*    DEFINITION OF UNLOCK                                                     */!0
                                                                                           !
422         CONSTANT AST_UNLOCK_FLAG = AST_UNLOCK_MASK;                                    !0
                                                                                           !
423         /*    BIT 0 OF CPL IS WIRED DOWN BIT                                           */!0
                                                                                           !
424         CONSTANT WIRED_DOWN_MASK = "8000";                                             !0
425         CONSTANT WIRED_DOWN_NOTMASK = "7FFF";                                          !0
426         CONSTANT WIRED_DOWN = WIRED_DOWN_MASK;                                         !0
                                                                                           !
427         /*    ASTE0 CONSTANTS                                                          */!0
                                                                                           !
428         CONSTANT ROOT_ASTE0 = 1;                                                       !0
                                                                                           !
429         /*    END OF AST                                                               */!0
                                                                                           !
430         /*    DEFINITION OF IPC ELEMENT POOL - REQUIRES0. 5K BYTES                     */!0
                                                                                           !
431         CONSTANT IPC_MAX = 127;                                                        !0
432         ABSOLUTE ("1900") ARRAY ((0 TO IPC_MAX)) OF BYTE IPC_LINK;                     !0
433         ABSOLUTE ("1980") ARRAY ((0 TO IPC_MAX)) OF BYTE IPC_PROCESS0;                 !0
434         ABSOLUTE ("1A00") ARRAY ((0 TO IPC_MAX)) OF WORD IPC_DATA;                     !
```

```
435       /*    RECEIVING PROCESSES ARE RESTRICTED TO 8 MESSAGE ELEMENTS                  */;0
436       CONSTANT IPC_QUOTA = 8;                                                           ;0
437       /*    END OF IPC POOL                                                           */;0
438       /*    DECLARATION OF BIT MAPS                                                   */;0
439       /*    BIT MAPS ARE USED BY THE DISK SPACE ALLOCATION MECHANISM.  THERE IS AN AREA  * ;0
440        * ON THE DISK FOR EACH SEGMENT SIZE, A BIT MAP FOR EACH DISK AREA, AND A BIT MAP  * ;0
441        * TABLE FOR EACH BIT MAP                                                        */;0
442       /*    A BIT MAP TABLE CONTAINS 4 WORDS - START AND END ADDRESS OF THE BIT MAP,    * ;0
443        * BASE ADDRESS OF THE DISK AREA, AND A SHIFT FACTOR                             */;0
444       /*    THE INITIAL IMPLEMENTATION ALLOCATES THE ENTIRE DISK TO 1K BYTE SEGMENTS -  * ;0
445        * 512 DISK PAGES REQUIRE A 64 BYTE BIT MAP                                      */;0
446       ABSOLUTE ("1B00") ARRAY ((0 TO 3)) OF WORD BMT_SIZE1;                             ;0
447       ABSOLUTE ("1B08") ARRAY ((0 TO 3)) OF WORD BMT_SIZE2;                             ;0
448       ABSOLUTE ("1B10") ARRAY ((0 TO 3)) OF WORD BMT_SIZE3;                             ;0
449       ABSOLUTE ("1B19") ARRAY ((0 TO 31)) OF WORD BIT_MAP2;                             ;0
450       ABSOLUTE ("1B58") ARRAY ((0 TO 1)) OF WORD BIT_MAP1;                              ;0
451       ABSOLUTE ("1B5A") ARRAY ((0 TO 1)) OF WORD BIT_MAP3;                              ;0
452       CONSTANT BMT_SIZE1_ADR = "1B00";                                                  ;0
453       CONSTANT BMT_SIZE2_ADR = "1B08";                                                  ;0
454       CONSTANT BMT_SIZE3_ADR = "1B10";                                                  ;0
455       /*    END OF BIT MAPS                                                           */;0
456       /*    DEFINITION OF SEMAPHORES                                                  */;0
457       /*    FIRST 255 SEMAPHORES ARE ASSOCIATED WITH ACTIVE SEGMENTS                  */;0
458       CONSTANT KERNEL_SMPR = 256;                                                       ;0
459       CONSTANT DISK_SMPR = 257;                                                         ;0
460       CONSTANT SMPR_MAX = DISK_SMPR;                                                    ;0
461       ABSOLUTE ("1B5C") ARRAY ((0 TO SMPR_MAX)) OF BYTE SMPR_COUNT;                     ;0
462       ABSOLUTE ("1C60") ARRAY ((0 TO SMPR_MAX)) OF BYTE SMPR_POINTER;                   ;0
463       /*    END OF SEMAPHORES                                                         */;0
464       /*    MISCELLANOUS                                                              */;0
465       ABSOLUTE ("1D64") WORD THE_CURRENT_PROCESS;                                       ;0
466       ABSOLUTE ("1D66") ARRAY ((0 TO FUNCTION_CODE_MAX)) OF WORD FUNCTION_ARRAY;        ;0
467       CONSTANT SEG0_FLAG = "8000";                                                      ;0
468       CONSTANT OFFSET_FLAG = "4000";                                                    ;0
469       CONSTANT CLASS_FLAG = "2000";                                                     ;0
470       CONSTANT REG_FLAG = "1000";                                                       ;0
471       CONSTANT PROCESS0_FLAG = "0800";                                                  ;0
472       CONSTANT MODE_FLAG = "0400";                                                      ;0
473       /*    THE PROCESS TABLE - ONE ENTRY IN THE PT FOR EACH PROCESS                  */;0
```

```
474       /*    SDR AND SAR ARRAYS MUST BE SEPERATED BY "20"                                          */
475       CONSTANT PT_KSDR1_ADR = "1E00";
476       CONSTANT PT_KSDR2_ADR = "1E40";
477       ABSOLUTE ("1E00") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSDR1;
478       ABSOLUTE ("1E20") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSAR1;
479       ABSOLUTE ("1E40") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSDR2;
480       ABSOLUTE ("1E60") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSAR2;
481       ABSOLUTE ("1E80") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSDR3;
482       ABSOLUTE ("1EA0") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KSAR3;
483       ABSOLUTE ("1EC0") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_R4;
484       ABSOLUTE ("1EE0") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_R5;
485       ABSOLUTE ("1F00") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_R6;
486       ABSOLUTE ("1F20") ARRAY ((0 TO PROCESS#_MAX)) OF BYTE PT_FLAGS;
487       ABSOLUTE ("1F20") ARRAY ((0 TO PROCESS#_MAX)) OF BYTE PT_LINK;
488       ABSOLUTE ("1F30") ARRAY ((0 TO PROCESS#_MAX)) OF BYTE PT_CUR_CLASS;
489       ABSOLUTE ("1F40") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_CUR_CAT;
490       ABSOLUTE ("1F60") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_KS_ASTE#;
491       ABSOLUTE ("1F80") ARRAY ((0 TO PROCESS#_MAX)) OF WORD PT_PS_ASTE#;
492       ABSOLUTE ("1FA0") ARRAY ((0 TO PROCESS#_MAX)) OF BYTE PT_IPC_QUEUE_HEAD;
493       ABSOLUTE ("1FB0") ARRAY ((0 TO PROCESS#_MAX)) OF BYTE PT_IPC_QUOTA;

494       /*    PT_FLAGS AND PT_LINK SHARE A BYTE - IT IS IMPORTANT THAT LINK IS ONLY
495        * MEANINGFUL WHEN FLAGS = BLOCKED                                                        */

496       CONSTANT PT_FLAGS_MASK = "C0";
497       CONSTANT PT_LINK_MASK = "3F";

498       /*    DEFINITION OF FLAGS FIELD                                                          */

499       CONSTANT BLOCKED = "00";
500       CONSTANT READY = "40";
501       CONSTANT INACTIVE = "80";

502       /*    IF PROCESS IS WAITING FOR A MESSAGE                                                */

503       CONSTANT IPC_WAIT = "FF";

504       /*    END OF PT                                                                          */

505       /*    SEGMENTATION REGISTER 1 - PROCESS SEGMENTS                                         */

506       /*    DECLARATION OF THE PROCESS SEGMENTS                                                */

507       /*    ONE PS FOR EACH PROCESS - PROCESS SEGMENTS ARE ALWAYS ACCESSED THROUGH KSR1        */

508       /*    SDR AND SAR ARRAYS MUST BE SEPERATED BY "20"                                       */

509       CONSTANT PS_KSR_ADR = "F4C2";
510       CONSTANT PS_SDR_ADR = "2000";
511       ABSOLUTE ("2000") ARRAY ((0 TO 15)) OF WORD PS_SDR;
512       ABSOLUTE ("2020") ARRAY ((0 TO 15)) OF WORD PS_SAR;
513       ABSOLUTE ("2040") WORD PS_CURRENT_PROCESS;
514       ABSOLUTE ("2042") WORD PS_PROCESS_MASK;
515       ABSOLUTE ("2044") WORD PS_PROCESS_NOTMASK;
```

```
516         ABSOLUTE ("2046") WORD PS_USER_ID;
517         ABSOLUTE ("2048") WORD PS_PROJECT_ID;
518         ABSOLUTE ("204A") BYTE PS_CUR_CLASS;
519         ABSOLUTE ("204C") WORD PS_CUR_CAT;
520         ABSOLUTE ("204E") BYTE PS_MEM_QUOTA;
521         ABSOLUTE ("2080") ARRAY ((0 TO SEG#_MAX)) OF WORD PS_SEG;
522         CONSTANT SEG_FLAG = "8000";
523         CONSTANT SEG_MASK = "7FF?";

524         /*    PROCESSES ARE RESTRICTED TO 9K WORDS OF MEMORY - EXEC IS VIRTUALLY
525          * UNRESTRICTED

526         CONSTANT MEM_QUOTA = "24";
527         CONSTANT EXEC_MEM_QUOTA = "7F";

528         /*    SEGMENTATIION REGISTER 2 - THE STACK

529         CONSTANT STACK_KSR_ADR = "F4C4";

530         /*    SEGMENTATION REGISTER 3 - DIRECTORIES

531         /*    DIRECTORIES ARE 1K BYTES IN SIZE AND ARE ALWAYS ACCESSED THROUGH
532          * SEGMENTATION REGISTER 3

533         CONSTANT DIR_KSR_ADR = "F4C6";
534         CONSTANT ACL_MAX = 127; /* NUMBER OF ACL ELEMENTS TO BE SHARED */
535         ABSOLUTE ("6000") ARRAY ((0 TO OFFSET_MAX)) OF BYTE DIR_TYPE;
536         ABSOLUTE ("6040") ARRAY ((0 TO OFFSET_MAX)) OF BYTE DIR_STATUS;
537         ABSOLUTE ("6080") ARRAY ((0 TO OFFSET_MAX)) OF BYTE DIR_CLASS;
538         ABSOLUTE ("60C0") ARRAY ((0 TO OFFSET_MAX)) OF BYTE DIR_ACL_HEAD;
539         ABSOLUTE ("6080") ARRAY ((0 TO OFFSET_MAX)) OF WORD DIR_CAT;
540         ABSOLUTE ("6100") ARRAY ((0 TO OFFSET_MAX)) OF WORD DIR_DISK;
541         ABSOLUTE ("6180") ARRAY ((0 TO OFFSET_MAX)) OF WORD DIR_SIZE;
542         ABSOLUTE ("6200") ARRAY ((0 TO ACL_MAX)) OF WORD ACL_MODE;
543         ABSOLUTE ("6200") ARRAY ((0 TO ACL_MAX)) OF WORD ACL_USER;
544         ABSOLUTE ("6300") ARRAY ((0 TO ACL_MAX)) OF BYTE ACL_PROJECT;
545         ABSOLUTE ("6380") ARRAY ((0 TO ACL_MAX)) OF BYTE ACL_CHAIN;

546         /*    TYPE, STATUS, AND CLASS SHARE A BYTE

547         CONSTANT DIR_TYPE_MASK = "80";
548         CONSTANT DIR_STATUS_MASK = "40";
549         CONSTANT DIR_STATUS_NOTMASK = "BF";
550         CONSTANT DIR_CLASS_MASK = "0F";
551         CONSTANT DIR_CLASS_NOTMASK = "F0";

552         /*    DEFINITION OF TYPE AND STATUS

553         CONSTANT DIR_TYPE_DIRECTORY = "80";
554         CONSTANT DIR_UNINITIALIZED = "40";

555         /*    MODE AND USER SHARE A WORD

556         CONSTANT ACL_MODE_MASK = "C000";
557         CONSTANT ACL_USER_MASK = "3FFF";
```

```
558       /*    KSR3 IS ALSO USED FOR INITIALIZING SEGMENTS                    */;
559       ABSOLUTE ("6000") ARRAY ((0 TO 1)) OF WORD ZERO_ARRAY;
560       /*    SEGMENTATION REGISTERS 4, 5 & 6 - PROGRAM CODE OF THE KERNEL   */;
561       /*    SEGMENTATION REGISTER 7 - CONTROL REGISTERS IN HIGH MEMORY      */;
562       /*    DEFINITION OF SUPERVISOR & USER SEGMENTATION REGISTERS         */;
563       /*    S SR0 IS AT 3F480, U SR0 IS A 3FFA0                            */;
564       /*    REG_CONSTANT = ((3FFA0-3F480)/2)-8 = "578"                     */;
565       CONSTANT REG_CONSTANT = "578";
566       CONSTANT P_REG#_MAX = "597";
567       CONSTANT CROSS_REG# = 7;
568       CONSTANT SDR_ADR = "F480";
569       ABSOLUTE ("F480") ARRAY ((0 TO P_REG#_MAX)) OF WORD SDR;
570       ABSOLUTE ("F480") WORD SDR0;
571       ABSOLUTE ("F4A0") ARRAY ((0 TO P_REG#_MAX)) OF WORD SAR;
572       ABSOLUTE ("F4A0") WORD SAR0;
573       /*    DEFINITION OF DESCRIPTOR REGISTER FIELDS                       */;
574       CONSTANT SDR_WRITE_ACCESS = "0006";
575       CONSTANT SDR_READ_ACCESS = "0002";
576       CONSTANT SDR_CHANGE_MASK = "0040";
577       CONSTANT SDR_CHANGED = SDR_CHANGE_MASK;
578       /*    KERNEL SEGMENTATION REGISTERS                                  */;
579       ABSOLUTE ("F4C0") WORD KSDR0;
580       ABSOLUTE ("F4C2") WORD KSDR1;
581       ABSOLUTE ("F4C4") WORD KSDR2;
582       ABSOLUTE ("F4C6") WORD KSDR3;
583       ABSOLUTE ("F4C8") WORD KSDR4;
584       ABSOLUTE ("F4CA") WORD KSDR5;
585       ABSOLUTE ("F4CC") WORD KSDR6;
586       ABSOLUTE ("F4CE") WORD KSDR7;
587       ABSOLUTE ("F4E0") WORD KSAR0;
588       ABSOLUTE ("F4E2") WORD KSAR1;
589       ABSOLUTE ("F4E4") WORD KSAR2;
590       ABSOLUTE ("F4E6") WORD KSAR3;
591       ABSOLUTE ("F4E8") WORD KSAR4;
592       ABSOLUTE ("F4EA") WORD KSAR5;
593       ABSOLUTE ("F4EC") WORD KSAR6;
594       ABSOLUTE ("F4EE") WORD KSAR7;
595       /*    MISC                                                           */;
596       ABSOLUTE ("FF7A") WORD STATUS_REG0;
597       ABSOLUTE ("FFFE") WORD PSW;
598       CONSTANT PREV_MODE_MASK = "3000";
599       CONSTANT PREV_MODE_SUPERV = "1000";
```

```
600         /*    DISK COMMANDS                                                                  */ 10
                                                                                                   10
601         CONSTANT DISK_WRITE = "0043";                                                          10
602         CONSTANT DISK_READ = "0045";                                                           10
                                                                                                   10
603         /*    THE KERNEL ACCESSES THE USER'S SRO STACK THROUGH KSR3 TO GET THE KERNPL       * 10
604          * PARAMETERS                                                                     */ 10
                                                                                                   10
605         ABSOLUTE ("63FE") WORD FUNCTION_CODE_APARM;                                            10
606         ABSOLUTE ("63FC") WORD SEG#_APARM;                                                     10
607         ABSOLUTE ("63FA") WORD OFFSET_APARM;                                                   10
608         ABSOLUTE ("63F8") WORD CLASS_APARM;                                                    10
609         ABSOLUTE ("63F6") WORD CAT_APARM;                                                      10
610         ABSOLUTE ("63F4") WORD SEG_TYPE_APARM;                                                 10
611         ABSOLUTE ("63F2") WORD SIZE_APARM;                                                     10
612         ABSOLUTE ("63F0") WORD MODE_APARM;                                                     10
613         ABSOLUTE ("63EE") WORD USER_APARM;                                                     10
614         ABSOLUTE ("63EC") WORD PROJECT_APARM;                                                  10
615         ABSOLUTE ("63EA") WORD PRG#_APARM;                                                     10
616         ABSOLUTE ("63E8") WORD PROCESS#_APARM;                                                 10
617         ABSOLUTE ("63E6") WORD MESSAGE_APARM;                                                  10
618         ABSOLUTE ("63E4") WORD KRC;                                                            10
619         ABSOLUTE ("63E2") WORD KRC2                                                            10
                261 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
DATA GATE
 620   _!_                                                                                         10
```

```
                    358 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
DATA KERNEL -L
                    261 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
DATA GATE L
 620   _I_
```

```
621   DATA GATE;                                                                    !0
                                                                                    !
622         /*    MUST CHECK INITIALIZATION OF INTERRUPT VECTORS IN STARTUP EVERY TIME A      * !0
623          * CHANGE IS MADE TO GATE (DATA OR PROGRAM)!                             */!0
                                                                                    !
624         MACRO KERNEL_ENTRY;                                                     !0
625               INLINE(MTPIR6);                                                   !0
626               INLINE(MOV, 0, 5, 4, 6);                                          !0
627               INLINE(MOV, 0, 4, 4, 6);                                          !0
628               INLINE(MOV, 0, 3, 4, 6);                                          !0
629               INLINE(MOV, 0, 2, 4, 6);                                          !0
630               INLINE(MOV, 0, 1, 4, 6);                                          !0
631               INLINE(MOV, 0, 0, 4, 6);                                          !0
632               INLINE(MOV, 0, 6, 0, 4);                                          !0
633               INLINE(MOV, 0, 6, 0, 5);                                          !0
634               INLINE(MOV, 2, 7, 6, 5, "800A", - 2);                             !0
635               INLINE(SUB, 2, 7, 0, 6, 6)                                        !0
636         END MACRO;                                                              !
                                                                                    !0
637         MACRO KERNEL_EXIT;                                                      !0
638               INLINE(ADD, 2, 7, 0, 6, 6);                                       !0
639               INLINE(MOV, 2, 6, 0, 0);                                          !0
640               INLINE(MOV, 2, 6, 0, 1);                                          !0
641               INLINE(MOV, 2, 6, 0, 2);                                          !0
642               INLINE(MOV, 2, 6, 0, 3);                                          !0
643               INLINE(MOV, 2, 6, 0, 4);                                          !0
644               INLINE(MOV, 2, 6, 0, 5);                                          !0
645               INLINE(MTPIR6);                                                   !0
646               INLINE(RTI)                                                       !0
647         END MACRO;                                                              !
                                                                                    !0
648         DECLARE                                                                 !0
649               WORD (PC);                                                        !
                                                                                    */!0
650         /*    EXTERNAL KERNEL FUNCTIONS                                         !
                                                                                    !0
651         DECLARE                                                                 !0
652               PROCEDURE (STOPP),                                                !0
653               PROCEDURE ACCEPTS (WORD) (DISABLE),                               !0
654               PROCEDURE ACCEPTS (WORD, WORD) (DCONNECT),                        !0
655               PROCEDURE ACCEPTS (WORD, WORD, WORD) (IPCSEND),                   !0
656               PROCEDURE RETURNS (WORD) (IPCRCV),                                !0
657               PROCEDURE ACCEPTS (WORD) RETURNS (WORD) (OUTSPP, OUTSRV),         !0
658               PROCEDURE ACCEPTS (WORD, WORD) RETURNS (WORD) (DELETE, GETW, GETR, ENABLE,  !0
659                     READIP),                                                    !0
660               PROCEDURE ACCEPTS (WORD, WORD, WORD) RETURNS (WORD) (INITH),      !0
661               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD) RETURNS (WORD) (RESCIND, CHANGEO),  !0
662               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD, WORD) RETURNS (WORD) (GIVE),  !0
663               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD, WORD, WORD) RETURNS (WORD) (CREATE,  !0
664                     STARTP);                                                    !
                                                                                    */!0
665         /*    INTERNAL KERNEL FUNCTIONS                                         !
                                                                                    !0
666         DECLARE                                                                 !0
667               PROCEDURE (SLEEP),
```

```
668             PROCEDURE ACCEPTS (WORD) (DFACT, P, V, SWAPIN, SWAPOUT, RUN), *                10
669             PROCEDURE ACCEPTS (WORD, WORD) (GOADD),                                        10
670             PROCEDURE ACCEPTS (WORD, WORD, WORD) (LSD),                                    10
671             PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD) (DISKIO),                           10
672             PROCEDURE RETURNS (WORD) (FCHECK),                                             10
673             PROCEDURE ACCEPTS (WORD) RETURNS (WORD) (WRITEDIR, HASH, PREHASH),             10
674             PROCEDURE ACCEPTS (WORD, WORD, WORD) RETURNS (WORD) (DSEARCH, CONNECT),        10
675             PROCEDURE ACCEPTS (WORD, WORD) RETURNS (WORD) (ACI);                           10
              56 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
DATA CREATE     F
676   -!-                                                                                      10
```

```
677   DATA_CREATE(ASTE#, OFFSET, CLASS, CAT, SEG_TYPE, SIZE) RETURNS (PC);          10
678         DECLARE                                                                 10
679               PROCEDURE ACCEPTS (WORD) RETURNS (WORD) (DALLOC);                 10
                  4 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
680   _!_                                                                           10
```

```
681   PROGRAM CREATE;
682         DECLARE
683               WORD (DISK_ADR);

684         /*    SECURITY CHECKS FIRST

685         /*    CREATE IS INTERPRETATIVE DIRECTORY WRITE

686         RC := WRITEDIR(ASTE#);

687         /*    CHECK COMPATIBILITY REQUIREMENTS

688         IF CLASS < (AST_CLASS(ASTE#) & AST_CLASS_MASK);

689           THEN: RC := ERR_FLAG;
FIXUP LOCATION 124 TO 6
690         END;

691         IF CAT ¬= (AST_CAT(ASTE#) | CAT);
```

```
  |24
  |24
  |24
  |
*/|24
  |
*/|24
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #16,5
CODE(10): MOV 7,(5)+
CODE(12): BR  2
CODE(14): =Y(CONTAINED PROCEDURE)
CODE(20): SUB #0,6
CODE(24): CLR -(6)
CODE(26): MOV 5,-(6)
CODE(30): MOV (5),-(6)
CODE(32): CLR -(6)
CODE(34): MOV -4(5),-(6)
CODE(40): MOV -2(4),2
CODE(44): JSR 7,@170(2)
CODE(50): MOV 5,6
CODE(52): CLR (6)+
CODE(54): MOV (6)+,5
CODE(56): MOV (6)+,4(5)
  |62
  |
*/|62
  |
  |126
CODE(62): MOV -4(5),-(6)
CODE(66): MOV #4400,2
CODE(72): ADD (6)+,2
CODE(74): MOVB0(2),1
CODE(100): MOV 1,-(6)
CODE(102): MOV #17,-(6)
CODE(106): COM (6)
CODE(110): BIC (6)+,(6)
CODE(112): MOV -10(5),3
CODE(116): CMP (6)+,3
CODE(120): BGT 2
CODE(122): JMP 0(7)
CODE(126): MOV #-2,4(5)
  |134

  |134
  |
  |174
CODE(134): MOV -4(5),-(6)
CODE(140): ASL (6)
CODE(142): MOV #13400,2
CODE(146): ADD (6)+,2
CODE(150): MOV 0(2),-(6)
CODE(154): BIS -12(5),(6)
CODE(160): MOV -12(5),3
CODE(164): CMP (6)+,3
CODE(166): BNE 2
```

```
 692          THEN: RC := ERR_FLAG;
FIXUP LOCATION 172 TO 6
 693          END;

 694          /*    IMPLEMENTATION CHECKS

 695          /*    CHECK SIZE PARAMETER - ONLY SIZE2 ALLOWED AT THIS TIME

 696          IF SIZE ¬= SIZE2;



 697          THEN: RC := ERR_FLAG;
FIXUP LOCATION 216 TO 6
 698          END;

 699          /*    CHECK OFFSET PARAMETER

 700          IF DIR_SIZE(OFFSET) ¬= 0;



 701          THEN: RC := ERR_FLAG;
FIXUP LOCATION 256 TO 6
 702          END;

 703          IF RC = ERR_FLAG;



 704          THEN;
 705  ....        RETURN;
FIXUP LOCATION 302 TO 4
 706          END;

 707          /*    ALLOCATE A DISK AREA FOR THE SEGMENT



 708          DISK_ADR := DALLOC(BMT_SIZE2_ADR);

 709          /*    CHECKING COMPLETE - PERFORM STATE CHANGE

 710          /*    FILL IN DIRECTORY ENTRY
```

```
CODE(170): JMP 0(7)
CODE(174): MOV #-2,#(5)
  |202

  |202
  |
*/|202
  |
*/|202
  |
  |220
CODE(202): MOV -16(5),-(6)
CODE(206): CMP #4,(6)+
CODE(212): BNE 2
CODE(214): JMP 0(7)
CODE(220): MOV #-2,#(5)
  |226

  |226
  |
*/|226
  |
  |260
CODE(226): MOV -6(5),-(6)
CODE(232): ASL (6)
CODE(234): MOV #60600,2
CODE(240): ADD (6)+,2
CODE(242): MOV 0(2),-(6)
CODE(246): CLR -(6)
CODE(250): CMP (6)+,(6)+
CODE(252): BNE 2
CODE(254): JMP 0(7)
CODE(260): MOV #-2,#(5)
  |266

  |266
  |
  |304
CODE(266): MOV #(5),-(6)
CODE(272): CMP #-2,(6)+
CODE(276): BEQ 2
CODE(300): JMP 0(7)
CODE(304): JMP #-20(5)
  |310
  |310

  |310
  |
*/|310
  |
CODE(310): CLR -(6)
CODE(312): MOV 5,-(6)
CODE(314): MOV 5,-(6)
CODE(316): CLR -(6)
CODE(320): MOV #15410,-(6)
CODE(324): MOV -2(5),2
CODE(330): JSR 7,#4(2)
CODE(334): MOV 5,6
CODE(336): CLR (6)+
CODE(340): MOV (6)+,5
CODE(342): MOV (6)+,-22(5)
  |346
  |
*/|346
  |
*/|346
```

```
711         DIR_TYPE(OFFSET) := (SEG_TYPE | DIR_UNINITIALIZED | CLASS);


712         DIR_CAT(OFFSET) := CAT;


713         DIR_DISK(OFFSET) := DISK_ADR;


714         DIR_SIZE(OFFSET) := SIZE;

FIXUP LOCATION 22 TO 2
              35 LINES WERE COMPILED, GENERATING 328 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA DELETE      5
 715   _!_
```

```
  !
CODE(346):  MOV -6(5),-(6)
CODE(352):  MOV -14(5),-(6)
CODE(356):  BIS #100,(6)
CODE(362):  BIS -10(5),(6)
CODE(366):  MOV #60000,2
CODE(372):  MOV (6)+,0
CODE(374):  ADD (6)+,2
CODE(376):  MOV#0,0(2)
  !402
CODE(402):  MOV -6(5),-(6)
CODE(406):  ASL (6)
CODE(410):  MOV -12(5),-(6)
CODE(414):  MOV #60200,2
CODE(420):  MOV (6)+,0
CODE(422):  ADD (6)+,2
CODE(426):  MOV 0,0(2)
  !430
CODE(430):  MOV -6(5),-(6)
CODE(434):  ASL (6)
CODE(436):  MOV -22(5),-(6)
CODE(442):  MOV #60400,2
CODE(446):  MOV (6)+,0
CODE(450):  ADD (6)+,2
CODE(452):  MOV 0,0(2)
  !456
CODE(456):  MOV -6(5),-(6)
CODE(462):  ASL (6)
CODE(464):  MOV -16(5),-(6)
CODE(470):  MOV #60600,2
CODE(474):  MOV (6)+,0
CODE(476):  ADD (6)+,2
CODE(500):  MOV 0,0(2)
  !504
CODE(504):  JMP #-20(5)
```

```
                                                                          10
 716   DATA DELETE(ASTE#, OFFSET) RETURNS (BC);                           10
 717        DECLARE                                                       10
 718             PROCEDURE ACCEPTS (WORD) (GETDIR),                       10
 719             PROCEDURE ACCEPTS (WORD, WORD) (DELETSEG);
                5 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
DATA GETDIR       *                                                       10
 720   _I_


                                                                          10
 721   DATA GETDIR(ASTE#);
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                  10
 722   _I_
```

```
723   PROGRAM GETDIR;
724         /*    LOAD SEGMENT DESCRIPTOR FOR DIRECTORY
725         IF AST_ADR(ASTE#) = 0;
726           THEN: SWAPIN(ASTE#);
FIXUP LOCATION 46 TO 32
727           END;
728         LSD(ASTE#, DIR_FSR_ADR, SDR_WRITE_ACCESS);
FIXUP LOCATION 14 TO 0
              7 LINES WERE COMPILED, GENERATING 104 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA DELETSEG     E
729   _!_
```

```
  |16
  |
*/|16
  |
  |50
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #7400,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-(6)
CODE(36): CLR -(6)
CODE(40): CMP (6)+,(6)+
CODE(42): BEQ 2
CODE(44): JMP 0(7)
CODE(50): MOV 5,-(6)
CODE(52): MOV (5),0
CODE(54): MOV (0),-(6)
CODE(56): CLR -(6)
CODE(60): MOV -4(5),-(6)
CODE(64): MOV -2(4),2
CODE(70): JSR 7,@134(2)
CODE(74): MOV 5,6
CODE(76): CLR (6)+
CODE(100): MOV (6)+,5
  |102
  |
  |102
  |
CODE(102): MOV 5,-(6)
CODE(104): MOV (5),0
CODE(106): MOV (0),-(6)
CODE(110): CLR -(6)
CODE(112): MOV -4(5),-(6)
CODE(116): MOV #-5472,-(6)
CODE(122): MOV #6,-(6)
CODE(126): MOV -2(4),2
CODE(132): JSR 7,@154(2)
CODE(136): MOV 5,6
CODE(140): CLR (6)+
CODE(142): MOV (6)+,5
  |144
CODE(144): JMP @-6(5)
```

```
730   DATA DELETSEG(ASTER, OFFSET);                                    10
731         DECLARE                                                    10
732               PROCEDURE ACCEPTS (WORD, WORD) (OFFSET);             10
                  4 LINES WERE COMPILED.                               10
NO ERRORS WERE DETECTED.
733   -I-
```

```
734   PROGRAM DELETSEG;
735         DECLARE
736               WORD (INDEX, OASIS0);

737         /*    REMOVE ANY ELEMENTS THAT MAY BE ON ACL CHAIN









734         INDEX := DIR_ACL_HEAD(OFFSET);

739         IF INDEX ¬= 0;
740           THEN;

741               CYCLE




FIXUP LOCATION 114 TO 4
742               .... EXIT WHEN ACL_CHAIN(INDEX) = 0;



743                     INDEX := ACL_CHAIN(INDEX);
FIXUP LOCATION 120 TO 24

744               END;
```

```
  124
  124
  124
  |
*/124
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): BR  2
CODE(14): <V(CONTAINED PO      RE)
CODE(20): SUB #0,6
CODE(24): MOV -6(5),-(6)
CODE(30): MOV #60100,2
CODE(34): ADD (6)+,2
CODE(36): MOVB0(2),1
CODE(42): MOV 1,-12(5)
  146
  |
  154
  154
  |
  164
CODE(46): MOV -12(5),-(6)
CODE(52): CLR -(6)
CODE(54): CMP (6)+,(6)+
CODE(56): BNE 2
CODE(60): JMP 0(7)
CODE(64): MOV -12(5),-(5)
CODE(70): MOV #61600,2
CODE(74): ADD (6)+,2
CODE(76): MOVB0(2),1
CODE(102): MOV 1,-(6)
CODE(104): CLR -(6)
CODE(106): CMP (6)+,(6)+
CODE(110): BEQ 2
CODE(112): JMP 0(7)
CODE(116): JMP #53(7)

  122
CODE(122): MOV -12(5),-(6)
CODE(126): MOV #61600,2
CODE(132): ADD (6)+,2
CODE(134): MOVB0(2),1
CODE(140): MOV 1,-12(5)
  144

CODE(144): BR  -31
  146
  |
CODE(146): MOV -12(5),-(6)
CODE(152): CLR -(6)
CODE(154): MOV #61600,2
CODE(160): ADD (6)+,2
CODE(162): MOVB0(2),1
CODE(166): MOV 1,-(6)
CODE(170): MOV #61600,2
```

```
745             ACL_CHAIN(INDEX) := ACL_CHAIN(0);

746             ACL_CHAIN(0) := DIR_ACL_HEAD(OFFSET);

747             DIR_ACL_HEAD(OFFSET) := 0;
FIXUP LOCATION 62 TO 200
748         END;
749         /*    NOW BUMP EVERYBODY OFF

750         SOADD(ASTE#, OFFSET);
751         /*    DEACTIVATE IF ASTE# OF OFFSET (OASTE#) IS AGED

752         OASTE# := HASH(DIR_DISK(OFFSET));
753         IF OASTE# >= 0;
754           THEN:
755             /*    SET CHANGE BIT TO UNCHANGED AND STATUS BIT TO INITIALIZED
```

```
CODE(174): MOV (6)+,0
CODE(176): ADD (6)+,2
CODE(200): MOVB0,0(2)
  |204
CODE(204): CLR -(6)
CODE(206): MOV -6(5),-(6)
CODE(212): MOV #60100,2
CODE(216): ADD (6)+,2
CODE(220): MOVB0(2),1
CODE(224): MOV 1,-(6)
CODE(226): MOV #61600,2
CODE(232): MOV (6)+,0
CODE(234): ADD (6)+,2
CODE(236): MOVB0,0(2)
  |242
CODE(242): MOV -6(5),-(6)
CODE(246): CLR -(6)
CODE(250): MOV #60100,2
CODE(254): MOV (6)+,0
CODE(256): ADD (6)+,2
CODE(260): MOVB0,0(2)
  |264

  |264
  |
*/|264
  |
CODE(264): MOV 5,-(6)
CODE(266): MOV (5),0
CODE(270): MOV (0),-(6)
CODE(272): CLR -(6)
CODE(274): MOV -4(5),-(6)
CODE(300): MOV -6(5),-(6)
CODE(304): MOV -2(4),2
CODE(310): JSR 7,@150(2)
CODE(314): MOV 5,6
CODE(316): CLR (6)+
CODE(320): MOV (6)+,5
  |322
  |
*/|322
  |
CODE(322): CLR -(6)
CODE(324): MOV 5,-(6)
CODE(326): MOV (5),0
CODE(330): MOV (0),-(6)
CODE(332): CLR -(6)
CODE(334): MOV -6(5),-(6)
CODE(340): ASL (6)
CODE(342): MOV #60400,2
CODE(346): ADD (6)+,2
CODE(350): MOV 0(2),-(6)
CODE(354): MOV -2(4),2
CODE(360): JSR 7,@174(2)
CODE(364): MOV 5,6
CODE(366): CLR (6)+
CODE(370): MOV (6)+,5
CODE(372): MOV (6)+,-14(5)
  |376
  |
  |414
  |414
  |
*/|414
  |
CODE(376): MOV -14(5),-(6)
```

```
 756            AST_CHANGE(OASTE#) := (AST_CHANGE(OASTE#) & AST_UNCHANGED_MASK);

 757            AST_STATUS(OASTE#) := (AST_STATUS(OASTE#) & AST_STATUS_NOTMASK);

 758            DEACT(OASTE#);
FIXUP LOCATION 412 TO 452
 759        END;

 760        /*    FREE UP RESOURCES - DISK SPACE AND DIRECTORY ENTRY

 761        DFREE(DIR_DISK(OFFSET), BMT_SIZE2_ADR);
```

```
CODE(402): CLR -(6)
CODE(404): CMP (6)+,(6)+
CODE(406): BNE 2
CODE(410): JMP 0(7)
CODE(414): MOV -14(5),-(6)
CODE(420): MOV -14(5),-(6)
CODE(424): MOV #4400,2
CODE(430): ADD (6)+,2
CODE(432): MOVB0(2),1
CODE(436): MOV 1,-(6)
CODE(440): MOV #337,-(6)
CODE(444): COM (6)
CODE(446): BIC (6)+,(6)
CODE(450): MOV #4400,2
CODE(454): MOV (6)+,0
CODE(456): ADD (6)+,2
CODE(460): MOVB0,0(2)
  |464
CODE(464): MOV -14(5),-(6)
CODE(470): MOV -14(5),-(6)
CODE(474): MOV #4400,2
CODE(500): ADD (6)+,2
CODE(502): MOVB0(2),1
CODE(506): MOV 1,-(6)
CODE(510): MOV #277,-(6)
CODE(514): COM (6)
CODE(516): BIC (6)+,(6)
CODE(520): MOV #4400,2
CODE(524): MOV (6)+,0
CODE(526): ADD (6)+,2
CODE(530): MOVB0,0(2)
  |534
CODE(534): MOV 5,-(6)
CODE(536): MOV (5),0
CODE(540): MOV (0),-(6)
CODE(542): CLR -(6)
CODE(544): MOV -14(5),-(6)
CODE(550): MOV -2(4),2
CODE(554): JSR 7,#120(2)
CODE(560): MOV 5,6
CODE(562): CLR (6)+
CODE(564): MOV (6)+,5
  |566

  |566
  |
*/|566
  |
CODE(566): MOV 5,-(6)
CODE(570): MOV 5,-(6)
CODE(572): CLR -(6)
CODE(574): MOV -6(5),-(6)
CODE(600): ASL (6)
CODE(602): MOV #60400,2
CODE(606): ADD (6)+,2
CODE(610): MOV 0(2),-(6)
CODE(614): MOV #15410,-(6)
CODE(620): MOV -2(5),2
CODE(624): JSR 7,#4(2)
CODE(630): MOV 5,6
CODE(632): CLR (6)+
CODE(634): MOV (6)+,5
  |636
CODE(636): MOV -6(5),-(6)
CODE(642): ASL (6)
CODE(644): CLR -(6)
```

```
762         DIR_DISK(OFFSET) := 0;

763         DIR_SIZE(OFFSET) := 0;

764         /*    SET CHANGE BIT IN PARENT

765         ASC_CHANGE(ASIZE#) := (ASC_CHANGE(ASIZE#) | ASC_CHANGED);

PICKUP LOCATION 22 TO 4
                33 LINES WERE COMPILED, GENERATING 494 BYTES OF CODE.
NO ERRORS WERE DETECTED.
PROGRAM DELETE   5
766   _!_
```

```
CODE(646): MOV #60400,2
CODE(652): MOV (6)+,0
CODE(654): ADD (6)+,2
CODE(656): MOV 0,0(2)
  |662
CODE(662): MOV -6(5),-(6)
CODE(666): ASL (6)
CODE(670): CLR -(6)
CODE(672): MOV #60600,2
CODE(676): MOV (6)+,0
CODE(700): ADD (6)+,2
CODE(702): MOV 0,0(2)
  |706
  |
*/|706
  |
CODE(706): MOV -8(5),-(6)
CODE(712): MOV -8(5),-(6)
CODE(716): MOV #4400,2
CODE(722): ADD (6)+,2
CODE(724): MOVB0(2),1
CODE(730): MOV 1,-(6)
CODE(732): BIS #40,(6)
CODE(736): MOV #4400,2
CODE(742): MOV (6)+,0
CODE(744): ADD (6)+,2
CODE(746): MOVB0,0(2)
  |752
CODE(752): JMP 8-10(5)
```

```
767   PROGRAM DELETE;
768         DECLARE
769               WORD (SPASTE#, SASTE#, SOFFSET, IOFFSET);

772         /*    SECURITY CHECKS FIRST

771         /*    DELETE IS AN INTERPRETATIVE DIRECTORY WRITE

772         IF #PRIVDIR(ASTE#) = ERR_FLAG;
```

```
773           THEN;
774   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 72 TO 12
775         END;

776         /*    IMPLEMENTATION CHECKS

777         IF DIR_SIZE(OFFSET) = 0;
```

```
778           THEN;
779   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 136 TO 12
780         END;

781         /*    ELIMINATE CASE OF DATA SEGMENT
```

```
                |30
                |30
                |30
                |
*/              |30
                |
*/              |30
                |
                |74
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): BR  4
CODE(14): #V(CONTAINED PROCEDURE)
CODE(20): #V(CONTAINED PROCEDURE)
CODE(24): SUB #0,6
CODE(30): CLR -(6)
CODE(32): MOV 5,-(6)
CODE(34): MOV (5),-(6)
CODE(36): CLR -(6)
CODE(40): MOV -4(5),-(6)
CODE(44): MOV -2(4),2
CODE(50): JSR 7,@170(2)
CODE(54): MOV 5,6
CODE(56): CLR (6)+
CODE(60): MOV (6)+,5
CODE(62): CMP #-2,(6)+
CODE(66): BEQ 2
CODE(70): JMP 0(7)
CODE(74): MOV #-2,4(5)
CODE(102): JMP @-10(5)
                |106
                |106

                |106
                |
*/              |106
                |
                |140
CODE(106): MOV -6(5),-(6)
CODE(112): ASL (6)
CODE(114): MOV #60600,2
CODE(120): ADD (6)+,2
CODE(122): MOV 0(2),-(6)
CODE(126): CLR -(6)
CODE(130): CMP (6)+,(6)+
CODE(132): BEQ 2
CODE(134): JMP 0(7)
CODE(140): MOV #-2,4(5)
CODE(146): JMP @-10(5)
                |152
                |152

                |152
                |
*/              |152
                |
```

```
782         IF (DIR_TYPE(OFFSET) & DIR_TYPE_MASK) >= DIR_TYPE_DIRECTORY;
783           THEN: DELETSEG(ASID#, OFFSET);
FIXUP LOCATION 212 TO 40
784           ELSE: <OUTER_CYCLE>
785             CYCLE
786                   /*    START AT THE TOP OF THE CHAIN AND WORK DOWN
787                   SPASID# := ASID#;
788                   SOFFSET := OFFSET;
789                   <INNER_CYCLE>
790                   CYCLE
791                        /*    ASSURE DIRECTORY TO BE DELETED IS ACTIVE
792                        SASID# := HASH(DIR_DISK(SOFFSET));
793                        IF SASID# = 0;
```

```
  |214
CODE(152): MOV -6(5),-(6)
CODE(156): MOV #60000,2
CODE(162): ADD (6)+,2
CODE(164): MOVB0(2),1
CODE(170): MOV 1,-(6)
CODE(172): MOV #200,-(6)
CODE(176): COM (6)
CODE(200): BIC (6)+,(6)
CODE(202): CMP #200,(6)+
CODE(206): BNE 2
CODE(210): JMP 0(7)
CODE(214): MOV 5,-(6)
CODE(216): MOV 5,-(6)
CODE(220): CLR -(6)
CODE(222): MOV -4(5),-(6)
CODE(226): MOV -6(5),-(6)
CODE(232): MOV -2(5),2
CODE(236): JSR 7,#10(2)
CODE(242): MOV 5,6
CODE(244): CLR (6)+
CODE(246): MOV (6)+,5
  |250
  |254
  |254
  |
*/|254
  |
CODE(250): JMP 0(7)
CODE(254): MOV -4(5),-12(5)
  |262
CODE(262): MOV -6(5),-16(5)
  |270
  |270
  |270
  |
*/|270
  |
CODE(270): CLR -(6)
CODE(272): MOV 5,-(6)
CODE(274): MOV (5),-(6)
CODE(276): CLR -(6)
CODE(300): MOV -16(5),-(6)
CODE(304): ASL (6)
CODE(306): MOV #60400,2
CODE(312): ADD (6)+,2
CODE(314): MOV 0(2),-(6)
CODE(320): MOV -2(4),2
CODE(324): JSR 7,#174(2)
CODE(330): MOV 5,6
CODE(332): CLR (6)+
CODE(334): MOV (6)+,5
CODE(336): MOV (6)+,-14(5)
  |342
  |
  |360
CODE(342): MOV -14(5),-(6)
CODE(346): CLR -(6)
CODE(350): CMP (6)+,(6)+
CODE(352): BEQ 2
CODE(354): JMP 0(7)
CODE(360): CLR -(6)
CODE(362): MOV 5,-(6)
CODE(364): MOV (5),-(6)
CODE(366): CLR -(6)
```

```
                                                                                CODE(370): MOV -14(5),-(6)
                                                                                CODE(374): MOV -16(5),-(6)
                                                                                CODE(400): MOV -2(4),2
                                                                                CODE(404): JSR 7,@214(2)
                                                                                CODE(410): MOV 5,6
                                                                                CODE(412): CLR (6)+
                                                                                CODE(414): MOV (6)+,5
                                                                                CODE(416): MOV (6)+,-14(5)
 794                      THEN: SASTE# := ACT(SPASTE#, SOFFSET);                   |422
FIXUP LOCATION 356 TO 42
 795                  END;                                                       |422
                                                                                 |
 796                  /*    LOAD SEGMENT DESCRIPTORS                           */|422
                                                                                 |
                                                                                CODE(422): MOV 5,-(6)
                                                                                CODE(424): MOV 5,-(6)
                                                                                CODE(426): CLR -(6)
                                                                                CODE(430): MOV -14(5),-(6)
                                                                                CODE(434): MOV -2(5),2
                                                                                CODE(440): JSR 7,@4(2)
                                                                                CODE(444): MOV 5,6
                                                                                CODE(446): CLR (6)+
                                                                                CODE(450): MOV (6)+,5
 797                  GETDIR(SASTE#);                                            |452
                                                                                 |
 798                  /*    SEARCH THE DIRECTORY FOR ENTRIES OF NON-ZERO SIZE   */|452
                                                                                 |
                                                                                CODE(452): MOV #77,-(6)
                                                                                CODE(456): MOV #1,-(6)
                                                                                CODE(462): JMP 453(7)
                                                                                CODE(466): INC (6)
FIXUP LOCATION 464 TO 2
 799                  DO TOFFSET := OFFSET_MIN TO OFFSET_MAX;                    |506
                                                                                 |
 800                      IF DIR_SIZE(TOFFSET) ~= 0;                             |540
 801                        THEN:                                                |540
 802                          IF (DIR_TYPE(TOFFSET) & DIR_TYPE_MASK) = DIR_TYPE_DIRECTORY;|602
 803                            THEN: /* GAIN ACCESS TO A DIRECTORY ENTRY--PREPARE TO   |602
 804                                     SEARCH */                               |602
                                                                                CODE(470): CMP (6),2(6)
                                                                                CODE(474): BLE 2
                                                                                CODE(476): JMP 453(7)
                                                                                CODE(502): MOV (6),-20(5)
                                                                                CODE(506): MOV -20(5),-(6)
                                                                                CODE(512): ASL (6)
                                                                                CODE(514): MOV #60600,2
                                                                                CODE(520): ADD (6)+,2
                                                                                CODE(522): MOV 0(2),-(6)
                                                                                CODE(526): CLR -(6)
                                                                                CODE(530): CMP (6)+,(6)+
                                                                                CODE(532): BNE 2
                                                                                CODE(534): JMP 0(7)
                                                                                CODE(540): MOV -20(5),-(6)
                                                                                CODE(544): MOV #60000,2
                                                                                CODE(550): ADD (6)+,2
                                                                                CODE(552): MOVB0(2),1
                                                                                CODE(556): MOV 1,-(6)
                                                                                CODE(560): MOV #200,-(6)
                                                                                CODE(564): COM (6)
                                                                                CODE(566): BIC (6)+,(6)
                                                                                CODE(570): CMP #200,(6)+
                                                                                CODE(574): BEQ 2
                                                                                CODE(576): JMP 0(7)
                                                                                CODE(602): MOV -14(5),-12(5)
```

```
 805                                  SPASTE# := SASTE#;
 806                                  SOFFSET := TOFFSET;
 807                          ....    REPEAT <INNER_CYCLE>;
FIXUP LOCATION 600 TO 26
 808                                ELSE: /* DELETE A DATA SEGMENT ENTRY */
 809                                  DELETSEG(SASTE#, TOFFSET);
FIXUP LOCATION 626 TO 34
 810                                END;
FIXUP LOCATION 536 TO 124
 811                              END;
FIXUP LOCATION 500 TO 164
 812                        END;
 813                        /*    THIS DIRECTORY IS EMPTY--DELETE IT
 814                        GETDIR(SPASTE#);
 815                        DELETSEG(SPASTE#, SOFFSET);
 816                        /*    FINISHED?
```

```
  |610
CODE(610): MOV -20(5),-16(5)
  |616
CODE(616): CMP (6)+,(6)+
CODE(620): JMP @53(7)
  |624
  |630
CODE(624): JMP 0(7)
CODE(630): MOV 5,-(6)
CODE(632): MOV 5,-(6)
CODE(634): CLR -(6)
CODE(636): MOV -14(5),-(6)
CODE(642): MOV -20(5),-(6)
CODE(646): MOV -2(5),2
CODE(652): JSR 7,@10(2)
CODE(656): MOV 5,6
CODE(660): CLR (6)+
CODE(662): MOV (6)+,5
  |664
  |664
  |
  |664
  |
CODE(664): BR  -100
CODE(666): CMP (6)+,(6)+
  |670
  |
*/|670
  |
CODE(670): MOV 5,-(6)
CODE(672): MOV 5,-(6)
CODE(674): CLR -(6)
CODE(676): MOV -12(5),-(6)
CODE(702): MOV -2(5),2
CODE(706): JSR 7,@4(2)
CODE(712): MOV 5,6
CODE(714): CLR (6)+
CODE(716): MOV (6)+,5
  |720
CODE(720): MOV 5,-(6)
CODE(722): MOV 5,-(6)
CODE(724): CLR -(6)
CODE(726): MOV -12(5),-(6)
CODE(732): MOV -16(5),-(6)
CODE(736): MOV -2(5),2
CODE(742): JSR 7,@10(2)
CODE(746): MOV 5,6
CODE(750): CLR (6)+
CODE(752): MOV (6)+,5
  |754
  |
*/|754
  |
CODE(754): MOV -12(5),-(6)
```

```
FIXUP LOCATION 770 TO 0
 817                    .... EXIT <OUTER_CYCLE> WHEN SPASIE* = ASTE*;

 818                         /*    LOAD SEGMENT DESCRIPTORS OF ORIGINAL DIRECTORY

 819                         GETDIR(ASTE*);

 820                    .... REPEAT <OUTER_CYCLE>;
FIXUP LOCATION 622 TO -334

 821                    END <INNER_CYCLE>;

FIXUP LOCATION 774 TO 44
FIXUP LOCATION 1030 TO -556

 822               END <OUTER_CYCLE>;

FIXUP LOCATION 252 TO 566
 823          END;


 824          RC := OK_FLAG;

FIXUP LOCATION 26 TO 10
                 59 LINES WERE COMPILED, GENERATING 556 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA GIVE          *
 825   _!_
```

```
CODE(760): CMP -4(5),(6)+
CODE(764): BEQ 2
CODE(766): JMP 0(7)
CODE(772): JMP 453(7)

  |776
  |
*/|776
  |
CODE(776): MOV 5,-(6)
CODE(1000): MOV 5,-(6)
CODE(1002): CLR -(6)
CODE(1004): MOV -4(5),-(6)
CODE(1010): MOV -2(5),2
CODE(1014): JSR 7,@4(2)
CODE(1020): MOV 5,6
CODE(1022): CLR (6)+
CODE(1024): MOV (6)+,5
  |1026
CODE(1026): JMP 453(7)
  |1032

CODE(1032): JMP -546(7)
  |1036
  |


CODE(1036): JMP -566(7)
  |1042
  |

  |1042
  |
CODE(1042): MOV @-1,4(5)
  |1050
CODE(1050): JMP @-10(5)



  |0
```

```
826   DATA GIVE(ASTER, OFFSET, MODE, USER, PROJECT) RETURNS (BIT);                    10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
827   _I_                                                                             10
```

```
828   PROGRAM GIVP;                                                                  |16
829         DECLARE                                                                  |16
830               WORD (ACLE#, INDEX, POSITION);                                     |16
                                                                                     |
831         /*    SECURITY CHECKS FIRST                                            */|16
                                                                                     |
832         /*    GIVE WRITES INTO DIRECTORY (INTERPRETATIVE WRITE)                */|16
                                                                                     |
833         IF WRITEDIR(ASTE#) = ERR_FLAG;                                           |62
                                                                                     CODE(0):      2,-(0)
                                                                                     CODE(2): MOV 6,5
                                                                                     CODE(4): ADD #14,5
                                                                                     CODE(10): MOV 7,(5)+
                                                                                     CODE(12): SUB #0,6
                                                                                     CODE(16): CLR -(6)
                                                                                     CODE(20): MOV 5,-(6)
                                                                                     CODE(22): MOV (5),-(6)
                                                                                     CODE(24): CLR -(6)
                                                                                     CODE(26): MOV -4(5),-(6)
                                                                                     CODE(32): MOV -2(4),2
                                                                                     CODE(36): JSR 7,@170(2)
                                                                                     CODE(42): MOV 5,6
                                                                                     CODE(44): CLR (6)+
                                                                                     CODE(46): MOV (6)+,5
                                                                                     CODE(50): CMP #-2,(6)+
                                                                                     CODE(54): BEQ 2
                                                                                     CODE(56): JMP 0(7)
                                                                                     CODE(62): MOV #-2,4(5)
                                                                                     CODE(70): JMP @-16(5)
834           THEN:                                                                  |74
835   ....      RETURN WITH SPE_FLAG;                                                |74
*IXUP LOCATION 60 TO 12                                                              |
836         END;                                                                     |74
                                                                                     |
837         /*    IMPLEMENTATION CHECKS                                            */|74
                                                                                     |
838         /*    SEGMENT MUST EXIST                                               */|74
                                                                                     |
839         IF DIR_DISK(OFFSET) = 0;                                                 |126
                                                                                     CODE(74): MOV -6(5),-(6)
                                                                                     CODE(100): ASL (6)
                                                                                     CODE(102): MOV #60400,2
                                                                                     CODE(106): ADD (6)+,2
                                                                                     CODE(110): MOV 0(2),-(6)
                                                                                     CODE(114): CLR -(6)
                                                                                     CODE(116): CMP (6)+,(6)+
                                                                                     CODE(120): BEQ 2
                                                                                     CODE(122): JMP 0(7)
                                                                                     CODE(126): MOV #-2,4(5)
                                                                                     CODE(134): JMP @-16(5)
840           THEN:                                                                  |140
841   ....      RETURN WITH EPP_FLAG;                                                |140
*IXMP LOCATION 124 TO 12                                                             |
842         END;                                                                     |140
                                                                                     |
843         /*    SEARCH FOR DUPLICATE ACL ELEMENT                                 */|140
                                                                                     |
                                                                                     CODE(140): MOV -6(5),-(6)
```

```
                                                
```

```
855         ACL#0 := ACL_CHAIN(0);

856         IF ACL#0 = 0;
                                                                    CODE(364): MOV #61600,2
                                                                    CODE(370): ADD (6)+,2
                                                                    CODE(372): MOVB0(2),1
                                                                    CODE(376): MOV 1,-20(5)
                                                                      |402
                                                                      |
                                                                      |420
                                                                    CODE(402): MOV -20(5),-(6)
                                                                    CODE(406): CLR -(6)
                                                                    CODE(410): CMP (6)+,(6)+
                                                                    CODE(412): BEQ 2
                                                                    CODE(414): JMP 0(7)
                                                                    CODE(420): MOV #-2,4(5)
                                                                    CODE(426): JMP #-16(5)
                                                                      |432
                                                                      |432

857           THEN:
858   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 416 TO 12
859         END;
                                                                      |432
                                                                      |
                                                                    */|432
                                                                      |
860         /*    CHECKING COMPLETE - PERFORM STATE CHANGE          CODE(432): CLR -(6)
                                                                    CODE(434): MOV -20(5),-(6)
                                                                    CODE(440): MOV #61600,2
                                                                    CODE(444): ADD (6)+,2
                                                                    CODE(446): MOVB0(2),1
                                                                    CODE(452): MOV 1,-(6)
                                                                    CODE(456): MOV #61600,2
                                                                    CODE(460): MOV (6)+,0
                                                                    CODE(462): ADD (6)+,2
                                                                    CODE(464): MOVB0,0(2)
                                                                      |470
                                                                      |
                                                                    */|470
                                                                      |
861         ACL_CHAIN(0) := ACL_CHAIN(ACL#0);                       CODE(470): CLR -(6)
                                                                    CODE(472): MOV (6)+,-24(5)
                                                                      |476
                                                                      |
862         /*    DETERMINE CORRECT POSITION FOR NEW ACL ELEMENT

863         POSITION := 0;
```

```
864         IF DIR_ACL_HEAD(OFFSET) ~= 0;
865           THEN:
866             IF ((USER = ALL_USERS) & (PROJECT = ALL_PROJECTS));
```

```
867               THEN: POSITION := DIR_ACL_HEAD(OFFSET);

869                  CYCLE
```

```
FIXUP LOCATION 660 TO 4
869                  .... EXIT WHEN ACL_CHAIN(POSITION) = 0;
```

```
870                        POSITION := ACL_CHAIN(POSITION);
FIXUP LOCATION 664 TO 24
871                  END;
```

```
  1530
  1530
  1606
CODE(476): MOV -6(5),-(6)
CODE(502): MOV #60100,2
CODE(506): ADD (6)+,2
CODE(510): MOVB0(2),1
CODE(514): MOV 1,-(6)
CODE(516): CLR -(6)
CODE(520): CMP (6)+,(6)+
CODE(522): BNE 2
CODE(524): JMP 0(7)
CODE(530): MOV -12(5),-(6)
CODE(534): CMP #37777,(6)+
CODE(540): BEQ 2
CODE(542): CLR -(6)
CODE(544): BR  2
CODE(546): MOV #1,-(6)
CODE(552): MOV -14(5),-(6)
CODE(556): MOV #177,-(6)
CODE(562): CMP (6)+,(6)+
CODE(564): BEQ 2
CODE(566): CLR -(6)
CODE(570): BR  2
CODE(572): MOV #1,-(6)
CODE(576): BIT (6)+,(6)+
CODE(600): BNE 2
CODE(602): JMP 0(7)
CODE(606): MOV -6(5),-(6)
CODE(612): MOV #60100,2
CODE(616): ADD (6)+,2
CODE(620): MOVB0(2),1
CODE(624): MOV 1,-24(5)
  1630
  |
  1630
CODE(630): MOV -24(5),-(6)
CODE(634): MOV #61600,2
CODE(640): ADD (6)+,2
CODE(642): MOVB0(2),1
CODE(646): MOV 1,-(6)
CODE(650): CLR -(6)
CODE(652): CMP (6)+,(6)+
CODE(654): BEQ 2
CODE(656): JMP 0(7)
CODE(662): JMP 453(7)

  1666
CODE(666): MOV -24(5),-(6)
CODE(672): MOV #61600,2
CODE(676): ADD (6)+,2
CODE(700): MOVB0(2),1
CODE(704): MOV 1,-24(5)
  1710

CODE(710): BR  -31
  1712
  |
```

```
FIXUP LOCATION 604 TO 110
 872                 ELSE:
 873                    IF ((USER = ALL_USERS) | (PROJECT = ALL_PROJECTS));
 874                       THEN:
 875                          IF (ACL_USER(DIR_ACL_HEAD(OFFSET)) & ACL_USER_MASK) -= ALL_USERS;
```

```
                                                
```

```
                                                                                                                    CODE(1170): MOV #37777,-(6)
                                                                                                                    CODE(1174): COM (6)
                                                                                                                    CODE(1176): BIC (6)+,(6)
                                                                                                                    CODE(1200): MOV #37777,-(6)
                                                                                                                    CODE(1204): CMP (6)+,(6)+
                                                                                                                    CODE(1206): BEQ 2
                                                                                                                    CODE(1210): CLR -(6)
                                                                                                                    CODE(1212): BR  2
                                                                                                                    CODE(1214): MOV #1,-(6)
                                                                                                                    CODE(1220): BIS (6)+,(6)
                                                                                                                    CODE(1222): ROR (6)+
                                                                                                                    CODE(1224): BCS 2
                                                                                                                    CODE(1226): JMP 0(7)
                                                                                                                    CODE(1232): JMP 453(7)

FIXUP LOCATION 1230 TO 4
 878.                          .... EXIT WHEN (ACL_CHAIN(POSITION) = 0) | ((ACL_USER(ACL_CHAIN( |1236
 879                                    POSITION)) & ACL_USER_MASK) = ALL_USERS);                  |1236
                                                                                                                    CODE(1236): MOV -24(5),-(6)
                                                                                                                    CODE(1242): MOV #61600,2
                                                                                                                    CODE(1246): ADD (6)+,2
                                                                                                                    CODE(1250): MOVB0(2),1
                                                                                                                    CODE(1254): MOV 1,-24(5)
 880                                        POSITION := ACL_CHAIN(POSITION);                          |1260
FIXUP LOCATION 1234 TO 24
                                                                                                                    CODE(1260): BR  -72
 881                                  END;                                                             |1262
                                                                                                       |
FIXUP LOCATION 1052 TO 206                                                                             |1262
 882                             END;                                                                  |
                                                                                                       |1262
FIXUP LOCATION 774 TO 264                                                                              |
 883                           END;                                                                    |1262
                                                                                                       |
FIXUP LOCATION 714 TO 344                                                                              |1262
 884                     END;                                                                          |
                                                                                                       |1262
FIXUP LOCATION 526 TO 532                                                                              |
 885         END;                                                                                      |1262
                                                                                                       |
 886         /*    FILL IN NEW ACL ELEMENT AND ADD TO CHAIN                                            */|1262
                                                                                                       |
                                                                                                                    CODE(1262): MOV -20(5),-(6)
                                                                                                                    CODE(1266): ASL (6)
                                                                                                                    CODE(1270): MOV -12(5),-(6)
                                                                                                                    CODE(1274): BIS -10(5),(6)
                                                                                                                    CODE(1300): MOV #61000,2
                                                                                                                    CODE(1304): MOV (6)+,0
                                                                                                                    CODE(1306): ADD (6)+,2
                                                                                                                    CODE(1310): MOV 0,0(2)
                                                                                                                      |1314
 887         ACL_USER(ACLE#) := (USER | MODE);                                                                   CODE(1314): MOV -20(5),-(6)
                                                                                                                    CODE(1320): MOV -14(5),-(6)
                                                                                                                    CODE(1324): MOV #61400,2
                                                                                                                    CODE(1330): MOV (6)+,0
                                                                                                                    CODE(1332): ADD (6)+,2
                                                                                                                    CODE(1336): MOVB0,0(2)
 888         ACL_PPOJECT(ACLE#) := PROJECT;                                                                       |1340
                                                                                                                  |
 889         IF POSITION = 0;                                                                                   |1356
                                                                                                                    CODE(1340): MOV -24(5),-(6)
                                                                                                                    CODE(1344): CLR -(6)
                                                                                                                    CODE(1346): CMP (6)+,(6)+
                                                                                                                    CODE(1350): BEQ 2
                                                                                                                    CODE(1352): JMP 0(7)
                                                                                                                    CODE(1356): MOV -20(5),-(6)
                                                                                                                    CODE(1362): MOV -6(5),-(6)
```

```
 890          THEN: ACL_CHAIN(ACLE#) := DIR_ACL_HEAD(OFFSET);

 891             DIR_ACL_HEAD(OFFSET) := ACLE#;
FIXUP LOCATION 1354 TO 7C

 892          ELSE: ACL_CHAIN(ACLE#) := ACL_CHAIN(POSITION);

 893             ACL_CHAIN(POSITION) := ACLE#;
FIXUP LOCATION 1444 TO 6A
 894         END;

 895         /*    "GIVEN" MAY RESCIND ACCESS RIGHTS

 896         IF "ODE >= WRITE#READ#EXECUTE_ACCESS;

 897          THEN: SDADD(ASTE#, OFFSET);
FIXUP LOCATION 1546 TO 34
 898         END;

 899         RC := OK_FLAG;

FIXUP LOCATION 14 TO 6
                73 LINES WERE COMPILED, GENERATING 910 BYTES OF CODE.
NO ERRORS WERE DETECTED.
```

```
CODE(1366): MOV #60100,2
CODE(1372): ADD (6)+,2
CODE(1374): MOVB0(2),1
CODE(1400): MOV 1,-(6)
CODE(1402): MOV #61600,2
CODE(1406): MOV (6)+,0
CODE(1410): ADD (6)+,2
CODE(1412): MOVB0,0(2)
  |1416
CODE(1416): MOV -6(5),-(6)
CODE(1422): MOV -20(5),-(6)
CODE(1426): MOV #60100,2
CODE(1432): MOV (6)+,0
CODE(1434): ADD (6)+,2
CODE(1436): MOVB0,0(2)
  |1442

CODE(1442): JMP 0(7)
CODE(1446): MOV -20(5),-(6)
CODE(1452): MOV -24(5),-(6)
CODE(1456): MOV #61600,2
CODE(1462): ADD (6)+,2
CODE(1464): MOVB0(2),1
CODE(1470): MOV 1,-(6)
CODE(1472): MOV #61600,2
CODE(1476): MOV (6)+,0
CODE(1500): ADD (6)+,2
CODE(1502): MOVB0,0(2)
  |1506
CODE(1506): MOV -24(5),-(6)
CODE(1512): MOV -20(5),-(6)
CODE(1516): MOV #61600,2
CODE(1522): MOV (6)+,0
CODE(1524): ADD (6)+,2
CODE(1526): MOVB0,0(2)
  |1532

  |1532
  |
*/|1532
  |
  |1550
CODE(1532): MOV -10(5),-(6)
CODE(1536): CMP #-40000,(6)+
CODE(1542): BNE 2
CODE(1544): JMP 0(7)
CODE(1550): MOV 5,-(6)
CODE(1552): MOV (5),-(6)
CODE(1554): CLR -(6)
CODE(1556): MOV -4(5),-(6)
CODE(1562): MOV -6(5),-(6)
CODE(1566): MOV -2(4),2
CODE(1572): JSR 7,@150(2)
CODE(1576): MOV 5,6
CODE(1600): CLR (6)+
CODE(1602): MOV (6)+,5
  |1604

  |1604
  |
CODE(1604): MOV #-1,4(5)
  |1612
CODE(1612): JMP #-16(5)
```

```
DATA RESCIND     E                                                    10
 900   _!_

 901   DATA RESCIND(ASTR#, OFFSET, USER, PROJECT) RETURNS (PC):       10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                              10
 902   _!_
```

```
903   PROGRAM RESCIND;
904        DECLARE
905             WORD (INDEX, SAVE_LAST);

906        /*    SECURITY CHECKS FIRST

907        /*    RESCIND IS INTERPRETATIVE DIRECTORY WRITE

908        IF WRITEDIR(ASTE#) = ERR_FLAG;

909          THEN;
910   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 50 TO 12
911        END;

912        /*    IMPLEMENTATION CHECKS

913        /*    SEARCH FOR SPECIFIED ACL ELEMENT

914        INDEX := DIR_ACL_HEAD(OFFSET);

915        CYCLE

916             IF INDEX = 0;

917               THEN;
918   ....          RETURN WITH ERR_FLAG;
FIXUP LOCATION 132 TO 12
```

```
  |16
  |16
  |16
  |
*/|16
  |
*/|16
  |
  |62
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #12,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): MOV 5,-(6)
CODE(22): MOV (5),-(6)
CODE(24): CLR -(6)
CODE(26): MOV -4(5),-(6)
CODE(32): MOV -2(4),2
CODE(36): JSR 7,@170(2)
CODE(42): MOV 5,6
CODE(44): CLR (6)+
CODE(46): MOV (6)+,5
CODE(50): CMP #-2,(6)+
CODE(54): BEQ 2
CODE(56): JMP 0(7)
CODE(62): MOV #-2,4(5)
CODE(70): JMP #-14(5)
  |74
  |74

  |74
  |
*/|74
  |
*/|74
  |
CODE(74): MOV -6(5),-(6)
CODE(100): MOV #60100,2
CODE(104): ADD (6)+,2
CODE(106): MOVB0(2),1
CODE(112): MOV 1,-16(5)
  |116
  |
  |116
  |
  |134
CODE(116): MOV -16(5),-(6)
CODE(122): CLR -(6)
CODE(124): CMP (6)+,(6)+
CODE(126): BEQ 2
CODE(130): JMP 0(7)
CODE(134): MOV #-2,4(5)
CODE(142): JMP #-14(5)
  |146
  |146
```

```
919             END;                                                                                                    |146
                                                                                                                        |
920             IF ((USER = (ACL_USER(INDEX) & ACL_USER_MASK)) & (PROJECT = ACL_PROJECT(INDEX)) |266
921                 );                                                                                                  |266
                                                                                                                CODE(146): MOV -16(5),-(6)
                                                                                                                CODE(152): ASL (6)
                                                                                                                CODE(154): MOV #61000,2
                                                                                                                CODE(160): ADD (6)+,2
                                                                                                                CODE(162): MOV 0(2),-(6)
                                                                                                                CODE(166): MOV #37777,-(6)
                                                                                                                CODE(172): COM (6)
                                                                                                                CODE(174): BIC (6)+,(6)
                                                                                                                CODE(176): MOV -10(5),3
                                                                                                                CODE(202): CMP (6)+,3
                                                                                                                CODE(204): BEQ 2
                                                                                                                CODE(206): CLR -(6)
                                                                                                                CODE(210): BR  2
                                                                                                                CODE(212): MOV #1,-(6)
                                                                                                                CODE(216): MOV -16(5),-(6)
                                                                                                                CODE(222): MOV #61400,2
                                                                                                                CODE(226): ADD (6)+,2
                                                                                                                CODE(230): MOVB0(2),1
                                                                                                                CODE(234): MOV 1,-(6)
                                                                                                                CODE(236): MOV -12(5),3
                                                                                                                CODE(242): CMP (6)+,3
                                                                                                                CODE(244): BEQ 2
                                                                                                                CODE(246): CLR -(6)
                                                                                                                CODE(250): BR  2
                                                                                                                CODE(252): MOV #1,-(6)
                                                                                                                CODE(256): BIT (6)+,(6)+
                                                                                                                CODE(260): BNE 2
                                                                                                                CODE(262): JMP 0(7)
                                                                                                                CODE(266): JMP 453(7)
922               THEN;                                                                                           |272
923         ....      *EXIT;                                                                                      |272
FIXUP LOCATION 264 TO 4
924             END;                                                                                                |272
                                                                                                                    |
925             SAVE_LAST := INDEX;                                                                             CODE(272): MOV -16(5),-20(5)
                                                                                                                    |300
                                                                                                                CODE(300): MOV -16(5),-(6)
                                                                                                                CODE(304): MOV #61600,2
                                                                                                                CODE(310): ADD (6)+,2
                                                                                                                CODE(312): MOVB0(2),1
926             INDEX := ACL_CHAIN(INDEX);                                                                      CODE(316): MOV 1,-16(5)
FIXUP LOCATION 270 TO 32                                                                                            |322
927         END;                                                                                                CODE(322): BR  -103
                                                                                                                    |324
                                                                                                                    |
928         /*    CHECKING COMPLETE - PERFORM STATE CHANGE                                                      */|324
                                                                                                                    |
929         /*    REMOVE FOUND ACL ELEMENT FROM CHAIN                                                           */|324
                                                                                                                    |
930         IF INDEX = DIR_ACL_HEAD(OFFSET);                                                                        |360
                                                                                                                CODE(324): MOV -6(5),-(6)
                                                                                                                CODE(330): MOV #60100,2
                                                                                                                CODE(334): ADD (6)+,2
                                                                                                                CODE(336): MOVB0(2),1
                                                                                                                CODE(342): MOV 1,-(6)
                                                                                                                CODE(344): MOV -16(5),3
                                                                                                                CODE(350): CMP (6)+,3
                                                                                                                CODE(352): BEQ 2
                                                                                                                CODE(354): JMP 0(7)
                                                                                                                CODE(360): MOV -6(5),-(6)
                                                                                                                CODE(364): MOV -16(5),-(6)
```

```
 931          THEN: DIR_ACL_HEAD(OFFSET) := ACL_CHAIN(INDEX);
FIXUP LOCATION 356 TO 40




 932          ELSE: ACL_CHAIN(SAVE_LAST) := ACL_CHAIN(INDEX);
FIXUP LOCATION 422 TO 40
 933        END;

 934        /*    AND PUT ON FREE CHAIN









 935        ACL_CHAIN(INDEX) := ACL_CHAIN(0);









 936        ACL_CHAIN(0) := INDEX;

 937        /*    NOW FOR THE MESSY PART









 938        SOADD(ASTE#, OFFSET); /* SEARCH OUT AND DESTROY DESCRIPTORS */

 939        RC := OK_FLAG;

FIXUP LOCATION 14 TO 4
                39 LINES WERE COMPILED, GENERATING 394 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA READIN      E
```

```
CODE(370): MOV #61600,2
CODE(374): ADD (6)+,2
CODE(376): MOVB0(2),1
CODE(402): MOV 1,-(6)
CODE(404): MOV #60100,2
CODE(410): MOV (6)+,0
CODE(412): ADD (6)+,2
CODE(414): MOVB0,0(2)
  1420

CODE(420): JMP 0(7)
CODE(424): MOV -20(5),-(6)
CODE(430): MOV -16(5),-(6)
CODE(434): MOV #61600,2
CODE(440): ADD (6)+,2
CODE(442): MOVB0(2),1
CODE(446): MOV 1,-(6)
CODE(450): MOV #61600,2
CODE(454): MOV (6)+,0
CODE(456): ADD (6)+,2
CODE(460): MOVB0,0(2)
  1464

  1464
   |
*/1464
   |
CODE(464): MOV -16(5),-(6)
CODE(470): CLR -(6)
CODE(472): MOV #61600,2
CODE(476): ADD (6)+,2
CODE(500): MOVB0(2),1
CODE(504): MOV 1,-(6)
CODE(506): MOV #61600,2
CODE(512): MOV (6)+,0
CODE(514): ADD (6)+,2
CODE(516): MOVB0,0(2)
  1522
CODE(522): CLR -(6)
CODE(524): MOV -16(5),-(6)
CODE(530): MOV #61600,2
CODE(534): MOV (6)+,0
CODE(536): ADD (6)+,2
CODE(540): MOVB0,0(2)
  1544
   |
*/1548
   |
CODE(544): MOV 5,-(6)
CODE(546): MOV (5),-(6)
CODE(550): CLR -(6)
CODE(552): MOV -4(5),-(6)
CODE(556): MOV -6(5),-(6)
CODE(562): MOV -2(4),2
CODE(566): JSR 7,@150(2)
CODE(572): MOV 5,6
CODE(574): CLR (6)+
CODE(576): MOV (6)+,5
  1600
CODE(600): MOV @-1,4(5)
  1606
CODE(606): JMP @-14(5)
```

```
940   _!_                                                              10


941   DATA READIR(ASSE@, OFFSET) RETURNS (PC):                         10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
942   _!_                                                              10
```

```
943   PROGRAM READIR;
944         DECLARE
945               WORD (CLASS, CAT, SEG_TYPE, SIZE);

946         /*    IMPLEMENTATION CHECKS

947         /*    CHECK THAT SPECIFIED SEGMENT IS A DIRECTORY

948         IF (AST_TYPE(ASTE#) & AST_TYPE_MASK) ~= AST_TYPE_DIRECTORY;

949           THEN;
950   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 56 TO 12
951         END;

952         /*    GAIN ACCESS TO THE DIRECTORY

953         IF AST_ADP(ASTE#) = 0;

954           THEN: SWAPIN(ASTE#);
FIXUP LOCATION 122 TO 30
955         END;
```

```
  |16
  |16
  |16
  |
*/|16
  |
*/|16
  |
  |60
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): MOV #4400,2
CODE(26): ADD (6)+,2
CODE(30): MOVB0(2),1
CODE(34): MOV 1,-(6)
CODE(36): MOV #200,-(6)
CODE(42): COM (6)
CODE(44): BIC (6)+,(6)
CODE(46): CMP #200,(6)+
CODE(52): BNE 2
CODE(54): JMP 0(7)
CODE(60): MOV #-2,4(5)
CODE(66): JMP #-10(5)
  |72
  |72

  |72
  |
*/|72
  |
  |124
CODE(72): MOV -4(5),-(6)
CODE(76): ASL (6)
CODE(100): MOV #7400,2
CODE(104): ADD (6)+,2
CODE(106): MOV 0(2),-(6)
CODE(112): CLR -(6)
CODE(114): CMP (6)+,(6)+
CODE(116): BEQ 2
CODE(120): JMP 0(7)
CODE(124): MOV 5,-(6)
CODE(126): MOV (5),-(6)
CODE(130): CLR -(6)
CODE(132): MOV -4(5),-(6)
CODE(136): MOV -2(4),2
CODE(142): JSR 7,#134(2)
CODE(146): MOV 5,6
CODE(150): CLR (6)+
CODE(152): MOV (6)+,5
  |154

  |154
  |
CODE(154): MOV 5,-(6)
```

```
956         LSD(ASTE#, DIR_KSR_ADR, SDR_READ_ACCESS);

957         /*    CHECK THAT SPECIFIED OFFSET EXISTS

958         IF DIR_SIZE(OFFSET) = 0;

959           THEN;
960   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 244 TO 12
961         END;

962         /*    SAVE CLASS, CAT, SEG_TYPE, AND SIZE

963         CLASS := DIR_CLASS(OFFSET) & DIR_CLASS_MASK;

964         CAT := DIR_CAT(OFFSET);

965         SEG_TYPE := DIR_TYPE(OFFSET) & DIR_TYPE_MASK;

966         SIZE := DIR_SIZE(OFFSET);
```

```
CODE(156): MOV (5),-(6)
CODE(160): CLR -(6)
CODE(162): MOV -8(5),-(6)
CODE(166): MOV #-5872,-(6)
CODE(172): MOV #2,-(6)
CODE(176): MOV -2(#),2
CODE(202): JSR 7,#154(2)
CODE(206): MOV 5,6
CODE(210): CLR (6)+
CODE(212): MOV (6)+,5
  |214
  |
*/|214
  |
  |246
CODE(214): MOV -6(5),-(6)
CODE(220): ASL (6)
CODE(222): MOV #60600,2
CODE(226): ADD (6)+,2
CODE(230): MOV 0(2),-(6)
CODE(234): CLR -(6)
CODE(236): CMP (6)+,(6)+
CODE(240): BEQ 2
CODE(242): JMP 0(7)
CODE(246): MOV #-2,#(5)
CODE(254): JMP #-10(5)
  |260
  |260

  |260
  |
*/|260
  |
CODE(260): MOV -6(5),-(6)
CODE(264): MOV #60000,2
CODE(270): ADD (6)+,2
CODE(272): MOVB0(2),1
CODE(276): MOV 1,-(6)
CODE(300): MOV #17,-(6)
CODE(304): COM (6)
CODE(306): BIC (6)+,(6)
CODE(310): MOV (6)+,-12(5)
  |314
CODE(3 4): MOV -6(5),-(6)
CODE(320): ASL (6)
CODE(322): MOV #60200,2
CODE(326): ADD (6)+,2
CODE(330): MOV 0(2),-14(5)
  |336
CODE(336): MOV -6(5),-(6)
CODE(342): MOV #60000,2
CODE(345): ADD (6)+,2
CODE(350): MOVB0(2),1
CODE(354): MOV 1,-(6)
CODE(356): MOV #200,-(6)
CODE(362): COM (6)
CODE(364): BIC (6)+,(6)
CODE(366): MOV (6)+,-16(5)
  |372
CODE(372): MOV -6(5),-(6)
CODE(376): ASL (6)
CODE(400): MOV #60600,2
CODE(404): ADD (6)+,2
CODE(406): MOV 0(2),-20(5)
  |414
  |
```

```
967         /*    REGAIN ACCESS TO USERS SB0 STACK AND INSERT DATA

968         KSDB3 := SDB0;

969         KSAB3 := SAB0;

970         CLASS_APARM := CLASS;

971         CAT_APARM := CAT;

972         SEG_TYPE_APARM := SEG_TYPE;

973         SIZE_APARM := SIZE;

974         RC := OK_FLAG;

FIXUP LOCATION 14 TO 10
          33 LINES WERE COMPILED, GENERATING 358 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA WRITTEN:    8
975   _!_
```

```
*/1414
   |
CODE(414): MOV #-5600,2
CODE(420): MOV 0(2),-(6)
CODE(424): MOV #-5472,2
CODE(430): MOV (6)+,0(2)
  |434
CODE(434): MOV #-5540,2
CODE(440): MOV 0(2),-(6)
CODE(444): MOV #-5432,2
CODE(450): MOV (6)+,0(2)
  |454
CODE(454): MOV -12(5),-(6)
CODE(460): MOV #61770,2
CODE(464): MOV (6)+,0(2)
  |470
CODE(470): MOV -14(5),-(6)
CODE(474): MOV #61766,2
CODE(500): MOV (6)+,0(2)
  |504
CODE(504): MOV -16(5),-(6)
CODE(510): MOV #61764,2
CODE(514): MOV (6)+,0(2)
  |520
CODE(520): MOV -20(5),-(6)
CODE(524): MOV #61762,2
CODE(530): MOV (6)+,0(2)
  |534
CODE(534): MOV #-1,4(5)
  |542
CODE(542): JMP #-10(5)
```

```
976   DATA WRITEDIP(ASTE0) RETURNS (PC);                                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                    10
977   _!_
```

```
978   PROGRAM WRITEDIR;

979         /*    CHECKS THAT SPECIFIED SEGMENT IS A DIRECTORY AND THAT IT IS IN CURRENT
980          * PROCESS'S R IN WRITE MODE
981         IF (AST_TYPE(ASTE0) & AST_TYPE_MASK) ~= AST_TYPE_DIRECTORY;
982           THEN;
983   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 56 TO 12
984         END;

985         IF (AST_WAL(ASTE0) & PS_PROCESS_MASK) = 0;
986           THEN;
987   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 136 TO 12
988         END;

989         IF AST_ADR(ASTE0) = 0;
```

```
  |16
  |
* |16
*/|16
  |
  |60
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): MOV #4400,2
CODE(26): ADD (6)+,2
CODE(30): MOVB0(2),1
CODE(34): MOV 1,-(6)
CODE(36): MOV #200,-(6)
CODE(42): COM (6)
CODE(44): BIC (6)+,(6)
CODE(46): CMP #200,(6)+
CODE(52): BNE 2
CODE(54): JMP 0(7)
CODE(60): MOV #-2,4(5)
CODE(66): JMP #-6(5)
  |72
  |72

  |72
  |
  |140
CODE(72): MOV -4(5),-(6)
CODE(76): ASL (6)
CODE(100): MOV #12400,2
CODE(104): ADD (6)+,2
CODE(106): MOV 0(2),-(6)
CODE(112): MOV #20102,2
CODE(116): MOV 0(2),-(6)
CODE(122): COM (6)
CODE(124): BIC (6)+,(6)
CODE(126): CLR -(6)
CODE(130): CMP (6)+,(6)+
CODE(132): BEQ 2
CODE(134): JMP 0(7)
CODE(140): MOV #-2,4(5)
CODE(146): JMP #-6(5)
  |152
  |152

  |152
  |
  |204
CODE(152): MOV -4(5),-(6)
CODE(156): ASL (6)
CODE(160): MOV #7400,2
CODE(164): ADD (6)+,2
CODE(166): MOV 0(2),-(6)
CODE(172): CLR -(6)
CODE(174): CMP (6)+,(6)+
```

```
990             THEN: SWAPIN(ASTE0);
FIXUP LOCATION 202 TO 30
991             END;

992             /*    GAIN ACCESS TO THE DIRECTORY




993             LSD(ASTE0, DIR_MSP_ADR, SDR_WRITE_ACCESS);

994             /*    DIRECTORY WILL BE CHANGED - SET CHANGE BIT




995             AST_CHANGE(ASTE0) := (AST_CHANGE(ASTE0) | AST_CHANGED);

996             RC := OK_FLAG;

FIXUP LOCATION 14 TO 2
                20 LINES WERE COMPILED, GENERATING 234 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA SCALE        E
997   _!_
```

```
CODE(176): BEQ 2
CODE(200): JMP 0(7)
CODE(204): MOV 5,-(6)
CODE(206): MOV (5),-(6)
CODE(210): CLR -(6)
CODE(212): MOV -4(5),-(6)
CODE(216): MOV -2(4),2
CODE(222): JSR 7,@134(2)
CODE(226): MOV 5,6
CODE(230): CLR (6)+
CODE(232): MOV (6)+,5
  |234

  |234
  |
*/|234
  |
CODE(234): MOV 5,-(6)
CODE(236): MOV (5),-(6)
CODE(240): CLR -(6)
CODE(242): MOV -4(5),-(6)
CODE(246): MOV @-5472,-(6)
CODE(252): MOV @6,-(6)
CODE(256): MOV -2(4),2
CODE(262): JSR 7,@154(2)
CODE(266): MOV 5,6
CODE(270): CLR (6)+
CODE(272): MOV (6)+,5
  |274
  |
*/|274
  |
CODE(274): MOV -4(5),-(6)
CODE(300): MOV -4(5),-(6)
CODE(304): MOV @4400,2
CODE(310): ADD (6)+,2
CODE(312): MOVB0(2),1
CODE(316): MOV 1,-(6)
CODE(320): BIS @40,(6)
CODE(324): MOV @4400,2
CODE(330): MOV (6)+,0
CODE(332): ADD (6)+,2
CODE(334): MOVB0,0(2)
  |340
CODE(340): MOV @-1,4(5)
  |346
CODE(346): JMP @-6(5)
```

```
999   DATA SOADD(PARITY, OFFSET);                                    10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
999   _I_                                                            10
```

```
1000   PROGRAM SOADD;
1001       DECLARE
1002            WORD (ASTE#, PROCESS#, MODE, REG#, SEG#, DUMMY);

1003       /*    DETERMINE IF SEGMENT TO WHICH ACCESS HAS BEEN RESCINDED IS ACTIVE

1004       ASTE# := HASH(DIR_DISK(OFFSET));

1005       IF (ASTE# ¬= 0) & (AST_CPL(ASTE#) ¬= 0);
1006         THEN:

1007            /*    LOOP THROUGH ALL PROCESS'S
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): MOV 5,-(6)
CODE(22): MOV (5),-(6)
CODE(24): CLR -(6)
CODE(26): MOV -6(5),-(6)
CODE(32): ASL (6)
CODE(34): MOV #60400,2
CODE(40): ADD (6)+,2
CODE(42): MOV 0(2),-(6)
CODE(46): MOV -2(4),2
CODE(52): JSR 7,@174(2)
CODE(56): MOV 5,6
CODE(60): CLR (6)+
CODE(62): MOV (6)+,5
CODE(64): MOV (6)+,-12(5)
  |70
  |
  |160
  |160
  |
*/|160
  |
CODE(70): MOV -12(5),-(6)
CODE(74): CLR -(6)
CODE(76): CMP (6)+,(6)+
CODE(100): BNE 2
CODE(102): CLR -(6)
CODE(104): BR  2
CODE(106): MOV #1,-(6)
CODE(112): MOV -12(5),-(6)
CODE(116): ASL (6)
CODE(120): MOV #11400,2
CODE(124): ADD (6)+,2
CODE(126): MOV 0(2),-(6)
CODE(132): CLR -(6)
CODE(134): CMP (6)+,(6)+
CODE(136): BNE 2
CODE(140): CLR -(6)
CODE(142): BR  2
CODE(144): MOV #1,-(6)
CODE(150): BIT (6)+,(6)+
CODE(152): BNE 2
CODE(154): JMP 0(7)
CODE(160): MOV #7,-(6)
CODE(164): MOV #1,-(6)
CODE(170): JMP 453(7)
CODE(174): INC (6)
```

```
FIXUP LOCATION 172 TO 2
1008              DO PROCESS# := PROCESS#_MIN TO PROCESS#_MAX;
1009                   IF (PT_FLAGS(PROCESS#) & PT_FLAGS_MASK) ¬= INACTIVE;
1010                   THEN: LSD(PT_PS_ASTE#(PROCESS#), PS_KSR_ADR, SDF_WRITE_ACCESS);
1011                      /*    IS THIS PROCESS CONNECTED TO THE SEGMENT?
1012                      IF (AST_CPL(ASTE#) & PS_PROCESS_MASK) ¬= 0;
1013                        THEN: /* YES - DETERMINE MODE OF ACCESS */
1014                            IF (AST_WAL(ASTE#) & PS_PROCESS_MASK) = 0;
```

```
  |214
  |
  |256
CODE(176): CMP (6),2(6)
CODE(202): BLE 2
CODE(204): JMP #53(7)
CODE(210): MOV (6),-14(5)
CODE(214): MOV -14(5),-(6)
CODE(220): MOV #17#40,2
CODE(224): ADD (6)+,2
CODE(226): MOVB0 (2),1
CODE(232): MOV 1,-(6)
CODE(236): MOV #300,-(6)
CODE(240): COM (6)
CODE(242): BIC (6)+,(6)
CODE(246): CMP #200,(6)+
CODE(250): BNE 2
CODE(252): JMP 0(7)
CODE(256): MOV 5,-(6)
CODE(260): MOV (5),-(6)
CODE(262): CLR -(6)
CODE(264): MOV -14(5),-(6)
CODE(270): ASL (6)
CODE(272): MOV #17600,2
CODE(276): ADD (6)+,2
CODE(300): MOV 0(2),-(6)
CODE(304): MOV #-5476,-(6)
CODE(310): MOV #6,-(6)
CODE(314): MOV -2(4),2
CODE(320): JSP 7,#154(2)
CODE(324): MOV 5,6
CODE(326): CLR (6)+
CODE(330): MOV (6)+,5
  |332
  |
*/|332
  |
  |400
  |400
  |
  |446
CODE(332): MOV -12(5),-(6)
CODE(336): ASL (6)
CODE(340): MOV #11400,2
CODE(344): ADD (6)+,2
CODE(346): MOV 0(2),-(6)
CODE(352): MOV #20102,2
CODE(356): MOV 0(2),-(6)
CODE(362): COM (6)
CODE(364): BIC (6)+,(6)
CODE(366): CLR -(6)
CODE(370): CMP (6)+,(6)+
CODE(372): BNE 2
CODE(374): JMP 0(7)
CODE(400): MOV -12(5),-(6)
CODE(404): ASL (6)
CODE(406): MOV #12400,2
CODE(412): ADD (6)+,2
CODE(414): MOV 0(2),-(6)
CODE(420): MOV #20102,2
CODE(424): MOV 0(2),-(6)
CODE(430): COM (6)
CODE(432): BIC (6)+,(6)
CODE(434): CLR -(6)
CODE(436): CMP (6)+,(6)+
```

```
1015                                    THEN: MODE := READ$EXECUTE_ACCESS;
FIXUP LOCATION 444 TO 12

1016                                    ELSE: MODE := WRITE$READ$EXECUTE_ACCESS;
FIXUP LOCATION 456 TO 6
1017                                END;

1018                                /*    DOES PROCESS STILL HAVE ACCESS RIGHTS?

1019                                IF DSEARCH(DASTE#, OFFSET, MODE) = ERR_FLAG;
1020                                  THEN:
```

```
CODE(440): BEQ 2
CODE(442): JMP 0(7)
CODE(446): MOV #40000,-16(5)
  |454

CODE(454): JMP 0(7)
CODE(460): MOV #-40000,-16(5)
  |466

  |466
  |
*/|466
  |
  |542
  |542
  |
CODE(466): CLR -(6)
CODE(470): MOV 5,-(6)
CODE(472): MOV (5),-(6)
CODE(474): CLR -(6)
CODE(476): MOV -4(5),-(6)
CODE(502): MOV -6(5),-(6)
CODE(506): MOV -16(5),-(6)
CODE(512): MOV -2(4),2
CODE(516): JSR 7,#204(2)
CODE(522): MOV 5,6
CODE(524): CLR (6)+
CODE(526): MOV (6)+,5
CODE(530): CMP #-2,(6)+
CODE(534): BEQ 2
CODE(536): JMP 0(7)
CODE(542): MOV #37,-(6)
CODE(546): MOV #1,-(6)
CODE(552): JMP 453(7)
CODE(556): INC (6)

  |576
  |
  |630
CODE(560): CMP (6),2(6)
CODE(564): BLE 2
CODE(566): JMP 453(7)
CODE(572): MOV (6),-22(5)
CODE(576): MOV -22(5),-(6)
CODE(602): ASL (6)
CODE(604): MOV #20200,2
CODE(610): ADD (6)+,2
CODE(612): MOV 0(2),-(6)
CODE(616): CMP -12(5),(6)+
CODE(622): BEQ 2
CODE(624): JMP 0(7)
CODE(630): MOV 5,-(6)
CODE(632): MOV (5),-(6)
CODE(634): CLR -(6)
CODE(636): MOV -22(5),-(6)
CODE(642): MOV -12(5),-(6)
CODE(646): MOV -2(4),2
CODE(652): JSR 7,#14(2)
CODE(656): MOV 5,6
CODE(660): CLR (6)+
CODE(662): MOV (6)+,5
  |664
CODE(664): JMP 453(7)
  |670

  |670
```

```
FIXUP LOCATION 554 TO 2
1021                                DO SEG# := SEG#_MIN TO SEG#_MAX;

1022                                    IF PS_SEG(SEG#) = ASTE#;
```

```
1023                                        THEN: DCONNECT(SEG#, ASTE#);

1024                                ....        EXIT;
FIXUP LOCATION 626 TO 40
1025                                    END;
```

```
FIXUP LOCATION 570 TO 100
FIXUP LOCATION 666 TO 2

1026                                    END;

FIXUP LOCATION 540 TO 132
1027                                END;

FIXUP LOCATION 376 TO 274
1028                            END;

FIXUP LOCATION 254 TO 416
1029                        END;

FIXUP LOCATION 206 TO 470

1030                END;

1031                /*    RESTORE KSR2

1032                LSD(PT_PS_ASIE#(THE_CURRENT_PROCESS), PS_KSR_ADR, SDP_WRITE_ACCESS);
FIXUP LOCATION 156 TO 602
1033        END;

FIXUP LOCATION 14 TO 14
                35 LINES WERE COMPILED, GENERATING 502 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA GETW           F
```

```
  |

CODE(670): BR  -46
CODE(672): CMP (6)+,(6)+
  |674
  |

  |674
  |

  |674
  |

  |674
  |

CODE(674): JMP -504(7)
CODE(700): CMP (6)+,(6)+
  |702
  |
*/|702
  |
CODE(702): MOV 5,-(6)
CODE(704): MOV (5),-(6)
CODE(706): CLR -(6)
CODE(710): MOV #16544,2
CODE(714): MOV 0(2),-(6)
CODE(720): ASL (6)
CODE(722): MOV #17600,2
CODE(726): ADD (6)+,2
CODE(730): MOV 0(2),-(6)
CODE(734): MOV #-5476,-(6)
CODE(740): MOV #6,-(6)
CODE(744): MOV -2(4),2
CODE(750): JSR 7,@154(2)
CODE(754): MOV 5,6
CODE(756): CLR (6)+
CODE(760): MOV (6)+,5
  |762
  |

  |762
  |
CODE(762): JMP @-10(5)
```

10

1034 -1-

```
1035   DATA GETW(ASTR*, OFFSET) RETURNS (BC);                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                         10
1036   _I_
```

```
1037   PROGRAM GETW;
1038         /*    SECURITY CHECKS FIRST
1039         /*    SEARCH DIRECTORY ACL
1040         IF DSEARCH(ASTE#, OFFSET, WRITE$READ$EXECUTE_ACCESS) = ERR_FLAG;
1041           THEN:
1042   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 70 TO 12
1043         END;
1044         IF PS_CURRENT_PROCESS ¬= EXEC_PROCESS#;
1045           THEN:
1046             /*    CHECK FOR PRESERVATION OF SECURITY AND * -PROPERTY
1047             IF PS_CUR_CLASS ¬= (DIR_CLASS(OFFSET) & DIR_CLASS_MASK);
```

```
  |16
  |
*/|16
  |
*/|16
  |
  |72
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): MOV 5,-(6)
CODE(22): MOV (5),-(6)
CODE(24): CLR -(6)
CODE(26): MOV -4(5),-(6)
CODE(32): MOV -6(5),-(6)
CODE(36): MOV #-40000,-(6)
CODE(42): MOV -2(4),2
CODE(46): JSR 7,@204(2)
CODE(52): MOV 5,6
CODE(54): CLR (6)+
CODE(56): MOV (6)+,5
CODE(60): CMP #-2,(6)+
CODE(64): BEQ 2
CODE(66): JMP 0(7)
CODE(72): MOV #-2,4(5)
CODE(100): JMP @-10(5)
  |104
  |104
  |
  |104
  |
  |126
  |126
  |
*/|126
  |
  |176
CODE(104): MOV #20100,2
CODE(110): MOV 0(2),-(6)
CODE(114): CMP #1,(6)+
CODE(120): BNE 2
CODE(122): JMP 0(7)
CODE(126): MOV -6(5),-(6)
CODE(132): MOV #60000,2
CODE(136): ADD (6)+,2
CODE(140): MOVB0(2),1
CODE(144): MOV 1,-(6)
CODE(146): MOV #17,-(6)
CODE(152): COM (6)
CODE(154): BIC (6)+,(6)
CODE(156): MOV #20112,2
CODE(162): MOVB0(2),3
CODE(166): CMP (6)+,3
CODE(170): BNE 2
CODE(172): JMP 0(7)
```

```
1048                THEN:
1049  ....             RETURN WITH ERR_FLAG;
FIXUP LOCATION 174 TO 12
1050                END;

1051                IF PS_CUR_CAT >= DIR_CAT(OFFSET);
```

```
CODE(176): MOV #-2,#(5)
CODE(204): JMP #-10(5)
  1210
  1210

  1210
  |
  1250
CODE(210): MOV -6(5),-(6)
CODE(214): ASL (6)
CODE(216): MOV #60200,2
CODE(222): ADD (6)+,2
CODE(224): MOV 0(2),-(6)
CODE(230): MOV #20114,2
CODE(234): MOV 0(2),3
CODE(240): CMP (6)+,3
CODE(242): BNE 2
CODE(244): JMP 0(7)
CODE(250): MOV #-2,#(5)
CODE(256): JMP #-10(5)
  1262
  1262

  1262
  |
  1266
  1336
CODE(262): JMP 0(7)
CODE(266): MOV -6(5),-(6)
CODE(272): MOV #60000,2
CODE(276): ADD (6)+,2
CODE(300): MOVB0(2),1
CODE(304): MOV 1,-(6)
CODE(306): MOV #17,-(6)
CODE(312): COM (6)
CODE(314): BIC (6)+,(6)
CODE(316): MOV #20112,2
CODE(322): MOVB0(2),3
CODE(326): CMP (6)+,3
CODE(330): BGT 2
CODE(332): JMP 0(7)
CODE(336): MOV #-2,#(5)
CODE(344): JMP #-10(5)
  1350
  1350

  1350
  |
  142
CODE(350): MOV -6(5),-(6)
CODE(354): ASL (6)
CODE(356): MOV #60200,2
CODE(362): ADD (6)+,2
CODE(364): MOV 0(2),-(6)
CODE(370): MOV #20114,2
CODE(374): BIS 0(2),(6)
CODE(400): MOV #20114,2
CODE(404): MOV 0(2),3
CODE(410): CMP (6)+,3
CODE(412): BNE 2
CODE(414): JMP 0(7)
CODE(420): MOV #-2,#(5)
CODE(426): JMP #-10(5)
  1432
  1432
```

```
1052                THEN:
1053  ....             RETURN WITH ERR_FLAG;
FIXUP LOCATION 246 TO 12
1054                END;

FIXUP LOCATION 124 TO 140
1055          ELSE:
1056                IF PS_CUR_CLASS < (DIR_CLASS(OFFSET) & DIR_CLASS_MASK);
```

```
1057                THEN:
1058  ....             RETURN WITH ERR_FLAG;
FIXUP LOCATION 334 TO 12
1059                END;

1060                IF PS_CUR_CAT >= (DIR_CAT(OFFSET) | PS_CUR_CAT);
```

```
1061                THEN:
1062  ....             RETURN WITH ERR_FLAG;
```

```
FIXUP LOCATION 416 TO 12
1063              END;

FIXUP LOCATION 264 TO 144
1064         END;

1065         /*    IMPLEMENTATION CHECKS

1066         /*    CONNECT PROCESS TO AST ENTRY FOR THIS SEGMENT

1067         PC := CONNECT(ASTE#, OFFSET, WRITE$READ$EXECUTE_ACCESS);

FIXUP LOCATION 14 TO 2
                  32 LINES WERE COMPILED, GENERATING 324 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA GETP          P
1068   _|_
```

```
  |432
  |

  |432
  |
*/|432
  |
*/|432
  |
CODE(432): CLR -(6)
CODE(434): MOV 5,-(6)
CODE(436): MOV (5),-(6)
CODE(440): CLR -(6)
CODE(442): MOV -4(5),-(6)
CODE(446): MOV -6(5),-(6)
CODE(452): MOV #-40000,-(6)
CODE(456): MOV -2(4),2
CODE(462): JSR 7,#210(2)
CODE(466): MOV 5,6
CODE(470): CLR (6)+
CODE(472): MOV (6)+,5
CODE(474): MOV (6)+,4(5)
  |500
CODE(500): JMP #-10(5)
```

```
1069   DATA GTIF(ADDR#, OFFSET) RETURNS (PC);                    10
                    2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                         10
1070   _I_
```

```
1071   PROGRAM GETR;
1072         /*    SECURITY CHECKS FIRST
1073         /*    SEARCH DIRECTORY ACL
1074         IF DSEARCH(ASTE#, OFFSET, READ$EXECUTE_ACCESS) = ERR_FLAG;
```

```
1075           THEN;
1076   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 70 TO 12
1077         END;
1078         /*    CHECK FOR PRESERVATION OF SECURITY AND * -PROPERTY
1079         IF PS_CUR_CLASS < (DIR_CLASS(OFFSET) & DIR_CLASS_MASK);
```

```
1080           THEN;
1081   ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 152 TO 12
1082         END;
1083         IF PS_CUR_CAT ¬= (DIR_CAT(OFFSET) | PS_CUR_CAT);
```

```
  |16
  |
*/|16
  |
*/|16
  |
  |72
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): MOV 5,-(6)
CODE(22): MOV (5),-(6)
CODE(24): CLR -(6)
CODE(26): MOV -4(5),-(6)
CODE(32): MOV -6(5),-(6)
CODE(36): MOV #40000,-(6)
CODE(42): MOV -2(4),2
CODE(46): JSR 7,@204(2)
CODE(52): MOV 5,6
CODE(54): CLR (6)+
CODE(56): MOV (6)+,5
CODE(60): CMP #-2,(6)+
CODE(64): BEQ 2
CODE(66): JMP 0(7)
CODE(72): MOV #-2,4(5)
CODE(100): JMP @-10(5)
  |104
  |104

  |104
  |
*/|104
  |
  |154
CODE(104): MOV -6(5),-(6)
CODE(110): MOV #60000,2
CODE(114): ADD (6)+,2
CODE(116): MOVB0(2),1
CODE(122): MOV 1,-(6)
CODE(124): MOV #17,-(6)
CODE(130): COM (6)
CODE(132): BIC (6)+,(6)
CODE(134): MOV #20112,2
CODE(140): MOVB0(2),3
CODE(144): CMP (6)+,3
CODE(146): BGT 2
CODE(150): JMP 0(7)
CODE(154): MOV #-2,4(5)
CODE(162): JMP @-10(5)
  |166
  |166

  |166
  |
  |236
```

```
1084          THEN:
1085  ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 234 TO 12
1086        END;

1087        /*    IMPLEMENTATION CHECKS

1088        /*    CONNECT PROCESS TO AST ENTRY FOR THIS SEGMENT

1089        RC := CONNECT(ASTE#, OFFSET, READ$EXECUTE_ACCESS);

FIXUP LOCATION 14 TO 2
                20 LINES WERE COMPILED, GENERATING 210 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA DSEARCH      E
1090  _!_
```

```
CODE(166): MOV -6(5),-(6)
CODE(172): ASL (6)
CODE(174): MOV #6C200,2
CODE(200): ADD (6)+,2
CODE(202): MOV 0(2),-(6)
CODE(206): MOV #20114,2
CODE(212): BIS 0(2),(6)
CODE(216): MOV #20114,2
CODE(222): MOV 0(2),3
CODE(226): CMP (6)+,3
CODE(230): BNE 2
CODE(232): JMP 0(7)
CODE(236): MOV #-2,4(5)
CODE(244): JMP #-10(5)
  |250
  |250

  |250
  |
*/|250
  |
*/|250
  |
CODE(250): CLR -(6)
CODE(252): MOV 5,-(6)
CODE(254): MOV (5),-(6)
CODE(256): CLR -(6)
CODE(260): MOV -4(5),-(6)
CODE(264): MOV -6(5),-(6)
CODE(270): MOV #40000,-(6)
CODE(274): MOV -2(4),2
CODE(300): JSR 7,#210(2)
CODE(304): MOV 5,6
CODE(306): CLR (6)+
CODE(310): MOV (6)+,5
CODE(312): MOV (6)+,4(5)
  |316
CODE(316): JMP #-10(5)
```

```
1091   DATA DSEARCH(ASTER, OFFSET, MODE) RETURNS (BC);                    10
                    2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                  10
1092   _I_
```

```
1093   PROGRAM DSEARCH;
1094         DECLARE
1095              WORD (INDEX, USER, PROJECT);

1096         IF (AST_TYPE(ASTE#) & AST_TYPE_MASK) ^= AST_TYPE_DIRECTORY;
                                                                                    |16
                                                                                    |16
                                                                                    |16
                                                                                    |
                                                                                    |60
                                                                                  CODE(0):      2,-(0)
                                                                                  CODE(2): MOV 6,5
                                                                                  CODE(4): ADD #10,5
                                                                                  CODE(10): MOV 7,(5)+
                                                                                  CODE(12): SUB #0,6
                                                                                  CODE(16): MOV -4(5),-(6)
                                                                                  CODE(22): MOV #4400,2
                                                                                  CODE(26): ADD (6)+,2
                                                                                  CODE(30): MOVB0(2),1
                                                                                  CODE(34): MOV 1,-(6)
                                                                                  CODE(36): MOV #200,-(6)
                                                                                  CODE(42): COM (6)
                                                                                  CODE(44): BIC (6)+,(6)
                                                                                  CODE(46): CMP #200,(6)+
                                                                                  CODE(52): BNE 2
                                                                                  CODE(54): JMP 0(7)
                                                                                  CODE(60): MOV #-2,4(5)
                                                                                  CODE(66): JMP #-12(5)
                                                                                    |72
                                                                                    |72

1097           THEN:                                                              |72
1098   ....        RETURN WITH ERR_FLAG;                                            |
FIXUP LOCATION 56 TO 12                                                         */172
1099         END;                                                                 |
                                                                                    |120
1100         /*    GAIN ACCESS TO DIRECTORY                                      CODE(72): MOV -4(5),-(6)
                                                                                  CODE(76): ASL (6)
1101         IF AST_ADR(ASTE#) = 0;                                             CODE(100): MOV #7400,2
                                                                                  CODE(104): ADD (6)+,2
                                                                                  CODE(106): MOV 0(2),-(6)
                                                                                  CODE(112): CLR -(6)
                                                                                  CODE(114): CMP (6)+,(6)+
                                                                                  CODE(116): BEQ 2
                                                                                  CODE(120): JMP 0(7)
                                                                                  CODE(124): MOV 5,-(6)
                                                                                  CODE(126): MOV (5),-(6)
                                                                                  CODE(130): CLR -(6)
                                                                                  CODE(132): MOV -4(5),-(6)
                                                                                  CODE(136): MOV -2(4),2
                                                                                  CODE(142): JSR 7,#134(2)
                                                                                  CODE(146): MOV 5,6
                                                                                  CODE(150): CLR (6)+
                                                                                  CODE(152): MOV (6)+,5
                                                                                    |154

1102           THEN: SWAPIN(ASTE#);                                               |154
FIXUP LOCATION 122 TO 30                                                            |
1103         END;                                                                 CODE(154): MOV 5,-(6)
                                                                                  CODE(156): MOV (5),-(6)
                                                                                  CODE(160): CLR -(6)
                                                                                  CODE(162): MOV -4(5),-(6)
                                                                                  CODE(166): MOV #-5472,-(6)
```

```
1104        LSD(ASTE#, DIR_KSR_ADR, SDR_WRITE_ACCESS);
1105        /*    NO NEED TO CHECK A USE BIT - ACL WILL BE EMPTY IF ENTRY IS NOT IN USE
1106        /*    SEARCH ACL FOR ELEMENT THAT GIVES CURRENT USER PERMISSION TO ACCESS
1107        INDEX := DIR_ACL_HEAD(OFFSET);
1108        CYCLE
1109             IF INDEX = 0;
1110               THEN;
1111  ....            RETURN WITH ERR_FLAG;
FIXUP LOCATION 252 TO 12
1112             END;
1113             USER := (ACL_USER(INDEX) & ACL_USER_MASK);
1114             PROJECT := ACL_PROJECT(INDEX);
```

```
CODE(172): MOV #6,-(6)
CODE(176): MOV -2(4),2
CODE(202): JSR 7,#154(2)
CODE(206): MOV 5,6
CODE(210): CLR (6)+
CODE(212): MOV (6)+,5
  |214
  |
*/|214
  |
*/|214
  |
CODE(214): MOV -6(5),-(6)
CODE(220): MOV #60100,2
CODE(224): ADD (6)+,2
CODE(226): MOVB0(2),1
CODE(232): MOV 1,-14(5)
  |236
  |
  |236
  |
  |254
CODE(236): MOV -14(5),-(6)
CODE(242): CLR -(6)
CODE(244): CMP (6)+,(6)+
CODE(246): BEQ 2
CODE(250): JMP 0(7)
CODE(254): MOV #-2,4(5)
CODE(262): JMP #-12(5)
  |266
  |266
  |
  |266
  |
CODE(266): MOV -14(5),-(6)
CODE(272): ASL (6)
CODE(274): MOV #61000,2
CODE(300): ADD (6)+,2
CODE(302): MOV 0(2),-(6)
CODE(306): MOV #37777,-(6)
CODE(312): COM (6)
CODE(314): BIC (6)+,(6)
CODE(316): MOV (6)+,-16(5)
  |322
CODE(322): MOV -14(5),-(6)
CODE(326): MOV #61400,2
CODE(332): ADD (6)+,2
CODE(334): MOVB0(2),1
CODE(340): MOV 1,-20(5)
  |344
CODE(344): MOV -16(5),-(6)
CODE(350): CMP #37777,(6)+
CODE(354): BEQ 2
CODE(356): CLR -(6)
CODE(360): BR  2
CODE(362): MOV #1,-(6)
CODE(366): MOV -16(5),-(6)
CODE(372): MOV #20106,2
CODE(376): MOV 0(2),-(6)
CODE(402): CMP (6)+,(6)+
CODE(404): BEQ 2
CODE(406): CLR -(6)
CODE(410): BR  2
CODE(412): MOV #1,-(6)
CODE(416): BIS (6)+,(6)
CODE(420): MOV -20(5),-(6)
```

```
FIXUP LOCATION 510 TO 4
1115        .... EXIT WHEN ((USER = ALL_USERS) | (USER = PS_USER_ID)) & ((PROJECT =
1116                    PS_PROJECT_ID) | (PROJECT = ALL_PROJECTS));



1117              INDEX := ACL_CHAIN(INDEX);
FIXUP LOCATION 514 TO 24

1118        END;

1119        IF (ACL_MODE(INDEX) & ACL_MODE_MASK) = NO_ACCESS;









1120          THEN;
1121    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 602 TO 12
1122        END;

1123        IF (MODE = WRITE$READ$EXECUTE_ACCESS) & ((ACL_MODE(INDEX) & ACL_MODE_MASK) ¬=
1124             WRITE$READ$EXECUTE_ACCESS);
```

```
CODE(424): MOV #20110,2
CODE(430): MOV 0(2),-(6)
CODE(434): CMP (6)+,(6)+
CODE(436): BEQ 2
CODE(440): CLR -(6)
CODE(442): BR  2
CODE(444): MOV #1,-(6)
CODE(450): MOV -20(5),-(6)
CODE(454): MOV #177,-(6)
CODE(460): CMP (6)+,(6)+
CODE(462): BEQ 2
CODE(464): CLR -(6)
CODE(466): BR  2
CODE(470): MOV #1,-(6)
CODE(474): BIS (6)+,(6)
CODE(476): COM (6)
CODE(500): BIC (6)+,(6)
CODE(502): BOR (6)+
CODE(504): BCS 2
CODE(506): JMP 0(7)
CODE(512): JMP 453(7)

  |516
  |516
CODE(516): MOV -14(5),-(6)
CODE(522): MOV #61600,2
CODE(526): ADD (6)+,2
CODE(530): MOVB0(2),1
CODE(534): MOV 1,-14(5)
  |540

CODE(540): BR  -142
  |542
  |
  |604
CODE(542): MOV -14(5),-(6)
CODE(546): ASL (6)
CODE(550): MOV #61000,2
CODE(554): ADD (6)+,2
CODE(556): MOV 0(2),-(6)
CODE(562): MOV #-40000,-(6)
CODE(566): COM (6)
CODE(570): BIC (6)+,(6)
CODE(572): CLR -(6)
CODE(574): CMP (6)+,(6)+
CODE(576): BEQ 2
CODE(600): JMP 0(7)
CODE(604): MOV #-2,#(5)
CODE(612): JMP #-12(5)
  |616
  |616

  |616
  |
  |720
  |720
CODE(616): MOV -10(5),-(6)
CODE(622): CMP #-40000,(6)+
CODE(626): BEQ 2
CODE(630): CLR -(6)
CODE(632): BR  2
CODE(634): MOV #1,-(6)
CODE(640): MOV -14(5),-(6)
CODE(644): ASL (6)
CODE(646): MOV #61000,2
CODE(652): ADD (6)+,2
```

```
1125          THEN:
1126  ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 716 TO 12
1127          END;


1128          RC := OK_FLAG;

FIXUP LOCATION 14 TO 6
                37 LINES WERE COMPILED, GENERATING 484 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA CONNECT      F
1129  _!_
```

```
CODE(654):  MOV 0(2),-(6)
CODE(660):  MOV #-40000,-(6)
CODE(664):  COM (6)
CODE(666):  BIC (6)+,(6)
CODE(670):  MOV #-40000,-(6)
CODE(674):  CMP (6)+,(6)+
CODE(676):  BNE 2
CODE(700):  CLR -(6)
CODE(702):  BR  2
CODE(704):  MOV #1,-(6)
CODE(710):  BIT (6)+,(6)+
CODE(712):  BNE 2
CODE(714):  JMP 0(7)
CODE(720):  MOV #-2,4(5)
CODE(726):  JMP @-12(5)
  |732
  |732

  |732
  |
CODE(732):  MOV #-1,4(5)
  |740
CODE(740):  JMP @-12(5)


  10
```

```
1130   DATA CONNECT(DASIF*, OFFSET, MODE) RETURNS (RC);                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1131   _I_                                                                 10
```

```
1132    PROGRAM CONNECT;
1133        DECLARE
1134            WORD (INDEX, NEXT, SEG#, ASTE#, HASH_VAL);

1135        /*    FIND A FREE SPG#

1136        SEG# := (PS_SPG(^) & SEG_MASK);

1137        IF SPG# = 0;

1138          THEN:
1139    ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 64 TO 12
1140        END;

1141        /*    DETERMINE IF SEGMENT IS ACTIVE

1142        ASTE# := HASH(DIR_DISK(OFFSET));

1143        IF ASTE# = 0; /* THEN: MUST ACTIVATE */
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #10,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): ASL (6)
CODE(22): MOV #20200,2
CODE(26): ADD (6)+,2
CODE(30): MOV 0(2),-(6)
CODE(34): MOV #77777,-(6)
CODE(40): COM (6)
CODE(42): BIC (6)+,(6)
CODE(44): MOV (6)+,-20(5)
  |50
  |
  |66
CODE(50): MOV -20(5),-(6)
CODE(54): CLR -(6)
CODE(56): CMP (6)+,(6)+
CODE(60): BEQ 2
CODE(62): JMP 0(7)
CODE(66): MOV #-2,4(5)
CODE(74): JMP @-12(5)
  |100
  |100

  |100
  |
*/|100
  |
CODE(100): CLR -(6)
CODE(102): MOV 5,-(6)
CODE(104): MOV (5),-(6)
CODE(106): CLR -(6)
CODE(110): MOV -6(5),-(6)
CODE(114): ASL (6)
CODE(116): MOV #60400,2
CODE(122): ADD (6)+,2
CODE(124): MOV 0(2),-(6)
CODE(130): MOV -2(4),2
CODE(134): JSR 7,@174(2)
CODE(140): MOV 5,6
CODE(142): CLR (6)+
CODE(144): MOV (6)+,5
CODE(146): MOV (6)+,-22(5)
  |152
  |
  |170
CODE(152): MOV -22(5),-(6)
CODE(156): CLR -(6)
CODE(160): CMP (6)+,(6)+
```

```
1144          THEN: ASTF# := ACT(DASTE#, OFFSET);
FIXUP LOCATION 144 TO 46
1145          ELSE:
1146             IF (AST_CPL(ASTF#) & PS_PROCESS_MASK) >= 0;
```

```
1147             THEN:
1148   ....          RETURN WITH ERR_FLAG;
FIXUP LOCATION 302 TO 12
1149             END;

FIXUP LOCATION 234 TO 60
1150       END;

1151       /*    UNAGE IF NECESSARY

1152       IF AST_CPL(ASTE#) = 0;
```

```
1153         THEN: INDEX := 0;

1154            CYCLE
```

```
1155                NEXT := AST_AGE_CHAIN(INDEX);
```

```
CODE(162): BEQ 2
CODE(164): JMP 0(7)
CODE(170): CLR -(6)
CODE(172): MOV 5,-(6)
CODE(174): MOV (5),-(6)
CODE(176): CLR -(6)
CODE(200): MOV -4(5),-(6)
CODE(204): MOV -6(5),-(6)
CODE(210): MOV -2(4),2
CODE(214): JSR 7,@214(2)
CODE(220): MOV 5,6
CODE(222): CLR (6)+
CODE(224): MOV (6)+,5
CODE(226): MOV (6)+,-22(5)
  |232

  |236
  |304
CODE(232): JMP 0(7)
CODE(236): MOV -22(5),-(6)
CODE(242): ASL (6)
CODE(244): MOV #11400,2
CODE(250): ADD (6)+,2
CODE(252): MOV 0(2),-(6)
CODE(256): MOV #20102,2
CODE(262): MOV 0(2),-(6)
CODE(266): COM (6)
CODE(270): BIC (6)+,(6)
CODE(272): CLR -(6)
CODE(274): CMP (6)+,(6)+
CODE(276): BNE 2
CODE(300): JMP 0(7)
CODE(304): MOV #-2,4(5)
CODE(312): JMP @-12(5)
  |316
  |316

  |316
  |

  |316
  |
*/|316
  |
  |350
CODE(316): MOV -22(5),-(6)
CODE(322): ASL (6)
CODE(324): MOV #11400,2
CODE(330): ADD (6)+,2
CODE(332): MOV 0(2),-(6)
CODE(336): CLR -(6)
CODE(340): CMP (6)+,(6)+
CODE(342): BEQ 2
CODE(344): JMP 0(7)
CODE(350): CLR -(6)
CODE(352): MOV (6)+,-14(5)
  |356
  |
  |356
CODE(356): MOV -14(5),-(6)
CODE(362): ASL (6)
CODE(364): MOV #12400,2
CODE(370): ADD (6)+,2
CODE(372): MOV 0(2),-16(5)
  |400
CODE(400): MOV -16(5),-(6)
```

```
FIXUP LOCATION 414 TO 4
1156                .... EXIT WHEN NEXT = ASTE#;

1157                    INDEX := NEXT;
FIXUP LOCATION 420 TO 10

1158                END;

1159                AST_AGR_CHAIN(INDEX) := AST_AGR_CHAIN(ASTE#);

1160                AST_AGR_CHAIN(ASTE#) := 0;
FIXUP LOCATION 346 TO 150
1161            END;

1162            /*    ADD THIS PROCESS TO CONNECTED PROCESS LIST

1163            AST_CPL(ASTE#) := (AST_CPL(ASTE#) | PS_PROCESS_MASK);

1164            IF MODE = WRITE$READ$EXECUTE_ACCESS;
```

```
CODE(404): CMP -22(5),(6)+
CODE(410): BEQ 2
CODE(412): JMP 0(7)
CODE(416): JMP 453(7)

  |422
CODE(422): MOV -16(5),-14(5)
  |430

CODE(430): BR  -26
  |432
  |
CODE(432): MOV -14(5),-(6)
CODE(436): ASL (6)
CODE(440): MOV -22(5),-(6)
CODE(444): ASL (6)
CODE(446): MOV #12400,2
CODE(452): ADD (6)+,2
CODE(454): MOV 0(2),-(6)
CODE(460): MOV #12400,2
CODE(464): MOV (6)+,0
CODE(466): ADD (6)+,2
CODE(470): MOV 0,0(2)
  |474
CODE(474): MOV -22(5),-(6)
CODE(500): ASL (6)
CODE(502): CLR -(6)
CODE(504): MOV #12400,2
CODE(510): MOV (6)+,0
CODE(512): ADD (6)+,2
CODE(516): MOV 0,0(2)
  |520

  |520
  |
*/|520
  |
CODE(520): MOV -22(5),-(6)
CODE(524): ASL (6)
CODE(526): MOV -22(5),-(6)
CODE(532): ASL (6)
CODE(534): MOV #11400,2
CODE(540): ADD (6)+,2
CODE(542): MOV 0(2),-(6)
CODE(546): MOV #20102,2
CODE(552): BIS 0(2),(6)
CODE(556): MOV #11400,2
CODE(562): MOV (6)+,0
CODE(564): ADD (6)+,2
CODE(566): MOV 0,0(2)
  |572
  |
  |610
CODE(572): MOV -10(5),-(6)
CODE(576): CMP #-40000,(6)+
CODE(602): BEQ 2
CODE(604): JMP 0(7)
CODE(610): MOV -22(5),-(6)
CODE(614): ASL (6)
CODE(616): MOV -22(5),-(6)
CODE(622): ASL (6)
CODE(624): MOV #12400,2
CODE(630): ADD (6)+,2
CODE(632): MOV 0(2),-(6)
CODE(636): MOV #20102,2
CODE(642): BIS 0(2),(6)
```

```
1165          THEN: AST_MAL(ASID#) := (AST_MAL(ASID#) ! PS_PROCESS_MASK);
FIXUP LOCATION 606 TO 52
1166        END;

1167        /*    AND UPDATE PS_SEG




1168        PS_SEG(0) := PS_SEG(SEG#);




1169        PS_SEG(SEG#) := ASID#;

1170        RC := SEG#;

FIXUP LOCATION 14 TO 12
                40 LINES WERE COMPILED, GENERATING 498 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA CONNECT    9
```

```
CODE(646): MOV #12400,2
CODE(652): MOV (6)+,0
CODE(654): ADD (6)+,2
CODE(656): MOV 0,0(2)
  !662

  !662
   !
*/!662
   !
CODE(662): CLR -(6)
CODE(664): ASL (6)
CODE(666): MOV -20(5),-(6)
CODE(672): ASL (6)
CODE(674): MOV #20200,2
CODE(700): ADD (6)+,2
CODE(702): MOV 0(2),-(6)
CODE(706): MOV #20200,2
CODE(712): MOV (6)+,C
CODE(714): ADD (6)+,2
CODE(716): MOV 0,0(2)
  !722
CODE(722): MOV -20(5),-(6)
CODE(726): ASL (6)
CODE(730): MOV -22(5),-(6)
CODE(734): MOV #20200,2
CODE(740): MOV (6)+,0
CODE(742): ADD (6)+,2
CODE(744): MOV 0,0(2)
  !750
CODE(750): MOV -20(5),4(5)
  !756
CODE(756): JMP @-12(5)
```

```
1172   DATA DCONNECT(SEG*, ASCII*);
                    2 LINES WERE COMPILED.                                  10
NO ERRORS WERE DETECTED.
1173   -I-                                                                  10
```

```
1174   PROGRAM DCONNECT;
1175         DECLARE
1176              WORD (BLOCK#, REG#);

1177         /*    PROCESS MUST NOT HAVE ANY DESCRIPTORS ON SEGMENT LOCKED IN

1178         BLOCK# := AST_ADR(ASTE#);

1179         IF (BLOCK# ¬= 0) & ((AST_UNLOCK(ASTE#) & AST_UNLOCK_MASK) ¬= AST_UNLOCK_FLAG);
1180           THEN;

1181              /*    MY BLOCKS TO THEIRS

1182              INLINE(ASL, BLOCK#);

1183              INLINE(ASL, BLOCK#);

FIXUP LOCATION 162 TO 2
1184              DO REG# := 0 TO REG#_MAX;
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(6): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #C,6
CODE(16): MOV -6(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #7400,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-12(5)
  |40
  |
  |142
  |142
  |
*/|142
  |
CODE(40): MOV -12(5),-(6)
CODE(44): CLR -(6)
CODE(46): CMP (6)+,(6)+
CODE(50): BNE 2
CODE(52): CLR -(6)
CODE(54): BR  2
CODE(56): MOV #1,-(6)
CODE(62): MOV -6(5),-(6)
CODE(66): MOV #4400,2
CODE(72): ADD (6)+,2
CODE(74): MOVBO(2),1
CODE(100): MOV 1,-(6)
CODE(102): MOV #20,-(6)
CODE(106): COM (6)
CODE(110): BIC (6)+,(6)
CODE(112): MOV #20,-(6)
CODE(116): CMP (6)+,(6)+
CODE(120): BNE 2
CODE(122): CLR -(6)
CODE(124): BR  2
CODE(126): MOV #1,-(6)
CODE(132): BIT (6)+,(6)+
CODE(134): BNE 2
CODE(136): JMP 0(7)
CODE(142): ASL -12(5)
  |146
CODE(146): ASL -12(5)
  |152
  |
CODE(152): MOV #17,-(6)
CODE(156): CLR -(6)
CODE(160): JMP 453(7)
CODE(164): INC (6)

  |204
```

```
1185                    IF PS_SAP(REG0) = BLOCK#;

1186                        THEN: DISABLE(REG0);
FIXUP LOCATION 234 TO 30
1187                        END;

FIXUP LOCATION 176 TO 70

1188                END;

FIXUP LOCATION 140 TO 130
1189           END;

1190           /*    DISCONNECT THIS PROCESS

1191           AST_CPL(ASTF0) := (AST_CPL(ASTF0) & PS_PROCESS_NOTMASK);
```

```
 |
 1236
CODE(166): CMP  (6),2(6)
CODE(172): BLE  2
CODE(174): JMP  453(7)
CODE(200): MOV  (6),-14(5)
CODE(204): MOV  -14(5),-(6)
CODE(210): ASL  (6)
CODE(212): MOV  #20040,2
CODE(216): ADD  (6)+,2
CODE(220): MOV  0(2),-(6)
CODE(224): CMP  -12(5),(6)+
CODE(230): BEQ  2
CODE(232): JMP  0(7)
CODE(236): MOV  5,-(6)
CODE(240): MOV  (5),-(6)
CODE(242): CLR  -(6)
CODE(244): MOV  -14(5),-(6)
CODE(250): MOV  -2(4),2
CODE(254): JSR  7,#10(2)
CODE(260): MOV  5,6
CODE(262): CLR  (6)+
CODE(264): MOV  (6)+,5
 |266

 |266
 |

CODE(266): BR   -42
CODE(270): CMP  (6)+,(6)+
 |272
 |

 |272
 |
*/|272
 |
CODE(272): MOV  -6(5),-(6)
CODE(276): ASL  (6)
CODE(300): MOV  -6(5),-(6)
CODE(304): ASL  (6)
CODE(306): MOV  #11400,2
CODE(312): ADD  (6)+,2
CODE(314): MOV  0(2),-(6)
CODE(320): MOV  #20104,2
CODE(324): MOV  0(2),-(6)
CODE(330): COM  (6)
CODE(332): BIC  (6)+,(6)
CODE(334): MOV  #11400,2
CODE(340): MOV  (6)+,0
CODE(342): ADD  (6)+,2
CODE(344): MOV  0,0(2)
 |350
CODE(350): MOV  -6(5),-(6)
CODE(354): ASL  (6)
CODE(356): MOV  -6(5),-(6)
CODE(362): ASL  (6)
CODE(364): MOV  #12400,2
CODE(370): ADD  (6)+,2
CODE(372): MOV  0(2),-(6)
CODE(376): MOV  #20104,2
CODE(402): MOV  0(2),-(6)
CODE(406): COM  (6)
CODE(410): BIC  (6)+,(6)
CODE(412): MOV  #12400,2
CODE(416): MOV  (6)+,0
```

```
1192        AST_WAL(ASTE#) := (AST_WAL(ASTE#) & PS_PROCESS_NOTMASK);

1193        /*    RESET SEMAPHORES FOR SEGMENT WITH EMPTY WRITE ACCESS LIST

1194        IF (AST_WAL(ASTE#) = 0) & ((AST_CPL(ASTE#) & WIRED_DOWN_MASK) ¬= WIRED_DOWN);
1195          THEN; /* FREE PROCESSES BLOCKED ON THE SEGMENT SEMAPHORE */

1196              CYCLE

FIXUP LOCATION 574 TO 8
1197              .... EXIT WHEN SMPR_COUNT(ASTE#) >= 0;

1198                    V(ASTE#);
FIXUP LOCATION 600 TO 32

1199              END;
```

```
CODE(420): ADD (6)+,2
CODE(422): MOV 0,0(2)
  |426
  |
*/|426
  |
  |544
  |544
  |
  |544
CODE(426): MOV -6(5),-(6)
CODE(432): ASL (6)
CODE(434): MOV #12400,2
CODE(440): ADD (6)+,2
CODE(442): MOV 0(2),-(6)
CODE(446): CLR -(6)
CODE(450): CMP (6)+,(6)+
CODE(452): BEQ 2
CODE(454): CLR -(6)
CODE(456): BR  2
CODE(460): MOV #1,-(6)
CODE(464): MOV -6(5),-(5)
CODE(470): ASL (6)
CODE(472): MOV #11400,2
CODE(476): ADD (6)+,2
CODE(500): MOV 0(2),-(6)
CODE(504): MOV #-100000,-(6)
CODE(510): COM (6)
CODE(512): BIC (6)+,(6)
CODE(514): MOV #-100000,-(6)
CODE(520): CMP (6)+,(6)+
CODE(522): BNE 2
CODE(524): CLR -(6)
CODE(526): BR  2
CODE(530): MOV #1,-(6)
CODE(534): BIT (6)+,(6)+
CODE(536): BNE 2
CODE(540): JMP 0(7)
CODE(544): MOV -6(5),-(6)
CODE(550): MOV #15534,2
CODE(554): ADD (6)+,2
CODE(556): MOVB0(2),1
CODE(562): MOV 1,-(6)
CODE(564): CLR -(6)
CODE(566): CMP (6)+,(6)+
CODE(570): BLE 2
CODE(572): JMP 0(7)
CODE(576): JMP 453(7)

  |602
CODE(602): MOV 5,-(6)
CODE(604): MOV (5),-(6)
CODE(606): CLR -(6)
CODE(610): MOV -6(5),-(6)
CODE(614): MOV -2(4),2
CODE(620): JSR 7,#130(2)
CODE(624): MOV 5,6
CODE(626): CLR (6)+
CODE(630): MOV (6)+,5
  |632

CODE(632): BR  -34
  |634
  |
CODE(634): MOV -6(5),-(6)
CODE(640): MOV #1,-(6)
```

```
1200              SMFB_COUNT(ASTE#) := 1;

1201              SMFB_POINTER(ASTE#) := 0;
FIXUP LOCATION 542 TO 136
1202          END;

1203          /*    AGE IF THIS IS THE LAST PROCESS TO DISCONNECT

1204          IF AST_CPL(ASTE#) = 0;

1205            THEN: AST_AGE_CHAIN(ASTE#) := AST_AGE_CHAIN(0);

1206              AST_AGE_CHAIN(0) := ASTE#;
FIXUP LOCATION 712 TO 64
1207          END;

1208          /*    PUT SEG# ON FREE CHAIN

1209          PS_SEG(SEG#) := PS_SEG(0);
```

```
CODE(644): MOV #15534,2
CODE(650): MOV (6)+,0
CODE(652): ADD (6)+,2
CODE(654): MOVB0,0(2)
  |660
CODE(660): MOV -6(5),-(6)
CODE(664): CLR -(6)
CODE(666): MOV #16140,2
CODE(672): MOV (6)+,0
CODE(674): ADD (6)+,2
CODE(676): MOVB0,0(2)
  |702

  |702
  |
*/|702
  |
  |734
CODE(702): MOV -6(5),-(6)
CODE(706): ASL (6)
CODE(710): MOV #11400,2
CODE(714): ADD (6)+,2
CODE(716): MOV 0(2),-(6)
CODE(722): CLR -(6)
CODE(724): CMP (6)+,(6)+
CODE(726): BEQ 2
CODE(730): JMP 0(7)
CODE(734): MOV -6(5),-(6)
CODE(740): ASL (6)
CODE(742): CLR -(6)
CODE(744): ASL (6)
CODE(746): MOV #12400,2
CODE(752): ADD (6)+,2
CODE(754): MOV 0(2),-(6)
CODE(760): MOV #12400,2
CODE(764): MOV (6)+,0
CODE(766): ADD (6)+,2
CODE(770): MOV 0,0(2)
  |774
CODE(774): CLR -(6)
CODE(776): ASL (6)
CODE(1000): MOV -6(5),-(6)
CODE(1004): MOV #12400,2
CODE(1010): MOV (6)+,0
CODE(1012): ADD (6)+,2
CODE(1014): MOV 0,0(2)
  |1020

  |1020
  |
*/|1020
  |
CODE(1020): MOV -6(5),-(6)
CODE(1024): ASL (6)
CODE(1026): CLR -(6)
CODE(1030): ASL (6)
CODE(1032): MOV #20200,2
CODE(1036): ADD (6)+,2
CODE(1040): MOV 0(2),-(6)
CODE(1044): MOV #20200,2
CODE(1050): MOV (6)+,0
CODE(1052): ADD (6)+,2
CODE(1054): MOV 0,0(2)
  |1060
CODE(1060): CLR -(6)
CODE(1062): ASL (6)
```

```
CODE(1064): MOV -8(3),-(6)
CODE(1070): BIS #-100000,(6)
CODE(1074): MOV #20200,2
CODE(1100): MOV (6)+,0
CODE(1102): ADD (6)+,2
CODE(1104): MOV 0,0(2)
```

```
1210         PS_SEG(0) := (SEG# | SEG_FLAG);

FIXUP LOCATION 14 TO 9
                  38 LINES WERE COMPILED, GENERATING 588 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA ACT          E
1211    _1_
```

```
 11110
CODE(1110): JMP $-10(5)

10
```

```
1212   DATA ACC(DASIO#, OFFSET) RETURNS (RC):                          10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                               10
1213   _I_
```

```
1210   PROGRAM ACT;
1215       DECLARE
1216           WORD (ASTE#, I, NEXT, HASH_VAL);

1217       /*    ALLOCATE AN ASTE ENTRY

1218       /*    HEAD OF FREE ASTE CHAIN IS IN ASTE 0

1219       IF AST_CHAIN(0) = 0;
1220         THEN:

1221           /*    NO FREE ASTE#'S - LOOK ON AGE CHAIN

1222           I := 0;

1223           CYCLE

1224               NEXT := AST_AGE_CHAIN(I);

PIXUP LOCATION 126 TO 4
1225           .... EXIT WHEN AST_AGE_CHAIN(NEXT) = 0;

1226               I := NEXT;
PIXUP LOCATION 132 TO 10

1227           END;
```

```
  |16
  |16
  |16
  |
*/|16
  |
*/|16
  |
  |46
  |46
  |
*/|46
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): CLR -(6)
CODE(20): ASL (6)
CODE(22): MOV #6400,2
CODE(26): ADD (6)+,2
CODE(30): MOV 0(2),-(6)
CODE(34): CLR -(6)
CODE(36): CMP (6)+,(6)+
CODE(40): BEQ 2
CODE(42): JMP 0(7)
CODE(46): CLR -(6)
CODE(50): MOV (6)+,-14(5)
  |54
  |
  |54
CODE(54): MOV -14(5),-(6)
CODE(60): ASL (6)
CODE(62): MOV #12400,2
CODE(66): ADD (6)+,2
CODE(70): MOV 0(2),-16(5)
  |76
CODE(76): MOV -16(5),-(6)
CODE(102): ASL (6)
CODE(104): MOV #12400,2
CODE(110): ADD (6)+,2
CODE(112): MOV 0(2),-(6)
CODE(116): CLR -(6)
CODE(120): CMP (6)+,(6)+
CODE(122): BEQ 2
CODE(124): JMP 0(7)
CODE(130): JMP 453(7)

  |134
CODE(134): MOV -16(5),-14(5)
  |142

CODE(142): BR  -34
  |144
  |
CODE(144): MOV 5,-(6)
CODE(146): MOV (5),-(6)
```

```
1228              DEACT(NEXT);
FIXUP LOCATION 44 TO 126
1229          END;

1230          /*    A FREE ASTE# EXISTS

1231          ASTE# := AST_CHAIN(0);

1232          /*    REMOVE THIS ASTE FROM THE FREE CHAIN AND AGE

1233          AST_CHAIN(0) := AST_CHAIN(ASTE#);

1234          AST_AGE_CHAIN(ASTE#) := AST_AGE_CHAIN(0);

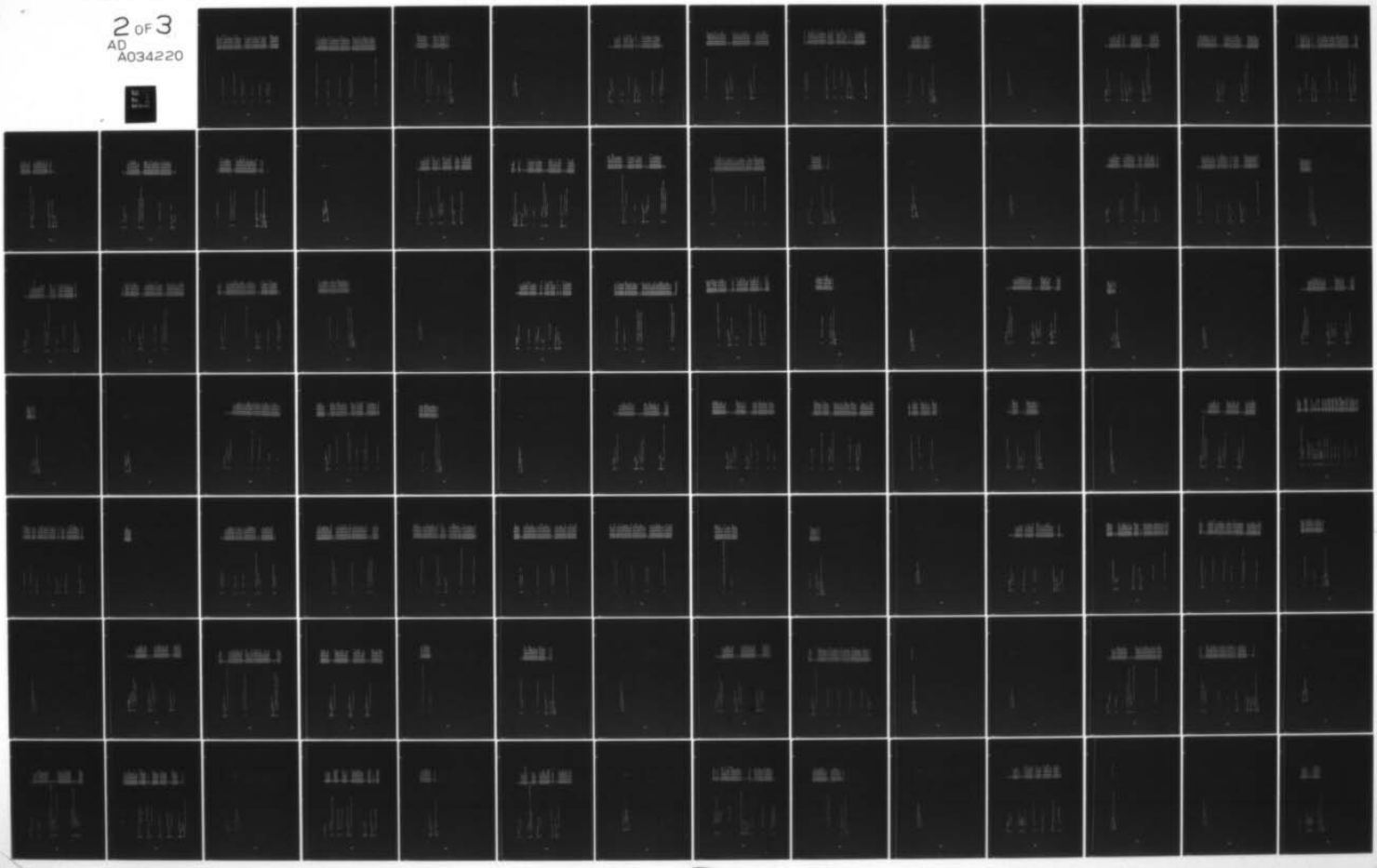
1235          AST_AGE_CHAIN(0) := ASTE#;

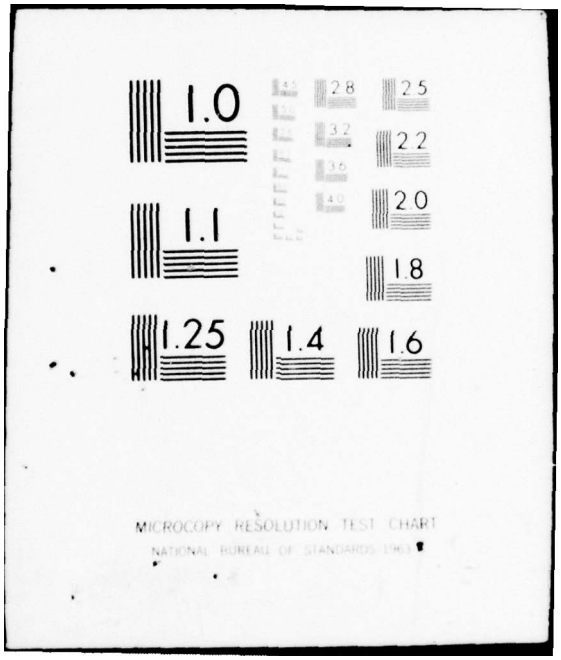
1236          /*    UPDATE HASH DATA BASE
```

```
CODE(150): CLR -(6)
CODE(152): MOV -16(5),-(6)
CODE(156): MOV -2(4),2
CODE(162): JSR 7,@120(2)
CODE(166): MOV 5,6
CODE(170): CLR (6)+
CODE(172): MOV (6)+,5
  |174
  |
  |174
  |
*/|174
  |
CODE(174): CLR -(6)
CODE(176): ASL (6)
CODE(200): MOV #6400,2
CODE(204): ADD (6)+,2
CODE(206): MOV 0(2),-12(5)
  |214
  |
*/|214
  |
CODE(214): CLR -(6)
CODE(216): ASL (6)
CODE(220): MOV -12(5),-(6)
CODE(224): ASL (6)
CODE(226): MOV #6400,2
CODE(232): ADD (6)+,2
CODE(234): MOV 0(2),-(6)
CODE(240): MOV #6400,2
CODE(244): MOV (6)+,0
CODE(246): ADD (6)+,2
CODE(250): MOV 0,0(2)
  |254
CODE(254): MOV -12(5),-(6)
CODE(260): ASL (6)
CODE(262): CLR -(6)
CODE(264): ASL (6)
CODE(266): MOV #12400,2
CODE(272): ADD (6)+,2
CODE(274): MOV 0(2),-(6)
CODE(300): MOV #12400,2
CODE(304): MOV (6)+,0
CODE(306): ADD (6)+,2
CODE(310): MOV 0,0(2)
  |314
CODE(314): CLR -(6)
CODE(316): ASL (6)
CODE(320): MOV -12(5),-(6)
CODE(324): MOV #12400,2
CODE(330): MOV (6)+,0
CODE(332): ADD (6)+,2
CODE(334): MOV 0,0(2)
  |340
  |
*/|340
  |
CODE(340): CLR -(6)
CODE(342): MOV 5,-(6)
CODE(344): MOV (5),-(6)
CODE(346): CLR -(6)
CODE(350): MOV -6(5),-(6)
CODE(354): ASL (6)
CODE(356): MOV #60400,2
CODE(362): ADD (6)+,2
CODE(364): MOV 0(2),-(6)
```

AD-A034 220 MITRE CORP BEDFORD MASS F/G 9/2
COMPUTER PROGRAM SPECIFICATION FOR THE SECURITY KERNEL FOR THE --ETC(U)
OCT 76 S R HARPER F19628-77-C-0001
UNCLASSIFIED MTR-3178-VOL-2 ESD-TR-76-288-VOL-2 NL

2 OF 3
AD
A034220




1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963

```
1237        HASH_VAL := PREHASH(DIR_DISK(OFFSET));

1238        AST_CHAIN(ASTP#) := HASH_TABLE(HASH_VAL);

1239        HASH_TABLE(HASH_VAL) := ASTP#;
1240        /*    CLEAN UP AST ENTRY

1241        AST_ADD(ASTP#) := 0;

1242        AST_DES_COUNT(ASTP#) := 0;

1243        AST_CPL(ASTP#) := 0;
1244        /*    FILL IN AST ENTRY
```

```
CODE(370): MOV -2(4),2
CODE(374): JSR 7,@200(2)
CODE(400): MOV 5,6
CODE(402): CLR (6)+
CODE(404): MOV (6)+,5
CODE(406): MOV (6)+,-20(5)
  |412
CODE(412): MOV -12(5),-(6)
CODE(416): ASL (6)
CODE(420): MOV -20(5),-(6)
CODE(424): ASL (6)
CODE(426): MOV #3400,2
CODE(432): ADD (6)+,2
CODE(434): MOV 0(2),-(6)
CODE(440): MOV #6400,2
CODE(444): MOV (6)+,0
CODE(446): ADD (6)+,2
CODE(450): MOV 0,0(2)
  |454
CODE(454): MOV -20(5),-(6)
CODE(460): ASL (6)
CODE(462): MOV -12(5),-(6)
CODE(466): MOV #3400,2
CODE(472): MOV (6)+,0
CODE(474): ADD (6)+,2
CODE(476): MOV 0,0(2)
  |502
  |
*/|502
  |
CODE(502): MOV -12(5),-(6)
CODE(506): ASL (6)
CODE(510): CLR -(6)
CODE(512): MOV #7400,2
CODE(516): MOV (6)+,0
CODE(520): ADD (6)+,2
CODE(522): MOV 0,0(2)
  |526
CODE(526): MOV -12(5),-(6)
CODE(532): ASL (6)
CODE(534): CLR -(6)
CODE(536): MOV #10400,2
CODE(542): MOV (6)+,0
CODE(544): ADD (6)+,2
CODE(546): MOV 0,0(2)
  |552
CODE(552): MOV -12(5),-(6)
CODE(556): ASL (6)
CODE(560): CLR -(6)
CODE(562): MOV #11400,2
CODE(566): MOV (6)+,0
CODE(570): ADD (6)+,2
CODE(572): MOV 0,0(2)
  |576
  |
*/|576
  |
CODE(576): MOV -12(5),-(6)
CODE(602): MOV -6(5),-(6)
CODE(606): MOV #60000,2
CODE(612): ADD (6)+,2
CODE(614): MOVB0(2),1
CODE(620): MOV 1,-(6)
CODE(622): MOV #4400,2
CODE(626): MOV (6)+,0
CODE(630): ADD (6)+,2
```

```
1245        AST_CLASS(ASTP#) := DIR_CLASS(OFFSET); /* SETS TYPE, STATUS */
1246        AST_CAT(ASTP#) := DIR_CAT(OFFSET);
1247        AST_DISK(ASTP#) := DIR_DISK(OFFSET);
1248        AST_SIZE(ASTP#) := DIR_SIZE(OFFSET);
1249        IF (DIR_STATUS(OFFSET) & DIR_STATUS_MASK) = DIR_UNINITIALIZED;
1250          THEN: DIR_STATUS(OFFSET) := (DIR_STATUS(OFFSET) & DIR_STATUS_NOTMASK);
```

```
CODE(632): MOVB0,0(2)
  |636
CODE(636): MOV -12(5),-(6)
CODE(642): ASL (6)
CODE(644): MOV -6(5),-(6)
CODE(650): ASL (6)
CODE(652): MOV #60200,2
CODE(656): ADD (6)+,2
CODE(660): MOV 0(2),-(6)
CODE(664): MOV #13400,2
CODE(670): MOV (6)+,0
CODE(672): ADD (6)+,2
CODE(676): MOV 0,0(2)
  |700
CODE(700): MOV -12(5),-(6)
CODE(704): ASL (6)
CODE(706): MOV -6(5),-(6)
CODE(712): ASL (6)
CODE(714): MOV #60400,2
CODE(720): ADD (6)+,2
CODE(722): MOV 0(2),-(6)
CODE(726): MOV #5400,2
CODE(732): MOV (6)+,0
CODE(734): ADD (6)+,2
CODE(736): MOV 0,0(2)
  |742
CODE(742): MOV -12(5),-(6)
CODE(746): MOV -6(5),-(6)
CODE(752): ASL (6)
CODE(754): MOV #60600,2
CODE(760): ADD (6)+,2
CODE(762): MOV 0(2),-(6)
CODE(766): MOV #5000,2
CODE(772): MOV (6)+,0
CODE(774): ADD (6)+,2
CODE(776): MOVB0,0(2)
  |1002
  |
  |1048
CODE(1002): MOV -6(5),-(6)
CODE(1006): MOV #60000,2
CODE(1012): ADD (6)+,2
CODE(1014): MOVB0(2),1
CODE(1020): MOV 1,-(6)
CODE(1022): MOV #100,-(6)
CODE(1026): COM (6)
CODE(1030): BIC (6)+,(6)
CODE(1032): CMP #100,(6)+
CODE(1036): BEQ 2
CODE(1040): JMP 0(7)
CODE(1044): MOV -6(5),-(6)
CODE(1050): MOV -6(5),-(6)
CODE(1054): MOV #60000,2
CODE(1060): ADD (6)+,2
CODE(1062): MOVB0(2),1
CODE(1066): MOV 1,-(6)
CODE(1070): MOV #277,-(6)
CODE(1074): COM (6)
CODE(1076): BIC (6)+,(6)
CODE(1100): MOV #60000,2
CODE(1104): MOV (6)+,0
CODE(1106): ADD (6)+,2
CODE(1110): MOVB0,0(2)
  |1114
  |
```

```
1251              /*    DIRECTORY HAS BEEN WRITTEN INTO - MUST SET CHANGE BIT

1252              AST_CHANGE(BASE#) := (AST_CHANGE(BASE#) | AST_CHANGED);
FIXUP LOCATION 1042 TO 114
1253          END;

1254          /*    INITIALIZE SEMAPHORE ASSOCIATED WITH SEGMENT

1255          SMPR_COUNT(ASTE#) := 1;

1256          SMPR_POINTER(ASTE#) := 0;

1257          TC := AST#;

FIXUP LOCATION 14 TO 12
                45 LINES WERE COMPILED, GENERATING 672 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA REACT        T
1258    _!_
```

```
*/|1114
  |
CODE(1114): MOV -4(5),-(6)
CODE(1120): MOV -4(5),-(6)
CODE(1124): MOV #4400,2
CODE(1130): ADD (6)+,2
CODE(1132): MOVB0(2),1
CODE(1136): MOV 1,-(6)
CODE(1140): BIS #40,(6)
CODE(1144): MOV #4400,2
CODE(1150): MOV (6)+,0
CODE(1152): ADD (6)+,2
CODE(1154): MOVB0,0(2)
  |1160

  |1160
  |
*/|1160
  |
CODE(1160): MOV -12(5),-(6)
CODE(1164): MOV #1,-(6)
CODE(1170): MOV #15534,2
CODE(1174): MOV (6)+,0
CODE(1176): ADD (6)+,2
CODE(1200): MOVB0,0(2)
  |1204
CODE(1204): MOV -12(5),-(6)
CODE(1210): CLR -(6)
CODE(1212): MOV #16140,2
CODE(1216): MOV (6)+,0
CODE(1220): ADD (6)+,2
CODE(1222): MOVB0,0(2)
  |1226
CODE(1226): MOV -12(5),4(5)
  |1234
CODE(1234): JMP @-10(5)
```

```
1259   DATA DEACT(ASCII0);
                    2 LINES WERE COMPILED.                                  10
NO ERRORS WERE DETECTED.
1260   _0_                                                                  10
```

```
1261   PROGRAM DEACT;
1262         DECLARE
1263               WORD (HASH_VAL, INDEX, NEXT);

1264         /*    REMOVE FROM AGE CHAIN




1265         INDEX := ^;

1266         CYCLE




1267               NEXT := AST_AGE_CHAIN(INDEX);



FIXUP LOCATION 62 TO 4
1268         .... EXIT WHEN NEXT = ASTE0;

1269               INDEX := NEXT;
FIXUP LOCATION 66 TO 10
1270         END;




1271         AST_AGE_CHAIN(INDEX) := AST_AGE_CHAIN(ASTE0);



1272         AST_AGE_CHAIN(ASTE0) := ^;

1273         /*    CAN NOT DEACTIVATE AN UNINITIALIZED SEGMENT
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(6)
CODE(2):  MOV 6,5
CODE(4):  ADD #4,5
CODE(10):  MOV 7,(5)+
CODE(12):  SUB #0,6
CODE(16):  CLR -(6)
CODE(20):  MOV (6)+,-12(5)
  |24
  |
  |24
CODE(24):  MOV -12(5),-(6)
CODE(30):  ASL (6)
CODE(32):  MOV #12400,2
CODE(36):  ADD (6)+,2
CODE(40):  MOV 0(2),-14(5)
  |46
CODE(46):  MOV -14(5),-(6)
CODE(52):  CMP -4(5),(6)+
CODE(56):  BEQ 2
CODE(60):  JMP 0(7)
CODE(64):  JMP #53(7)

  |70
CODE(70):  MOV -14(5),-12(5)
  |76

CODE(76):  BR  -26
  |100
  |
CODE(100):  MOV -12(5),-(6)
CODE(104):  ASL (6)
CODE(106):  MOV -4(5),-(6)
CODE(112):  ASL (6)
CODE(114):  MOV #12400,2
CODE(120):  ADD (6)+,2
CODE(122):  MOV 0(2),-(6)
CODE(126):  MOV #12400,2
CODE(132):  MOV (6)+,0
CODE(134):  ADD (6)+,2
CODE(136):  MOV 0,0(2)
  |142
CODE(142):  MOV -4(5),-(6)
CODE(146):  ASL (6)
CODE(150):  CLR -(6)
CODE(152):  MOV #12400,2
CODE(156):  MOV (6)+,0
CODE(160):  ADD (6)+,2
CODE(162):  MOV 0,0(2)
  |166
  |
*/|166
  |
```

```
1274        IF (AST_STATUS(ASTE#) & AST_STATUS_MASK) = AST_UNINITIALIZED;
1275          THEN: SWAPIN(ASTE#);
FIXUP LOCATION 224 TO 30
1276        END;
1277        /*    SWAPOUT IF IN MAIN MEMORY
1278        IF AST_ADR(ASTE#) >= 0;
1279          THEN: SWAPOUT(ASTE#);
FIXUP LOCATION 310 TO 30
1280        END;
1281        /*    REMOVE THIS ASTE FROM HASH TABLE OR HASH CHAIN
```

```
  1230
CODE(166): MOV -4(5),-(6)
CODE(172): MOV #4400,2
CODE(176): ADD (6)+,2
CODE(200): MOVB0(2),1
CODE(204): MOV 1,-(6)
CODE(206): MOV #100,-(6)
CODE(212): COM (6)
CODE(214): BIC (6)+,(6)
CODE(216): CMP #100,(6)+
CODE(222): BEQ 2
CODE(224): JMP 0(7)
CODE(230): MOV 5,-(6)
CODE(232): MOV (5),-(6)
CODE(234): CLR -(6)
CODE(236): MOV -4(5),-(6)
CODE(242): MOV -2(4),2
CODE(246): JSR 7,@134(2)
CODE(252): MOV 5,6
CODE(254): CLR (6)+
CODE(256): MOV (6)+,5
  1260

  1260
  |
*/1260
  |
  1312
CODE(260): MOV -4(5),-(6)
CODE(264): ASL (6)
CODE(266): MOV #7400,2
CODE(272): ADD (6)+,2
CODE(274): MOV 0(2),-(6)
CODE(300): CLR -(6)
CODE(302): CMP (6)+,(6)+
CODE(304): BNE 2
CODE(306): JMP 0(7)
CODE(312): MOV 5,-(6)
CODE(314): MOV (5),-(6)
CODE(316): CLR -(6)
CODE(320): MOV -4(5),-(6)
CODE(324): MOV -2(4),2
CODE(330): JSR 7,@140(2)
CODE(334): MOV 5,6
CODE(336): CLR (6)+
CODE(340): MOV (6)+,5
  1342

  1342
  |
*/1342
  |
CODE(342): CLR -(6)
CODE(344): MOV 5,-(6)
CODE(346): MOV (5),-(6)
CODE(350): CLR -(6)
CODE(352): MOV -4(5),-(6)
CODE(356): ASL (6)
CODE(360): MOV #5400,2
CODE(364): ADD (6)+,2
CODE(366): MOV 0(2),-(6)
CODE(372): MOV -2(4),2
CODE(376): JSR 7,@200(2)
CODE(402): MOV 5,6
CODE(404): CLR (6)+
CODE(406): MOV (6)+,5
```

```
1282        HASH_VAL := PREHASH(AST_DISK(ASTE#));
1283        IF HASH_TABLE(HASH_VAL) = ASTE#;
1284          THEN: HASH_TABLE(HASH_VAL) := AST_CHAIN(ASTE#);
FIXUP LOCATION 444 TO 46
1285          ELSE: HASH_VAL := HASH_TABLE(HASH_VAL);
1286             CYCLE
1287                 NEXT := AST_CHAIN(HASH_VAL);
FIXUP LOCATION 574 TO 4
1288             .... EXIT WHEN NEXT = ASTE#;
1289                 HASH_VAL := NEXT;
FIXUP LOCATION 600 TO 10
1290             END;
1291           AST_CHAIN(HASH_VAL) := AST_CHAIN(ASTE#);
FIXUP LOCATION 512 TO 140
```

```
CODE(410): MOV  (6)+,-10(5)
  |414
  |
  |446
CODE(414): MOV -10(5),-(6)
CODE(420): ASL  (6)
CODE(422): MOV #3400,2
CODE(426): ADD  (6)+,2
CODE(430): MOV 0(2),-(6)
CODE(434): CMP -4(5),(6)+
CODE(440): BEQ 2
CODE(442): JMP 0(7)
CODE(446): MOV -10(5),-(6)
CODE(452): ASL  (6)
CODE(454): MOV -4(5),-(6)
CODE(460): ASL  (6)
CODE(462): MOV #6400,2
CODE(466): ADD  (6)+,2
CODE(470): MOV 0(2),-(6)
CODE(474): MOV #3400,2
CODE(500): MOV  (6)+,0
CODE(502): ADD  (6)+,2
CODE(504): MOV 0,0(2)
  |510
CODE(510): JMP 0(7)
CODE(514): MOV -10(5),-(6)
CODE(520): ASL  (6)
CODE(522): MOV #3400,2
CODE(526): ADD  (6)+,2
CODE(530): MOV 0(2),-10(5)
  |536
  |
  |536
CODE(536): MOV -10(5),-(6)
CODE(542): ASL  (6)
CODE(544): MOV #6400,2
CODE(550): ADD  (6)+,2
CODE(552): MOV 0(2),-14(5)
  |560
CODE(560): MOV -14(5),-(6)
CODE(564): CMP -4(5),(6)+
CODE(570): BEQ 2
CODE(572): JMP 0(7)
CODE(576): JMP 453(7)

  |602
CODE(602): MOV -14(5),-10(5)
  |610

CODE(610): BR  -26
  |612
  |
CODE(612): MOV -10(5),-(6)
CODE(616): ASL  (6)
CODE(620): MOV -4(5),-(6)
CODE(624): ASL  (6)
CODE(626): MOV #6400,2
CODE(632): ADD  (6)+,2
CODE(636): MOV 0(2),-(6)
CODE(640): MOV #6400,2
CODE(644): MOV  (6)+,0
CODE(646): ADD  (6)+,2
CODE(650): MOV 0,0(2)
  |654
```

```
1292        END;
1293        /*    ADD THIS AST ENTRY TO THE FREE CHAIN
1294        AST_CHAIN(ASTE#) := AST_CHAIN(0);
1295        AST_CHAIN(0) := ASTE#;
FIXUP LOCATION 14 TO 6
            36 LINES WERE COMPILED, GENERATING 484 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA ENABLE      E
1296   _!_
```

```
  1654
  |
*/1654
  |
CODE(654): MOV -4(5),-(6)
CODE(660): ASL (6)
CODE(662): CLR -(6)
CODE(664): ASL (6)
CODE(666): MOV #6400,2
CODE(672): ADD (6)+,2
CODE(674): MOV 0(2),-(6)
CODE(700): MOV #6400,2
CODE(704): MOV (6)+,0
CODE(706): ADD (6)+,2
CODE(710): MOV 0,0(2)
  1714
CODE(714): CLR -(6)
CODE(716): ASL (6)
CODE(720): MOV -4(5),-(6)
CODE(724): MOV #6400,2
CODE(730): MOV (6)+,0
CODE(732): ADD (6)+,2
CODE(734): MOV 0,0(2)
  1740
CODE(740): JMP @-6(5)
```

```
1297   DATA VARBL"(ASIT*, BEG*) RETURNS (PC);                                10
                  2 LEVEL PROC COMPILED.
NO ERRORS WERE DETECTED.                                                     10
1298   _I_
```

```
1299    PROGRAM ENABLE;
1300        DECLARE
1301            WORD (P_REG#, MODE, REG_ADR, INDEX, NEXT);

1302        IF REG# > CROSS_REG#;
                                                                                    |16
                                                                                    |16
                                                                                    |16
                                                                                    |
                                                                                    |34
                                                                                CODE(0):    2,-(0)
                                                                                CODE(2): MOV 6,5
                                                                                CODE(6): ADD #6,5
                                                                                CODE(10): MOV 7,(5)+
                                                                                CODE(12): SUB #0,6
                                                                                CODE(16): MOV -6(5),-(6)
                                                                                CODE(22): CMP #7,(6)+
                                                                                CODE(26): BLT 2
                                                                                CODE(30): JMP 0(7)
                                                                                CODE(34): MOV -6(5),-(6)
                                                                                CODE(40): ADD #2570,(6)
                                                                                CODE(44): MOV (6)+,-12(5)
                                                                                  |50

1303            THEN: P_REG# := REG# * REG_CONSTANT;                            CODE(50): JMP 0(7)
FIXUP LOCATION 32 TO 20                                                         CODE(54): MOV -6(5),-12(5)
                                                                                  |62

1304            ELSE: P_REG# := REG#;                                             |62
FIXUP LOCATION 52 TO 6                                                              |
1305        END;                                                                *//162
                                                                                    |
1306        /*    REGISTER MUST BE FREE                                             |114
                                                                                CODE(62): MOV -6(5),-(6)
1307        IF PS_SAR(REG#) ~= 0;                                               CODE(66): ASL (6)
                                                                                CODE(70): MOV #20040,2
                                                                                CODE(74): ADD (6)+,2
                                                                                CODE(76): MOV 0(2),-(6)
                                                                                CODE(102): CLR -(6)
                                                                                CODE(104): CMP (6)+,(6)+
                                                                                CODE(106): BNE 2
                                                                                CODE(110): JMP 0(7)
                                                                                CODE(114): MOV #-2,4(5)
                                                                                CODE(122): JMP #-10(5)
                                                                                  |126
1308            THEN:                                                           |126
1309    ....        RETURN WITH ERR_FLAG;                                         |126
FIXUP LOCATION 112 TO 12                                                            |
1310        END;                                                                *//126
                                                                                    |
1311        /*    SPACE IN USER'S MEMORY MUST BE AVAILABLE                          |252
                                                                                    |252
1312        IF ((AST_CPL(ASTE#) & WIRED_DOWN_MASK) ~= WIRED_DOWN) & (AST_SIZE(ASTE#) >  CODE(126): MOV -4(5),-(6)
1313            PS_MEM_QUOTA);                                                  CODE(132): ASL (6)
                                                                                CODE(134): MOV #11400,2
                                                                                CODE(140): ADD (6)+,2
                                                                                CODE(142): MOV 0(2),-(6)
                                                                                CODE(146): MOV #-100000,-(6)
                                                                                CODE(152): COM (6)
                                                                                CODE(154): BIC (6)+,(6)
                                                                                CODE(156): CMP #-100000,(6)+
                                                                                CODE(162): BNE 2
```

```
1314          THEN:
1315  ....      RETURN WITH PRP_FLAG;
FIXUP LOCATION 250 TO 12
1316          END;

1317          /*    SWAPIN IF NECESSARY

1318          IF AST_ADR(ASTE#) = 0;
```

```
1319          THEN: SWAPIN(ASTE#);
FIXUP LOCATION 314 TO 30
1320          END;

1321          /*    REMOVE FROM SWAP CHAIN IF NECESSARY

1322          IF (AST_UNLOCK(ASTE#) & AST_UNLOCK_MASK) = AST_UNLOCK_FLAG;
```

```
CODE(164): CLR -(6)
CODE(166): BR  2
CODE(170): MOV #1,-(6)
CODE(174): MOV -4(5),-(6)
CODE(200): MOV #5000,2
CODE(204): ADD (6)+,2
CODE(206): MOVB0(2),1
CODE(212): MOV 1,-(6)
CODE(214): MOV #20116,2
CODE(220): MOVB0(2),1
CODE(224): MOV 1,-(6)
CODE(226): CMP (6)+,(6)+
CODE(230): BLT 2
CODE(232): CLR -(6)
CODE(234): BR  2
CODE(236): MOV #1,-(6)
CODE(242): BIT (6)+,(6)+
CODE(244): BNE 2
CODE(246): JMP 0(7)
CODE(252): MOV #-2,4(5)
CODE(260): JMP #-10(5)
  |264
  |264

  |264
  |
*/|264
  |
  |316
CODE(264): MOV -4(5),-(6)
CODE(270): ASL (6)
CODE(272): MOV #7400,2
CODE(276): ADD (6)+,2
CODE(300): MOV 0(2),-(6)
CODE(304): CLR -(6)
CODE(306): CMP (6)+,(6)+
CODE(310): BEQ 2
CODE(312): JMP 0(7)
CODE(316): MOV 5,-(6)
CODE(320): MOV (5),-(6)
CODE(322): CLR -(6)
CODE(324): MOV -4(5),-(6)
CODE(330): MOV -2(4),2
CODE(334): JSR 7,#134(2)
CODE(340): MOV 5,6
CODE(342): CLR (6)+
CODE(344): MOV (6)+,5
  |346

  |346
  |
*/|346
  |
  |410
CODE(346): MOV -4(5),-(6)
CODE(352): MOV #4400,2
CODE(356): ADD (6)+,2
CODE(360): MOVB0(2),1
CODE(364): MOV 1,-(6)
CODE(366): MOV #20,-(6)
CODE(372): COM (6)
CODE(374): BIC (6)+,(6)
CODE(376): CMP #20,(6)+
CODE(402): BEQ 2
CODE(404): JMP 0(7)
CODE(410): CLR -(6)
```

```
1323          THEN: INDEX := 0;
1324              CYCLE
1325                  NEXT := AST_SWAP_CHAIN(INDEX);
FIXUP LOCATION 454 TO 4
1326              .... EXIT WHEN NEXT = ASTE#;
1327                  INDEX := NEXT;
FIXUP LOCATION 460 TO 10
1328              END;
1329              AST_SWAP_CHAIN(INDEX) := AST_SWAP_CHAIN(ASTE#);
1330              AST_DES_COUNT(ASTE#) := 0;
1331              AST_UNLOCK(ASTE#) := (AST_UNLOCK(ASTE#) & AST_LOCK_MASK);
FIXUP LOCATION 406 TO 220
1332          END;
1333          /*    DETERMINE TYPE OF ACCESS PERMITTED
1334          IF (AST_TYPE(ASTE#) & AST_TYPE_MASK) = AST_TYPE_DIRECTORY;
```

```
CODE(412): MOV (6)+,-20(5)
  |416
  |
  |416
CODE(416): MOV -20(5),-(6)
CODE(422): ASL (6)
CODE(424): MOV #10400,2
CODE(430): ADD (6)+,2
CODE(432): MOV 0(2),-22(5)
  |440
CODE(440): MOV -22(5),-(6)
CODE(444): CMP -4(5),(6)+
CODE(450): BEQ 2
CODE(452): JMP 0(7)
CODE(456): JMP 453(7)

  |462
CODE(462): MOV -22(5),-20(5)
  |470

CODE(470): BR  -26
  |472
  |
CODE(472): MOV -20(5),-(6)
CODE(476): ASL (6)
CODE(500): MOV -4(5),-(6)
CODE(504): ASL (6)
CODE(506): MOV #10400,2
CODE(512): ADD (6)+,2
CODE(514): MOV 0(2),-(6)
CODE(520): MOV #10400,2
CODE(524): MOV (6)+,0
CODE(526): ADD (6)+,2
CODE(530): MOV 0,0(2)
  |534
CODE(534): MOV -4(5),-(6)
CODE(540): ASL (6)
CODE(542): CLR -(6)
CODE(544): MOV #10400,2
CODE(550): MOV (6)+,0
CODE(552): ADD (6)+,2
CODE(554): MOV 0,0(2)
  |560
CODE(560): MOV -4(5),-(6)
CODE(564): MOV -4(5),-(6)
CODE(570): MOV #4400,2
CODE(574): ADD (6)+,2
CODE(576): MOVB0(2),1
CODE(602): MOV 1,-(6)
CODE(604): MOV #357,-(6)
CODE(610): COM (6)
CODE(612): BIC (6)+,(6)
CODE(614): MOV #4400,2
CODE(620): MOV (6)+,0
CODE(622): ADD (6)+,2
CODE(624): MOVB0,0(2)
  |630

  |630
  |
*/|630
  |
  |672
CODE(630): MOV -4(5),-(6)
CODE(634): MOV #4400,2
CODE(640): ADD (6)+,2
```

```
1335          THEN: MODE := 0; /* DIRECTORY ACCESSES MUST BE INTERPRETIVE */
FIXUP LOCATION 670 TO 12
1336          ELSE:
1337            IF (AST_WAL(ASTE#) & PS_PROCESS_MASK) = 0;
```

```
1338              THEN: MODE := SDR_READ_ACCESS;
FIXUP LOCATION 750 TO 12

1339              ELSE: MODE := SDR_WRITE_ACCESS;
FIXUP LOCATION 762 TO 6
1340            END;

FIXUP LOCATION 792 TO 66
```

```
CODE(642): MOVB0(2),1
CODE(646): MOV 1,-(6)
CODE(650): MOV #200,-(6)
CODE(654): COM (6)
CODE(656): BIC (6)+,(6)
CODE(660): CMP #200,(6)+
CODE(664): BEQ 2
CODE(666): JMP 0(7)
CODE(672): CLR -(6)
CODE(674): MOV (6)+,-14(5
  |700

  |704
  |752
CODE(700): JMP 0(7)
CODE(704): MOV -4(5),-(6)
CODE(710): ASL (6)
CODE(712): MOV #12400,2
CODE(716): ADD (6)+,2
CODE(720): MOV 0(2),-(6)
CODE(724): MOV #20102,2
CODE(730): MOV 0(2),-(6)
CODE(734): COM (6)
CODE(736): BIC (6)+,(6)
CODE(740): CLR -(6)
CODE(742): CMP (6)+,(6)+
CODE(748): BEQ 2
CODE(746): JMP 0(7)
CODE(752): MOV #2,-14(5)
  |760

CODE(760): JMP 0(7)
CODE(764): MOV #6,-14(5)
  |772

  |772
  |
```

```
1341        END;
1342        /*    LOAD SEGMENT DESCRIPTOR
1343        INLINE(SPLHIGH);
1344        LSD(ASTE#, PS_SDR_ADR + REG# + REG#, MODE);
1345        /*    IF THIS IS THE CURRENT PROCESS LOAD HARDWARE REGS ALSO
1346        IF PS_CURRENT_PROCESS = THE_CURRENT_PROCESS;
1347          THEN: SDR(P_REG#) := PS_SDR(REG#);
1348              SAR(P_REG#) := PS_SAR(REG#);
FIXUP LOCATION 1070 TO 104
1349        END;
1350        INLINE(SPLLOW);
```

```
  |772
  |
*/|772
  |
  |774
CODE(774): MOV 5,-(6)
CODE(776): MOV (5),-(6)
CODE(1000): CLR -(6)
CODE(1002): MOV -4(5),-(6)
CODE(1006): MOV #20000,-(6)
CODE(1012): ADD -6(5),(6)
CODE(1016): ADD -6(5),(6)
CODE(1022): MOV -14(5),-(6)
CODE(1026): MOV -2(4),2
CODE(1032): JSR 7,#154(2)
CODE(1036): MOV 5,6
CODE(1040): CLR (6)+
CODE(1042): MOV (6)+,5
  |1044
  |
*/|1044
  |
  |1072
CODE(1044): MOV #20100,2
CODE(1050): MOV 0(2),-(6)
CODE(1054): MOV #16544,2
CODE(1060): CMP 0(2),(6)+
CODE(1064): BEQ 2
CODE(1066): JMP 0(7)
CODE(1072): MOV -12(5),-(6)
CODE(1076): ASL (6)
CODE(1100): MOV -6(5),-(6)
CODE(1104): ASL (6)
CODE(1106): MOV #20000,2
CODE(1112): ADD (6)+,2
CODE(1114): MOV 0(2),-(6)
CODE(1120): MOV #-5600,2
CODE(1124): MOV (6)+,0
CODE(1126): ADD (6)+,2
CODE(1130): MOV 0,0(2)
  |1134
CODE(1134): MOV -12(5),-(6)
CODE(1140): ASL (6)
CODE(1142): MOV -6(5),-(6)
CODE(1146): ASL (6)
CODE(1150): MOV #20040,2
CODE(1154): ADD (6)+,2
CODE(1156): MOV 0(2),-(6)
CODE(1162): MOV #-5540,2
CODE(1166): MOV (6)+,0
CODE(1170): ADD (6)+,2
CODE(1172): MOV 0,0(2)
  |1176

  |1176
  |
  |1200
  |
```

```
1351        /*    INCREMENT DESCRIPTOR COUNT

1352        AST_DES_COUNT(ASTE#) := AST_DES_COUNT(ASTE#) + 1;
1353        /*    ADJUST USER'S QUOTA
1354        IF (AST_CPL(ASTE#) & WIRED_DOWN_MASK) ¬= WIRED_DOWN;

1355            THEN; PS_MEM_QUOTA := PS_MEM_QUOTA - AST_SIZE(ASTE#);
FIXUP LOCATION 1306 TO 44
1356        END;

1357        RC := OK_FLAG;

FIXUP LOCATION 14 TO 12
            60 LINES WERE COMPILED, GENERATING 758 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA DISABLE     E
1358  _!_
```

```
*/!1200
  !
CODE(1200): MOV -4(5),-(6)
CODE(1204): ASL (6)
CODE(1206): MOV -4(5),-(6)
CODE(1212): ASL (6)
CODE(1214): MOV #10400,2
CODE(1220): ADD (6)+,2
CODE(1222): MOV 0(2),-(6)
CODE(1226): ADD #1,(6)
CODE(1232): MOV #10400,2
CODE(1236): MOV (6)+,0
CODE(1240): ADD (6)+,2
CODE(1242): MOV 0,0(2)
  !1246
  !
*/!1246
  !
  !1310
CODE(1246): MOV -4(5),-(6)
CODE(1252): ASL (6)
CODE(1254): MOV #11400,2
CODE(1260): ADD (6)+,2
CODE(1262): MOV 0(2),-(6)
CODE(1266): MOV #-100000,-(6)
CODE(1272): COM (6)
CODE(1274): BIC (6)+,(6)
CODE(1276): CMP #-100000,(6)+
CODE(1302): BNE 2
CODE(1304): JMP 0(7)
CODE(1310): MOV -4(5),-(6)
CODE(1314): MOV #5000,2
CODE(1320): ADD (6)+,2
CODE(1322): MOVB0(2),1
CODE(1326): MOV 1,-(6)
CODE(1330): MOV #20116,2
CODE(1334): MOVB0(2),3
CODE(1340): SUB 3,(6)
CODE(1342): NEG (6)
CODE(1344): MOV #20116,2
CODE(1350): MOVB(6)+,0(2)
  !1354

  !1354
  !
CODE(1354): MOV #-1,4(5)
  !1362
CODE(1362): JMP #-10(5)
```

```
1359   DATA DISABLE(REG0);                                                  10
1360         DECLARE                                                        10
1361               WORD (BLOCK0, ASTF0);                                    10
                 4 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                    10
1362   _!_
```

```
1363   PROGRAM DISABLE;
1364         /*    TRANSLATE REG# TO ASTE# (AND CHECK FOR DESCRIPTOR IN REGISTER)
1365         BLOCK# := PS_SAR(REG#);
1366         IF BLOCK# ¬= 0;
1367           THEN: INLINE(ASR, BLOCK#); /* THRIR BLOCKS TO MINE */
1368                 INLINE(ASR, BLOCK#);
1369                 IF BLOCK# < FWD_BLOCK#;
1370                   THEN: ASTE# := NBT_ASTF(BLOCK#);
FIXUP LOCATION 102 TO 26
1371                   ELSE:
FIXUP LOCATION 144 TO 2
1372                        DO ASTE# := ASTE#_MIN TO ASTE#_MAX;
```

```
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #20040,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-10(5)
  |40
  |
  |56
CODE(40): MOV -10(5),-(6)
CODE(44): CLR -(6)
CODE(46): CMP (6)+,(6)+
CODE(50): BNE 2
CODE(52): JMP 0(7)
CODE(56): ASR -10(5)
  |62
CODE(62): ASR -10(5)
  |66
  |
  |104
CODE(66): MOV -10(5),-(6)
CODE(72): CMP #1740,(6)+
CODE(76): BGT 2
CODE(100): JMP 0(7)
CODE(104): MOV -10(5),-(6)
CODE(110): ASL (6)
CODE(112): MOV #400,2
CODE(116): ADD (6)+,2
CODE(120): MOV 0(2),-12(5)
  |126

  |132
  |
CODE(126): JMP 0(7)
CODE(132): MOV #377,-(6)
CODE(136): MOV #1,-(6)
CODE(142): JMP 453(7)
CODE(146): INC (6)

  |166
CODE(150): CMP (6),2(6)
CODE(154): BLE 2
CODE(156): JMP 453(7)
CODE(162): MOV (6),-12(5)
CODE(166): MOV -12(5),-(6)
CODE(172): ASL (6)
CODE(174): MOV #7400,2
CODE(200): ADD (6)+,2
CODE(202): MOV 0(2),-(6)
CODE(206): CMP -10(5),(6)+
```

```
FIXUP LOCATION 216 TO 4
1373                    .... EXIT WHEN AST_ADR(ASTE#) = BLOCK#;
FIXUP LOCATION 160 TO 44
FIXUP LOCATION 222 TO 2

1374                END;

FIXUP LOCATION 130 TO 76
1375            END;

1376            /*    DESTROY DESCRIPTOR

1377            INLINE(SPLHIGH);

1378            PS_SDR(PEG#) := 0;

1379            PS_SAR(REG#) := 0;

1380            /*    IF FOR CURRENT PROCESS ALSO CLEAR SEGMENTATION REGISTER

1381            IF PS_CURRENT_PROCESS = THE_CURRENT_PROCESS;
1382              THEN:
1383                IF REG# > CROSS_REG#;

1384                  THEN: REG# := REG# + REG_CONSTANT;
FIXUP LOCATION 344 TO 14
1385                END;

1386                /*    CHECK FOR CHANGE BIT BEING SET

1387                IF (SDR(REG#) & SDR_CHANGE_MASK) = SDR_CHANGED;
```

```
CODE(212): BEQ 2
CODE(214): JMP 0(7)
CODE(220): JMP 453(7)

  |224


CODE(224): BR  -30
CODE(226): CMP (6)+,(6)+
  |230
  |

  |230
  |
*/|230
  |
  |232
CODE(232): MOV -4(5),-(6)
CODE(236): ASL (6)
CODE(240): CLR -(6)
CODE(242): MOV #20000,2
CODE(246): MOV (6)+,0
CODE(250): ADD (6)+,2
CODE(252): MOV 0,0(2)
  |256
CODE(256): MOV -4(5),-(6)
CODE(262): ASL (6)
CODE(264): CLR -(6)
CODE(266): MOV #20040,2
CODE(272): MOV (6)+,0
CODE(274): ADD (6)+,2
CODE(276): MOV 0,0(2)
  |302
  |
*/|302
  |
  |330
  |330
  |346
CODE(302): MOV #20100,2
CODE(306): MOV 0(2),-(6)
CODE(312): MOV #16544,2
CODE(316): CMP 0(2),(6)+
CODE(322): BEQ 2
CODE(324): JMP 0(7)
CODE(330): MOV -4(5),-(6)
CODE(334): CMP #7,(6)+
CODE(340): BLT 2
CODE(342): JMP 0(7)
CODE(346): MOV -4(5),-(6)
CODE(352): ADD #2570,(6)
CODE(356): MOV (6)+,-4(5)
  |362

  |362
  |
*/|362
  |
  |424
CODE(362): MOV -4(5),-(6)
CODE(366): ASL (6)
CODE(370): MOV #-5600,2
CODE(374): ADD (6)+,2
CODE(376): MOV 0(2),-(6)
CODE(402): MOV #100,-(6)
CODE(406): COM (6)
```

```
1388                    THEN: AST_CHANGE(ASTE#) := (AST_CHANGE(ASTE#) | AST_CHANGED);
FIXUP LOCATION 422 TO 44
1389                    END;

1390                    /*    CLEAR SEGMENTATION REGISTER

1391                    SDP(REG#) := 0;

1392                    SAP(REG#) := 0;
FIXUP LOCATION 326 TO 210
1393                END;

1394                INLINE(SPLLOW);

1395                /*    DECREMENT DESCRIPTOR COUNT

1396                AST_DES_COUNT(ASTE#) := AST_DES_COUNT(ASTE#) - 1;

1397                /*    THE FOLLOWING DOES NOT APPLY TO WIRED DOWN SEGMENTS

1398                IF (AST_CPL(ASTE#) & WIRED_DOWN_MASK) = 0;
```

```
CODE(410): BIC  (6)+,(6)
CODE(412): CMP #100,(6)+
CODE(416): BEQ 2
CODE(420): JMP 0(7)
CODE(424): MOV -12(5),-(6)
CODE(430): MOV -12(5),-(6)
CODE(434): MOV #4400,2
CODE(440): ADD (6)+,2
CODE(442): MOVB0(2),1
CODE(446): MOV 1,-(6)
CODE(450): BIS #40,(6)
CODE(454): MOV #4400,2
CODE(460): MOV (6)+,0
CODE(462): ADD (6)+,2
CODE(466): MOVB0,0(2)
  |470

  |470
  |
*/|470
  |
CODE(470): MOV -4(5),-(6)
CODE(474): ASL (6)
CODE(476): CLR -(6)
CODE(500): MOV #-5600,2
CODE(504): MOV (6)+,0
CODE(506): ADD (6)+,2
CODE(510): MOV 0,0(2)
  |514
CODE(514): MOV -4(5),-(6)
CODE(520): ASL (6)
CODE(522): CLR -(6)
CODE(524): MOV #-5540,2
CODE(530): MOV (6)+,0
CODE(532): ADD (6)+,2
CODE(536): MOV 0,0(2)
  |540

  |540
  |
  |542
  |
*/|542
  |
CODE(542): MOV -12(5),-(6)
CODE(546): ASL (6)
CODE(550): MOV -12(5),-(6)
CODE(554): ASL (6)
CODE(556): MOV #10400,2
CODE(562): ADD (6)+,2
CODE(564): MOV 0(2),-(6)
CODE(570): SUB #1,(6)
CODE(574): MOV #10400,2
CODE(600): MOV (6)+,0
CODE(602): ADD (6)+,2
CODE(604): MOV 0,0(2)
  |610
  |
*/|610
  |
  |652
```

```
1398                THEN:                                                                    |652
                                                                                             |
1400                   /*    IF NO DESCRIPTORS LEFT THEN UNLOCK AND ADD TO SWAP CHAIN       */|652
                                                                                             |
1401                   IF (AST_DES_COUNT(ASTE#) = 0);                                        |704
                                                                                    CODE(610): MOV -12(5),-(6)
                                                                                    CODE(614): ASL (6)
                                                                                    CODE(616): MOV #11400,2
                                                                                    CODE(622): ADD (6)+,2
                                                                                    CODE(624): MOV 0(2),-(6)
                                                                                    CODE(630): MOV #-100000,-(6)
                                                                                    CODE(634): COM (6)
                                                                                    CODE(636): BIC (6)+,(6)
                                                                                    CODE(640): CLR -(6)
                                                                                    CODE(642): CMP (6)+,(6)+
                                                                                    CODE(644): BEQ 2
                                                                                    CODE(646): JMP 0(7)
                                                                                    CODE(652): MOV -12(5),-(6)
                                                                                    CODE(656): ASL (6)
                                                                                    CODE(660): MOV #10400,2
                                                                                    CODE(664): ADD (6)+,2
                                                                                    CODE(666): MOV 0(2),-(6)
                                                                                    CODE(672): CLR -(6)
                                                                                    CODE(674): CMP (6)+,(6)+
                                                                                    CODE(676): BEQ 2
                                                                                    CODE(700): JMP 0(7)
                                                                                    CODE(704): MOV -12(5),-(6)
                                                                                    CODE(710): ASL (6)
                                                                                    CODE(712): CLR -(6)
                                                                                    CODE(714): ASL (6)
                                                                                    CODE(716): MOV #10400,2
                                                                                    CODE(722): ADD (6)+,2
                                                                                    CODE(724): MOV 0(2),-(6)
                                                                                    CODE(730): MOV #10400,2
                                                                                    CODE(734): MOV (6)+,0
                                                                                    CODE(736): ADD (6)+,2
                                                                                    CODE(740): MOV 0,0(2)
1402                      THEN: AST_SWAP_CHAIN(ASTE#) := AST_SWAP_CHAIN(0);             |744
                                                                                    CODE(744): CLR -(6)
                                                                                    CODE(746): ASL (6)
                                                                                    CODE(750): MOV -12(5),-(6)
                                                                                    CODE(754): MOV #10400,2
                                                                                    CODE(760): MOV (6)+,0
                                                                                    CODE(762): ADD (6)+,2
                                                                                    CODE(764): MOV 0,0(2)
1403                         AST_SWAP_CHAIN(0) := ASTE#;                                |770
                                                                                    CODE(770): MOV -12(5),-(6)
                                                                                    CODE(774): MOV -12(5),-(6)
                                                                                    CODE(1000): MOV #4400,2
                                                                                    CODE(1004): ADD (6)+,2
                                                                                    CODE(1006): MOVB0(2),1
                                                                                    CODE(1012): MOV 1,-(6)
                                                                                    CODE(1014): BIS #20,(6)
                                                                                    CODE(1020): MOV #4400,2
                                                                                    CODE(1024): MOV (6)+,0
                                                                                    CODE(1026): ADD (6)+,2
                                                                                    CODE(1030): MOVB0,0(2)
1404                         AST_UNLOCK(ASTE#) := (AST_UNLOCK(ASTE#) | AST_UNLOCK_FLAG);  |1034
```

```
FIXUP LOCATION 702 TO 130
1405                    END;

1406                    /*    ADJUST MEMORY QUOTA

1407                    PS_MEM_QUOTA := PS_MEM_QUOTA + AST_SIZE(ASTPR);
FIXUP LOCATION 450 TO 224
1408                END;

FIXUP LOCATION 54 TO 1020
1409        END;

FIXUP LOCATION 14 TO 8
                44 LINES WERE COMPILED, GENERATING 578 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA SWAPIN      5
1410   _|_
```

```
  |1034
  |
*/|1034
  |
CODE(1034): MOV -12(5),-(6)
CODE(1040): MOV #5000,2
CODE(1044): ADD (6)+,2
CODE(1046): MOVB0(2),1
CODE(1052): MOV 1,-(6)
CODE(1054): MOV #20116,2
CODE(1060): MOVB0(2),3
CODE(1064): ADD 3,(6)
CODE(1066): MOV #20116,2
CODE(1072): MOVB(6)+,0(2)
  |1076

  |1076
  |

  |1076
  |
CODE(1076): JMP @-6(5)



  10
```

```
                                                                    10
                                                                    15
                                                                    16
1411   DATA SWAPIN(ASTEP0);                                          10
1412        DECLARE
                PROCEDURE ACCEPTS (WORD, WORD) (INITSEG);
1413        0 LINES WERE COMPILED.

NO ERRORS WERE DETECTED.
DATA INITSEG
1418   _I_
```

```
1415   DATA INITSEG(ASTER#, BLOCK#);                                    10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1416   _I_                                                              10
```

```
1417    PROGRAM INITSEG;
1418        DECLARE
1419            WORD (I);




1420        ASI_ADR(ASTF#) := BLOCK#;

1421        /*    GAIN ACCESS TO SEGMENT




1422        LSD(ASTF#, DIR_KSR_ADR, SDR_WRITE_ACCESS);

1423        /*    THIS ASSUMES ALL SDRS ARE 1K BYTES - WILL CHANGE LATER


FIXUP LOCATION 116 TO 2
1424        DO I := 2 TO 511;




1425            ZERO_ARRAY(I) := 0;
FIXUP LOCATION 132 TO 32

1426        END;
```

```
  |16
  |16
  |16
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #6,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV -6(5),-(6)
CODE(30): MOV #7400,2
CODE(34): MOV (6)+,0
CODE(36): ADD (6)+,2
CODE(40): MOV 0,0(2)
  |44
  |
*/|44
  |
CODE(44): MOV 5,-(6)
CODE(46): MOV (5),0
CODE(50): MOV (0),-(6)
CODE(52): CLR -(6)
CODE(54): MOV -4(5),-(6)
CODE(60): MOV #-5472,-(6)
CODE(64): MOV #6,-(6)
CODE(70): MOV -2(4),2
CODE(74): JSR 7,#154(2)
CODE(100): MOV 5,6
CODE(102): CLR (6)+
CODE(104): MOV (6)+,5
  |106
  |
*/|106
  |
CODE(106): MOV #777,-(6)
CODE(112): CLR -(6)
CODE(114): JMP 453(7)
CODE(120): INC (6)

  |140
CODE(122): CMP (6),2(6)
CODE(126): BLE 2
CODE(130): JMP 453(7)
CODE(134): MOV (6),-12(5)
CODE(140): MOV -12(5),-(6)
CODE(144): ASL (6)
CODE(146): CLR -(6)
CODE(150): MOV #60000,2
CODE(154): MOV (6)+,0
CODE(156): ADD (6)+,2
CODE(160): MOV 0,0(2)
  |164

CODE(164): BR -23
CODE(166): CMP (6)+,(6)+
  |170
  |
```

```
1427        IF (AST_TYPE(ASTE#) & AST_TYPE_MASK) = AST_TYPE_DIRECTORY;
1428          THEN;
1429              /*    PUT ALL ACL ELEMENTS ON FREE CHAIN
```

```
FIXUP LOCATION 242 TO ?
1430              DO I := 0 TO ACL_MAX;
```

```
1431                    ACL_CHAIN(I) := I + 1;
FIXUP LOCATION 256 TO 36
1432              END;
```

```
1433              ACL_CHAIN(ACL_MAX) := 0;
FIXUP LOCATION 230 TO 110
1434          END;
1435          /*    UPDATE AST
```

```
1436        AST_STATUS(ASTE#) := (AST_STATUS(ASTE#) & AST_STATUS_NOTMASK);
```

```
  1232
  1232
   |
*/1232
   |
CODE(170): MOV -4(5),-(6)
CODE(174): MOV #4400,2
CODE(200): ADD (6)+,2
CODE(202): MOVB0(2),1
CODE(206): MOV 1,-(6)
CODE(210): MOV #200,-(6)
CODE(214): COM (6)
CODE(216): BIC (6)+,(6)
CODE(220): CMP #200,(6)+
CODE(224): BEQ 2
CODE(226): JMP 0(7)
CODE(232): MOV #177,-(6)
CODE(236): CLR -(6)
CODE(240): JMP 453(7)
CODE(244): INC (6)

  1264
CODE(246): CMP (6),2(6)
CODE(252): BLE 2
CODE(254): JMP 453(7)
CODE(260): MOV (6),-12(5)
CODE(264): MOV -12(5),-(6)
CODE(270): MOV -12(5),-(6)
CODE(274): ADD #1,(6)
CODE(300): MOV #61600,2
CODE(304): MOV (6)+,0
CODE(306): ADD (6)+,2
CODE(310): MOVB0,0(2)
  1314

CODE(314): BR  -25
CODE(316): CMP (6)+,(6)+
  1320
   |
CODE(320): MOV #177,-(6)
CODE(324): CLR -(6)
CODE(326): MOV #61600,2
CODE(332): MOV (6)+,0
CODE(334): ADD (6)+,2
CODE(336): MOVB0,0(2)
  1342

  1342
   |
*/1342
   |
CODE(342): MOV -4(5),-(6)
CODE(346): MOV -4(5),-(6)
CODE(352): MOV #4400,2
CODE(356): ADD (6)+,2
CODE(360): MOVB0(2),1
CODE(364): MOV 1,-(6)
CODE(366): MOV #277,-(6)
CODE(372): COM (6)
CODE(374): BIC (6)+,(6)
CODE(376): MOV #4400,2
CODE(402): MOV (6)+,0
CODE(404): ADD (6)+,2
CODE(406): MOVB0,0(2)
  1412
CODE(412): MOV -4(5),-(6)
```

```
1437        AST_CHANGF(ASTF#) := (AST_CHANGF(ASTF#) | AST_CHANGFD);

FIXUP LOCATION 14 TO 2
              22 LINES WERE COMPILED, GENERATING 306 BYTES OF CODE.
NO ERRORS WERE DETECTED.
PROGRAM SWAPIN    F
1438    _!_
```

```
CODE(416): MOV -4(5),-(6)
CODE(422): MOV #4400,2
CODE(426): ADD (6)+,2
CODE(430): MOVB0(2),1
CODE(434): MOV 1,-(6)
CODE(436): BIS #4C,(6)
CODE(442): MOV #4400,2
CODE(446): MOV (6)+,C
CODE(450): ADD (6)+,2
CODE(452): MOVB0,C(2)
  1456
CODE(456): JMP @-10(5)
```

```
1439   PROGRAM SWAPIN;
1440         DECLARE
1441               WORD (BLOCK#, I, NEXT);

1442         /*    LOOK FOR A FREE BLOCK

1443         BLOCK# := (MRT_CHAIN(0) & MRT_CHAIN_MASK);

1444         IF BLOCK# = END_BLOCK#;

1445               /*    NO FREE BLOCKS - LOOK AT SWAP CHAIN FOR SOMETHING TO SWAPOUT

1446            THEN: I := 0;

1447               CYCLE

1448                     NEXT := AST_SWAP_CHAIN(I);

FIXUP LOCATION 154 TO 4
1449                  .... EXIT WHEN AST_SWAP_CHAIN(NEXT) = 0;

1450                     I := NEXT;
FIXUP LOCATION 160 TO 10
```

```
  |24
  |24
  |24
  |
*/|24
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): BR  2
CODE(14): =V(CONTAINED PROCEDURE)
CODE(20): SUB #0,6
CODE(24): CLR -(6)
CODE(26): ASL (6)
CODE(30): MOV #400,2
CODE(34): ADD (6)+,2
CODE(36): MOV 0(2),-(6)
CODE(42): MOV #37777,-(6)
CODE(46): COM (6)
CODE(50): BIC (6)+,(6)
CODE(52): MOV (6)+,-10(5)
  |56
  |
  |74
  |
*/|74
  |
CODE(56): MOV -10(5),-(6)
CODE(62): CMP #1740,(6)+
CODE(66): BEQ 2
CODE(70): JMP 0(7)
CODE(74): CLR -(6)
CODE(76): MOV (6)+,-12(5)
  |102
  |
  |102
CODE(102): MOV -12(5),-(6)
CODE(106): ASL (6)
CODE(110): MOV #10400,2
CODE(114): ADD (6)+,2
CODE(116): MOV 0(2),-14(5)
  |124
CODE(124): MOV -14(5),-(6)
CODE(130): ASL (6)
CODE(132): MOV #10400,2
CODE(136): ADD (6)+,2
CODE(140): MOV 0(2),-(6)
CODE(144): CLR -(6)
CODE(146): CMP (6)+,(6)+
CODE(150): BEQ 2
CODE(152): JMP 0(7)
CODE(156): JMP 453(7)

  |162
CODE(162): MOV -14(5),-12(5)
  |170
```

```
1451            END;

1452            BLOCK# := AST_ADR(NEXT);

1453            SWAPOUT(NEXT);
PTASP LOCATION 72 TO 150
1454        END;

1455        /*    REMOVE BLOCK FROM FREE CHAIN

1456        MBT_CHAIN(0) := MBT_CHAIN(BLOCK#);

1457        MBT_FLAGS(BLOCK#) := ALLOCATED;

1458        /*    IF SEGMENT IS UNINITIALIZED DO NOT PERFORM DISK I/O

1459        IF (AST_STATUS(ASTP#) & AST_STATUS_MASK) = AST_UNINITIALIZED;
```

```
CODE(170): BR  -34
  |172
  |
CODE(172): MOV -14(5),-(6)
CODE(176): ASL (6)
CODE(200): MOV #7400,2
CODE(204): ADD (6)+,2
CODE(206): MOV 0(2),-10(5)
  |214
CODE(214): MOV 5,-(6)
CODE(216): MOV (5),-(6)
CODE(220): CLR -(6)
CODE(222): MOV -14(5),-(6)
CODE(226): MOV -2(#),2
CODE(232): JSR 7,#140(2)
CODE(236): MOV 5,6
CODE(240): CLR (6)+
CODE(242): MOV (6)+,5
  |244

  |244
  |
*/|244
  |
CODE(244): CLR -(6)
CODE(246): ASL (6)
CODE(250): MOV -10(5),-(6)
CODE(254): ASL (6)
CODE(256): MOV #400,2
CODE(262): ADD (6)+,2
CODE(264): MOV 0(2),-(6)
CODE(270): MOV #400,2
CODE(274): MOV (6)+,0
CODE(276): ADD (6)+,2
CODE(300): MOV 0,0(2)
  |304
CODE(304): MOV -10(5),-(6)
CODE(310): ASL (6)
CODE(312): CLR -(6)
CODE(314): MOV #400,2
CODE(320): MOV (6)+,0
CODE(322): ADD (6)+,2
CODE(324): MOV 0,0(2)
  |330
  |
*/|330
  |
  |372
CODE(330): MOV -4(5),-(6)
CODE(334): MOV #4400,2
CODE(340): ADD (6)+,2
CODE(342): MOVB0(2),1
CODE(346): MOV 1,-(6)
CODE(350): MOV #100,-(6)
CODE(354): COM (6)
CODE(356): BIC (6)+,(6)
CODE(360): CMP #100,(6)+
CODE(364): BEQ 2
CODE(366): JMP 0(7)
CODE(372): MOV 5,-(6)
CODE(374): MOV 5,-(6)
CODE(376): CLR -(6)
CODE(400): MOV -4(5),-(6)
CODE(404): MOV -10(5),-(6)
CODE(410): MOV -2(5),2
CODE(414): JSR 7,#4(2)
```

```
1460          THEN: INITSEG(ASTE#, BLOCK#);
FIXUP LOCATION 370 TO 40
1461          ELSE:
1462               /*    START DISK I/O AND WAIT FOR COMPLETION
1463               DISKIO(AST_DISK(ASTE#), BLOCK#, AST_SIZE(ASTE#), DISK_READ);
1464               F(DISK_SMFR);
FIXUP LOCATION 430 TO 124
1465          END;
1466          /*    UPDATE AST
1467          AST_ADR(ASTE#) := BLOCK#;
1468          SBT_ASTE(BLOCK#) := (ASTE# | ALLOCATED);
1469          /*    PUT SEGMENT ON SWAP CHAIN
```

```
CODE(420): MOV 5,6
CODE(422): CLR (6)+
CODE(424): MOV (6)+,5
  |426
  |432
  |
*/|432
  |
CODE(426): JMP 0(7)
CODE(432): MOV 5,-(6)
CODE(434): MOV (5),-(6)
CODE(436): CLR -(6)
CODE(440): MOV -4(5),-(6)
CODE(444): ASL (6)
CODE(446): MOV #5400,2
CODE(452): ADD (6)+,2
CODE(454): MOV 0(2),-(6)
CODE(460): MOV -10(5),-(6)
CODE(464): MOV -4(5),-(6)
CODE(470): MOV #5000,2
CODE(474): ADD (6)+,2
CODE(476): MOVB0(2),1
CODE(502): MOV 1,-(6)
CODE(504): MOV #105,-(6)
CODE(510): MOV -2(4),2
CODE(514): JSR 7,@160(2)
CODE(520): MOV 5,6
CODE(522): CLR (6)+
CODE(524): MOV (6)+,5
  |526
CODE(526): MOV 5,-(6)
CODE(530): MOV (5),-(6)
CODE(532): CLR -(6)
CODE(534): MOV #401,-(6)
CODE(540): MOV -2(4),2
CODE(544): JSR 7,@124(2)
CODE(550): MOV 5,6
CODE(552): CLR (6)+
CODE(554): MOV (6)+,5
  |556
  |556
  |
*/|556
  |
CODE(556): MOV -4(5),-(6)
CODE(562): ASL (6)
CODE(564): MOV -10(5),-(6)
CODE(570): MOV #7400,2
CODE(574): MOV (6)+,0
CODE(576): ADD (6)+,2
CODE(600): MOV 0,0(2)
  |604
CODE(604): MOV -10(5),-(6)
CODE(610): ASL (6)
CODE(612): MOV -4(5),-(6)
CODE(616): CLR -(6)
CODE(620): BIS (6)+,(6)
CODE(622): MOV #400,2
CODE(626): MOV (6)+,0
CODE(630): ADD (6)+,2
CODE(632): MOV 0,0(2)
  |636
  |
*/|636
```

```
1470        AST_SWAP_CHAIN(ASTE*) := AST_SWAP_CHAIN(*);

1471        AST_SWAP_CHAIN*(*) := ASTE*;

1472        AST_UNLOCK(ASTE*) := (AST_UNLOCK(ASTE*) | AST_UNLOCK_FLAG);

FIXUP LOCATION 22 TO 6
                 45 LINES WERE COMPILED, GENERATING 504 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA SWAPOUT      5
1473    _!_
```

```
    !
CODE(636): MOV -4(5),-(6)
CODE(642): ASL (6)
CODE(644): CLR -(6)
CODE(646): ASL (6)
CODE(650): MOV #10400,2
CODE(654): ADD (6)+,2
CODE(656): MOV 0(2),-(6)
CODE(662): MOV #10400,2
CODE(666): MOV (6)+,0
CODE(670): ADD (6)+,2
CODE(672): MOV 0,0(2)
    !676
CODE(676): CLR -(6)
CODE(700): ASL (6)
CODE(702): MOV -4(5),-(6)
CODE(706): MOV #10400,2
CODE(712): MOV (6)+,0
CODE(714): ADD (6)+,2
CODE(716): MOV 0,0(2)
    !722
CODE(722): MOV -4(5),-(6)
CODE(726): MOV -4(5),-(6)
CODE(732): MOV #4400,2
CODE(736): ADD (6)+,2
CODE(740): MOVB0(2),1
CODE(744): MOV 1,-(6)
CODE(746): BIS #20,(6)
CODE(752): MOV #4400,2
CODE(756): MOV (6)+,0
CODE(760): ADD (6)+,2
CODE(762): MOVB0,0(2)
    !766
CODE(766): JMP @-6(5)
```

```
1474    DATA SWAPOUT(ASTF0);
                    2 LINES WERE COMPILED.                                  17
NO ERRORS WERE DETECTED.
1475    _|_                                                                 19
```

```
1476   PROGRAM SWAPOUT;
1477        DECLARE
1478             WORD (BLOCK0, INDEX, NEXT);

1479        BLOCK0 := AST_ADR(ASTE0);

1480        AST_ADR(ASTP0) := 0;
1481        /*    REMOVE FROM SWAP CHAIN

1482        INDEX := 0;
1483        CYCLE

1484             NEXT := AST_SWAP_CHAIN(INDEX);

FIXUP LOCATION 130 TO 4
1485        .... EXIT WHEN NEXT = ASTE0;

1486             INDEX := NEXT;
FIXUP LOCATION 134 TO 10
1487        END;
```

```
  |16
  |16
  |16
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #7400,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-10(5)
  |40
CODE(40): MOV -4(5),-(6)
CODE(44): ASL (6)
CODE(46): CLR -(6)
CODE(50): MOV #7400,2
CODE(54): MOV (6)+,0
CODE(56): ADD (6)+,2
CODE(60): MOV 0,0(2)
  |64
  |
*/|64
  |
CODE(64): CLR -(6)
CODE(66): MOV (6)+,-12(5)
  |72
  |
  |72
CODE(72): MOV -12(5),-(6)
CODE(76): ASL (6)
CODE(100): MOV #10400,2
CODE(104): ADD (6)+,2
CODE(106): MOV 0(2),-14(5)
  |114
CODE(114): MOV -14(5),-(6)
CODE(120): CMP -4(5),(6)+
CODE(124): BEQ 2
CODE(126): JMP 0(7)
CODE(132): JMP #53(7)

  |136
CODE(136): MOV -14(5),-12(5)
  |144

CODE(144): BR  -26
  |146
  |
CODE(146): MOV -12(5),-(6)
CODE(152): ASL (6)
CODE(154): MOV -4(5),-(6)
CODE(160): ASL (6)
CODE(162): MOV #10400,2
CODE(166): ADD (6)+,2
CODE(170): MOV 0(2),-(6)
CODE(174): MOV #10400,2
CODE(200): MOV (6)+,0
```

```
1488        AST_SWAP_CHAIN(INDEX) := AST_SWAP_CHAIN(ASTE0);

1489        AST_SWAP_CHAIN(ASTE0) := 0;

1490        AST_UNLOCK(ASTE0) := (AST_UNLOCK(ASTE0) & AST_LOCK_MASK);
1491        /*    ONLY PERFORM DISK WRITE IF SEGMENT HAS CHANGED
1492        IF (AST_CHANGE(ASTE0) & AST_CHANGE_MASK) = AST_CHANGED;

1493          THEN: DISKIO(AST_DISK(ASTE0), BLOCK0, AST_SIZE(ASTE0), DISK_WRITE);
1494             /*    DO BOOKKEEPING AND WAIT FOR I/O TO COMPLETE
```

```
CODE(202): ADD (6)+,2
CODE(204): MOV 0,0(2)
  |210
CODE(210): MOV -8(5),-(6)
CODE(214): ASL (6)
CODE(216): CLR -(6)
CODE(220): MOV #10400,2
CODE(224): MOV (6)+,0
CODE(226): ADD (6)+,2
CODE(230): MOV 0,0(2)
  |234
CODE(234): MOV -8(5),-(6)
CODE(240): MOV -8(5),-(6)
CODE(244): MOV #4400,2
CODE(250): ADD (6)+,2
CODE(252): MOVB0(2),1
CODE(256): MOV 1,-(6)
CODE(260): MOV #357,-(6)
CODE(264): COM (6)
CODE(266): BIC (6)+,(6)
CODE(270): MOV #4400,2
CODE(274): MOV (6)+,0
CODE(276): ADD (6)+,2
CODE(300): MOVB0,0(2)
  |304
  |
*/|304
  |
  |346
CODE(304): MOV -8(5),-(6)
CODE(310): MOV #4400,2
CODE(314): ADD (6)+,2
CODE(316): MOVBG(2),1
CODE(322): MOV 1,-(6)
CODE(324): MOV #40,-(6)
CODE(330): COM (6)
CODE(332): BIC (6)+,(6)
CODE(334): CMP #40,(6)+
CODE(340): BEQ 2
CODE(342): JMP 0(7)
CODE(346): MOV 5,-(6)
CODE(350): MOV (5),-(6)
CODE(352): CLR -(6)
CODE(354): MOV -8(5),-(6)
CODE(360): ASL (6)
CODE(362): MOV #5400,2
CODE(366): ADD (6)+,2
CODE(370): MOV 0(2),-(6)
CODE(374): MOV -10(5),-(6)
CODE(400): MOV -8(5),-(6)
CODE(404): MOV #5000,2
CODE(410): ADD (6)+,2
CODE(412): MOVB0(2),1
CODE(416): MOV 1,-(6)
CODE(420): MOV #103,-(6)
CODE(424): MOV -2(8),2
CODE(430): JSR 7,#160(2)
CODE(434): MOV 5,6
CODE(436): CLR (6)+
CODE(440): MOV (6)+,5
  |442
  |
*/|442
  |
CODE(442): MOV -8(5),-(6)
CODE(446): MOV -8(5),-(6)
```

```
1495              AST_CHANGE(ASTE0) := AST_CHANGE(ASTE0) & AST_UNCHANGED_MASK;

1496              P(DISK_SMPR);
FIXUP LOCATION 344 TO 174
1497          END;

1498          /*    FREE MEMORY ALLOCATED TO SEGMENT

1499          INDEX := 0;

1500          CYCLE

FIXUP LOCATION 610 TO 4
1501              .... EXIT WHEN (MBT_CHAIN(INDEX) & MBT_CHAIN_MASK) > BLOCK0;

1502              INDEX := (MBT_CHAIN(INDEX) & MBT_CHAIN_MASK);
FIXUP LOCATION 614 TO 36
1503          END;

1504          /*    PLACE BLOCK TO BE FREED IN THE CHAIN
```

```
CODE(452): MOV #4400,2
CODE(456): ADD (6)+,2
CODE(460): MOVB0(2),1
CODE(464): MOV 1,-(6)
CODE(466): MOV #337,-(6)
CODE(472): COM (6)
CODE(474): BIC (6)+,(6)
CODE(476): MOV #4400,2
CODE(502): MOV (6)+,0
CODE(504): ADD (6)+,2
CODE(506): MOVB0,0(2)
  |512
CODE(512): MOV 5,-(6)
CODE(514): MOV (5),-(6)
CODE(516): CLR -(6)
CODE(520): MOV #401,-(6)
CODE(524): MOV -2(4),2
CODE(530): JSP 7,#124(2)
CODE(534): MOV 5,6
CODE(536): CLR (6)+
CODE(540): MOV (6)+,5
  |542

  |542
  |
*/|542
  |
CODE(542): CLR -(6)
CODE(544): MOV (6)+,-12(5)
  |550
  |
  |550
CODE(550): MOV -12(5),-(6)
CODE(554): ASL (6)
CODE(556): MOV #400,2
CODE(562): ADD (6)+,2
CODE(564): MOV 0(2),-(6)
CODE(570): MOV #37777,-(6)
CODE(574): COM (6)
CODE(576): BIC (6)+,(6)
CODE(600): CMP -10(5),(6)+
CODE(604): BLT 2
CODE(606): JMP 0(7)
CODE(612): JMP #53(7)

  |616
CODE(616): MOV -12(5),-(6)
CODE(622): ASL (6)
CODE(624): MOV #400,2
CODE(630): ADD (6)+,2
CODE(632): MOV 0(2),-(6)
CODE(636): MOV #37777,-(6)
CODE(642): COM (6)
CODE(644): BIC (6)+,(6)
CODE(646): MOV (6)+,-12(5)
  |652

CODE(652): BR -42
  |654
  |
*/|654
  |
CODE(654): MOV -10(5),-(6)
CODE(660): ASL (6)
CODE(662): MOV -12(5),-(6)
CODE(666): ASL (6)
```

```
1505        MBT_CHAIN(BLOCK#) := MBT_CHAIN(INDEX);

1506        MBT_CHAIN(INDEX) := (BLOCK# ! FREE_MEM);

FIXUP LOCATION 14 TO 6
              32 LINES WERE COMPILED, GENERATING 492 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA OUTERP      R
1507    -!-
```

```
CODE(670):  MOV  #400,2
CODE(674):  ADD  (6)+,2
CODE(676):  MOV  0(2),-(6)
CODE(702):  MOV  #400,2
CODE(706):  MOV  (6)+,0
CODE(710):  ADD  (6)+,2
CODE(712):  MOV  0,0(2)
  !716
CODE(716):  MOV  -12(5),-(6)
CODE(722):  ASL  (6)
CODE(724):  MOV  -10(5),-(6)
CODE(730):  BIS  #-100000,(6)
CODE(734):  MOV  #400,2
CODE(740):  MOV  (6)+,0
CODE(742):  ADD  (6)+,2
CODE(744):  MOV  0,0(2)
  !750
CODE(750):  JMP  @-6(5)
```

```
1508   DATA OUTERP(ASTER) RETURNS (RC);                                    10
                    2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                   10
1509   _I_
```

```
1510   PROGRAM OUTERP;

1511         /*    SECURITY CHECKS FIRST

1512         /*    PROCESS MUST HAVE WRITE$READ ACCESS TO SEMAPHORE

1513         IF (AST_WAL(ASTE#) & PS_PROCESS_MASK) = 0;
```

```
1514           THEN:
1515    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 62 TO 12
1516         END;

1517         /*    IMPLEMENTATION CHECKS

1518         INLINE(SPLHIGH);

1519         IF SMPR_COUNT(ASTE#) = - 128;
```

```
1520           THEN:
1521    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 130 TO 12
1522         END;

1523         /*    CHECKING COMPLETE - PERFORM STATE CHANGE
```

```
  |16
  |
*/|16
  |
*/|16
  |
  |64
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #12400,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-(6)
CODE(36): MOV #20102,2
CODE(42): MOV 0(2),-(6)
CODE(46): COM (6)
CODE(50): BIC (6)+,(6)
CODE(52): CLR -(6)
CODE(54): CMP (6)+,(6)+
CODE(56): BEQ 2
CODE(60): JMP 0(7)
CODE(64): MOV #-2,4(5)
CODE(72): JMP @-6(5)
  |76
  |76

  |76
  |
*/|76
  |
  |100
  |
  |132
CODE(100): MOV -4(5),-(6)
CODE(104): MOV #15534,2
CODE(110): ADD (6)+,2
CODE(112): MOVB0(2),1
CODE(116): MOV 1,-(6)
CODE(120): CMP #-200,(6)+
CODE(124): BEQ 2
CODE(126): JMP 0(7)
CODE(132): MOV #-2,4(5)
CODE(140): JMP @-6(5)
  |144
  |144

  |144
  |
*/|144
  |
CODE(144): MOV 5,-(6)
CODE(146): MOV (5),-(6)
CODE(150): CLR -(6)
CODE(152): MOV -4(5),-(6)
```

```
1524          P(ASTER);

1525          BC := OK_FLAG;

FIXUP LOCATION 14 TO 2
                17 LINES WERE COMPILED, GENERATING 134 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA OUTERV       5
1526   _I_
```

```
CODE(156): MOV -2(4),2
CODE(162): JSR 7,#124(2)
CODE(166): MOV 5,6
CODE(170): CLR (6)+
CODE(172): MOV (6)+,5
  |174
CODE(174): BOV #-1,4(5)
  |202
CODE(202): JMP #-6(5)
```

```
1527   DATA OUTERV(ASTER#) RETURNS (RC);                                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                    10
1528   _I_
```

```
1529   PROGRAM OUTERV;

1530         /*    SECURITY CHECKS FIRST

1531         /*    PROCESS MUST HAVE WRITE&READ ACCESS TO SEMAPHORE

1532         IF (AST_VAL(ASTE#) & PS_PROCESS_MASK) = 0;
```

```
1533           THEN;
1534    ....       RETURN WITH ERR_FLAG;
FIXUP LOCATION 62 TO 12
1535         END;

1536         /*    IMPLEMENTATION CHECKS

1537         INLINE(SPLHIGH);

1538         IF SMPR_COUNT(ASTE#) = 127;
```

```
1539           THEN;
1540    ....       RETURN WITH ERR_FLAG;
FIXUP LOCATION 130 TO 12
1541         END;

1542         /*    CHECKING COMPLETE - PERFORM STATE CHANGE
```

```
  |16
  |
*/|16
  |
*/|16
  |
  |64
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),-(6)
CODE(22): ASL (6)
CODE(24): MOV #12400,2
CODE(30): ADD (6)+,2
CODE(32): MOV 0(2),-(6)
CODE(36): MOV #20102,2
CODE(42): MOV 0(2),-(6)
CODE(46): COM (6)
CODE(50): BIC (6)+,(6)
CODE(52): CLR -(6)
CODE(54): CMP (6)+,(6)+
CODE(56): BEQ 2
CODE(60): JMP 0(7)
CODE(64): MOV #-2,4(5)
CODE(72): JMP @-6(5)
  |76
  |76

  |76
  |
*/|76
  |
  |100
  |
  |132
CODE(100): MOV -4(5),-(6)
CODE(104): MOV #15534,2
CODE(110): ADD (6)+,2
CODE(112): MOVB0(2),1
CODE(116): MOV 1,-(6)
CODE(120): CMP #177,(6)+
CODE(124): BEQ 2
CODE(126): JMP 0(7)
CODE(132): MOV #-2,4(5)
CODE(140): JMP @-6(5)
  |144
  |144

  |144
  |
*/|144
  |
CODE(144): MOV 5,-(6)
CODE(146): MOV (5),-(6)
CODE(150): CLR -(6)
CODE(152): MOV -4(5),-(6)
```

```
1543          V(ASIFB);

1544          RC := OK_FLAG;

FIXUP LOCATION 14 TO 2
                17 LINES WERE COMPILED, GENERATING 134 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA IPCRCV       E
1545   _|_
```

```
CODE(156): MOV -2(4),2
CODE(162): JSR 7,@130(2)
CODE(166): MOV 5,6
CODE(170): CLR (6)+
CODE(172): MOV (6)+,5
  |174
CODE(174): MOV #-1,0(5)
  |202
CODE(202): JMP @-6(5)
```

```
1546   DATA IPCRCV RETURNS (RC);                                        10
1547        DECLARE                                                     10
1548                WORD (INDEX);                                       10
                  0 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                               10
1549   _f_
```

```
1550    PROGRAM IPCRCV;
1551        /*    NO SECURITY CHECKING
1552        /*    ANYTHING THERE
1553        IF PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS) = 0;
1554          THEN: PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS) := IPC_WAIT;
1555            PT_FLAGS(PS_CURRENT_PROCESS) := BLOCKED;
1556            V(KERNEL_SEMP);
1557            SLEEP;
```

```
                |116
                |
*/              |116
                |
*/              |116
                |
                |154
CODE(0):        2,-(0)
CODE(2):  MOV 6,5
CODE(4):  ADD #2,5
CODE(10):  MOV 7,(5)+
CODE(12):  SUB #0,6
CODE(16):  MOV #20100,2
CODE(22):  MOV 0(2),-(6)
CODE(26):  MOV #17640,2
CODE(32):  ADD (6)+,2
CODE(34):  MOVB0(2),1
CODE(40):  MOV 1,-(6)
CODE(42):  CLR -(6)
CODE(44):  CMP (6)+,(6)+
CODE(46):  BEQ 2
CODE(50):  JMP 0(7)
CODE(54):  MOV #20100,2
CODE(60):  MOV 0(2),-(6)
CODE(64):  MOV #377,-(6)
CODE(70):  MOV #17640,2
CODE(74):  MOV (6)+,0
CODE(76):  ADD (6)+,2
CODE(100):  MOVB0,0(2)
                |104
CODE(104):  MOV #20100,2
CODE(110):  MOV 0(2),-(6)
CODE(114):  CLR -(6)
CODE(116):  MOV #17640,2
CODE(122):  MOV (6)+,0
CODE(124):  ADD (6)+,2
CODE(126):  MOVB0,0(2)
                |132
CODE(132):  MOV 5,-(6)
CODE(134):  MOV (5),-(6)
CODE(136):  CLR -(6)
CODE(140):  MOV #400,-(6)
CODE(144):  MOV -2(4),2
CODE(150):  JSR 7,#130(2)
CODE(154):  MOV 5,6
CODE(156):  CLR (6)+
CODE(160):  MOV (6)+,5
                |162
CODE(162):  MOV 5,-(6)
CODE(164):  MOV (5),-(6)
CODE(166):  CLR -(6)
CODE(170):  MOV -2(4),2
CODE(174):  JSR 7,#114(2)
CODE(200):  MOV 5,6
CODE(202):  CLR (6)+
CODE(204):  MOV (6)+,5
                |206
CODE(206):  MOV 5,-(6)
```

```
1558              P(KERNEL_SEMP);
FIXUP LOCATION 52 TO 162
1559          END;
1560          /*    REMOVE FIRST MESSAGE ELEMENT

1561          INDEX := PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS);

1562          PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS) := IPC_LINK(INDEX);
1563          /*    TAKE STUFF OUT OF IPC MESSAGE ELEMENT

1564          RC := IPC_PROCESS#(INDEX);

1565          MSG2 := IPC_DATA(INDEX);
1566          /*    PUT BACK ON FREE CHAIN AND INCREMENT QUOTA

1567          IPC_LINK(INDEX) := IPC_LINK(0);
```

```
CODE(210): MOV (5),-(6)
CODE(212): CLR -(6)
CODE(214): MOV #400,-(6)
CODE(220): MOV -2(5),2
CODE(224): JSR 7,#124(2)
CODE(230): MOV 5,6
CODE(232): CLR (6)+
CODE(234): MOV (6)+,5
  |236
  |
  |236
  |
*/|236
  |
CODE(236): MOV #20100,2
CODE(242): MOV 0(2),-(6)
CODE(246): MOV #17640,2
CODE(252): ADD (6)+,2
CODE(254): MOVB0(2),1
CODE(260): MOV 1,-6(5)
  |264
CODE(264): MOV #20100,2
CODE(270): MOV 0(2),-(6)
CODE(274): MOV -6(5),-(6)
CODE(300): MOV #14400,2
CODE(304): ADD (6)+,2
CODE(306): MOVB0(2),1
CODE(312): MOV 1,-(6)
CODE(314): MOV #17640,2
CODE(320): MOV (6)+,0
CODE(322): ADD (6)+,2
CODE(324): MOVB0,0(2)
  |330
  |
*/|330
  |
CODE(330): MOV -6(5),-(6)
CODE(334): MOV #14600,2
CODE(340): ADD (6)+,2
CODE(342): MOVB0(2),1
CODE(346): MOV 1,4(5)
  |352
CODE(352): MOV -6(5),-(6)
CODE(356): ASL (6)
CODE(360): MOV #15000,2
CODE(364): ADD (6)+,2
CODE(366): MOV 0(2),-(6)
CODE(372): MOV #61742,2
CODE(376): MOV (6)+,0(2)
  |402
  |
*/|402
  |
CODE(402): MOV -6(5),-(6)
CODE(406): CLR -(6)
CODE(410): MOV #14400,2
CODE(414): ADD (6)+,2
CODE(416): MOVB0(2),1
CODE(422): MOV 1,-(6)
CODE(424): MOV #14400,2
CODE(430): MOV (6)+,0
CODE(432): ADD (6)+,2
CODE(434): MOVB0,0(2)
  |440
CODE(440): CLR -(6)
CODE(442): MOV -6(5),-(6)
```

```
1569          IPC_LINK(0) := INDEX;

1569          PT_IPC_QUOTA(PS_CURRENT_PROCESS) := PT_IPC_QUOTA(PS_CURRENT_PROCESS) + 1;

FIXUP LOCATION 14 TO 2
                21 LINES WERE COMPILED, GENERATING 354 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA IPCSEND     F
1570   _|_
```

```
CODE(446): MOV #14400,2
CODE(452): MOV (6)+,0
CODE(454): ADD (6)+,2
CODE(456): MOVB0,0(2)
  1462
CODE(462): MOV #20100,2
CODE(466): MOV 0(2),-(6)
CODE(472): MOV #20100,2
CODE(476): MOV 0(2),-(6)
CODE(502): MOV #17660,2
CODE(506): ADD (6)+,2
CODE(510): MOVB0(2),1
CODE(514): MOV 1,-(6)
CODE(516): ADD #1,(6)
CODE(522): MOV #17660,2
CODE(526): MOV (6)+,0
CODE(530): ADD (6)+,2
CODE(532): MOVB0,0(2)
  1536
CODE(536): JMP @-4(5)
```

```
1571   DATA IPCSEND(PROCESS#, MESSAGE, DOMAIN);                    10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1572   _I_                                                         10
```

```
1573   PROGRAM IPCSEND;                                                                    |16
1574         DECLARE                                                                       |16
1575               WORD (INDEX, INDEX2);                                                   |16
                                                                                           |
1576         IF (PT_FLAGS(PROCESS#) & PT_FLAGS_MASK) = INACTIVE;                           |60
                                                                                    CODE(0):      2,-(0)
                                                                                    CODE(2): MOV 6,5
                                                                                    CODE(4): ADD #10,5
                                                                                    CODE(10): MOV 7,(5)+
                                                                                    CODE(12): SUB #0,6
                                                                                    CODE(16): MOV -4(5),-(6)
                                                                                    CODE(22): MOV #17440,2
                                                                                    CODE(26): ADD (6)+,2
                                                                                    CODE(30): MOVB0(2),1
                                                                                    CODE(34): MOV 1,-(6)
                                                                                    CODE(36): MOV #300,-(6)
                                                                                    CODE(42): COM (6)
                                                                                    CODE(44): BIC (6)+,(6)
                                                                                    CODE(46): CMP #200,(6)+
                                                                                    CODE(52): BEQ 2
                                                                                    CODE(54): JMP 0(7)
                                                                                    CODE(60): JMP @-12(5)
1577           THEN;                                                                 |64
1578   ....      RETURN;                                                               |64
FIXUP LOCATION 56 TO 4
1579         END;                                                                      |64
                                                                                       |
1580         /*    SECURITY CHECK                                                   */|64
                                                                                       |
1581         IF PS_CURRENT_PROCESS <= EXEC_PROCESS#;                                   |106
1582           THEN;                                                                   |106
1583             IF PT_CUR_CLASS(PROCESS#) < PS_CUR_CLASS;                             |146
                                                                                    CODE(64): MOV #20100,2
                                                                                    CODE(70): MOV 0(2),-(6)
                                                                                    CODE(74): CMP #1,(6)+
                                                                                    CODE(100): BNE 2
                                                                                    CODE(102): JMP 0(7)
                                                                                    CODE(106): MOV -4(5),-(6)
                                                                                    CODE(112): MOV #17460,2
                                                                                    CODE(116): ADD (6)+,2
                                                                                    CODE(120): MOVB0(2),1
                                                                                    CODE(124): MOV 1,-(6)
                                                                                    CODE(126): MOV #20112,2
                                                                                    CODE(132): MOVB0(2),1
                                                                                    CODE(136): CMP 1,(6)+
                                                                                    CODE(140): BGT 2
                                                                                    CODE(142): JMP 0(7)
                                                                                    CODE(146): JMP @-12(5)
1584               THEN;                                                             |152
1585   ....          RETURN;                                                           |152
FIXUP LOCATION 144 TO 4
1586             END;                                                                |152
                                                                                     |
1587             IF PT_CUR_CAT(PROCESS#) <= (PT_CUR_CAT(PROCESS#) | PS_CUR_CAT);     |240
                                                                                    CODE(152): MOV -4(5),-(6)
                                                                                    CODE(156): ASL (6)
                                                                                    CODE(160): MOV -4(5),-(6)
                                                                                    CODE(164): ASL (6)
```

```
1588                THEN;
1589  ....              RETURN;
FIXUP LOCATION 236 TO 4
1590              END;

FIXUP LOCATION 104 TO 136
1591        END;

1592        /*    IMPLEMENTATION CHECK

1593        IF PT_IPC_QUOTA(PROCESS#) = 0;
```

```
1594          THEN;
1595  ....      RETURN;
FIXUP LOCATION 274 TO 4
1596        END;

1597        /*    ALLOCATE AN IPC QUEUE ELEMENT

1598        INDEX := IPC_LINK(0);

1599        IPC_LINK(0) := IPC_LINK(INDEX);

1600        IPC_LINK(INDEX) := 0;
```

```
CODE(166): MOV #17500,2
CODE(172): ADD (6)+,2
CODE(174): MOV 0(2),-(6)
CODE(200): MOV #20114,2
CODE(204): BIS 0(2),(6)
CODE(210): MOV #17500,2
CODE(214): MOV 2,1
CODE(216): ADD 2(6),1
CODE(222): MOV (6)+,(6)
CODE(224): MOV 0(1),3
CODE(230): CMP (6)+,3
CODE(232): BNE 2
CODE(234): JSP 0(7)
CODE(240): JMP @-12(5)
  |244
  |244

  |244
  |

  |244
  |
*/|244
  |
  |276
CODE(244): MOV -4(5),-(6)
CODE(250): MOV #17660,2
CODE(254): ADD (6)+,2
CODE(256): MOVB0(2),1
CODE(262): MOV 1,-(6)
CODE(264): CLR -(6)
CODE(266): CMP (6)+,(6)+
CODE(270): BEQ 2
CODE(272): JMP 0(7)
CODE(276): JMP @-12(5)
  |302
  |302

  |302
  |
*/|302
  |
CODE(302): CLR -(6)
CODE(304): MOV #14400,2
CODE(310): ADD (6)+,2
CODE(312): MOVB0(2),1
CODE(316): MOV 1,-14(5)
  |322
CODE(322): CLR -(6)
CODE(324): MOV -14(5),-(6)
CODE(330): MOV #14400,2
CODE(334): ADD (6)+,2
CODE(336): MOVB0(2),1
CODE(342): MOV 1,-(6)
CODE(344): MOV #14400,2
CODE(350): MOV (6)+,0
CODE(352): ADD (6)+,2
CODE(354): MOVB0,0(2)
  |360
CODE(360): MOV -14(5),-(6)
CODE(364): CLR -(6)
CODE(366): MOV #14400,2
CODE(372): MOV (6)+,0
CODE(374): ADD (6)+,2
CODE(376): MOVB0,0(2)
  |402
```

```
1601        /*    FILL IN IPC ELEMENT

1602        IPC_PROCESS#(INDEX) := (PS_CURRENT_PROCESS | DOMAIN);

1603        IPC_DATA(INDEX) := MESSAGE;
1604        /*    IS PROCESS WAITING?
1605        IF (PT_IPC_QUEUE_HEAD(PROCESS#) & BYTE_MASK) = IPC_WAIT;

1606          THEN: PT_IPC_QUEUE_HEAD(PROCESS#) := INDEX;

1607              PT_FLAGS(PROCESS#) := READY;
FIXUP LOCATION 524 TO 54
1608          FLSE:
1609              IF PT_IPC_QUEUE_HEAD(PROCESS#) = 0;
```

```
  |
*/1402
  |
CODE(402): MOV -14(5),-(6)
CODE(406): MOV #20100,2
CODE(412): MOV 0(2),-(6)
CODE(416): BIS -10(5),(6)
CODE(422): MOV #14600,2
CODE(426): MOV (6)+,0
CODE(430): ADD (6)+,2
CODE(432): MOVB0,0(2)
  |436
CODE(436): MOV -14(5),-(6)
CODE(442): ASL (6)
CODE(444): MOV -6(5),-(6)
CODE(450): MOV #15000,2
CODE(454): MOV (6)+,0
CODE(456): ADD (6)+,2
CODE(460): MOV 0,0(2)
  |464
  |
*/|464
  |
  |526
CODE(464): MOV -4(5),-(6)
CODE(470): MOV #17640,2
CODE(474): ADD (6)+,2
CODE(476): MOVB0(2),1
CODE(502): MOV 1,-(6)
CODE(504): MOV #377,-(6)
CODE(510): COM (6)
CODE(512): BIC (6)+,(6)
CODE(514): CMP #377,(6)+
CODE(520): BEQ 2
CODE(522): JMP 0(7)
CODE(526): MOV -4(5),-(6)
CODE(532): MOV -14(5),-(6)
CODE(536): MOV #17640,2
CODE(542): MOV (6)+,0
CODE(544): ADD (6)+,2
CODE(546): MOVB0,0(2)
  |552
CODE(552): MOV -4(5),-(6)
CODE(556): MOV #100,-(6)
CODE(562): MOV #17440,2
CODE(566): MOV (6)+,0
CODE(570): ADD (6)+,2
CODE(572): MOVB0,0(2)
  |576

  |602
  |634
CODE(576): JMP 0(7)
CODE(602): MOV -4(5),-(6)
CODE(606): MOV #17640,2
CODE(612): ADD (6)+,2
CODE(614): MOVB0(2),1
CODE(620): MOV 1,-(6)
CODE(622): CLR -(6)
CODE(624): CMP (6)+,(6)+
CODE(626): BEQ 2
CODE(630): JMP 0(7)
CODE(634): MOV -4(5),-(6)
CODE(640): MOV -14(5),-(6)
CODE(644): MOV #17640,2
CODE(650): MOV (6)+,0
```

```
1610                THEN: PT_IPC_QUEUE_HEAD(PROCESS#) := INDEX;
FIXUP LOCATION 632 TO 30
```

```
1611                ELSE: INDEX2 := PT_IPC_QUEUE_HEAD(PROCESS#);
1612                   CYCLE
```

```
FIXUP LOCATION 736 TO 4
1613                   .... EXIT WHEN IPC_LINK(INDEX2) = 0;
```

```
CODE(652): ADD  (6)+,2
CODE(656): MOVBO,0(2)
  |660

CODE(660): JMP 0(7)
CODE(664): MOV  -4(5),-(6)
CODE(670): MOV  #17640,2
CODE(674): ADD  (6)+,2
CODE(676): MOVBO(2),1
CODE(702): MOV  1,-16(5)
  |706
  |
  |706
CODE(706): MOV  -16(5),-(6)
CODE(712): MOV  #14400,2
CODE(716): ADD  (6)+,2
CODE(720): MOVBO(2),1
CODE(724): MOV  1,-(6)
CODE(726): CLR  -(6)
CODE(730): CMP  (6)+,(6)+
CODE(732): BEQ  2
CODE(734): JMP 0(7)
CODE(740): JMP  453(7)

  |744
CODE(744): MOV  -16(5),-(6)
CODE(750): MOV  #14400,2
CODE(754): ADD  (6)+,2
CODE(756): MOVBO(2),1
CODE(762): MOV  1,-16(5)
```

```
1614                        INDEX2 := IPC_LINK(INDEX2);
FIXUP LOCATION 742 TO 24

1615                    END;




1616                    IPC_LINK(INDEX2) := INDEX;
FIXUP LOCATION 662 TO 130
1617                END;

FIXUP LOCATION 600 TO 212
1618            END;

1619            /*    ADJUST QUOTA




1620            PT_IPC_QUOTA(PROCESS#) := PT_IPC_QUOTA(PROCESS#) - 1;

FIXUP LOCATION 14 TO 4
                    49 LINES WERE COMPILED, GENERATING 564 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA STARTP       F
1621    _!_
```

```
  |766

CODE(766): BR  -31
  |770
  |
CODE(770): MOV -16(5),-(6)
CODE(774): MOV -14(5),-(6)
CODE(1000): MOV #14400,2
CODE(1004): MOV (6)+,0
CODE(1006): ADD (6)+,2
CODE(1010): MOVB0,0(2)
  |1014

  |1014
  |

  |1014
  |
*/|1014
  |
CODE(1014): MOV -4(5),-(6)
CODE(1020): MOV -4(5),-(6)
CODE(1024): MOV #17660,2
CODE(1030): ADD (6)+,2
CODE(1032): MOVB0(2),1
CODE(1036): MOV 1,-(6)
CODE(1040): SUB #1,(6)
CODE(1044): MOV #17660,2
CODE(1050): MOV (6)+,0
CODE(1052): ADD (6)+,2
CODE(1056): MOVB0,0(2)
  |1060
CODE(1060): JMP @-12(5)
```

```
1622   DATA STARTP(USER, PROJECT, CLASS, CAT, PROCESS#, PROC_OFFSET) RETURNS (RC);          10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1623   _I_                                                                                  10
```

```
1624   PROGRAM STARTP;
1625         DECLARE
1626               WORD (I, PDD_SEG0, PD_SEG0, CD_SEG0, SS_SEG0, KS_ASTF0, PROC_SEG0, DUMMY);

1627         /*    ONLY EXECUTIVE CAN CALL THIS FUNCTION

1628         IF PS_CURRENT_PROCESS ¬= EXEC_PROCESS0;

1629           THEN;
1630   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 36 TO 12
1631         END;

1632         /*    PROCESS MUST BE FREE

1633         IF (PT_FLAGS(PROCESS0) & PT_FLAGS_MASK) ¬= INACTIVE;

1634           THEN;
1635   ....      ?RETURN WITH ERR_FLAG;
FIXUP LOCATION 112 TO 12
1636         END;

1637         /*    MAKE "PARTIAL" SWITCH TO USER PROCESS
```

```
  |16
  |16
  |16
  |
*/|16
  |
  |40
CODE(0):       2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #16,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV #20100,2
CODE(22): MOV 0(2),-(6)
CODE(26): CMP #1,(6)+
CODE(32): BNE 2
CODE(34): JMP 0(7)
CODE(40): MOV #-2,4(5)
CODE(46): JMP #-20(5)
  |52
  |52

  |52
  |
*/|52
  |
  |114
CODE(52): MOV -14(5),-(6)
CODE(56): MOV #17440,2
CODE(62): ADD (6)+,2
CODE(66): MOVB0(2),1
CODE(70): MOV 1,-(6)
CODE(72): MOV #300,-(6)
CODE(76): COM (6)
CODE(100): BIC (6)+,(6)
CODE(102): CMP #200,(6)+
CODE(106): BNE 2
CODE(110): JMP 0(7)
CODE(114): MOV #-2,4(5)
CODE(122): JMP #-20(5)
  |126
  |126

  |126
  |
*/|126
  |
CODE(126): MOV 5,-(6)
CODE(130): MOV (5),-(6)
CODE(132): CLR -(6)
CODE(134): MOV -14(5),-(6)
CODE(140): ASL (6)
CODE(142): MOV #17600,2
CODE(146): ADD (6)+,2
CODE(150): MOV 0(2),-(6)
CODE(154): MOV #-5476,-(6)
CODE(160): MOV #6,-(6)
CODE(164): MOV -2(4),2
```

```
1638        LSD(PT_PS_ASTE#(PROCESS#), PS_KSB_ADR, SDR_WRITE_ACCESS);
1639        /*    INITIALIZE PS
1640        PS_CURRENT_PROCESS := PROCESS#;
1641        /*    NEED MASK + NOTMASK
1642        INLINE(MOV, PROCESS#, 0, 3);
1643        INLINE(DEC, 0, 3);
1644        INLINE(NEG, 0, 3);
1645        INLINE(MOV, 2, 7, 0, 0, "4000");
1646        INLINE(MOV, 2, 7, 0, 1, "BFFF");
1647        INLINE(ASHR0R3);
1648        INLINE(ASHR1R3);
1649        INLINE(MOV, 0, 0, PS_PROCESS_MASK);
1650        INLINE(MOV, 0, 1, PS_PROCESS_NOTMASK);
1651        PS_USER_ID := USER;
1652        PS_PROJECT_ID := PROJECT;
1653        PS_CUR_CLASS := CLASS;
1654        PT_CUR_CLASS(PROCESS#) := CLASS;
1655        PS_CUR_CAT := CAT;
1656        PT_CUR_CAT(PROCESS#) := CAT;
```

```
CODE(170): JSR 7,@154(2)
CODE(174): MOV 5,6
CODE(176): CLR (6)+
CODE(200): MOV (6)+,5
  |202
  |
*/|202
  |
CODE(202): MOV -14(5),-(6)
CODE(206): MOV @20100,2
CODE(212): MOV (6)+,0(2)
  |216
  |
*/|216
  |
CODE(216): MOV -14(5),3
  |222
CODE(222): DEC 3
  |224
CODE(224): NEG 3
  |226
CODE(226): MOV @40000,0
  |232
CODE(232): MOV @-40001,1
  |236
  |240
  |242
CODE(242): MOV @20102,2
CODE(246): MOV 0,0(2)
  |252
CODE(252): MOV @20104,2
CODE(256): MOV 1,0(2)
  |262
CODE(262): MOV -4(5),-(6)
CODE(266): MOV @20106,2
CODE(272): MOV (6)+,0(2)
  |276
CODE(276): MOV -6(5),-(6)
CODE(302): MOV @20110,2
CODE(306): MOV (6)+,0(2)
  |312
CODE(312): MOV -10(5),-(6)
CODE(316): MOV @20112,2
CODE(322): MOVB(6)+,0(2)
  |326
CODE(326): MOV -14(5),-(6)
CODE(332): MOV -10(5),-(6)
CODE(336): MOV @17460,2
CODE(342): MOV (6)+,0
CODE(344): ADD (6)+,2
CODE(346): MOVB0,0(2)
  |352
CODE(352): MOV -12(5),-(6)
CODE(356): MOV @20114,2
CODE(362): MOV (6)+,0(2)
  |366
CODE(366): MOV -14(5),-(6)
CODE(372): ASL (6)
CODE(374): MOV -12(5),-(6)
CODE(400): MOV @17500,2
CODE(404): MOV (6)+,0
CODE(406): ADD (6)+,2
CODE(410): MOV 0,0(2)
  |414
CODE(414): MOV @44,-(6)
CODE(420): MOV @20116,2
```

```
1657        PS_MEM_QUOTA := MEM_QUOTA;


1658        PT_IPC_QUOTA(PROCESS#) := IPC_QUOTA;


FIXUP LOCATION 464 TO 2
1659        DO I := 0 TO 15;


1660            PS_SAR(I) := 0;


1661            PS_SDR(I) := 0;
FIXUP LOCATION 500 TO 56

1662        END;


FIXUP LOCATION 572 TO 2
1663        DO I := 0 TO SEG#_MAX;


1664            PS_SEG(I) := ((I + 1) | SEG_FLAG);
FIXUP LOCATION 606 TO 44

1665        END;
```

```
CODE(424): MOVB(6)+,0(2)
  |430
CODE(430): MOV -14(5),-(6)
CODE(434): MOV #10,-(6)
CODE(440): MOV #17660,2
CODE(444): MOV (6)+,0
CODE(446): ADD (6)+,2
CODE(450): MOVB0,0(2)
  |454
  |
CODE(454): MOV #17,-(6)
CODE(460): CLR -(6)
CODE(462): JMP 453(7)
CODE(466): INC (6)

  |506
CODE(470): CMP (6),2(6)
CODE(474): BLE 2
CODE(476): JMP 453(7)
CODE(502): MOV (6),-22(5)
CODE(506): MOV -22(5),-(6)
CODE(512): ASL (6)
CODE(514): CLR -(6)
CODE(516): MOV #20040,2
CODE(522): MOV (6)+,0
CODE(524): ADD (6)+,2
CODE(526): MOV 0,0(2)
  |532
CODE(532): MOV -22(5),-(6)
CODE(536): ASL (6)
CODE(540): CLR -(6)
CODE(542): MOV #20000,2
CODE(546): MOV (6)+,0
CODE(550): ADD (6)+,2
CODE(552): MOV 0,0(2)
  |556

CODE(556): BR  -35
CODE(560): CMP (6)+,(6)+
  |562
  |
CODE(562): MOV #37,-(6)
CODE(566): CLR -(6)
CODE(570): JMP 453(7)
CODE(574): INC (6)

  |614
CODE(576): CMP (6),2(6)
CODE(602): BLE 2
CODE(604): JMP 453(7)
CODE(610): MOV (6),-22(5)
CODE(614): MOV -22(5),-(6)
CODE(620): ASL (6)
CODE(622): MOV -22(5),-(6)
CODE(626): ADD #1,(6)
CODE(632): BIS #-100000,(6)
CODE(636): MOV #20200,2
CODE(642): MOV (6)+,0
CODE(644): ADD (6)+,2
CODE(646): MOV 0,0(2)
  |652

CODE(652): BR  -30
CODE(654): CMP (6)+,(6)+
  |656
  |
```

```
CODE(656): MOV #37,-(6)
CODE(662): ASL (6)
CODE(664): MOV #-100000,-(6)
CODE(670): MOV #20200,2
CODE(674): MOV (6)+,0
CODE(676): ADD (6)+,2
CODE(700): MOV 0,0(2)
```

```
1666        PS_SEG(SEG0_MAX) := SEG_FLAG;                                                    |704
                                                                                             |
1667        /*    PUT ROOT INTO "8"                                                        */|704
                                                                                             |
                                                                                           CODE(704): CLR -(6)
                                                                                           CODE(706): ASL (6)
                                                                                           CODE(710): MOV #1,-(6)
                                                                                           CODE(714): ASL (6)
                                                                                           CODE(716): MOV #20200,2
                                                                                           CODE(722): ADD (6)+,2
                                                                                           CODE(724): MOV 0(2),-(6)
                                                                                           CODE(730): MOV #20200,2
                                                                                           CODE(734): MOV (6)+,0
                                                                                           CODE(736): ADD (6)+,2
                                                                                           CODE(740): MOV 0,0(2)
                                                                                             |744
1668        PS_SEG(0) := PS_SEG(1);                                                        CODE(744): MOV #1,-(6)
                                                                                           CODE(750): ASL (6)
                                                                                           CODE(752): MOV #1,-(6)
                                                                                           CODE(756): MOV #20200,2
                                                                                           CODE(762): MOV (6)+,0
                                                                                           CODE(764): ADD (6)+,2
                                                                                           CODE(766): MOV 0,0(2)
                                                                                             |772
1669        PS_SEG(1) := ROOT_ASTE#;                                                       CODE(772): MOV #1,-(6)
                                                                                           CODE(776): ASL (6)
                                                                                           CODE(1000): MOV #1,-(6)
                                                                                           CODE(1004): ASL (6)
                                                                                           CODE(1006): MOV #11400,2
                                                                                           CODE(1012): ADD (6)+,2
                                                                                           CODE(1014): MOV 0(2),-(6)
                                                                                           CODE(1020): MOV #20102,2
                                                                                           CODE(1024): BIS 0(2),(6)
                                                                                           CODE(1030): MOV #11400,2
                                                                                           CODE(1034): MOV (6)+,0
                                                                                           CODE(1036): ADD (6)+,2
                                                                                           CODE(1040): MOV 0,0(2)
                                                                                             |1044
1670        AST_CPL(ROOT_ASTE#) := (AST_CPL(ROOT_ASTE#) | PS_PROCESS_MASK);                  |
                                                                                             |
1671        /*    GAIN ACCESS TO STACKS                                                    */|1044
                                                                                             |
1672        /*    FIRST PDD                                                                */|1044
                                                                                             |
                                                                                           CODE(1044): CLR -(6)
                                                                                           CODE(1046): MOV 5,-(6)
                                                                                           CODE(1050): MOV (5),-(6)
                                                                                           CODE(1052): CLR -(6)
                                                                                           CODE(1054): MOV #1,-(6)
                                                                                           CODE(1060): MOV #1,-(6)
                                                                                           CODE(1064): MOV -2(4),2
                                                                                           CODE(1070): JSR 7,#SC(2)
                                                                                           CODE(1074): MOV 5,6
                                                                                           CODE(1076): CLR (6)+
                                                                                           CODE(1100): MOV (6)+,5
                                                                                           CODE(1102): MOV (6)+,-24(5)
                                                                                             |1106
1673        PDD_SEG# := GETR(ROOT_ASTE#, PDD_OFFSET);                                      |
                                                                                             |
1674        /*    NEXT EXEC'S PD                                                           */|1106
```

```
1675        PD_SEG# := GETR(PS_SEG(PDD_SEG#), EXEC_PROCESS#);
1676        /*    NOW STACKS - FIRST S STACK
1677        SS_SEG# := GETW(PS_SEG(PD_SEG#), PROCESS#);
1678        DUMMY := ENABLE(PS_SEG(SS_SEG#), 0);
1679        /*    K STACK IS AWKWARD BECAUSE GETW MUST FAIL
1680        /*    HOWEVER EXEC HAS DONE THE GETW AND AN ENABLE
```

```
  |
CODE(1106): CLR -(6)
CODE(1110): MOV 5,-(6)
CODE(1112): MOV (5),-(6)
CODE(1116): CLR -(6)
CODE(1116): MOV -24(5),-(6)
CODE(1122): ASL (6)
CODE(1124): MOV #20200,2
CODE(1130): ADD (6)+,2
CODE(1132): MOV 0(2),-(6)
CODE(1136): MOV #1,-(6)
CODE(1142): MOV -2(4),2
CODE(1146): JSR 7,#50(2)
CODE(1152): MOV 5,6
CODE(1154): CLR (6)+
CODE(1156): MOV (6)+,5
CODE(1160): MOV (6)+,-26(5)
  |1164
  |
*/|1164
  |
CODE(1164): CLR -(6)
CODE(1166): MOV 5,-(6)
CODE(1170): MOV (5),-(6)
CODE(1172): CLR -(6)
CODE(1174): MOV -26(5),-(6)
CODE(1200): ASL (6)
CODE(1202): MOV #20200,2
CODE(1206): ADD (6)+,2
CODE(1210): MOV 0(2),-(6)
CODE(1214): MOV -14(5),-(6)
CODE(1220): MOV -2(4),2
CODE(1224): JSR 7,#44(2)
CODE(1230): MOV 5,6
CODE(1232): CLR (6)+
CODE(1234): MOV (6)+,5
CODE(1236): MOV (6)+,-32(5)
  |1242
CODE(1242): CLR -(6)
CODE(1244): MOV 5,-(6)
CODE(1246): MOV (5),-(6)
CODE(1250): CLR -(6)
CODE(1252): MOV -32(5),-(6)
CODE(1256): ASL (6)
CODE(1260): MOV #20200,2
CODE(1264): ADD (6)+,2
CODE(1266): MOV 0(2),-(6)
CODE(1272): CLR -(6)
CODE(1274): MOV -2(4),2
CODE(1300): JSR 7,#54(2)
CODE(1304): MOV 5,6
CODE(1306): CLR (6)+
CODE(1310): MOV (6)+,5
CODE(1312): MOV (6)+,-40(5)
  |1316
  |
*/|1316
  |
*/|1316
  |
CODE(1316): MOV 5,-(6)
CODE(1320): MOV (5),-(6)
CODE(1322): CLR -(6)
CODE(1324): MOV -26(5),-(6)
CODE(1330): ASL (6)
CODE(1332): MOV #20200,2
```

```
1681        LSD(PS_SEG(PD_SEG#), DIR_KSP_ADR, SDR_READ_ACCESS);

1682        KS_ASTE# := HASH(DIR_DISK(PROCESS#_MAX + PROCESS#));
1683        IF KS_ASTE# = 0;

1684          THEN: INLINE(0);
FIXUP LOCATION 1464 TO 2
1685        END;

1686        LSD(KS_ASTE#, PT_KSDR2_ADR + PPOCESS# + PPOCESS#, SDR_WRITE_ACCESS);

1687        AST_DES_COUNT(KS_ASTE#) := AST_DES_COUNT(KS_ASTE#) + 1;
```

```
CODE(1336): ADD (6)+,2
CODE(1340): MOV 0(2),-(6)
CODE(1344): MOV #-5472,-(6)
CODE(1350): MOV #2,-(6)
CODE(1354): MOV -2(4),2
CODE(1360): JSR 7,#154(2)
CODE(1364): MOV 5,6
CODE(1366): CLR (6)+
CODE(1370): MOV (6)+,5
  |1372
CODE(1372): CLR -(6)
CODE(1374): MOV 5,-(6)
CODE(1376): MOV (5),-(6)
CODE(1400): CLR -(6)
CODE(1402): MOV #7,-(6)
CODE(1406): ADD -14(5),(6)
CODE(1412): ASL (6)
CODE(1414): MOV #60400,2
CODE(1420): ADD (6)+,2
CODE(1422): MOV 0(2),-(6)
CODE(1426): MOV -2(4),2
CODE(1432): JSR 7,#174(2)
CODE(1436): MOV 5,6
CODE(1440): CLR (6)+
CODE(1442): MOV (6)+,5
CODE(1444): MOV (6)+,-34(5)
  |1450
  |
  |1466
CODE(1450): MOV -34(5),-(6)
CODE(1454): CLR -(6)
CODE(1456): CMP (6)+,(6)+
CODE(1460): BEQ 2
CODE(1462): JMP @(7)
  |1470

  |1470
  |
CODE(1470): MOV 5,-(6)
CODE(1472): MOV (5),-(6)
CODE(1474): CLR -(6)
CODE(1476): MOV -34(5),-(6)
CODE(1502): MOV #17100,-(6)
CODE(1506): ADD -14(5),(6)
CODE(1512): ADD -14(5),(6)
CODE(1516): MOV #6,-(6)
CODE(1522): MOV -2(4),2
CODE(1526): JSR 7,#154(2)
CODE(1532): MOV 5,6
CODE(1534): CLR (6)+
CODE(1536): MOV (6)+,5
  |1540
CODE(1540): MOV -34(5),-(6)
CODE(1544): ASL (6)
CODE(1546): MOV -34(5),-(6)
CODE(1552): ASL (6)
CODE(1554): MOV #10400,2
CODE(1560): ADD (6)+,2
CODE(1562): MOV 0(2),-(6)
CODE(1566): ADD #1,(6)
CODE(1572): MOV #10400,2
CODE(1576): MOV (6)+,0
CODE(1600): ADD (6)+,2
CODE(1602): MOV 0,0(2)
  |1606
CODE(1606): MOV -14(5),-(6)
```

```
1688        PT_KS_ASTE0(PROCESS0) := KS_ASTE0;
1689        /*    CLEAN UP A BIT
1690        DCONNECT(PE_SEG0, PS_SEG(PD_SEG0));
1691        DCONNECT(PDD_SEG0, PS_SEG(PDE_SEG0));
1692        /*    NOW FOR INITIAL PROC
1693        CD_SEG0 := GETP(BOOT_ASTE0, CD_OFFSET);
```

```
CODE(1612): ASL (6)
CODE(1614): MOV -30(5),-(6)
CODE(1620): MOV #17540,2
CODE(1624): MOV (6)+,0
CODE(1626): ADD (6)+,2
CODE(1630): MOV 0,0(2)
  |1634
  |
*/|1634
  |
CODE(1634): MOV 5,-(6)
CODE(1636): MOV (5),-(6)
CODE(1640): CLR -(6)
CODE(1642): MOV -26(5),-(6)
CODE(1646): MOV -26(5),-(6)
CODE(1652): ASL (6)
CODE(1654): MOV #20200,2
CODE(1660): ADD (6)+,2
CODE(1662): MOV 0(2),-(6)
CODE(1666): MOV -2(4),2
CODE(1672): JSR 7,#14(2)
CODE(1676): MOV 5,6
CODE(1700): CLR (6)+
CODE(1702): MOV (6)+,5
  |1704
CODE(1704): MOV 5,-(6)
CODE(1706): MOV (5),-(6)
CODE(1710): CLR -(6)
CODE(1712): MOV -24(5),-(6)
CODE(1716): MOV -24(5),-(6)
CODE(1722): ASL (6)
CODE(1724): MOV #20200,2
CODE(1730): ADD (6)+,2
CODE(1732): MOV 0(2),-(6)
CODE(1736): MOV -2(4),2
CODE(1742): JSR 7,#10(2)
CODE(1746): MOV 5,6
CODE(1750): CLR (6)+
CODE(1752): MOV (6)+,5
  |1754
  |
*/|1754
  |
CODE(1754): CLR -(6)
CODE(1756): MOV 5,-(6)
CODE(1760): MOV (5),-(6)
CODE(1762): CLR -(6)
CODE(1764): MOV #1,-(6)
CODE(1770): MOV #3,-(6)
CODE(1774): MOV -2(4),2
CODE(2000): JSR 7,#50(2)
CODE(2004): MOV 5,6
CODE(2006): CLR (6)+
CODE(2010): MOV (6)+,5
CODE(2012): MOV (6)+,-30(5)
  |2016
CODE(2016): CLR -(6)
CODE(2020): MOV 5,-(6)
CODE(2022): MOV (5),-(6)
CODE(2024): CLR -(6)
CODE(2026): MOV -30(5),-(6)
CODE(2032): ASL (6)
CODE(2034): MOV #20200,2
CODE(2040): ADD (6)+,2
CODE(2042): MOV 0(2),-(6)
CODE(2046): MOV -16(5),-(6)
```

```
1694        PROC_SEG# := GETR(PS_SEG(CD_SEG#), PROC_OFFSET);

1695        DUMMY := ENABLE(PS_SEG(PROC_SEG#), 2);

1696        DCONNECT(CD_SEG#, PS_SEG(CD_SEG#));
1697        /*    SWITCH BACK TO EXECUTIVE

1698        LSD(PT_PS_ASTF#(EXEC_PROCESS#), PS_KSR_ADR, SDR_WRITE_ACCESS);
```

```
CODE(2052):  MOV -2(4),2
CODE(2056):  JSR 7,#50(2)
CODE(2062):  MOV 5,6
CODE(2064):  CLR (6)+
CODE(2066):  MOV (6)+,5
CODE(2070):  MOV (6)+,-36(5)
  |2074
CODE(2074):  CLR -(6)
CODE(2076):  MOV 5,-(6)
CODE(2100):  MOV (5),-(6)
CODE(2102):  CLR -(6)
CODE(2104):  MOV -36(5),-(6)
CODE(2110):  ASL (6)
CODE(2112):  MOV #20200,2
CODE(2116):  ADD (6)+,2
CODE(2120):  MOV 0(2),-(6)
CODE(2124):  MOV #2,-(6)
CODE(2130):  MOV -2(4),2
CODE(2134):  JSR 7,#54(2)
CODE(2140):  MOV 5,6
CODE(2142):  CLR (6)+
CODE(2144):  MOV (6)+,5
CODE(2146):  MOV (6)+,-40(5)
  |2152
CODE(2152):  MOV 5,-(6)
CODE(2154):  MOV (5),-(6)
CODE(2156):  CLR -(6)
CODE(2160):  MOV -30(5),-(6)
CODE(2164):  MOV -30(5),-(6)
CODE(2170):  ASL (6)
CODE(2172):  MOV #20200,2
CODE(2176):  ADD (6)+,2
CODE(2200):  MOV 0(2),-(6)
CODE(2204):  MOV -2(4),2
CODE(2210):  JSR 7,#14(2)
CODE(2214):  MOV 5,6
CODE(2216):  CLR (6)+
CODE(2220):  MOV (6)+,5
  |2222
  |
*/|2222
  |
CODE(2222):  MOV 5,-(6)
CODE(2224):  MOV (5),-(6)
CODE(2226):  CLR -(6)
CODE(2230):  MOV #1,-(6)
CODE(2234):  ASL (6)
CODE(2236):  MOV #17600,2
CODE(2242):  ADD (6)+,2
CODE(2244):  MOV 0(2),-(6)
CODE(2250):  MOV #-5476,-(6)
CODE(2254):  MOV #6,-(6)
CODE(2260):  MOV -2(4),2
CODE(2264):  JSR 7,#154(2)
CODE(2270):  MOV 5,6
CODE(2272):  CLR (6)+
CODE(2274):  MOV (6)+,5
  |2276
CODE(2276):  MOV 5,-(6)
CODE(2300):  MOV (5),-(6)
CODE(2302):  CLR -(6)
CODE(2304):  MOV -14(5),-(6)
CODE(2310):  ASL (6)
CODE(2312):  MOV #17600,2
CODE(2316):  ADD (6)+,2
CODE(2320):  MOV 0(2),-(6)
```

```
                                                                                CODE(2324):  MOV  #17000,-(6)
                                                                                CODE(2330):  ADD  -14(5),(6)
                                                                                CODE(2334):  ADD  -14(5),(6)
                                                                                CODE(2340):  MOV  #6,-(6)
                                                                                CODE(2344):  MOV  -2(4),2
                                                                                CODE(2350):  JSR  7,#158(2)
                                                                                CODE(2354):  MOV  5,6
                                                                                CODE(2356):  CLR  (6)+
                                                                                CODE(2360):  MOV  (6)+,5
                                                                                  12362
1699        LSD(PT_PS_ASTR#(PROCESS#), PT_KSDR1_ADR + PROCESS# + PROCESS#, SDR_WRITE_ACCESS);   CODE(2362):  MOV  -14(5),-(6)
                                                                                CODE(2366):  ASL  (6)
                                                                                CODE(2370):  CLR  -(6)
                                                                                CODE(2372):  MOV  #17340,2
                                                                                CODE(2376):  MOV  (6)+,0
                                                                                CODE(2400):  ADD  (6)+,2
                                                                                CODE(2402):  MOV  0,0(2)
                                                                                  12406
1700        PT_P5(PROCESS#) := 0;                                               CODE(2406):  MOV  -14(5),-(6)
                                                                                CODE(2412):  MOV  #100,-(6)
                                                                                CODE(2416):  MOV  #17440,2
                                                                                CODE(2422):  MOV  (6)+,0
                                                                                CODE(2424):  ADD  (6)+,2
                                                                                CODE(2426):  MOVB0,0(2)
```

```
1701        PT_FLAGS(PROCESS#) := READY;

1702        PT_IPC_QUEUE_HEAD(PROCESS#) := 0;

1703        RC := OK_FLAG;

FIXUP LOCATION 14 TO 20
              81 LINES WERE COMPILED, GENERATING 1334 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA STOPP       5
1704   _!_
```

```
  12432
CODE(2432): MOV -14(5),-(6)
CODE(2436): CLR -(6)
CODE(2440): MOV #17640,2
CODE(2444): MOV (6)+,0
CODE(2446): ADD (6)+,2
CODE(2450): MOVB0,0(2)
  12454
CODE(2454): MOV #-1,4(5)
  12462
CODE(2462): JMP @-20(5)
```

10

```
1705    DATA STOPP;                                                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1706    _I_                                                            10
```

```
1707   PROGRAM STOPP;
1708         DECLARE
1709               WORD (I, ASTE#, DUMMY);

1710         /*    CLEAR OUT "B"




FIXUP LOCATION 30 TO 2
1711         DO I := SEG#_MIN TO SEG#_MAX;




1712               ASTE# := PS_SEG(I);

1713               IF (ASTE# & SEG_FLAG) = ^;




1714                  THEN: DCONNECT(I, ASTE#);
FIXUP LOCATION 120 TO 34
1715               END;

FIXUP LOCATION 44 TO 112

1716         END;

1717         /*    CLEAR OUT IPC QUEUE
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #2,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV #37,-(6)
CODE(22): MOV #1,-(6)
CODE(26): JMP 453(7)
CODE(32): INC (6)

  |52
CODE(34): CMP (6),2(6)
CODE(40): BLE 2
CODE(42): JMP 453(7)
CODE(46): MOV (6),-6(5)
CODE(52): MOV -6(5),-(6)
CODE(56): ASL (6)
CODE(60): MOV #20200,2
CODE(64): ADD (6)+,2
CODE(66): MOV 0(2),-10(5)
  |74
  |
  |122
CODE(74): MOV -10(5),-(6)
CODE(100): MOV #-100000,-(6)
CODE(104): COM (6)
CODE(106): BIC (6)+,(6)
CODE(110): CLR -(6)
CODE(112): CMP (6)+,(6)+
CODE(114): BEQ 2
CODE(116): JMP 0(7)
CODE(122): MOV 5,-(6)
CODE(124): MOV (5),-(6)
CODE(126): CLR -(6)
CODE(130): MOV -6(5),-(6)
CODE(134): MOV -10(5),-(6)
CODE(140): MOV -2(4),2
CODE(144): JSR 7,#14(2)
CODE(150): MOV 5,6
CODE(152): CLR (6)+
CODE(154): MOV (6)+,5
  |156

  |156
  |

CODE(156): BR  -53
CODE(160): CMP (6)+,(6)+
  |162
  |
*/|162
  |
```

```
1718        I := PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS);

1719        IF I <> 0;
1720          THEN:

1721            CYCLE




PIKUP LOCATION 256 TO 4
1722            .... EXIT WHEN IPC_LINK(I) = 0;



1723                I := IPC_LINK(I);
PIKUP LOCATION 262 TO 24

1724            END;




1725            IPC_LINK(I) := IPC_LINK(0);




1726            IPC_LINK(0) := PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS);
```

```
CODE(162): MOV #20100,2
CODE(166): MOV 0(2),-(6)
CODE(172): MOV #17640,2
CODE(176): ADD (6)+,2
CODE(200): MOVB0(2),1
CODE(204): MOV 1,-6(5)
  |210
  |
  |226
  |226
  |
  |226
CODE(210): MOV -6(5),-(6)
CODE(214): CLR -(6)
CODE(216): CMP (6)+,(6)+
CODE(220): BNE 2
CODE(222): JMP 0(7)
CODE(226): MOV -6(5),-(6)
CODE(232): MOV #14400,2
CODE(236): ADD (6)+,2
CODE(240): MOVB0(2),1
CODE(244): MOV 1,-(6)
CODE(246): CLR -(6)
CODE(250): CMP (6)+,(6)+
CODE(252): BEQ 2
CODE(254): JMP 0(7)
CODE(260): JMP 453(7)

  |264
CODE(264): MOV -6(5),-(6)
CODE(270): MOV #14400,2
CODE(274): ADD (6)+,2
CODE(276): MOVB0(2),1
CODE(302): MOV 1,-6(5)
  |306

CODE(306): BR -31
  |310
  |
CODE(310): MOV -6(5),-(6)
CODE(314): CLR -(6)
CODE(316): MOV #14400,2
CODE(322): ADD (6)+,2
CODE(324): MOVB0(2),1
CODE(330): MOV 1,-(6)
CODE(332): MOV #14400,2
CODE(336): MOV (6)+,0
CODE(340): ADD (6)+,2
CODE(342): MOVB0,0(2)
  |346
CODE(346): CLR -(6)
CODE(350): MOV #20100,2
CODE(354): MOV 0(2),-(6)
CODE(360): MOV #17640,2
CODE(364): ADD (6)+,2
CODE(366): MOVB0(2),1
CODE(372): MOV 1,-(6)
CODE(374): MOV #14400,2
CODE(400): MOV (6)+,0
CODE(402): ADD (6)+,2
CODE(404): MOVB0,0(2)
  |410
CODE(410): MOV #20100,2
CODE(414): MOV 0(2),-(6)
CODE(420): CLR -(6)
CODE(422): MOV #17640,2
```

```
1727                PT_IPC_QUEUE_HEAD(PS_CURRENT_PROCESS) := 0;
FIXUP LOCATION 224 TO 230
1728            END;

1729            /*    GET RID OF K STACK

1730            ASTE# := PT_KS_ASTE#(PS_CURRENT_PROCESS);

1731            AST_SWAP_CHAIN(ASTE#) := AST_SWAP_CHAIN(0);

1732            AST_SWAP_CHAIN(0) := ASTE#;

1733            AST_UNLOCK(ASTE#) := (AST_UNLOCK(ASTE#) | AST_UNLOCK_FLAG);
1734            /*    LET EXEC KNOW WHAT'S HAPPENING

1735            IPCSEND(EXEC_PROCESS#, 0, KERNEL_DOMAIN);
```

```
CODE(426): MOV (6)+,0
CODE(430): ADD (6)+,2
CODE(432): MOVB0,0(2)
  |436

  |436
  |
*/|436
  |
CODE(436): MOV #20100,2
CODE(442): MOV 0(2),-(6)
CODE(446): ASL (6)
CODE(450): MOV #17540,2
CODE(454): ADD (6)+,2
CODE(456): MOV 0(2),-10(5)
  |464
CODE(464): MOV -10(5),-(6)
CODE(470): ASL (6)
CODE(472): CLR -(6)
CODE(474): ASL (6)
CODE(476): MOV #10400,2
CODE(5'2): ADD (6)+,2
CODE(504): MOV 0(2),-(6)
CODE(510): MOV #10400,2
CODE(514): MOV (6)+,0
CODE(516): ADD (6)+,2
CODE(520): MOV 0,0(2)
  |524
CODE(524): CLR -(6)
CODE(526): ASL (6)
CODE(530): MOV -10(5),-(6)
CODE(534): MOV #10400,2
CODE(540): MOV (6)+,0
CODE(542): ADD (6)+,2
CODE(544): MOV 0,0(2)
  |550
CODE(550): MOV -10(5),-(6)
CODE(554): MOV -10(5),-(6)
CODE(560): MOV #4400,2
CODE(564): ADD (6)+,2
CODE(566): MOVB0(2),1
CODE(572): MOV 1,-(6)
CODE(574): BIS #20,(6)
CODE(600): MOV #4400,2
CODE(604): MOV (6)+,0
CODE(606): ADD (6)+,2
CODE(610): MOVB0,0(2)
  |614
  |
*/|614
  |
CODE(614): MOV 5,-(6)
CODE(616): MOV (5),-(6)
CODE(620): CLR -(6)
CODE(622): MOV #1,-(6)
CODE(626): CLR -(6)
CODE(630): MOV #200,-(6)
CODE(634): MOV -2(4),2
CODE(640): JSR 7,#20(2)
CODE(644): MOV 5,6
CODE(646): CLR (6)+
CODE(650): MOV (6)+,5
  |652
CODE(652): MOV #20100,2
CODE(656): MOV 0(2),-(6)
CODE(662): MOV #200,-(6)
```

```
1736          PI_FLAGS(PS_CURRENT_PROCESS) := INACTIVE;

1737          V(KERNEL_SMPR);

1738          SLEEP;

FIXUP LOCATION 14 TO 6
                33 LINES WERE COMPILED, GENERATING 498 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA CHANGED      *
1739    _!_
```

```
CODE(666): MOV #1/440,2
CODE(672): MOV (6)+,0
CODE(674): ADD (6)+,2
CODE(676): MOVB0,0(2)
  1702
CODE(702): MOV 5,-(6)
CODE(704): MOV (5),-(6)
CODE(706): CLR -(6)
CODE(710): MOV #400,-(6)
CODE(714): MOV -2(4),2
CODE(720): JSR 7,@130(2)
CODE(724): MOV 5,6
CODE(726): CLR (6)+
CODE(730): MOV (6)+,5
  1732
CODE(732): MOV 5,-(6)
CODE(734): MOV (5),-(6)
CODE(736): CLR -(6)
CODE(740): MOV -2(4),2
CODE(744): JSR 7,@114(2)
CODE(750): MOV 5,6
CODE(752): CLR (6)+
CODE(754): MOV (6)+,5
  1756
CODE(756): JMP @-4(5)
```

```
1740   DATA CHANGED(ASTE#, OFFSET, CLASS, CAT) RETURNS (RC):                    10
                2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                                        10
1741   _!_
```

```
1742    PROGRAM CHANGEO;
1743          DECLARE
1744                WORD (OASTE0);

1745          /*    SECURITY CHECKS

1746          /*    ONLY TRUSTED SUBJECTS CAN USE THIS FUNCTION

1747          IF PS_CURRENT_PROCESS ¬= EXEC_PROCESS0;

1748             THEN:
1749    ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 36 TO 12
1750          END;

1751          /*    INTERPRETIVE DIRECTORY WRITE CHECK

1752          IF WRITEDIR(ASTE0) ¬= OK_FLAG;

1753             THEN:
1754    ....        RETURN WITH ERR_FLAG;
FIXUP LOCATION 114 TO 12
1755          END;

1756          IF DIR_SIZE(OFFSET) = 0;
```

```
  |16
  |16
  |16
  |
*/|16
  |
*/|16
  |
  |40
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #12,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV #20100,2
CODE(22): MOV 0(2),-(6)
CODE(26): CMP #1,(6)+
CODE(32): BNE 2
CODE(34): JMP 0(7)
CODE(40): MOV #-2,4(5)
CODE(46): JMP @-14(5)
  |52
  |52

  |52
  |
*/|52
  |
  |116
CODE(52): CLR -(6)
CODE(54): MOV 5,-(6)
CODE(56): MOV (5),-(6)
CODE(60): CLR -(6)
CODE(62): MOV -4(5),-(6)
CODE(66): MOV -2(4),2
CODE(72): JSR 7,@170(2)
CODE(76): MOV 5,6
CODE(100): CLR (6)+
CODE(102): MOV (6)+,5
CODE(104): CMP #-1,(6)+
CODE(110): BNE 2
CODE(112): JMP 0(7)
CODE(116): MOV #-2,4(5)
CODE(124): JMP @-14(5)
  |130
  |130

  |130
  |
  |162
CODE(130): MOV -6(5),-(6)
CODE(134): ASL (6)
CODE(136): MOV #60600,2
CODE(142): ADD (6)+,2
CODE(144): MOV 0(2),-(6)
CODE(150): CLR -(6)
CODE(152): CMP (6)+,(6)+
CODE(154): BEQ 2
```

```
1757          THEN:
1758   ....       RETURN WITH ERR_FLAG;
FIXUP LOCATION 160 TO 12
1759          END;

1760          /*    IF OBJECT IS INACTIVE THEN NO NEED TO DO SECURITY & * -PROPERTY CHECK
```

```
CODE(156): JMP 0(7)
CODE(162): MOV #-2,#(5)
CODE(170): JMP #-14(5)
  |174
  |174

  |174
  |
*/|174
  |
CODE(174): CLR -(6)
CODE(176): MOV 5,-(6)
CODE(200): MOV (5),-(6)
CODE(202): CLR -(6)
CODE(204): MOV -6(5),-(6)
CODE(210): ASL (6)
CODE(212): MOV #60400,2
CODE(216): ADD (6)+,2
CODE(220): MOV 0(2),-(6)
CODE(224): MOV -2(6),2
CODE(230): JSR 7,#174(2)
CODE(234): MOV 5,6
CODE(236): CLR (6)+
CODE(240): MOV (6)+,5
CODE(242): MOV (6)+,-16(5)
  |246
  |
  |306
CODE(246): MOV -16(5),-(6)
CODE(252): CLR -(6)
CODE(254): CMP (6)+,(6)+
CODE(256): BNE 2
CODE(260): CLR -(6)
CODE(262): BR  2
CODE(264): MOV #1,-(6)
CODE(270): MOV -16(5),-(6)
CODE(274): ASL (6)
CODE(276): MOV #11400,2
CODE(302): ADD (6)+,2
CODE(304): MOV 0(2),-(6)
CODE(310): MOV #77777,-(6)
CODE(314): COM (6)
CODE(316): BIC (6)+,(6)
CODE(320): CLR -(6)
CODE(322): CMP (6)+,(6)+
CODE(324): BNE 2
CODE(326): CLR -(6)
CODE(330): BR  2
CODE(332): MOV #1,-(6)
CODE(336): BIT (6)+,(6)+
CODE(340): BNE 2
CODE(342): JMP 0(7)
CODE(346): MOV #-2,#(5)
CODE(354): JMP #-14(5)
  |360
  |360

  |360
  |
*/|360
  |
  |422
CODE(360): MOV -6(5),-(6)
CODE(364): MOV #60000,2
CODE(370): ADD (6)+,2
CODE(372): MOVB0(2),1
```

```
1761          OASTE# := HASH(DIR_DISK(OFFSET));

1762          IF (OASTE# >= 0) & ((AST_CPL(OASTE#) & WIRED_DOWN_NOTMASK) >= 0);
```

```
1763          THEN:
1764   ....       RETURN WITH ERR_FLAG;
FIXUP LOCATION 344 TO 12
1765          END;

1766          /*    COMPATABILITY CHECK SIMPLIFIED IF OBJECT IS NOT A DIRECTORY

1767          IF (DIR_TYPE(OFFSET) & DIR_TYPE_MASK) = DIR_TYPE_DIRECTORY;
```

```
1768          THEN:
1769    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 420 TO 12
1770          END;

1771          IF CLASS < (AST_CLASS(ASTE#) & AST_CLASS_MASK);

1772          THEN:
1773    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 476 TO 12
1774          END;

1775          IF CAT >= (AST_CAT(ASTE#) | CAT);

1776          THEN:
1777    ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 550 TO 12
1778          END;

1779          /*    CHECKING COMPLETE - PERFORM STATE CHANGE
```

```
CODE(376):  MOV 1,-(6)
CODE(400):  MOV #200,-(6)
CODE(404):  COM (6)
CODE(406):  BIC (6)+,(6)
CODE(410):  CMP #200,(6)+
CODE(414):  BEQ 2
CODE(416):  JMP 0(7)
CODE(422):  MOV #-2,4(5)
CODE(430):  JMP a-14(5)
  |434
  |434

  |434
  |
  |500
CODE(434):  MOV -4(5),-(6)
CODE(440):  MOV #4400,2
CODE(444):  ADD (6)+,2
CODE(446):  MOVB0(2),1
CODE(452):  MOV 1,-(6)
CODE(454):  MOV #17,-(6)
CODE(460):  COM (6)
CODE(462):  BIC (6)+,(6)
CODE(466):  MOV -10(5),3
CODE(470):  CMP (6)+,3
CODE(472):  BGT 2
CODE(474):  JMP 0(7)
CODE(500):  MOV #-2,4(5)
CODE(506):  JMP a-14(5)
  |512
  |512

  |512
  |
  |552
CODE(512):  MOV -4(5),-(6)
CODE(516):  ASL (6)
CODE(520):  MOV #13400,2
CODE(524):  ADD (6)+,2
CODE(526):  MOV 0(2),-(6)
CODE(532):  BIS -12(5),(6)
CODE(536):  MOV -12(5),3
CODE(542):  CMP (6)+,3
CODE(544):  BNE 2
CODE(546):  JMP 0(7)
CODE(552):  MOV #-2,4(5)
CODE(560):  JMP a-14(5)
  |564
  |564

  |564
  |
*/|564
  |
CODE(564):  MOV -6(5),-(6)
CODE(570):  MOV -6(5),-(6)
CODE(574):  MOV #60000,2
CODE(600):  ADD (6)+,2
CODE(602):  MOVB0(2),1
CODE(606):  MOV 1,-(6)
CODE(610):  MOV #360,-(6)
CODE(614):  COM (6)
CODE(616):  BIC (6)+,(6)
CODE(620):  BIS -10(5),(6)
CODE(624):  MOV #60000,2
CODE(630):  MOV (6)+,0
```

```
1780        DIR_CLASS(OFFSET) := ((DIR_CLASS(OFFSET) & DIR_CLASS_NOTMASK) | CLASS);

1781        DIR_CAT(OFFSET) := CAT;
```

```
CODE(632): ADD (6)+,2
CODE(634): MOVB0,0(2)
  |640
CODE(640): MOV -6(5),-(6)
CODE(644): ASL (6)
CODE(646): MOV -12(5),-(6)
CODE(652): MOV #60200,2
CODE(656): MOV (6)+,0
CODE(660): ADD (6)+,2
CODE(662): MOV 0,0(2)
  |666
  |
```

```
1782        /*    IF WIRED DOWN CHANGE ASTE CLASS AND CAT ALSO
1783        IF OASTE# ¬= 0;
1784          THEN: AST_CLASS(OASTE#) := DIR_CLASS(OFFSET);
1785            AST_CAT(OASTE#) := CAT;
FIXUP LOCATION 702 20 66
1786        END;
1787        RC := OK_FLAG;

FIXUP LOCATION 14 TO 2
              47 LINES WERE COMPILED, GENERATING 516 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA INITH       5
1788   _!_
```

```
*/1666
  |
  1704
CODE(666): MOV -16(5),-(6)
CODE(672): CLR -(6)
CODE(674): CMP (6)+,(6)+
CODE(676): BNE 2
CODE(700): JMP 0(7)
CODE(704): MOV -16(5),-(6)
CODE(710): MOV -6(5),-(6)
CODE(714): MOV #60000,2
CODE(720): ADD (6)+,2
CODE(722): MOVB0(2),1
CODE(726): MOV 1,-(6)
CODE(730): MOV #4400,2
CODE(734): MOV (6)+,0
CODE(736): ADD (6)+,2
CODE(740): MOVB0,0(2)
  1744
CODE(744): MOV -16(5),-(6)
CODE(750): ASL (6)
CODE(752): MOV -12(5),-(6)
CODE(756): MOV #13400,2
CODE(762): MOV (6)+,0
CODE(764): ADD (6)+,2
CODE(766): MOV 0,0(2)
  1772

  1772
  |
CODE(772): MOV #-1,#(5)
  1000
CODE(1000): JMP #-14(5)


 10
```

```
1789    DATA INITH(DASTE@, OFFSET, ASTE@) RETURNS (PC);                    10
                  2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1790    _!_                                                                10
```

```
1791   PROGRAM INITH;
1792         /*    SECURITY CHECKS
1793         /*    ONLY TRUSTED SUBJECTS CAN USE THIS FUNCTION
1794         IF PS_CURRENT_PROCESS ¬= EXEC_PROCESS#;
1795           THEN;
1796   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 34 TO 12
1797         END;
1798         /*    INTERPRETIVE DIRECTORY WRITE CHECK
1799         IF WRITEDIR(DASIR#) ¬= OK_FLAG;
1800           THEN;
1801   ....      RETURN WITH ERR_FLAG;
FIXUP LOCATION 114 TO 12
1802         END;
1803         /*    IMPLEMENTATION CHECKS
1804         IF DIR_SIZE(OFFSET) ¬= 0;
```

```
  |16
  |
*/|16
  |
*/|16
  |
  |40
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #10,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV #2C100,2
CODE(22): MOV 0(2),-(6)
CODE(26): CMP #1,(6)+
CODE(32): BNE 2
CODE(34): JMP 0(7)
CODE(40): MOV #-2,#(5)
CODE(46): JMP #-12(5)
  |52
  |52

  |52
  |
*/|52
  |
  |116
CODE(52): CLR -(6)
CODE(54): MOV 5,-(6)
CODE(56): MOV (5),-(6)
CODE(60): CLR -(6)
CODE(62): MOV -4(5),-(6)
CODE(66): MOV -2(4),2
CODE(72): JSR 7,#17C(2)
CODE(75): MOV 5,6
CODE(100): CLR (6)+
CODE(102): MOV (6)+,5
CODE(104): CMP #-1,(6)+
CODE(110): BNE 2
CODE(112): JMP 0(7)
CODE(116): MOV #-2,#(5)
CODE(124): JMP #-12(5)
  |130
  |130

  |130
  |
*/|130
  |
  |162
CODE(130): MOV -4(5),-(6)
CODE(134): ASL (6)
CODE(136): MOV #6C600,2
CODE(142): ADD (6)+,2
CODE(144): MOV 0(2),-(6)
CODE(150): CLR -(6)
CODE(152): CMP (6)+,(6)+
CODE(154): BNE 2
```

```
1805          THEN;
1806    ....      RETURN WITH PRB_FLAG;
FIXUP LOCATION 160 TO 12
1807          END;

1808          /*    PERFORM STATE CHANGE - COPY ASTE ATTRIBUTES INTO DIRECTORY ENTRY

1809          DIR_CLASS(OFFSET) := AST_CLASS(ASTE#);

1810          DIR_CAT(OFFSET) := AST_CAT(ASTE#);

1811          DIR_DISK(OFFSET) := AST_DISK(ASTE#);

1812          DIR_SIZE(OFFSET) := AST_SIZE(ASTE#);

1813          DIR_ACL_HEAD(OFFSET) := 0;

1814          RC := OK_FLAG;
```

```
CODE(156): JMP 0(7)
CODE(162): MOV #-2,4(5)
CODE(170): JMP #-12(5)
  |174
  |174

  |174
  |
*/|174
  |
CODE(174): MOV -6(5),-(6)
CODE(200): MOV -10(5),-(6)
CODE(204): MOV #4400,2
CODE(210): ADD (6)+,2
CODE(212): MOVB0(2),1
CODE(216): MOV 1,-(6)
CODE(220): MOV #60000,2
CODE(224): MOV (6)+,0
CODE(226): ADD (6)+,2
CODE(230): MOVB0,0(2)
  |234
CODE(234): MOV -6(5),-(6)
CODE(240): ASL (6)
CODE(242): MOV -10(5),-(6)
CODE(246): ASL (6)
CODE(250): MOV #13400,2
CODE(254): ADD (6)+,2
CODE(256): MOV 0(2),-(6)
CODE(262): MOV #60200,2
CODE(266): MOV (6)+,0
CODE(270): ADD (6)+,2
CODE(272): MOV 0,0(2)
  |276
CODE(276): MOV -6(5),-(6)
CODE(302): ASL (6)
CODE(304): MOV -10(5),-(6)
CODE(310): ASL (6)
CODE(312): MOV #5400,2
CODE(316): ADD (6)+,2
CODE(320): MOV 0(2),-(6)
CODE(324): MOV #60400,2
CODE(330): MOV (6)+,0
CODE(332): ADD (6)+,2
CODE(334): MOV 0,0(2)
  |340
CODE(340): MOV -6(5),-(6)
CODE(344): ASL (6)
CODE(346): MOV -10(5),-(6)
CODE(352): MOV #5000,2
CODE(356): ADD (6)+,2
CODE(360): MOVB0(2),1
CODE(364): MOV 1,-(6)
CODE(366): MOV #60600,2
CODE(372): MOV (6)+,0
CODE(374): ADD (6)+,2
CODE(376): MOV 0,0(2)
  |402
CODE(402): MOV -6(5),-(6)
CODE(406): CLR -(6)
CODE(410): MOV #60100,2
CODE(414): MOV (6)+,0
CODE(416): ADD (6)+,2
CODE(420): MOVB0,0(2)
  |424
CODE(424): MOV #-1,4(5)
  |432
```

```
PICKUP LOCATION 14 TO 2                                          CODE(432): JMP 0-12(5)
                25 LINES WERE COMPILED, GENERATING 286 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA P           F
1915   _1_                                                       10
```

```
1016   DATA P(SNFR0);                                                    I0
                    2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1017   _I_                                                               I0
```

```
1818   PROGRAM P;                                                          |16
                                                                           |
1819        /*    BLOCK INTERRUPTS                                       */|16
                                                                           |
                                                                    CODE(0):      2,-(0)
                                                                    CODE(2): MOV 6,5
                                                                    CODE(4): ADD #4,5
                                                                    CODE(10): MOV 7,(5)+
                                                                    CODE(12): SUB #0,6
                                                                           |20
1820        INLINE(SPLHIGH);                                           CODE(20): MOV -4(5),-(6)
                                                                    CODE(24): MOV -4(5),-(6)
                                                                    CODE(30): MOV #15534,2
                                                                    CODE(34): ADD (6)+,2
                                                                    CODE(36): MOVB0(2),1
                                                                    CODE(42): MOV 1,-(6)
                                                                    CODE(44): SUB #1,(6)
                                                                    CODE(50): MOV #15534,2
                                                                    CODE(54): MOV (6)+,0
                                                                    CODE(56): ADD (6)+,2
                                                                    CODE(60): MOVB0,0(2)
                                                                           |64
1821        SMFR_COUNT(SMFR#) := SMFR_COUNT(SMFR#) - 1;                    |
                                                                           |
1822        /*    IF SEMAPHORE COUNT IS NEGATIVE, THEN PROCESS BECOMES BLOCKED */|64
                                                                           |
1823        IF SMFR_COUNT(SMFR#) < 0;                                      |116
1824          THEN;                                                        |116
                                                                           |
1825             /*    ADD CURRENT PROCESS TO QUEUE OF PROCESSES BLOCKED ON SEMAPHORE */|116
                                                                           |
                                                                    CODE(64): MOV -4(5),-(6)
                                                                    CODE(70): MOV #15534,2
                                                                    CODE(74): ADD (6)+,2
                                                                    CODE(76): MOVB0(2),1
                                                                    CODE(102): MOV 1,-(6)
                                                                    CODE(104): CLR -(6)
                                                                    CODE(106): CMP (6)+,(6)+
                                                                    CODE(110): BGT 2
                                                                    CODE(112): JMP 0(7)
                                                                    CODE(116): MOV #16544,2
                                                                    CODE(122): MOV 0(2),-(6)
                                                                    CODE(126): MOV -4(5),-(6)
                                                                    CODE(132): MOV #16140,2
                                                                    CODE(136): ADD (6)+,2
                                                                    CODE(140): MOVB0(2),1
                                                                    CODE(144): MOV 1,-(6)
                                                                    CODE(146): CLR -(6)
                                                                    CODE(150): BIS (6)+,(6)
                                                                    CODE(152): MOV #17440,2
                                                                    CODE(156): MOV (6)+,0
                                                                    CODE(160): ADD (6)+,2
                                                                    CODE(162): MOVB0,0(2)
                                                                           |166
1826             PT_FLAGS(THE_CURRENT_PROCESS) := (SMFR_POINTER(SMFR#) | BLOCKED);  CODE(166): MOV -4(5),-(6)
                                                                    CODE(172): MOV #16544,2
                                                                    CODE(176): MOV 0(2),-(6)
                                                                    CODE(202): MOV #16140,2
                                                                    CODE(206): MOV (6)+,0
```

```
1827              SMFR_POINTER(SMFR#) := THE_CURRENT_PROCESS;

1828              /*    START A NEW PROCESS RUNNING

1829              IF SMFR# < KRNNEL_SMFR;

1830                THEN: V(KERNEL_SMFR);

1831                   SLEEP;

1832                   P(KERNEL_SMFR);
PIXUP LOCATION 232 TO 110

1833                DLSP: SLEEP;
PIXUP LOCATION 342 TO 24
1834              END;

PIXUP LOCATION 114 TO 252
1835        END;

1836        INLINF(SPLLOW);

PIXUP LOCATION 14 TO 0
          20 LINES WERE COMPILED, GENERATING 254 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA V          5
1837  _!_
```

```
CODE(210): ADD (6)+,2
CODE(212): MOVB0,0(2)
  |216
  |
*/|216
  |
  |234
CODE(216): MOV -4(5),-(6)
CODE(222): CMP #400,(6)+
CODE(226): BGT 2
CODE(230): JMP 0(7)
CODE(234): MOV 5,-(6)
CODE(236): MOV (5),-(6)
CODE(240): CLR -(6)
CODE(242): MOV #400,-(6)
CODE(246): MOV -2(4),2
CODE(252): JSR 7,#130(2)
CODE(256): MOV 5,6
CODE(260): CLR (6)+
CODE(262): MOV (6)+,5
  |264
CODE(264): MOV 5,-(6)
CODE(266): MOV (5),-(6)
CODE(270): CLR -(6)
CODE(272): MOV -2(4),2
CODE(276): JSR 7,#114(2)
CODE(302): MOV 5,6
CODE(304): CLR (6)+
CODE(306): MOV (6)+,5
  |310
CODE(310): MOV 5,-(6)
CODE(312): MOV (5),-(6)
CODE(314): CLR -(6)
CODE(316): MOV #400,-(6)
CODE(322): MOV -2(4),2
CODE(326): JSR 7,#124(2)
CODE(332): MOV 5,6
CODE(334): CLR (6)+
CODE(336): MOV (6)+,5
  |340

CODE(340): JMP 0(7)
CODE(344): MOV 5,-(6)
CODE(346): MOV (5),-(6)
CODE(350): CLR -(6)
CODE(352): MOV -2(4),2
CODE(356): JSR 7,#114(2)
CODE(362): MOV 5,6
CODE(364): CLR (6)+
CODE(366): MOV (6)+,5
  |370

  |370
  |

  |370
  |
  |372
CODE(372): JMP @-6(5)


  10
```

```
1838   DATA V(SMPP#);                                                    10
1839         DECLARE                                                     10
1840               #ORD (PROCESS_A, PROCESS_B);                          10
                  # LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1841   _I_                                                               10
```

```
1842   PROGRAM V;                                                                  |16
                                                                                   |
1843         /*    BLOCK INTERRUPTS                                               */|16
                                                                                   |
                                                                                   CODE(0):      2,-(0)
                                                                                   CODE(2): MOV 6,5
                                                                                   CODE(4): ADD #4,5
                                                                                   CODE(10): MOV 7,(5)+
                                                                                   CODE(12): SUB #0,6
                                                                                   |20
1844         INLINE(SPLHIGH);                                                      CODE(20): MOV -4(5),-(6)
                                                                                   CODE(24): MOV -4(5),-(6)
                                                                                   CODE(30): MOV #15534,2
                                                                                   CODE(34): ADD (6)+,2
                                                                                   CODE(36): MOVB0(2),1
                                                                                   CODE(42): MOV 1,-(6)
                                                                                   CODE(44): ADD #1,(6)
                                                                                   CODE(50): MOV #15534,2
                                                                                   CODE(54): MOV (6)+,0
                                                                                   CODE(56): ADD (6)+,2
                                                                                   CODE(60): MOVB0,0(2)
                                                                                   |64
1845         SMFR_COUNT(SMFR#) := SMFR_COUNT(SMFR#) + 1;                           |
                                                                                   * |64
1846         /*    IF SEMAPHORE COUNT IS NON-POSITIVE, THEN A PROCESS BLOCKED ON THE SEMAPHORE
1847          * MUST BE UNCHEHED                                                   */|64
                                                                                   |
1848         IF SMFR_COUNT(SMFR#)<= 0;                                             |116
1849           THEN:                                                               |116
                                                                                   |
1850              /*    THE MOST RECENT PROCESS ADDED TO THIS SEMAPHORE'S QUEUE    */|116
                                                                                   |
                                                                                   CODE(64): MOV -4(5),-(6)
                                                                                   CODE(70): MOV #15534,2
                                                                                   CODE(74): ADD (6)+,2
                                                                                   CODE(76): MOVB0(2),1
                                                                                   CODE(102): MOV 1,-(6)
                                                                                   CODE(104): CLR -(6)
                                                                                   CODE(106): CMP (6)+,(6)+
                                                                                   CODE(110): BGE 2
                                                                                   CODE(112): JMP 0(7)
                                                                                   CODE(116): MOV -4(5),-(6)
                                                                                   CODE(122): MOV #16140,2
                                                                                   CODE(126): ADD (6)+,2
                                                                                   CODE(130): MOVB0(2),1
                                                                                   CODE(134): MOV 1,-10(5)
                                                                                   |140
1851              PROCESS_A := SMFR_POINTFR(SMFR#);                                |
                                                                                   * |140
1852              /*    IF THERE IS MORE THAN ONE PROCESS ON QUEUE, FOLLOW CHAIN THROUGH
1853               * PROCESS TABLE                                                 */|140
                                                                                   |
1854              IF SMFR_COUNT(SMFR#) ~= 0;                                       |172
1855                THEN:                                                          |172
                                                                                   |
1856                   CYCLE                                                       |172
                                                                                   CODE(140): MOV -4(5),-(6)
                                                                                   CODE(144): MOV #15534,2
                                                                                   CODE(150): ADD (6)+,2
                                                                                   CODE(152): MOVB0(2),1
```

```
FIXUP LOCATION 222 TO 4
1857                    .... EXIT WHEN PT_LINK(PROCESS_A) = 0;

1858                         PROCESS_B := PROCESS_A;



1859                         PROCESS_A := PT_LINK(PROCESS_B);
FIXUP LOCATION 226 TO 32

1860                    END;

1861                    /*    REMOVE PROCESS_A FROM END OF QUEUE



1862                    PT_LINK(PROCESS_B) := 0;
FIXUP LOCATION 170 TO 116



1863                  ELSE: SHEP_POINTR(SHEP#) := 0;
FIXUP LOCATION 306 TO 22
1864               END;

1865               /*    PROCESS_A BECOMES READY



1866               PT_FLAGS(PROCESS_A) := READY;
FIXUP LOCATION 114 TO 240
1867          END;

1868          INLINE(SPLLOW);

FIXUP LOCATION 14 TO 4
                 28 LINES WERE COMPILED, GENERATING 244 BYTES OF CODE.
```

```
CODE(156): MOV 1,-(6)
CODE(160): CLR -(6)
CODE(162): CMP (6)+,(6)+
CODE(164): BNE 2
CODE(166): JMP 0(7)
CODE(172): MOV -10(5),-(6)
CODE(176): MOV #17440,2
CODE(202): ADD (6)+,2
CODE(204): MOVB0(2),1
CODE(210): MOV 1,-(6)
CODE(212): CLR -(6)
CODE(214): CMP (6)+,(6)+
CODE(216): BEQ 2
CODE(220): JMP 0(7)
CODE(224): JMP 453(7)

  |230
CODE(230): MOV -10(5),-12(5)
  |236
CODE(236): MOV -12(5),-(6)
CODE(242): MOV #17440,2
CODE(246): ADD (6)+,2
CODE(250): MOVB0(2),1
CODE(254): MOV 1,-10(5)
  |260

CODE(260): BR  -34
  |262
  |
*/|262
  |
CODE(262): MOV -12(5),-(6)
CODE(266): CLR -(6)
CODE(270): MOV #17440,2
CODE(274): MOV (6)+,0
CODE(276): ADD (6)+,2
CODE(300): MOVB0,0(2)
  |304

CODE(304): JMP 0(7)
CODE(310): MOV -4(5),-(6)
CODE(314): CLR -(6)
CODE(316): MOV #16140,2
CODE(322): MOV (6)+,0
CODE(324): ADD (6)+,2
CODE(326): MOVB0,0(2)
  |332

  |332
  |
*/|332
  |
CODE(332): MOV -10(5),-(6)
CODE(336): MOV #100,-(6)
CODE(342): MOV #17440,2
CODE(346): MOV (6)+,0
CODE(350): ADD (6)+,2
CODE(352): MOVB0,0(2)
  |356

  |356
  |
  |360
CODE(360): JMP @-6(5)
```

```
NO ERRORS WERE DETECTED.
1969 DATA SLEEP          I                                              10
1969   _|_

1970   DATA SLEEP;                                                      10
1971        DECLARE                                                     10
1972                *ORD (NEXT_PROCESS);                                10
                  4 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.                                               10
1973   _|_
```

```
1874    PROGRAM SLEEP;


1875        INLINE(SPLLOW);
1876        /*    FIND NEXT PROCESS THAT CAN RUN - ROUND ROBIN SCHEDULER


1877        NEXT_PROCESS := THE_CURRENT_PROCESS + 1;
1878        CYCLE
1879            IF NEXT_PROCESS > PROCESS#_MAX;


1880              THEN: NEXT_PROCESS := PROCESS#_MIN;
FIXUP LOCATION 54 TO 6
1881            END;
1882            IF (PT_FLAGS(NEXT_PROCESS) & PT_FLAGS_MASK) = READY;


1883              THEN;
1884                EXIT;
FIXUP LOCATION 124 TO 4
1885            END;


1886            NEXT_PROCESS := NEXT_PROCESS + 1;
FIXUP LOCATION 130 TO 16
1887        END;
1888        IF NEXT_PROCESS >= THE_CURRENT_PROCESS;
```

```
  |16
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(6): ADD #2,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
  |20
  |
*/120
  |
CODE(20): MOV #16544,2
CODE(24): MOV 0(2),-(6)
CODE(30): ADD #1,(6)
CODE(34): MOV (6)+,-6(5)
  |40
  |
  |40
  |
  |56
CODE(40): MOV -6(5),-(6)
CODE(44): CMP #7,(6)+
CODE(50): BLT 2
CODE(52): JMP 0(7)
CODE(56): MOV #1,-6(5)
  |64

  |64
  |
  |126
CODE(64): MOV -6(5),-(6)
CODE(70): MOV #17440,2
CODE(74): ADD (6)+,2
CODE(76): MOVB0(2),1
CODE(102): MOV 1,-(6)
CODE(106): MOV #300,-(6)
CODE(110): COM (6)
CODE(112): BIC (6)+,(6)
CODE(114): CMP #100,(6)+
CODE(120): BEQ 2
CODE(122): JMP 0(7)
CODE(126): JMP 453(7)
  |132
  |132

  |132
  |
CODE(132): MOV -6(5),-(6)
CODE(136): ADD #1,(6)
CODE(142): MOV (6)+,-6(5)
  |146

CODE(146): BR  -44
  |150
  |
  |172
CODE(150): MOV -6(5),-(6)
CODE(154): MOV #16544,2
CODE(160): CMP 0(2),(6)+
```

```
1889          THEN: RUN(NEXT_PROCESS);
FIXUP LOCATION 170 TO 30
1890        END;


FIXUP LOCATION 14 TO 2
                 19 LINES WERE COMPILED, GENERATING 151 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA RUN
1891   _!_
```

```
CODE(164): BNE 4
CODE(166): JMP @(7)
CODE(172): MOV 5,-(6)
CODE(174): MOV (5),-(6)
CODE(176): CLR -(6)
CODE(200): MOV -6(5),-(6)
CODE(204): MOV -2(4),2
CODE(210): JSR 7,@144(2)
CODE(214): MOV 5,6
CODE(216): CLR (6)+
CODE(220): MOV (6)+,5
  1222

  1222
  |
CODE(222): JMP @-4(5)


 10
```

```
1897   PROGRAM RUN;                                                                  |24
                                                                                     |
1898          /*    BLOCK INTERRUPTS                                                */|24
                                                                                     |
                                                                                     CODE(0):      2,-(0)
                                                                                     CODE(2): MOV 6,5
                                                                                     CODE(4): ADD #4,5
                                                                                     CODE(10): MOV 7,(5)+
                                                                                     CODE(12): BR  2
                                                                                     CODE(14): =V(CONTAINED PROCEDURE)
                                                                                     CODE(20): SUB #0,6
                                                                                       |26
1899          INLINE(SPLHIGH);                                                       |
                                                                                     * 126
1900          /*    NO NEED TO SAVE PROCESS REGISTERS BECAUSE A COPY IS IN THE PROCESS SEGMENT,
1901           * BUT CHANGE BIT MUST BE INSPECTED                                  */|26
                                                                                     |
                                                                                     CODE(26): MOV #17,-(6)
                                                                                     CODE(32): CLR -(6)
                                                                                     CODE(34): JMP 453(7)
                                                                                     CODE(40): INC (6)

FIXUP LOCATION 36 TO 2                                                                 |60
1902          DO REG# := 2 TO REG#_MAX;                                                |
                                                                                       |76
1903             IF REG# > CROSS_REG#;                                               CODE(42): CMP (6),2(6)
                                                                                     CODE(46): BLE 2
                                                                                     CODE(50): JMP 453(7)
                                                                                     CODE(54): MOV (6),-10(5)
                                                                                     CODE(60): MOV -10(5),-(6)
                                                                                     CODE(64): CMP #7,(6)+
                                                                                     CODE(70): BLT 2
                                                                                     CODE(72): JMP 0(7)
                                                                                     CODE(76): MOV -10(5),-(6)
                                                                                     CODE(102): ADD #257C,(6)
                                                                                     CODE(106): MOV (6)+,-12(5)
                                                                                       |112
1904                THEN: X := REG# * REG_CONSTANT;                                  |
FIXUP LOCATION 74 TO 20                                                              CODE(112): JMP 0(7)
                                                                                     CODE(116): MOV -10(5),-12(5)
                                                                                       |124
1905                ELSE: X := REG#;                                                 |
FIXUP LOCATION 114 TO 6                                                                |124
1906             END;                                                                |
                                                                                       |166
1907             IF (SDR(X) & SDR_CHANGE_MASK) = SDR_CHANGED;                        CODE(124): MOV -12(5),-(6)
                                                                                     CODE(130): ASL (6)
                                                                                     CODE(132): MOV #-5600,2
                                                                                     CODE(136): ADD (6)+,2
                                                                                     CODE(140): MOV 0(2),-(6)
                                                                                     CODE(144): MOV #100,-(6)
                                                                                     CODE(150): COM (6)
                                                                                     CODE(152): BIC (6)+,(6)
                                                                                     CODE(154): CMP #100,(6)+
                                                                                     CODE(160): BEQ 2
                                                                                     CODE(162): JMP 0(7)
                                                                                     CODE(166): MOV -12(5),-(6)
                                                                                     CODE(172): ASL (6)
                                                                                     CODE(174): MOV #-5540,2
```

```
1892   DATA RUN(NEXT_PROCESS);
1893        DECLARE                                                          10
1894             PROCEDURE ACCEPTS (#WORD, WORD) (S#AF),                     10
1895             WORD (REG#, X, VAR);                                        10
                 5 LINES WERE COMPILED.                                      10
NO ERRORS WERE DETECTED.
1896   _I_                                                                   10
```

```
1908                THEN: VAR := SAR(X);
1909                   INLINE(ASR, VAR); /* THEIR BLOCKS TO MINE */
1910                   INLINE(ASR, VAR);
1911                   IF VAR < END_BLOCK#;
1912                      THEN: VAR := NBT_ASTR(VAR);
1913                         AST_CHANGE(VAR) := (AST_CHANGE(VAR) | AST_CHANGED);
FIXUP LOCATION 234 TO 66
1914                   END;
FIXUP LOCATION 164 TO 136
1915             END;
FIXUP LOCATION 52 TO 252
1916        END;
1917        /*    SAVE KSR1
1918        PT_KSAR1(THE_CURRENT_PROCESS) := KSAR1;
1919        PT_KSDR1(THE_CURRENT_PROCESS) := KSDR1;
1920        /*    LOAD DESCRIPTOR FOR NEXT PROCESS'S PS
```

```
CODE(200): ADD (6)+,2
CODE(202): MOV 0(2),-14(5)
  |210
CODE(210): ASR -14(5)
  |214
CODE(214): ASR -14(5)
  |220
  |
  |236
CODE(220): MOV -14(5),-(6)
CODE(224): CMP #1740,(6)+
CODE(230): BGT 2
CODE(232): JMP 0(7)
CODE(236): MOV -14(5),-(6)
CODE(242): ASL (6)
CODE(244): MOV #400,2
CODE(250): ADD (6)+,2
CODE(252): MOV 0(2),-14(5)
  |260
CODE(260): MOV -14(5),-(6)
CODE(264): MOV -14(5),-(6)
CODE(270): MOV #4400,2
CODE(274): ADD (6)+,2
CODE(276): MOVB0(2),1
CODE(302): MOV 1,-(6)
CODE(304): BIS #40,(6)
CODE(310): MOV #4400,2
CODE(314): MOV (6)+,0
CODE(316): ADD (6)+,2
CODE(320): MOVB0,0(2)
  |324
  |324
  |
  |324
  |
CODE(324): BR  -133
CODE(326): CMP (6)+,(6)+
  |330
  |
*/|330
  |
CODE(330): MOV #16544,2
CODE(334): MOV 0(2),-(6)
CODE(340): ASL (6)
CODE(342): MOV #-5476,2
CODE(346): MOV 0(2),-(6)
CODE(352): MOV #17040,2
CODE(356): MOV (6)+,0
CODE(360): ADD (6)+,2
CODE(362): MOV 0,0(2)
  |366
CODE(366): MOV #16544,2
CODE(372): MOV 0(2),-(6)
CODE(376): ASL (6)
CODE(400): MOV #-5476,2
CODE(404): MOV 0(2),-(6)
CODE(410): MOV #17000,2
CODE(414): MOV (6)+,0
CODE(416): ADD (6)+,2
CODE(420): MOV 0,0(2)
  |424
  |
*/|424
```

```
1421        ENQ(PT_PS_ADDR#(NEXT_PROCESS), PS_MSR_ADR, SDR_WRITE_ACCESS);
1422        /*    TRANSFER (AND ENABLE) TO DO REAL SWAP IN PAL

1423        SWAP(ONE_CURRENT_PROCESS, NEXT_PROCESS);
1424        IMITMT(SPILL#);

ERROR LOCATION AS TO P
                24 LINES READ COMPILED, GENERATING 358 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA HASH          T
1325    _I_
```

```
  |
CODE(424): MOV 5,-(6)
CODE(426): MOV (5),-(6)
CODE(430): CLR -(6)
CODE(432): MOV -4(5),-(6)
CODE(436): ASL (6)
CODE(440): MOV #17630,2
CODE(444): ADD (6)+,2
CODE(446): MOV 0(2),-(6)
CODE(452): MOV #-5476,-(6)
CODE(456): MOV #6,-(6)
CODE(462): MOV -2(4),2
CODE(466): JSR 7,#154(2)
CODE(472): MOV 5,6
CODE(474): CLR (6)+
CODE(476): MOV (6)+,5
  |500
  |
*/|500
  |
CODE(500): MOV 5,-(6)
CODE(502): MOV 5,-(6)
CODE(504): CLR -(6)
CODE(506): MOV #16544,2
CODE(512): MOV 0(2),-(6)
CODE(516): MOV -4(5),-(6)
CODE(522): MOV -2(5),2
CODE(526): JSR 7,#4(2)
CODE(532): MOV 5,6
CODE(534): CLR (6)+
CODE(536): MOV (6)+,5
  |540
  |542
CODE(542): JMP #-6(5)
```

```
1926   DATA HASH(DISK_ADR) RETURNS (ASTE#);                              10
                    2 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
1927   _|_                                                               10
```

```
1928   PROGRAM HASH;
1929         DECLARE
1930               WORD (HASH_VAL);

1931         /*    HASH DISK_ADR




1932         INLINE(MOV, DISK_ADR, 0, 0);

1933         INLINE(MOV, 0, 0, 0, 1);
1934         INLINE(ASHR1);
1935         INLINE("PFFRM");
1936         INLINE(XOR10);

1937         INLINE(MOV, 0, 0, HASH_VAL);




1939         ASTE# := HASH_TABLE(HASH_VAL & MODFRM);

1939         CYCLE




FIXUP LOCATION 104 TO 4
1940   ....       RETURN WHEN ASTE# = 0;




FIXUP LOCATION 142 TO 4
1941   ....       RETURN WHEN AST_DISK(ASTE#) = DISK_ADR;



1942               ASTE# := AST_CHAIN(ASTE#);

1943         END;
```

```
  |16
  |16
  |16
  |
*/|16
  |
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),0
  |22
CODE(22): MOV 0,1
  |24
  |26
  |30
  |32
CODE(32): MOV 0,-10(5)
  |36
CODE(36): MOV -10(5),-(6)
CODE(42): MOV #377,-(6)
CODE(46): COM (6)
CODE(50): BIC (6)+,(6)
CODE(52): ASL (6)
CODE(54): MOV #3400,2
CODE(60): ADD (6)+,2
CODE(62): MOV 0(2),4(5)
  |70
  |
  |70
CODE(70): MOV 4(5),-(6)
CODE(74): CLR -(6)
CODE(76): CMP (6)+,(6)+
CODE(100): BEQ 2
CODE(102): JMP 0(7)
CODE(106): JMP @-6(5)

  |112
CODE(112): MOV 4(5),-(6)
CODE(116): ASL (6)
CODE(120): MOV #5400,2
CODE(124): ADD (6)+,2
CODE(126): MOV 0(2),-(6)
CODE(132): CMP -4(5),(6)+
CODE(136): BEQ 2
CODE(140): JMP 0(7)
CODE(144): JMP @-6(5)

  |150
CODE(150): MOV 4(5),-(6)
CODE(154): ASL (6)
CODE(156): MOV #6400,2
CODE(162): ADD (6)+,2
CODE(164): MOV 0(2),4(5)
  |172
CODE(172): BR  -42
  |174
```

```
FIXUP LOCATION 14 TO 2
                17 LINES WERE COMPILED, GENERATING 128 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA PREHASH     5
1948   _I_
```

```
1
CODE(174): JMP 0-6(5)
```

10

```
1945   DATA PREHASH(DISK_ADR) RETURNS (HASH_VAL);                    10
                  2 LINES WERE COMPILED.
NO PROBS WERE DETECTED.
1946   _|_                                                           10
```

```
1947   PROGRAM PREHASH;



1948         INLINE(MOV, DISK_ADR, 0, 0);

1949         INLINE(MOV, 0, 0, 0, 1);
1950         INLINE(ASHP1);
1951         INLINE("PERFM");
1952         INLINE(XOS10);

1953         INLINE(MOV, 0, 0, HASH_VAL);



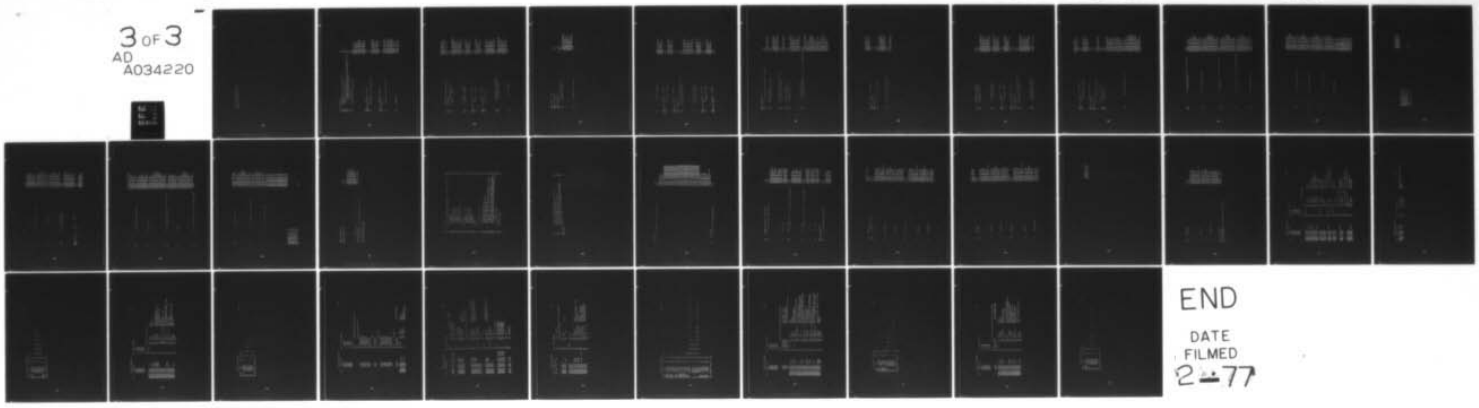
1954         HASH_VAL := (HASH_VAL & 7FFFH);

FIXUP LOCATION 14 TO 2
                0 LINES WERE COMPILED, GENERATING 52 BYTES OF CODE.
NO ERRORS WERE DETECTED.
DATA PCHECK      F
1955   _!_
```

```
  !16
CODE(0):      2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #4,5
CODE(10): MOV 7,(5)+
CODE(12): SUB #0,6
CODE(16): MOV -4(5),0
  !22
CODE(22): MOV 0,1
  !24
  !26
  !30
  !32
CODE(32): MOV 0,4(5)
  !36
CODE(36): MOV 4(5),-(6)
CODE(42): MOV #377,-(6)
CODE(46): COM (6)
CODE(50): BIC (6)+,(6)
CODE(52): MOV (6)+,4(5)
  !56
CODE(56): JMP @-6(5)
```

AD-A034 220 MITRE CORP BEDFORD MASS F/G 9/2
COMPUTER PROGRAM SPECIFICATION FOR THE SECURITY KERNEL FOR THE --ETC(U)
OCT 76 S R HARPER F19628-77-C-0001
UNCLASSIFIED MTR-3178-VOL-2 ESD-TR-76-288-VOL-2 NL

3 OF 3
AD
A034220



END
DATE
FILMED
2-77

```
1956   DATA PCHECK RETURNS (PC);                                    10
                    2 LINES ●ERF COMPILED.
NO ERRORS ●ERE DETECTED.
1957   _!_                                                          10
```

```
1958   PROGRAM PCHECK;                                                                                                  |16
                                                                                                                        |
1959         TYPE FUNCTION_CODE_TYPE = (0 TO FUNCTION_CODE_MAX);                                                        |16
1960         DECLARE                                                                                                    |16
1961              FUNCTION_CODE_TYPE (FUNCTION_CODE);                                                                   |16
1962         DECLARE                                                                                                    |16
1963              WORD (SEG0_PARM, OFFSET_PARM, CLASS_PARM, CAT_PARM, SEG_TYPE_PARM, SIZE_PARM,                         |16
1964                    MODE_PARM, USER_PARM, PROJECT_PARM, SEG#_PARM, PROCESS#_PARM,                                   |16
1965                    MESSAGE_PARM, ASTE#_PARM, PARM_FLAGS);                                                          |16
                                                                                                                        |
1966         /*    CHECK FUNCTION CODE                                                                                 */|16
                                                                                                                        |
1967         IF FUNCTION_CODE_APARM < FUNCTION_CODE_MIN;                                                                |40
                                                                                                                      CODE(0):      2,-(C)
                                                                                                                      CODE(2): MOV 6,5
                                                                                                                      CODE(4): ADD #2,5
                                                                                                                      CODE(10): MOV 7,(5)+
                                                                                                                      CODE(12): SUB #0,6
                                                                                                                      CODE(16): MOV #61776,2
                                                                                                                      CODE(22): MOV 0(2),-(6)
                                                                                                                      CODE(26): CMP #1,(6)+
                                                                                                                      CODE(32): BGT 2
                                                                                                                      CODE(34): JMP 0(7)
                                                                                                                      CODE(40): MOV #-3,4(5)
                                                                                                                      CODE(46): JMP #-4(5)
1968           THEN;                                                                                                  |52
1969   ....      RETURN WITH SEVERE_FLAG;                                                                             |52
FIXUP LOCATION 36 TO 12
1970         END;                                                                                                       |52
                                                                                                                        |
1971         IF FUNCTION_CODE_APARM > FUNCTION_CODE_MAX;                                                                |74
                                                                                                                      CODE(52): MOV #61776,2
                                                                                                                      CODE(56): MOV 0(2),-(6)
                                                                                                                      CODE(62): CMP #24,(6)+
                                                                                                                      CODE(66): BLT 2
                                                                                                                      CODE(70): JMP 0(7)
                                                                                                                      CODE(74): MOV #-3,4(5)
                                                                                                                      CODE(102): JMP #-4(5)
1972           THEN;                                                                                                  |106
1973   ....      RETURN WITH SEVERE_FLAG;                                                                             |106
FIXUP LOCATION 72 TO 12
1974         END;                                                                                                       |106
                                                                                                                        |
                                                                                                                      CODE(106): MOV #61776,2
                                                                                                                      CODE(112): MOVB0(2),-5(5)
1975         FUNCTION_CODE := FUNCTION_CODE_APARM;                                                                      |120
                                                                                                                      CODE(120): MOV 5,-(6)
                                                                                                                      CODE(122): MOV (5),-(6)
                                                                                                                      CODE(124): CLR -(6)
                                                                                                                      CODE(126): MOV #400,-(6)
                                                                                                                      CODE(132): MOV -2(4),2
                                                                                                                      CODE(136): JSB 7,#124(2)
                                                                                                                      CODE(142): MOV 5,6
                                                                                                                      CODE(144): CLR (6)+
                                                                                                                      CODE(146): MOV (6)+,5
1976         P(KERNEL_SMFP);                                                                                            |150
                                                                                                                      CODE(150): MOVB-5(5),1
                                                                                                                      CODE(154): MOV 1,-(6)
```

```
1977          PARM_FLAGS := FUNCTION_ARRAY(FUNCTION_CODE);
1978          RC := OK_FLAG;
1979          /*    CHECK SEG# PARAMETER IF REQUIRED
1980          IF (PARM_FLAGS & SEG#_FLAG) >= 0;
1981             THEN: SEG#_PARM := SEG#_APARM;
1982                IF SEG#_PARM < SEG#_MIN;
1983                   THEN: RC := SEVERE_FLAG;
PIKUP LOCATION 254 TO 6
1984                END;
1985                IF SEG#_PARM > SEG#_MAX;
1986                   THEN: RC := SEVERE_FLAG;
PIKUP LOCATION 312 TO 6
1987                END;
1988                IF RC >= SEVERE_FLAG;
1989                   THEN: ASTE#_PARM := PS_SEG(SEG#_PARM);
1990                      IF (ASTE#_PARM & SEG_FLAG) >= 0;
1991                         THEN: RC := SEVERE_FLAG;
```

```
CODE(156): ASL (6)
CODE(160): MOV #16546,2
CODE(164): ADD (6)+,2
CODE(166): MOV 0(2),-42(5)
  |174
CODE(174): MOV #-1,4(5)
  |202
  |
*/|202
  |
  |230
CODE(202): MOV -42(5),-(6)
CODE(206): MOV #-100000,-(6)
CODE(212): COM (6)
CODE(214): BIC (6)+,(6)
CODE(216): CLR -(6)
CODE(220): CMP (6)+,(6)+
CODE(222): BNE 2
CODE(224): JMP 0(7)
CODE(230): MOV #61774,2
CODE(234): MOV 0(2),-10(5)
  |242
  |
  |260
CODE(242): MOV -10(5),-(6)
CODE(246): CMP #1,(6)+
CODE(252): BGT 2
CODE(254): JMP 0(7)
CODE(260): MOV #-3,4(5)
  |266

  |266
  |
  |304
CODE(266): MOV -10(5),-(6)
CODE(272): CMP #37,(6)+
CODE(276): BLT 2
CODE(300): JMP 0(7)
CODE(304): MOV #-3,4(5)
  |312

  |312
  |
  |330
CODE(312): MOV 4(5),-(6)
CODE(316): CMP #-3,(6)+
CODE(322): BNE 2
CODE(324): JMP 0(7)
CODE(330): MOV -10(5),-(6)
CODE(334): ASL (6)
CODE(336): MOV #20200,2
CODE(342): ADD (6)+,2
CODE(344): MOV 0(2),-40(5)
  |352
  |
  |400
CODE(352): MOV -40(5),-(6)
CODE(356): MOV #-100000,-(6)
CODE(362): COM (6)
CODE(364): BIC (6)+,(6)
CODE(366): CLR -(6)
CODE(370): CMP (6)+,(6)+
CODE(372): BNE 2
CODE(374): JMP 0(7)
CODE(400): MOV #-3,4(5)
  |406
```

```
FIXUP LOCATION 376 TO 6
1992                    END;

FIXUP LOCATION 324 TO 56
1993                END;

FIXUP LOCATION 226 TO 156
1994            END;

1995            /*    CHECK OFFSET PARM IF REQUIRED

1996            IF (PARM_FLAGS & OFFSET_FLAG) >= 0;

1997              THEN: OFFSET_PARM := OFFSET_APARM;
```

```
                |406
                |

                |406
                |

                |406
                |
*/              |406
                |
                |434
CODE(406): MOV -42(5),-(6)
CODE(412): MOV #40000,-(6)
CODE(416): COM (6)
CODE(420): BIC (6)+,(6)
CODE(422): CLR -(6)
CODE(424): CMP (6)+,(6)+
CODE(426): BNE 2
CODE(430): JMP 0(7)
CODE(434): MOV #61772,2
CODE(440): MOV 0(2),-12(5)
                |446
                |
```

```
1998              IF OFFSET_PARM < OFFSET_MIN;
                  |464
CODE(446): MOV -12(5),-(6)
CODE(452): CMP #1,(6)+
CODE(456): BGT 2
CODE(460): JMP 0(7)
CODE(464): MOV #-3,4(5)
                  |472
1999                THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 462 TO 6
2000              END;
                  |472
                  |
2001              IF OFFSET_PARM > OFFSET_MAX;
                  |510
CODE(472): MOV -12(5),-(6)
CODE(476): CMP #77,(6)+
CODE(502): BLT 2
CODE(504): JMP 0(7)
CODE(510): MOV #-3,4(5)
                  |516
2002                THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 506 TO 6
2003              END;
                  |516
                  |
FIXUP LOCATION 432 TO 62
2004          END;
                  |516
                  |
2005          /*    CHECK CLASSIFICATION PARAMETER IF REQUIRED
                  |516
*/
2006          IF (PARM_FLAGS & CLASS_FLAG) <> 0;
                  |544
CODE(516): MOV -42(5),-(6)
CODE(522): MOV #20000,-(6)
CODE(526): COM (6)
CODE(530): BIC (6)+,(6)
CODE(532): CLR -(6)
CODE(534): CMP (6)+,(6)+
CODE(536): BNE 2
CODE(540): JMP 0(7)
CODE(544): MOV #61770,2
CODE(550): MOV 0(2),-14(5)
                  |556
                  |
                  |574
CODE(556): MOV -14(5),-(6)
CODE(562): CMP #1,(6)+
CODE(566): BGT 2
CODE(570): JMP 0(7)
CODE(574): MOV #-3,4(5)
                  |602
2007            THEN: CLASS_PARM := CLASS_ADDRM;
                  |602
                  |
2008              IF CLASS_PARM < CLASS_MIN;
                  |620
CODE(602): MOV -14(5),-(6)
CODE(606): CMP #4,(6)+
CODE(612): BLT 2
CODE(614): JMP 0(7)
CODE(620): MOV #-3,4(5)
                  |626
2009                THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 572 TO 6
2010              END;
                  |626
                  |
2011              IF CLASS_PARM > CLASS_MAX;
2012                THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 616 TO 6
2013              END;
```

```
FIXUP LOCATION 542 TO 62
2014        END;                                                                                    |626
                                                                                                    |
2015        /*    NO SYNTACTIC ERROR CHECKING FOR CATEGORY PARAMETER                              */|626
                                                                                                    |
                                                                                                    CODE(626): MOV #61766,2
                                                                                                    CODE(632): MOV 0(2),-16(5)
2016        CAT_PARM := CAT_APARM;                                                                  |640
                                                                                                    |
2017        /*    NO CHECKING OF SEG_TYPE PARAMETER                                               */|640
                                                                                                    |
                                                                                                    CODE(640): MOV #61764,2
                                                                                                    CODE(644): MOV 0(2),-(6)
                                                                                                    CODE(650): MOV #200,-(6)
                                                                                                    CODE(654): COM (6)
                                                                                                    CODE(656): BIC (6)+,(6)
                                                                                                    CODE(660): MOV (6)+,-20(5)
2018        SEG_TYPE_PARM := (SEG_TYPE_APARM & TYP_TYPE_MASK);                                      |664
                                                                                                    |
2019        /*    NO CHECKING OF SIZE PARAMETER                                                   */|664
                                                                                                    |
                                                                                                    CODE(664): MOV #61762,2
                                                                                                    CODE(670): MOV 0(2),-22(5)
2020        SIZE_PARM := SIZE_APARM;                                                                |676
                                                                                                    |
2021        /*    CHECK MODE PARM IF REQUIRED                                                     */|676
                                                                                                    |
2022        IF (PARM_FLAGS & MODE_FLAG) >= 0;                                                       |724
                                                                                                    CODE(676): MOV -42(5),-(6)
                                                                                                    CODE(702): MOV #2000,-(6)
                                                                                                    CODE(706): COM (6)
                                                                                                    CODE(710): BIC (6)+,(6)
                                                                                                    CODE(712): CLR -(6)
                                                                                                    CODE(714): CMP (6)+,(6)+
                                                                                                    CODE(716): BNE 2
                                                                                                    CODE(720): JMP 0(7)
                                                                                                    CODE(724): MOV #61760,2
                                                                                                    CODE(730): MOV 0(2),-24(5)
2023          THEN; MODE_PARM := MODE_APARM;                                                        |736
                                                                                                    |
2024              IF (MODE_PARM >= WRITE&READ&EXECUTE_ACCESS) & (MODE_PARM >= READ&EXECUTE_ACCESS |1052
2025                  ) & (MODE_PARM >= NO_ACCESS);                                                 |1052
                                                                                                    CODE(736): MOV -24(5),-(6)
                                                                                                    CODE(742): CMP #-40000,(6)+
                                                                                                    CODE(746): BNE 2
                                                                                                    CODE(750): CLR -(6)
                                                                                                    CODE(752): BR  2
                                                                                                    CODE(754): MOV #1,-(6)
                                                                                                    CODE(760): MOV -24(5),-(6)
                                                                                                    CODE(764): MOV #40000,-(6)
                                                                                                    CODE(770): CMP (6)+,(6)+
                                                                                                    CODE(772): BNE 2
                                                                                                    CODE(774): CLR -(6)
                                                                                                    CODE(776): BR  2
                                                                                                    CODE(1000): MOV #1,-(6)
                                                                                                    CODE(1004): BIT (6)+,(6)+
                                                                                                    CODE(1006): BNE 2
                                                                                                    CODE(1010): CLR -(6)
                                                                                                    CODE(1012): BR  2
                                                                                                    CODE(1014): MOV #1,-(6)
                                                                                                    CODE(1020): MOV -24(5),-(6)
                                                                                                    CODE(1024): CLR -(6)
                                                                                                    CODE(1026): CMP (6)+,(6)+
                                                                                                    CODE(1030): BNE 2
                                                                                                    CODE(1032): CLR -(6)
                                                                                                    CODE(1034): BR  2
```

```
2026                THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 1050 TO 4
2027                END;

FIXUP LOCATION 722 TO 134
2028        END;

2029        /*    NO CHECKING OF USER PARAMETER




2030        USER_PARM := (USER_APARM & ACL_USER_MASK);

2031        /*    NO CHECKING OF PROJECT PARAMETER
```

```
CODE(1036): MOV #1,-(6)
CODE(1042): BIT (6)+,(6)+
CODE(1044): BNE 2
CODE(1046): JMP 0(7)
CODE(1052): MOV 6-3,4(5)
  |1060

  |1060
  |

  |1060
  |
*/|1060
  |
CODE(1060): MOV #61756,2
CODE(1064): MOV 0(2),-(6)
CODE(1070): MOV #37777,-(6)
CODE(1074): COM (6)
CODE(1076): BIC (6)+,(6)
CODE(1100): MOV (6)+,-26(5)
  |1104
  |
*/|1104
  |
CODE(1104): MOV #61754,2
CODE(1110): MOV 0(2),-30(5)
```

```
2032        PROJECT_PARM := PROJECT_APARM;
2033        /*    CHECK REG0 IF REQUIRED
2034        IF (PARM_FLAGS & REG_FLAG) ¬= 0;
2035          THEN: REG0_PARM := REG0_APARM;
2036            IF REG0_PARM < REG0_MIN;
2037              THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 1172 TO 6
2038            END;
2039            IF REG0_PARM > REG0_MAX;
2040              THEN: RC := SEVERE_FLAG;
FIXUP LOCATION 1216 TO 6
2041            END;
FIXUP LOCATION 1142 TO 62
2042        END;
2043        /*    CHECK PROCESS0 PARAMETER IF REQUIRED
2044        IF (PARM_FLAGS & PROCESS0_FLAG) ¬= 0;
2045          THEN: PROCESS0_PARM := PROCESS0_APARM;
2046            IF PROCESS0_PARM < PROCESS0_MIN;
```

```
  |1116
  |
*/|1116
  |
  |1144
CODE(1116): MOV -42(5),-(6)
CODE(1122): MOV #10000,-(6)
CODE(1126): COM (6)
CODE(1130): BIC (6)+,(6)
CODE(1132): CLR -(6)
CODE(1134): CMP (6)+,(6)+
CODE(1136): BNE 2
CODE(1140): JMP 0(7)
CODE(1144): MOV #61752,2
CODE(1150): MOV 0(2),-32(5)
  |1156
  |
  |1174
CODE(1156): MOV -32(5),-(6)
CODE(1162): CLR -(6)
CODE(1164): CMP (6)+,(6)+
CODE(1166): BGT 2
CODE(1170): JMP 0(7)
CODE(1174): MOV #-3,#(5)
  |1202
  |
  |1202
  |
  |1220
CODE(1202): MOV -32(5),-(6)
CODE(1206): CMP #17,(6)+
CODE(1212): BLT 2
CODE(1214): JMP 0(7)
CODE(1220): MOV #-3,#(5)
  |1226
  |
  |1226
  |
  |1226
  |
*/|1226
  |
  |1254
CODE(1226): MOV -42(5),-(6)
CODE(1232): MOV #4000,-(6)
CODE(1236): COM (6)
CODE(1240): BIC (6)+,(6)
CODE(1242): CLR -(6)
CODE(1244): CMP (6)+,(6)+
CODE(1246): BNE 2
CODE(1250): JMP 0(7)
CODE(1254): MOV #61750,2
CODE(1260): MOV 0(2),-34(5)
  |1266
  |
  |1304
CODE(1266): MOV -34(5),-(6)
```

```
2047                THEN; PC := SEVERE_FLAG;
FIXUP LOCATION 1302 TO 6
2048              END;

2049              IF PROCESS0_PARM > PROCESS0_MAX;
```

```
2050                THEN; PC := SEVERE_FLAG;
FIXUP LOCATION 1326 TO 6
2051              END;

FIXUP LOCATION 1252 TO 62
2052          END;

2053          /*    NO CHECKING OF MESSAGE PARAMETER

2054          MESSAGE_PARM := MESSAGE_PARM;

2055          IF PC >= SEVERE_FLAG;
2056            THEN;
2057              IF FUNCTION_CODE < 10;
2058                THEN;
```

```
2059                  CASE FUNCTION_CODE_TYPE TAG FUNCTION_CODE;
```

```
CODE(1272): CMP #1,(6)+
CODE(1276): BGT 2
CODE(1300): JMP 0(7)
CODE(1304): MOV 0-3,-4(5)
  |1312

  |1312
  |
  |1330
CODE(1312): MOV -34(5),-(6)
CODE(1316): CMP #7,(6)+
CODE(1322): BLT 2
CODE(1324): JMP 0(7)
CODE(1330): MOV 0-3,-4(5)
  |1336

  |1336
  |

  |1336
  |
*/|1336
  |
CODE(1336): MOV #61746,2
CODE(1342): MOV 0(2),-36(5)
  |1350
  |
  |1366
  |1366
  |1406
  |1406
  |
CODE(1350): MOV 4(5),-(6)
CODE(1354): CMP #-3,(6)+
CODE(1360): BNE 2
CODE(1362): JMP 0(7)
CODE(1366): MOVB-5(5),1
CODE(1372): MOV 1,-(6)
CODE(1374): CMP #12,(6)+
CODE(1400): BGT 2
CODE(1402): JMP 0(7)
CODE(1406): MOVB-5(5),1
CODE(1412): MOV 1,-(6)
CODE(1414): ADD (6),(6)
CODE(1416): MOV (6)+,1
CODE(1420): ADD 7,1
CODE(1422): MOV 10(1),1
CODE(1426): ADD 7,1
CODE(1430): JMP (1)
  |1504
CODE(1504): CLR -(6)
CODE(1506): MOV 5,-(6)
CODE(1510): MOV (5),-(6)
CODE(1512): CLR -(6)
CODE(1514): MOV -40(5),-(6)
CODE(1520): MOV -12(5),-(6)
CODE(1524): MOV -14(5),-(6)
CODE(1530): MOV -16(5),-(6)
CODE(1534): MOV -20(5),-(6)
CODE(1540): MOV -22(5),-(6)
CODE(1544): MOV -2(4),2
CODE(1550): JSR 7,#1C04(2)
CODE(1554): MOV 5,6
CODE(1556): CLR (6)+
CODE(1560): MOV (6)+,5
CODE(1562): MOV (6)+,4(5)
```

```
2060                    1: RC := CREATE(ASTE#_PARM, OFFSET_PARM, CLASS_PARM, CAT_PARM,
2061                            SEG_TYPE_PARM, SIZE_PARM);
                                                                        |1566
                                                                      CODE(1566): JMP 0(7)
                                                                      CODE(1572): CLR -(6)
                                                                      CODE(1574): MOV 5,-(6)
                                                                      CODE(1576): MOV (5),-(6)
                                                                      CODE(1600): CLR -(6)
                                                                      CODE(1602): MOV -40(5),-(6)
                                                                      CODE(1606): MOV -12(5),-(6)
                                                                      CODE(1612): MOV -2(4),2
                                                                      CODE(1616): JSR 7,@40(2)
                                                                      CODE(1622): MOV 5,6
                                                                      CODE(1624): CLR (6)+
                                                                      CODE(1626): MOV (6)+,5
                                                                      CODE(1630): MOV (6)+,4(5)
2062                    2: RC := DELETE(ASTE#_PARM, OFFSET_PARM);
                                                                        |1634
                                                                      CODE(1634): JMP 0(7)
                                                                      CODE(1640): CLR -(6)
                                                                      CODE(1642): MOV 5,-(6)
                                                                      CODE(1644): MOV (5),-(6)
                                                                      CODE(1646): CLR -(6)
                                                                      CODE(1650): MOV -40(5),-(6)
                                                                      CODE(1654): MOV -12(5),-(6)
                                                                      CODE(1660): MOV -24(5),-(6)
                                                                      CODE(1664): MOV -26(5),-(6)
                                                                      CODE(1670): MOV -30(5),-(6)
                                                                      CODE(1674): MOV -2(4),2
                                                                      CODE(1700): JSR 7,@100(2)
                                                                      CODE(1704): MOV 5,6
                                                                      CODE(1706): CLR (6)+
                                                                      CODE(1710): MOV (6)+,5
                                                                      CODE(1712): MOV (6)+,4(5)
2063                    3: RC := GIVE(ASTE#_PARM, OFFSET_PARM, MODE_PARM, USER_PARM,
2064                            PROJECT_PARM);
                                                                        |1716
                                                                        |1716
                                                                      CODE(1716): JMP 0(7)
                                                                      CODE(1722): CLR -(6)
                                                                      CODE(1724): MOV 5,-(6)
                                                                      CODE(1726): MOV (5),-(6)
                                                                      CODE(1730): CLR -(6)
                                                                      CODE(1732): MOV -40(5),-(6)
                                                                      CODE(1736): MOV -12(5),-(6)
                                                                      CODE(1742): MOV -26(5),-(6)
                                                                      CODE(1746): MOV -30(5),-(6)
                                                                      CODE(1752): MOV -2(4),2
                                                                      CODE(1756): JSR 7,@70(2)
                                                                      CODE(1762): MOV 5,6
                                                                      CODE(1764): CLR (6)+
                                                                      CODE(1766): MOV (6)+,5
                                                                      CODE(1770): MOV (6)+,4(5)
2065                    4: RC := RESCIND(ASTE#_PARM, OFFSET_PARM, USER_PARM, PROJECT_PARM);
                                                                        |1774
                                                                      CODE(1774): JMP 0(7)
                                                                      CODE(2000): CLR -(6)
                                                                      CODE(2002): MOV 5,-(6)
                                                                      CODE(2004): MOV (5),-(6)
                                                                      CODE(2006): CLR -(6)
                                                                      CODE(2010): MOV -40(5),-(6)
                                                                      CODE(2014): MOV -12(5),-(6)
                                                                      CODE(2020): MOV -2(4),2
                                                                      CODE(2024): JSR 7,@44(2)
                                                                      CODE(2030): MOV 5,6
                                                                      CODE(2032): CLR (6)+
                                                                      CODE(2034): MOV (6)+,5
                                                                      CODE(2036): MOV (6)+,4(5)
2066                    5: RC := GET#(ASTE#_PARM, OFFSET_PARM);
                                                                        |2042
                                                                      CODE(2042): JMP 0(7)
                                                                      CODE(2046): CLR -(6)
```

```
2067                    6: RC := GETP(ASID#_PARM, OFFSET_PARM);

2068                    7: DCONNECT(REG#_PARM, ASID#_PARM);

2069                    8: RC := ENABLE(ASID#_PARM, REG#_PARM);

2070                    9: DISABLE(REG#_PARM);
```

```
CODE(2050): MOV 5,-(6)
CODE(2052): MOV (5),-(6)
CODE(2054): CLR -(6)
CODE(2056): MOV -40(5),-(6)
CODE(2062): MOV -12(5),-(6)
CODE(2066): MOV -2(4),2
CODE(2072): JSR 7,@50(2)
CODE(2076): MOV 5,6
CODE(2100): CLR (6)+
CODE(2102): MOV (6)+,5
CODE(2104): MOV (6)+,4(5)
  12110
CODE(2110): JMP 0(7)
CODE(2114): MOV 5,-(6)
CODE(2116): MOV (5),-(6)
CODE(2120): CLR -(6)
CODE(2122): MOV -10(5),-(6)
CODE(2126): MOV -40(5),-(6)
CODE(2132): MOV -2(4),2
CODE(2136): JSR 7,@14(2)
CODE(2142): MOV 5,6
CODE(2144): CLR (6)+
CODE(2146): MOV (6)+,5
  12150
CODE(2150): JMP 0(7)
CODE(2154): CLR -(6)
CODE(2156): MOV 5,-(6)
CODE(2160): MOV (5),-(6)
CODE(2162): CLR -(6)
CODE(2164): MOV -40(5),-(6)
CODE(2170): MOV -32(5),-(6)
CODE(2174): MOV -2(4),2
CODE(2200): JSR 7,@54(2)
CODE(2204): MOV 5,6
CODE(2206): CLR (6)+
CODE(2210): MOV (6)+,5
CODE(2212): MOV (6)+,4(5)
  12216
CODE(2216): JMP 0(7)
CODE(2222): MOV 5,-(6)
CODE(2224): MOV (5),-(6)
CODE(2226): CLR -(6)
CODE(2230): MOV -32(5),-(6)
CODE(2234): MOV -2(4),2
CODE(2240): JSR 7,@10(2)
CODE(2244): MOV 5,6
CODE(2246): CLR (6)+
CODE(2250): MOV (6)+,5
  12252
BRANCH TABLE(0): 622
BRANCH TABLE(1): 54
BRANCH TABLE(2): 142
BRANCH TABLE(3): 210
BRANCH TABLE(4): 272
BRANCH TABLE(5): 350
BRANCH TABLE(6): 416
BRANCH TABLE(7): 464
BRANCH TABLE(10): 524
BRANCH TABLE(11): 572
BRANCH TABLE(12): 622
BRANCH TABLE(13): 622
BRANCH TABLE(14): 622
BRANCH TABLE(15): 622
BRANCH TABLE(16): 622
BRANCH TABLE(17): 622
BRANCH TABLE(20): 622
```

```
FIXUP LOCATION 1570 TO 460
FIXUP LOCATION 1636 TO 412
FIXUP LOCATION 1720 TO 33C
FIXUP LOCATION 1776 TO 252
FIXUP LOCATION 2044 TO 204
FIXUP LOCATION 2112 TO 136
FIXUP LOCATION 2152 TO 76
FIXUP LOCATION 2220 TO 30
2071                    END;

FIXUP LOCATION 1404 TO 652
```

```
BRANCH TABLE(21): 622
BRANCH TABLE(22): 622
BRANCH TABLE(23): 622
BRANCH TABLE(24): 622
BRANCH TABLE(25): 622
```

```
12252
```

```
2072                ELSE:

2073                    CASE FUNCTION_CODE_TYPE TAG FUNCTION_CODE;

2074                      1: PC := OUTERR(ASTE#_PARM);

2075                      11: PC := OUTERR(ASTE#_PARM);
2076                      12:
2077                        IF SMER_COUNT(ASTE#_PARM) < 7;

2078                          THEN: PC := ERR_FLAG;
FIXUP LOCATION 2510 TO 6
2079                        END;
```

```
  12256
   |
CODE(2252): JMP 0(7)
CODE(2256): MOVB-5(5),1
CODE(2262): MOV 1,-(6)
CODE(2264): ADD (6),(6)
CODE(2266): MOV (6)+,1
CODE(2270): ADD 7,1
CODE(2272): MOV 10(1),1
CODE(2276): ADD 7,1
CODE(2300): JMP (1)
  12354
CODE(2354): CLR -(6)
CODE(2356): MOV 5,-(6)
CODE(2360): MOV (5),-(6)
CODE(2362): CLR -(6)
CODE(2364): MOV -40(5),-(6)
CODE(2370): MOV -2(4),2
CODE(2374): JSR 7,#30(2)
CODE(2400): MOV 5,6
CODE(2402): CLR (6)+
CODE(2404): MOV (6)+,5
CODE(2406): MOV (6)+,4(5)
  12412
CODE(2412): JMP 0(7)
CODE(2416): CLR -(6)
CODE(2420): MOV 5,-(6)
CODE(2422): MOV (5),-(6)
CODE(2424): CLR -(6)
CODE(2426): MOV -40(5),-(6)
CODE(2432): MOV -2(4),2
CODE(2436): JSR 7,#34(2)
CODE(2442): MOV 5,6
CODE(2444): CLR (6)+
CODE(2446): MOV (6)+,5
CODE(2450): MOV (6)+,4(5)
  12454
  12460
  12512
CODE(2454): JMP 0(7)
CODE(2460): MOV -40(5),-(6)
CODE(2464): MOV #15534,2
CODE(2470): ADD (6)+,2
CODE(2472): MOVB0(2),1
CODE(2476): MOV 1,-(6)
CODE(2500): CLR -(6)
CODE(2502): CMP (6)+,(6)+
CODE(2504): BGT 2
CODE(2506): JMP 0(7)
CODE(2512): MOV #-2,4(5)
  12520

  12520
   |
CODE(2520): JMP 0(7)
CODE(2524): MOV 5,-(6)
CODE(2526): MOV (5),-(6)
CODE(2530): CLR -(6)
```

```
                                                                    CODE(2532): MOV -34(5),-(6)
                                                                    CODE(2536): MOV -36(5),-(6)
                                                                    CODE(2542): CLR -(6)
                                                                    CODE(2544): MOV -2(4),2
                                                                    CODE(2550): JSR 7,@20(2)
                                                                    CODE(2554): MOV 5,6
                                                                    CODE(2556): CLR (6)+
                                                                    CODE(2560): MOV (6)+,5
                                                                      12562
2090                    13: IPCSEND(PROCESS@_PARM, MESSAGE_PARM, 0);  CODE(2562): JMP 0(7)
                                                                    CODE(2566): CLR -(6)
                                                                    CODE(2570): MOV 5,-(6)
                                                                    CODE(2572): MOV (5),-(6)
                                                                    CODE(2574): CLR -(6)
                                                                    CODE(2576): MOV -2(4),2
                                                                    CODE(2602): JSR 7,@24(2)
                                                                    CODE(2606): MOV 5,6
                                                                    CODE(2610): CLR (6)+
                                                                    CODE(2612): MOV (6)+,5
                                                                    CODE(2614): MOV (6)+,4(5)
                                                                      12620
2091                    14: RC := IPCRCV;                           CODE(2620): JMP 0(7)
                                                                    CODE(2624): CLR -(6)
                                                                    CODE(2626): MOV 5,-(6)
                                                                    CODE(2630): MOV (5),-(6)
                                                                    CODE(2632): CLR -(6)
                                                                    CODE(2634): MOV -26(5),-(6)
                                                                    CODE(2640): MOV -30(5),-(6)
                                                                    CODE(2644): MOV -14(5),-(6)
                                                                    CODE(2650): MOV -16(5),-(6)
                                                                    CODE(2654): MOV -34(5),-(6)
                                                                    CODE(2660): MOV -12(5),-(6)
                                                                    CODE(2664): MOV -2(4),2
                                                                    CODE(2670): JSR 7,@110(2)
                                                                    CODE(2674): MOV 5,6
                                                                    CODE(2676): CLR (6)+
                                                                    CODE(2700): MOV (6)+,5
                                                                    CODE(2702): MOV (6)+,4(5)
2092                    15: RC := START(USER_PARM, PROJECT_PARM, CLASS_PARM, CAT_PARM,
2093                            PROCESS@_PARM, OFFSET_PARM);          12706
                                                                      12706
                                                                    CODE(2706): JMP 0(7)
                                                                    CODE(2712): MOV 5,-(6)
                                                                    CODE(2714): MOV (5),-(6)
                                                                    CODE(2716): CLR -(6)
                                                                    CODE(2720): MOV -2(4),2
                                                                    CODE(2724): JSR 7,@4(2)
                                                                    CODE(2730): MOV 5,6
                                                                    CODE(2732): CLR (6)+
                                                                    CODE(2734): MOV (6)+,5
                                                                      12736
2094                    16: STOPP;                                  CODE(2736): JMP 0(7)
                                                                    CODE(2742): CLR -(6)
                                                                    CODE(2744): MOV 5,-(6)
                                                                    CODE(2746): MOV (5),-(6)
                                                                    CODE(2750): CLR -(6)
                                                                    CODE(2752): MOV -40(5),-(6)
                                                                    CODE(2756): MOV -12(5),-(6)
                                                                    CODE(2762): MOV -14(5),-(6)
                                                                    CODE(2766): MOV -16(5),-(6)
                                                                    CODE(2772): MOV -2(4),2
                                                                    CODE(2776): JSR 7,@74(2)
                                                                    CODE(3002): MOV 5,6
                                                                    CODE(3004): CLR (6)+
                                                                    CODE(3006): MOV (6)+,5
                                                                    CODE(3010): MOV (6)+,4(5)
2095                    17: RC := CHANGE@(ASID@_PARM, OFFSET_PARM, CLASS_PARM, CAT_PARM);  13014
```

```
2086                    19: PC := PS_CURRENT_PROCESS;

2087                    13: PC := INITH(ASTE#_PARM, OFFSET_PARM, CAT_PARM);

2088                    20: PC := READIP(ASTE#_PARM, OFFSET_PARM);

FIXUP LOCATION 2414 TO 534
FIXUP LOCATION 2456 TO 472
FIXUP LOCATION 2522 TO 426
FIXUP LOCATION 2564 TO 364
FIXUP LOCATION 2622 TO 326
FIXUP LOCATION 2710 TO 240
FIXUP LOCATION 2740 TO 210
FIXUP LOCATION 3016 TO 132
FIXUP LOCATION 3034 TO 114
FIXUP LOCATION 3106 TO 42
2089                  END;
```

```
CODE(3014): JMP 0(7)
CODE(3020): MOV #2C100,2
CODE(3024): MOV 0(2),4(5)
  |3032
CODE(3032): JMP 0(7)
CODE(3036): CLR -(6)
CODE(3040): MOV 5,-(6)
CODE(3042): MOV (5),-(6)
CODE(3044): CLR -(6)
CODE(3046): MOV -40(5),-(6)
CODE(3052): MOV -12(5),-(6)
CODE(3056): MOV -16(5),-(6)
CODE(3062): MOV -2(4),2
CODE(3066): JSR 7,#64(2)
CODE(3072): MOV 5,6
CODE(3074): CLR (6)+
CODE(3076): MOV (6)+,5
CODE(3100): MOV (6)+,4(5)
  |3104
CODE(3104): JMP 0(7)
CODE(3110): CLR -(6)
CODE(3112): MOV 5,-(6)
CODE(3114): MOV (5),-(6)
CODE(3116): CLR -(6)
CODE(3120): MOV -40(5),-(6)
CODE(3124): MOV -12(5),-(6)
CODE(3130): MOV -2(4),2
CODE(3134): JSR 7,#60(2)
CODE(3140): MOV 5,6
CODE(3142): CLR (6)+
CODE(3144): MOV (6)+,5
CODE(3146): MOV (6)+,4(5)
  |3152
BRANCH TABLE(0): 652
BRANCH TABLE(1): 652
BRANCH TABLE(2): 652
BRANCH TABLE(3): 652
BRANCH TABLE(4): 652
BRANCH TABLE(5): 652
BRANCH TABLE(6): 652
BRANCH TABLE(7): 652
BRANCH TABLE(10): 652
BRANCH TABLE(11): 652
BRANCH TABLE(12): 50
BRANCH TABLE(13): 116
BRANCH TABLE(14): 160
BRANCH TABLE(15): 224
BRANCH TABLE(16): 266
BRANCH TABLE(17): 324
BRANCH TABLE(20): 412
BRANCH TABLE(21): 502
BRANCH TABLE(22): 520
BRANCH TABLE(23): 536
BRANCH TABLE(24): 610
BRANCH TABLE(25): 652

|3152
```

```
FIXUP LOCATION 2254 TO 674
2090            END;

FIXUP LOCATION 1364 TO 1564
2091        END;




2092        V(KERNEL_SMEB);

FIXUP LOCATION 14 TO 26
                136 LINES WERE COMPILED, GENERATING 1470 BYTES OF CODE.
NO ERRORS WERE DETECTED.


 **    DEL PROGRAM LOADED AT 011E420
EL FP FB = 1460512 1319112 1319082
```

```
  |
  |3152
  |
  |3152
  |
CODE(3152): MOV 5,-(6)
CODE(3154): MOV (5),-(6)
CODE(3156): CLR -(6)
CODE(3160): MOV #400,-(6)
CODE(3164): MOV -2(4),2
CODE(3170): JSR 7,@130(2)
CODE(3174): MOV 5,6
CODE(3176): CLR (6)+
CODE(3200): MOV (6)+,5
  |3202
CODE(3202): JMP @-4(5)
```

```
621   DATA GATE;                                                                    10
                                                                                    |
622         /*    MUST CHECK INITIALIZATION OF INTERRUPT VECTORS IN STARTUP EVERY TIME A     * 10
623          * CHANGE IS MADE TO GATE (DATA OR PROGRAM)!                             */10
                                                                                    |
624         MACRO KERNEL_ENTRY;                                                     10
625               INLINE(#PID6);                                                    10
626               INLINE(MOV, 0, 6, 4, 6);                                          10
627               INLINE(MOV, 0, 4, 4, 6);                                          10
628               INLINE(MOV, 0, 3, 4, 6);                                          10
629               INLINE(MOV, 0, 2, 4, 6);                                          10
630               INLINE(MOV, 0, 1, 4, 6);                                          10
631               INLINE(MOV, 0, 0, 4, 6);                                          10
632               INLINE(MOV, 0, 6, 0, 4);                                          10
633               INLINE(MOV, 0, 5, 0, 5);                                          10
634               INLINE(MOV, 2, 7, 5, 5, #PROGAM, - 2);                            10
635               INLINE(SUB, 2, 7, 0, 6, 6)                                        10
636         END MACRO;                                                              10
                                                                                    |
637         MACRO KERNEL_EXIT;                                                      10
638               INLINE(ADD, 2, 7, 0, 6, 6);                                       10
639               INLINE(MOV, 2, 6, 0, 0);                                          10
640               INLINE(MOV, 2, 6, 0, 1);                                          10
641               INLINE(MOV, 2, 6, 0, 2);                                          10
642               INLINE(MOV, 2, 6, 0, 3);                                          10
643               INLINE(MOV, 2, 6, 0, 4);                                          10
644               INLINE(MOV, 2, 6, 0, 5);                                          10
645               INLINE(#TPIP4);                                                   10
646               INLINE(RTI)                                                       10
647         END MACRO;                                                              10
                                                                                    |
648         DECLARE                                                                 10
649               WORD (RC);                                                        10
                                                                                    |
650         /*    EXTERNAL KERNEL FUNCTIONS                                         */10
                                                                                    |
651         DECLARE                                                                 10
652               PROCEDURE (STOPP),                                                10
653               PROCEDURE ACCEPTS (WORD) (DISABLE),                               10
654               PROCEDURE ACCEPTS (WORD, WORD) (ECONNECT),                        10
655               PROCEDURE ACCEPTS (WORD, WORD, WORD) (IPCSEND),                   10
656               PROCEDURE RETURNS (WORD) (IPCRCV),                                10
657               PROCEDURE ACCEPTS (WORD) RETURNS (WORD) (OUTERP, OUTERV),         10
658               PROCEDURE ACCEPTS (WORD, WORD) RETURNS (WORD) (DELETE, GRTW, GETR, ENABLE,  10
659                     READIR),                                                    10
660               PROCEDURE ACCEPTS (WORD, WORD, WORD) RETURNS (WORD) (INITM),      10
661               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD) RETURNS (WORD) (RESCIND, CHANGEO),  10
662               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD, WORD) RETURNS (WORD) (GIVE),  10
663               PROCEDURE ACCEPTS (WORD, WORD, WORD, WORD, WORD, WORD) RETURNS (WORD) (CREATE,  10
664                     STARTP);                                                    10
                                                                                    |
665         /*    INTERNAL KERNEL FUNCTIONS                                         */10
                                                                                    |
666         DECLARE                                                                 10
667               PROCEDURE (SLEEP),                                                10
```

```
668             PROCEDURE ACCEPTS (#ORE) (DEACT, P, V, SWAPIN, SWAPOUT, RUN),                10
669             PROCEDURE ACCEPTS (#ORD, WORD) (SDADD),                                      10
670             PROCEDURE ACCEPTS (#ORD, #ORD, WORD) (LST),                                  10
671             PROCEDURE ACCEPTS (#ORD, WORE, WORE, WORD) (DISKIO),                         10
672             PROCEDURE RETURNS (#ORE) (PCHECK),                                           10
673             PROCEDURE ACCEPTS (WORD) RETURNS (#ORD) (#RITEDIR, HASH, PREHASH),           10
674             PROCEDURE ACCEPTS (#ORE, WORE, WORE) RETURNS (WORD) (DSEARCH, CONNECT),      10
675             PROCEDURE ACCEPTS (#ORD, WORD) RETURNS (#ORE) (ACT);                         10
                675 LINES WERE COMPILED.
NO ERRORS WERE DETECTED.
PROGRAM GATE     F
676   _I_                                                                                    10
```

```
677   PROGRAM GATE;
678         /*    PUSH SUPERVISOR R5 AND R0-R5 ONTO KERNEL'S STACK
679         KERNEL_ENTRY;
680         /*    IGNORE CALL IF NOT MADE FROM SUPERVISOR MODE
```

```
  1234
   |
*/1234
   |
CODE(0):       2,-(0)
CODE(2): MOV 6,5
CODE(4): ADD #2,5
CODE(10): MOV 7,(5)+
CODE(12): BR   106
CODE(14): =V(CONTAINED PROCEDURE)
CODE(20): =V(CONTAINED PROCEDURE)
CODE(24): =V(CONTAINED PROCEDURE)
CODE(30): =V(CONTAINED PROCEDURE)
CODE(34): =V(CONTAINED PROCEDURE)
CODE(40): =V(CONTAINED PROCEDURE)
CODE(44): =V(CONTAINED PROCEDURE)
CODE(50): =V(CONTAINED PROCEDURE)
CODE(54): =V(CONTAINED PROCEDURE)
CODE(60): =V(CONTAINED PROCEDURE)
CODE(64): =V(CONTAINED PROCEDURE)
CODE(70): =V(CONTAINED PROCEDURE)
CODE(74): =V(CONTAINED PROCEDURE)
CODE(100): =V(CONTAINED PROCEDURE)
CODE(104): =V(CONTAINED PROCEDURE)
CODE(110): =V(CONTAINED PROCEDURE)
CODE(114): =V(CONTAINED PROCEDURE)
CODE(120): =V(CONTAINED PROCEDURE)
CODE(124): =V(CONTAINED PROCEDURE)
CODE(130): =V(CONTAINED PROCEDURE)
CODE(134): =V(CONTAINED PROCEDURE)
CODE(140): =V(CONTAINED PROCEDURE)
CODE(144): =V(CONTAINED PROCEDURE)
CODE(150): =V(CONTAINED PROCEDURE)
CODE(154): =V(CONTAINED PROCEDURE)
CODE(160): =V(CONTAINED PROCEDURE)
CODE(164): =V(CONTAINED PROCEDURE)
CODE(170): =V(CONTAINED PROCEDURE)
CODE(174): =V(CONTAINED PROCEDURE)
CODE(200): =V(CONTAINED PROCEDURE)
CODE(204): =V(CONTAINED PROCEDURE)
CODE(210): =V(CONTAINED PROCEDURE)
CODE(214): =V(CONTAINED PROCEDURE)
CODE(220): =V(CONTAINED PROCEDURE)
CODE(224): =V(CONTAINED PROCEDURE)
CODE(230): SUB #0,6
CODE(236): MOV 5,-(6)
CODE(240): MOV 4,-(6)
CODE(242): MOV 3,-(6)
CODE(244): MOV 2,-(6)
CODE(246): MOV 1,-(6)
CODE(250): MOV 0,-(6)
CODE(252): MOV 6,#
CODE(254): MOV 6,5
CODE(256): MOV #-77766,-2(5)
CODE(264): SUB #6,6
  1270
   |
*/1270
```

```
681         IF (PSW & PREV_MODE_MASK) = PREV_MODE_SUPERV;                                  |
                                                                                           |322
682           THEN:                                                                        |322
                                                                                           |
683               /*    ACCESS SUPERVISOR'S STACK THROUGH KSR3                           */|322
                                                                                           |
                                                                                          CODE(270): MOV #-2,2
                                                                                          CODE(274): MOV 0(2),-(6)
                                                                                          CODE(300): MOV #30000,-(6)
                                                                                          CODE(304): COM (6)
                                                                                          CODE(306): BIC (6)+,(6)
                                                                                          CODE(310): CMP #10000,(6)+
                                                                                          CODE(314): BEQ .
                                                                                          CODE(316): JMP 0(7)
                                                                                          CODE(322): MOV #-560C,2
                                                                                          CODE(326): MOV 0(2),-(6)
                                                                                          CODE(332): MOV #-5472,2
                                                                                          CODE(336): MOV (6)+,C(2)
                                                                                            |342
684               KSDR3 := SDR0;                                                          CODE(342): MOV #-5540,2
                                                                                          CODE(346): MOV 0(2),-(6)
                                                                                          CODE(352): MOV #-5432,2
685               KSAR3 := SAR0;                                                          CODE(356): MOV (6)+,0(2)
                                                                                            |362
                                                                                            |
686               /*    CALL PARAMETER CHECKER - WHO IN TURN CALLS THE REQUESTED FUNCTION */|362
                                                                                            |
                                                                                          CODE(362): CLR -(6)
                                                                                          CODE(364): MOV 5,-(6)
                                                                                          CODE(366): MOV 5,-(6)
                                                                                          CODE(370): CLR -(6)
                                                                                          CODE(372): MOV -2(5),2
                                                                                          CODE(376): JSR 7,#164(2)
                                                                                          CODE(402): MOV 5,6
                                                                                          CODE(404): CLR (6)+
                                                                                          CODE(406): MOV (6)+,5
687               PC := PCHECK;                                                           CODE(410): MOV (6)+,-6(5)
                                                                                            |414
                                                                                            |
688               /*    REGAIN ACCESS TO SUPERVISOR'S SP0 STACK AND INSERT RETURN CODE */|414
                                                                                            |
                                                                                          CODE(414): MOV #-5600,2
                                                                                          CODE(420): MOV 0(2),-(6)
                                                                                          CODE(424): MOV #-5472,2
                                                                                          CODE(430): MOV (6)+,0(2)
                                                                                            |434
689               KSDR3 := SDR0;                                                          CODE(434): MOV #-5540,2
                                                                                          CODE(440): MOV 0(2),-(6)
                                                                                          CODE(444): MOV #-5432,2
690               KSAR3 := SAR0;                                                          CODE(450): MOV (6)+,0(2)
                                                                                            |454
                                                                                          CODE(454): MOV -6(5),-(6)
691               KPC := PC;                                                              CODE(460): MOV #61744,2
FIXUP LOCATION 320 TO 146                                                                 CODE(464): MOV (6)+,0(2)
692           END;                                                                          |470
                                                                                            |
                                                                                            |470
693           /*    RESTORE SUPERVISOR'S REGISTERS AND RETURN                        */|470
                                                                                            |
                                                                                          CODE(470): ADD #6,6
                                                                                          CODE(474): MOV (6)+,0
                                                                                          CODE(476): MOV (6)+,1
                                                                                          CODE(500): MOV (6)+,2
                                                                                          CODE(502): MOV (6)+,3
                                                                                          CODE(504): MOV (6)+,4
```

```
694         KERNEL_EXIT;
695         /*    INTERRUPT ENTRY POINTS
696         /*    DISK
697         KERNEL_ENTRY:
698         V(DISK_SEM);
699         KERNEL_EXIT;
700         /*    DEC WRITER
701         KERNEL_ENTRY:
702         V(DEC_PROCESS);
```

```
CODE(506): MOV (6)+,5
CODE(512):      0,2
  |514
  |
*/|514
  |
*/|514
  |
CODE(516): MOV 5,-(6)
CODE(520): MOV 4,-(6)
CODE(522): MOV 3,-(6)
CODE(524): MOV 2,-(6)
CODE(526): MOV 1,-(6)
CODE(530): MOV 0,-(6)
CODE(532): MOV 6,4
CODE(534): MOV 6,5
CODE(536): MOV @-77766,-2(5)
CODE(544): SUB #6,6
  |550
CODE(550): MOV 5,-(6)
CODE(552): MOV 5,-(6)
CODE(554): CLR -(6)
CODE(556): MOV #401,-(6)
CODE(562): MOV -2(5),2
CODE(566): JSR 7,@130(2)
CODE(572): MOV 5,6
CODE(574): CLR (6)+
CODE(576): MOV (6)+,5
  |600
CODE(600): ADD #6,6
CODE(604): MOV (6)+,0
CODE(606): MOV (6)+,1
CODE(610): MOV (6)+,2
CODE(612): MOV (6)+,3
CODE(614): MOV (6)+,4
CODE(616): MOV (6)+,5
CODE(622):      0,2
  |624
  |
*/|624
  |
CODE(626): MOV 5,-(6)
CODE(630): MOV 4,-(6)
CODE(632): MOV 3,-(6)
CODE(634): MOV 2,-(6)
CODE(636): MOV 1,-(6)
CODE(640): MOV 0,-(6)
CODE(642): MOV 6,4
CODE(646): MOV 6,5
CODE(606): MOV @-77766,-2(5)
CODE(650): SUB #6,6
  |660
CODE(660): MOV 5,-(6)
CODE(662): MOV 5,-(6)
CODE(664): CLR -(6)
CODE(666): MOV #3,-(6)
CODE(672): MOV -2(5),2
CODE(676): JSR 7,@130(2)
CODE(702): MOV 5,6
CODE(704): CLR (6)+
CODE(706): MOV (6)+,5
  |710
CODE(710): ADD #6,6
CODE(714): MOV (6)+,0
CODE(716): MOV (6)+,1
CODE(720): MOV (6)+,2
```

```
703         KERNEL_EXIT;
704         /*    TTY
705         KERNEL_ENTRY;
706         V(TTY_PROCESS#);
727         KERNEL_EXIT;
728         /*    SCOPE1
729         KERNEL_ENTRY;
730         V(SCOPE1_PROCESS#);
```

```
CODE(722): MOV (6)+,3
CODE(724): MOV (6)+,4
CODE(726): MOV (6)+,5
CODE(732):      0,2
  |734
  |
*/|734
  |
CODE(736): MOV 5,-(6)
CODE(740): MOV 4,-(6)
CODE(742): MOV 3,-(6)
CODE(744): MOV 2,-(6)
CODE(746): MOV 1,-(6)
CODE(750): MOV 0,-(6)
CODE(752): MOV 6,4
CODE(754): MOV 6,5
CODE(756): MOV #-77766,-2(5)
CODE(764): SUB #6,6
  |770
CODE(770): MOV 5,-(6)
CODE(772): MOV 5,-(6)
CODE(774): CLR -(6)
CODE(776): MOV #2,-(6)
CODE(1002): MOV -2(5),2
CODE(1006): JSR 7,@130(2)
CODE(1012): MOV 5,6
CODE(1014): CLR (6)+
CODE(1016): MOV (6)+,5
  |1020
CODE(1020): ADD #6,6
CODE(1024): MOV (6)+,0
CODE(1026): MOV (6)+,1
CODE(1030): MOV (6)+,2
CODE(1032): MOV (6)+,3
CODE(1034): MOV (6)+,4
CODE(1036): MOV (6)+,5
CODE(1042):      0,2
  |1044
  |
*/|1044
  |
CODE(1046): MOV 5,-(6)
CODE(1050): MOV 4,-(6)
CODE(1052): MOV 3,-(6)
CODE(1054): MOV 2,-(6)
CODE(1056): MOV 1,-(6)
CODE(1060): MOV 0,-(6)
CODE(1062): MOV 6,4
CODE(1064): MOV 6,5
CODE(1066): MOV #-77766,-2(5)
CODE(1074): SUB #6,6
  |1100
CODE(1100): MOV 5,-(6)
CODE(1102): MOV 5,-(6)
CODE(1104): CLR -(6)
CODE(1106): MOV #4,-(6)
CODE(1112): MOV -2(5),2
CODE(1116): JSR 7,@130(2)
CODE(1122): MOV 5,6
CODE(1124): CLR (6)+
CODE(1126): MOV (6)+,5
  |1130
CODE(1130): ADD #6,6
CODE(1134): MOV (6)+,0
CODE(1136): MOV (6)+,1
CODE(1140): MOV (6)+,2
```

```
CODE(1194): MOV (6)+,4
CODE(1198): MOV (6)+,6
CODE(119A): MOV (6)+,8
CODE(119E): MOV (6)+,A
CODE(11A2):    0,2
```

```
711         KERNEL_EXIT;
712         /*    SCOPE2
713         KERNEL_ENTRY;
714         V(SCOPE2_PROCESS#);
715         KERNEL_EXIT;
FIXUP LOCATION 232 TO 2
              40 LINES WERE COMPILED, GENERATING 696 BYTES OF CODE.
NO ERRORS WERE DETECTED.
```

```
  |1154
  |
*/|1154
  |
CODE(1156): MOV 5,-(6)
CODE(1160): MOV 4,-(6)
CODE(1162): MOV 3,-(6)
CODE(1164): MOV 2,-(6)
CODE(1166): MOV 1,-(6)
CODE(1170): MOV 0,-(6)
CODE(1172): MOV 6,4
CODE(1174): MOV 6,5
CODE(1176): MOV #-77766,-2(5)
CODE(1204): SUB #6,6
  |1210
CODE(1210): MOV 5,-(6)
CODE(1212): MOV 5,-(6)
CODE(1214): CLR -(6)
CODE(1216): MOV #5,-(6)
CODE(1222): MOV -2(5),2
CODE(1226): JSR 7,@130(2)
CODE(1232): MOV 5,6
CODE(1234): CLR (6)+
CODE(1236): MOV (6)+,5
  |1240
CODE(1240): ADD #6,6
CODE(1244): MOV (6)+,0
CODE(1246): MOV (6)+,1
CODE(1250): MOV (6)+,2
CODE(1252): MOV (6)+,3
CODE(1254): MOV (6)+,4
CODE(1256): MOV (6)+,5
CODE(1262):      0,2
  |1264
CODE(1264): JMP @-4(5)
```

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

        000000                     1  R0=%0
        000001                     2  R1=%1
        000002                     3  R2=%2
        000003                     4  R3=%3
        000004                     5  R4=%4
        000005                     6  R5=%5
        000006                     7  R6=%6
        000007                     8  PC=%7
                                   9  ;
                                  10  ;       DISK_ADR := DALLOC(BMT_ADR);
                                  11  ;
                                  12  ;
000000' 000240                    13  DALLOC: NOP
000002' 010605                    14          MOV   R6,R5             ;SAVE
000004' 062705  000004            15          ADD   #4,R5             ;ENTRY
000010' 010725                    16          MOV   PC,(R5)+          ;SEQUENCE
000012' 162706  000002            17          SUB   #2,R6
000016' 010446                    18          MOV   R4,-(R6)          ;DON'T NEED R4
000020' 016500  177774            19          MOV   -4(R5),R0         ;GET PARAMETER
000024' 010001                    20          MOV   R0,R1             ;ADDRESS OF BIT MAP TABLE
000026' 012102                    21          MOV   (R1)+,R2          ;START OF BIT MAP
000030' 011103                    22          MOV   (R1),R3           ;END OF BIT MAP
000032' 012704  177777            23          MOV   #177777,R4        ;ALL BITS SET
                                  24  ;
000036' 012201                    25  WLOOP:  MOV   (R2)+,R1          ;SEARCH BIT MAP
000040' 020104                    26          CMP   R1,R4             ;FOR A 0 BIT
000042' 001003                    27          BNE   WFOUND            ;FOUND ONE
000044' 020203                    28          CMP   R2,R3             ;END OF MAP?
000046' 002773                    29          BLT   WLOOP             ;NO
000050' 000000                    30          HALT                    ;UNRECOVERABLE ERROR
                                  31  ;
000052' 162702  000002            32  WFOUND: SUB   #2,R2             ;ADDRESS OF WORD WITH FREE BIT
000056' 010203                    33          MOV   R2,R3
000060' 161003                    34          SUB   (R0),R3           ;ITS WORD NUMBER (RELATIVE TO BASE)
                                  35  ;
000062' 005204                    36  BLOOP:  INC   R4                ;SEARCH
000064' 006201                    37          ASR   R1                ;FOR FIRST
000066' 103775                    38          BCS   BLOOP             ;0 BIT
                                  39  ;
                                  40  ;       NOW MUST TURN BIT ON IN MAP
                                  41  ;
000070' 012701  000001            42          MOV   #1,R1             ;ONE BIT ON
000074' 072104                    43          .WORD 072104 ;ASH R1,R4 ;SHIFT BIT OVER
000076' 050112                    44          BIS   R1,(R2)           ;SET USE BIT
                                  45  ;
                                  46  ;       TRANSLATE (WORD#, BIT#, SIZE) INTO A DISK ADDRESS
                                  47  ;
000100' 072327                    48          .WORD 072327 ;ASH R3,3  ;SHIFT WORD NUMBER OVER 3 BITS
000102' 000003                    49          .WORD 3
000104' 060403                    50          ADD   R4,R3             ;ADD IN BIT NUMBER
000106' 016002  000006            51          MOV   6(R0),R2          ;SHIFT FACTOR
000112' 072302                    52          .WORD 072302            ;ASH   R3,R2
```

```
LOC     OBJECT CODE          STMT  SOURCE STATEMENT

000114' 066003  000004        53          ADD   4(R0),R3       ;ADD IN BASE ADDRESS OF DISK AREA
000120' 010365  000004        54          MOV   R3,4(R5)       ;RETURN TO USER
000124' 012604                55          MOV   (R5)+,R4       ;RESTORE
000126' 000175  177772        56          JMP   @-6(R5)        ;DONE!!
                              57          .END  DALLOC
```

SYMBOL	VALUE	DEFN	REFERENCES							
ELCOP	0000628	36	38							
EALLOC	0000008	13	57							
EC	=%0000007	8	16							
R0	=%0000000	1	19	20	34	51	53			
R1	=%0000001	2	20	21	22	25	26	37	42	44
R2	=%0000002	3	21	25	28	32	33	46	51	
R3	=%0000003	4	22	29	33	34	50	53	54	
R4	=%0000004	5	18	23	26	36	50	55		
R5	=%0000005	6	14	15	16	17	54	56		
R6	=%0000006	7	14	17	18	55				
WPCUNC	000052R	32	27							
WLCCP	000036R	25	29							

TOTAL NUMBER OF ERRORS -- 00000

```
        PAL-11S/360 ASSEMBLER                                                   PAGE   1

LOC     OBJECT CODE             STMT  SOURCE STATEMENT

        000000                     1  R0=%0
        000001                     2  R1=%1
        000002                     3  R2=%2
        000003                     4  R3=%3
        000004                     5  R4=%4
        000005                     6  R5=%5
        000006                     7  R6=%6
        000007                     8  PC=%7
                                   9  ;
                                  10  ;       DFREE(DISK_ADR, BMT_ADR);
                                  11  ;
000000' 000240                    12  DFREE:  NOP
000002' 010605                    13          MOV   R6,R5             ;SUE
000004' 062705  000006            14          ADD   #6,R5             ;ENTRY
000010' 010725                    15          MOV   PC,(R5)+          ;SEQUENCE
000012' 162706  000002            16          SUB   #2,R6
000016' 016502  177774            17          MOV   -4(R5),R2         ;DISK ADDRESS
000022' 016500  177772            18          MOV   -6(R5),R0         ;BMT ADDRESS PARAMETER
000026' 166002  000004            19          SUB   4(R0),R2          ;SUBTRACT OUT BASE DISK ADDRESS
000032' 016001  000006            20          MOV   6(R0),R1          ;GET SHIFT FACTOR
000036' 005401                    21          NEG   R1                ;NEGATIVE FOR RIGHT SHIFT
000040' 072201                    22          .WORD 072201            ;SRL   R2,R1
000042' 073227                    23          .WORD 073227 ;ASHC R2,-4 ;SRDL R2,3,4
000044' 177774                    24          .WORD -4                ;SEPERATE WORD#, BIT#
000046' 072327                    25          .WORD 072327 ;ASH R3,-12 ;SRL R3,12
000050' 177764                    26          .WORD -14               ;GET BIT NUMBER OVER TO RIGHT
000052' 042703  177760            27          BIC   #177760,R3        ;ASH IS ARITHMETIC, NOT LOGICAL, SHIFT
000056' 012701  000001            28          MOV   #1,R1             ;ONE 1 BIT
000062' 072103                    29          .WORD 072103 ;ASH R1,R3  ;SRL R1,0(R3)
000064' 006302                    30          ASL   R2                ;WORD# TO BYTE#
000066' 061002                    31          ADD   (R0),R2           ;ADD IN START ADDRESS OF BIT MAP
000070' 040112                    32          BIC   R1,(R2)           ;CLEAR USE BIT
000072' 000175  177770            33          JMP   @-10(R5)          ;DONE
                                  34          .END  DFREE

219
```

SYMBOL	VALUE	DEFN	REFERENCES					
ERREE	0000008	12	34					
FC	=X0000007	4	15					
R0	=X0000000	1	18	19	20	31		
R1	=X0000001	2	20	21	28	32		
R2	=X0000002	3	17	19	30	31	32	
R3	=X0000003	4	27					
R4	=X0000004	5						
R5	=X0000005	6	13	14	15	17	18	33
R6	=X0000006	7	13	16				

TOTAL NUMBER OF ERRORS -- 00000

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

                                  1   R0=%0
                                  2   R1=%1
                                  3   R2=%2
                                  4   R3=%3
                                  5   R4=%4
                                  6   R5=%5
                                  7   R6=%6
                                  8   PC=%7
                                  9   ;
                                 10   ;        SWAP(CURRENT_PROCESS, NEXT_PROCESS);
                                 11   ;
                                 12   ;
                                 13   ;        RELEVANT DEFINITIONS IN PT
                                 14   ;
                                 15   PTKDB1=17000
                                 16   PTKAR1=17040
                                 17   PTKDB2=17100
                                 18   PTKAR2=17140
                                 19   PTKDB3=17200
                                 20   PTKAR3=17240
                                 21   PTDB4=17300
                                 22   PTDB5=17340
                                 23   PTDB6=17400
                                 24   ;
                                 25   ;        AND PS
                                 26   ;
                                 27   PSSDR=20000
                                 28   PSSAR=20040
                                 29   ;
                                 30   ;        HARDWARE REGS
                                 31   ;
                                 32   KSDR1=172302
                                 33   KSAR1=172342
                                 34   KSAR2=172344
                                 35   KSDR2=172304
                                 36   KSAR3=172346
                                 37   KSDR3=172306
                                 38   SAR0=172240
                                 39   SDR0=172200
                                 40   SAR8=177640
                                 41   SDR8=177600
                                 42   ;
                                 43   ;        AND FINALLY
                                 44   ;
                                 45   TCP=16544                    ;THE CURRENT PROCESS
                                 46   PSW=177776
                                 47   ;
000000' 010605                   48   SWAP:    MOV   R6,R5         ;SUE
000002' 062705  000006           49            ADD   #6,R5         ;ENTRY
000006' 010725                   50            MOV   PC,(R5)+      ;SEQUENCE
000010' 016500  177774           51            MOV   -4(R5),R0     ;CURRENT_PROCESS
000014' 006300                   52            ASL   R0            ;ARRAYS ARE OF WORDS
```

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

                                  53  ;
                                  54  ;       SAVE  KSR3    OF CURRENT PROCESS IN PT
                                  55  ;
000016' 013760  172306  017200    56          MOV   @#KSDR3,PIKDR3(R0)
000024' 013760  172346  017240    57          MOV   @#KSAR3,PIKAR3(R0)
                                  58  ;
                                  59  ;       SAVE  GPR4-6 IN PT
                                  60  ;
000032' 010460  017300            61          MOV   R4,PIR4(R0)
000036' 010560  017340            62          MOV   R5,PIR5(R0)
000042' 010660  017400            63          MOV   R6,PIR6(R0)
000046' 016500  177772            64          MOV   -6(R5),R0           ;NEXT_PROCESS
000052' 010037  016544            65          MOV   R0,@#SCP            ;SAVE
000056' 006300                    66          ASL   R0                  ;ARRAYS ARE WORDS
                                  67  ;
                                  68  ;       LOAD SR0-15 FOR NEXT PROCESS FROM ITS PS
                                  69  ;
                                  70  ;       FIRST SR0-7
                                  71  ;
000060' 012701  020040            72          MOV   #ISSAR,R1           ;BASE ADDRESS OF PS_SAR ARRAY
000064' 012702  020000            73          MOV   #ISSDR,R2           ;PS_SDR
000070' 012703  172240            74          MOV   #SAR0,R3            ;BASE ADDRESS OF SUPERVISOR
000074' 012704  172200            75          MOV   #SDR0,R4            ;SEGMENTATION REGISTERS (SR0-7)
000100' 012705  000010            76          MOV   #8.,R5              ;LOOP CONTROL
000104' 012123                    77  SR07:   MOV   (R1)+,(R3)+
000106' 012224                    78          MOV   (R2)+,(R4)+
000110' 005305                    79          DEC   R5
000112' 001374                    80          BNE   SR07
                                  81  ;
                                  82  ;       NOW   SR8-15
                                  83  ;
000114' 012703  177640            84          MOV   #SAPR,R3            ;BASE ADDRESS OF USER
000120' 012704  177600            85          MOV   #SDPR,R4            ;SEGMENTATION REGISTERS (SR8-15)
000124' 012705  000010            86          MOV   #8.,R5              ;LOOP CONTROL
000130' 012123                    87  SR815:  MOV   (R1)+,(R3)+
000132' 012224                    88          MOV   (R2)+,(R4)+
000134' 005305                    89          DEC   R5
000136' 001374                    90          BNE   SR815
                                  91  ;
                                  92  ;       LOAD KSR1, 2, & 3 FROM PT
                                  93  ;
000140' 016037  017000  172302    94          MOV   PIKDR1(R0),@#KSDR1
000146' 016037  017040  172342    95          MOV   PIKAR1(R0),@#KSAR1
000154' 016037  017100  172344    96          MOV   PIKAR2(R0),@#KSAR2
000162' 016037  017100  172304    97          MOV   PIKDR2(R0),@#KSDR2
000170' 016037  017240  172346    98          MOV   PIKAR3(R0),@#KSAR3
000176' 016037  017200  172306    99          MOV   PIKDR3(R0),@#KSDR3
                                 100  ;
                                 101  ;       AND GPR4-6
                                 102  ;
000204' 016004  017300           103          MOV   PIR4(R0),R4
000210' 016005  017340           104          MOV   PIR5(R0),R5
```

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

000214' 016006  017400          105           MOV   PIRR(SP),R6
                                106   ;
                                107   ;       IF THIS IS NOT A NEW PROCESS RETURN INTO KERNEL
                                108   ;       ELSE RETURN OUT TO USER
                                109   ;
000220' 010505                  110           MOV   R5,R5
000222' 001403                  111           BEQ   NEW
000224' 000230                  112           .WORD 230                 ;SPL LOW
000226' 000175  177770          113           JMP   @-10(R5)
                                114   ;
                                115   ;
000232' 005000                  116   NEW:    CLR   R0
000234' 005001                  117           CLR   R1
000236' 005002                  118           CLR   R2
000240' 005003                  119           CLR   R3
000242' 012704  001740          120           MOV   #1740,R4            ;"3E0"
000246' 012706  041776          121           MOV   #41776,R6           ;KERNEL STACK POINTER - "43FE"
000252' 012746  070000          122           MOV   #070000,-(R6)       ;USER PSW - CM=S, PM=U
000256' 012746  040000          123           MOV   #040000,-(R6)       ;PC = "4000"
000262' 012746  001734          124           MOV   #001734,-(R6)       ;USER R6
000266' 010446                  125           MOV   R4,-(R6)            ;POINTER TO STATIC LINK
000270' 012737  010340  177776  126           MOV   #010340,@#PSW       ;FOR NEXT INSTRUCTION
000276' 006637                  127           .WORD 006637 ;MTPI        ;SET STATIC LINK EQUAL TO
000300' 001740                  128           .WORD 1740                ;POINTER TO STATIC LINK
000302' 006606                  129           .WORD 006606 ;MTPI R6     ;SET SUPERVISOR MODE R6
000304' 000002                  130           RTI
                                131           .END  SWAP
```

SYMBOL	VALUE	DEFN	REFERENCES													
KSAR1	= 172342	33	95													
KSAR2	= 172344	34	96													
KSAR3	= 172346	36	57	98												
KSDR1	= 172302	32	94													
KSDR2	= 172304	35	97													
KSDR3	= 172306	37	56	99												
NEW	000232R	116	111													
PC	=%000007	8	50													
PSSAR	= 020040	28	72													
PSSDR	= 020000	27	73													
PSW	= 177776	46	126													
PTKAR1	= 017040	16	95													
PTKAR2	= 017140	18	96													
PTKAR3	= 017240	20	57	99												
PTKDR1	= 017000	15	94													
PTKDR2	= 017100	17	97													
PTKDR3	= 017200	19	56	99												
PTR4	= 017300	21	61	103												
PTR5	= 017340	22	62	104												
PTR6	= 017400	23	63	105												
R0	=%000000	1	51	52	56	57	61	62	63	64	65	66	94	95	96	97	98
			99	103	104	105	116										
R1	=%000001	2	72	77	87	117											
R2	=%000002	3	73	78	88	118											
R3	=%000003	4	74	77	84	87	119										
R4	=%000004	5	61	75	78	85	88	103	120	125							
R5	=%000005	6	48	49	50	51	62	64	76	79	86	89	104	110	112	113	
R6	=%000006	7	49	63	105	121	122	123	124	125							
SAR0	= 172240	38	74														
SDR0	= 177640	40	84														
SDR0	= 172200	39	75														
SER9	= 177600	41	85														
SR07	000104R	77	90														
SR15	000130R	87	90														
SWAP	000000R	48	131														
TCE	= 016544	45	65														

TOTAL NUMBER OF ERRORS -- 00000

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

        000000                     1  R0=%0
        000001                     2  R1=%1
        000002                     3  R2=%2
        000003                     4  R3=%3
        000004                     5  R4=%4
        000005                     6  R5=%5
        000006                     7  R6=%6
        000007                     8  PC=%7
                                   9  ;
                                  10  ;       DISKIO(DISK_ADDRESS, MEMORY_ADDRESS, SIZE, MODE);
                                  11  ;
        177460                    12  RFDSC=177460                    ;DISK CONTROL STATUS
        177462                    13  RFWC=177462                     ;WORD COUNT
        177464                    14  RFCMA=177464                    ;CURRENT MEMORY ADDRESS
        177466                    15  RFDAR=177466                    ;DISK ADDRESS REGISTER
        177470                    16  RFDAE=177470                    ;DISK ADDRESS EXTENSION ERROR REGISTER
                                  17  ;
C00000' 010605                    18  DISKIO: MOV   R6,R5             ;SUB
C00002' 062705  000012            19          ADD   #12,R5            ;ENTRY
C00006' 010725                    20          MOV   PC,(R5)+          ;SEQUENCE
C00010' 105737  177460            21          TSTB  @#RFDSC           ;DISK MUST BE READY
C00014' 002377                    22  ERROR:  BGE   ERROR             ;OTHERWISE INFINITE LOOP
C00016' 016501  177770            23          MOV   -10(R5),R1        ;SIZE OF SEGMENT IN 256 BYTE BLOCKS
000022' 072127                    24          .WORD 072127 ;ASH R1,#7 ;SLL R1,7
C00024' 000007                    25          .WORD 000007            ;SIZE OF SEGMENT IN WORDS
000026' 005401                    26          NEG   R1                ;DISK REQUIRES 2'S COMPLEMENT OF COUNT
C00030' 010137  177462            27          MOV   R1,@#RFWC         ;MOVE COUNT INTO WORD COUNT REGISTER
C00034' 005000                    28          CLR   R0                ;SET UP FOR SPLITTING DISK ADDRESS INTO 2
000036' 016501  177774            29          MOV   -4(R5),R1         ;DISK ADDRESS WITH 8 LOW ORDER 0'S REMOVED
000042' 073027                    30          .WORD 073027 ;ASHC R0,#10;SLDL R0,R1,8
C00044' 000010                    31          .WORD 000010            ;ADD IN THE LOW ORDER 0'S
C00046' 010037  177470            32          MOV   R0,@#RFDAE        ;MOVE DISK ADDRESS
000052' 010137  177466            33          MOV   R1,@#RFDAR        ;TO CONTROL REGISTERS
C00056' 005000                    34          CLR   R0                ;SET UP FOR SPLITTING MEM ADDRESS INTO 2
000060' 016501  177772            35          MOV   -6(R5),R1         ;MEM ADDRESS WITH 8 LOW ORDER 0'S REMOVED
000064' 073027                    36          .WORD 073027 ;ASHC R0,#10;SLDL R0,R1,8
C00066' 000010                    37          .WORD 000010            ;ADD IN THE LOW ORDER 0'S
C00070' 010137  177464            38          MOV   R1,@#RFCMA        ;MOVE LOW ORDER WORD INTO CONTROL REGISTER
C00074' 072027                    39          .WORD 072027 ;ASH R0,#4 ;SLL R0,4 - MOVE EXTENDED BITS TO RIGHT
000076' 000004                    40          .WORD 000004
000100' 056500  177766            41          BIS   -12(R5),R0        ;OR IN INT. ENABLE, FUNCTION, AND GO BITS
000104' 010037  177460            42          MOV   R0,@#RFDSC        ;START DISK
000110' 000175  177764            43          JMP   @-14(R5)          ;RETURN
                                  44          .END  DISKIO
```

```
SYMBOL   VALUE     DEFN  REFERENCES

DISKIO   000008R    18    44
ERRCR    000014R    22    22
EC      =%000007     8    20
RFCMA   = 177464    14    39
RFDAE   = 177470    16    32
RFDAR   = 177466    15    33
RFESC   = 177460    12    21    42
RFWC    = 177462    13    27
R0      =%000000     1    28    32    34    41    42
R1      =%000001     2    23    26    27    29    33    35    38
R2      =%000002     3
R3      =%000003     4
R4      =%000004     5
R5      =%000005     6    18    19    20    23    24    35    41    43
R6      =%000006     7    18

TOTAL NUMBER OF ERRORS -- 00000
```

```
LOC     OBJECT CODE             STMT  SOURCE STATEMENT

        000000                     1  R0=%0
        000001                     2  R1=%1
        000002                     3  R2=%2
        000003                     4  R3=%3
        000004                     5  R4=%4
        000005                     6  R5=%5
        000006                     7  R6=%6
        000007                     8  PC=%7
                                   9  ;
                                  10  ;       LSD: LOAD_SEGMENT_DESCRIPTOR(ASTE#, REG_LOC, MODE);
                                  11  ;
        007400                    12  ASTADR=007400                       ;BASE OF AST_ADR ARRAY
        005000                    13  ASTSIZ=005000                       ;BASE OF AST_SIZE ARRAY
                                  14  ;
000000' 010605                    15  LSD:    MOV   R6,R5                 ;SUE
000002' 062705  000010            16          ADD   #10,R5                ;ENTRY
000006' 010725                    17          MOV   PC,(R5)+              ;SEQUENCE
000010' 016500  177774            18          MOV   -4(R5),R0             ;ASTE#
000014' 116001  005000            19          MOVB  ASTSIZ(R0),R1         ;SIZE OF THE SEGMENT
000020' 006301                    20          ASL   R1                    ;MY BLOCKS TO THEIR'S
000022' 006301                    21          ASL   R1
000024' 005301                    22          DEC   R1                    ;MMU USES EXCESS ONES NOTATION
000026' 000301                    23          SWAB  R1                    ;SLF IS IN LEFT BYTE OF DESCRIPTOR REG
000030' 056501  177770            24          BIS   -10(R5),R1            ;OR IN A, W, ED, AND ACF BITS
000034' 016502  177772            25          MOV   -6(R5),R2             ;POINTER TO DESCRIPTOR REG
000040' 010112                    26          MOV   R1,(R2)               ;STORE DESCRIPTOR
000042' 062702  000040            27          ADD   #40,R2                ;POINT TO ADDRESS REGISTER
000046' 006300                    28          ASL   R0                    ;ARRAY OF WORDS
000050' 016001  007400            29          MOV   ASTADR(R0),R1         ;BASE ADDRESS OF SEGMENT
000054' 006301                    30          ASL   R1                    ;MY BLOCKS TO THEIR'S
000056' 006301                    31          ASL   R1
000060' 010112                    32          MOV   R1,(R2)               ;STORE IN SEGMENTATION REGISTER
000062' 000175  177766            33          JMP   @-12(R5)              ;DONE
                                  34          .END  LSD
```

SYMBOL	VALUE	DEFN	REFERENCES										
ASTACP	= 007400	12	29										
ASTSIZ	= 005000	13	19										
ISE	000000R	15	34										
FC	=X000007	8	17										
F0	=X000000	1	19	19	20	24							
F1	=X000001	2	19	20	21	22	23	25	26	29	30	31	32
F2	=X000002	3	25	26	27	32							
F3	=X000003	4											
F4	=X000004	5											
F5	=X000005	6	15	16	17	18	24	25	33				
F6	=X000006	7	15										

TOTAL NUMBER OF ERRORS -- 00000