```
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN         AAAAAAAAA            000000000
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN         AAAAAAAAA            000000000
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN         AAAAAAAAA            000000000
XXX         XXX      MMMMMM   MMMMMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
XXX         XXX      MMMMMM   MMMMMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
XXX         XXX      MMMMMM   MMMMMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
   XXX   XXX         MMM   MMM   MMM      OOO         OOO      NNNNNN      NNN      AAA         AAA      000      000000
   XXX   XXX         MMM   MMM   MMM      OOO         OOO      NNNNNN      NNN      AAA         AAA      000      000000
   XXX   XXX         MMM   MMM   MMM      OOO         OOO      NNNNNN      NNN      AAA         AAA      000      000000
      XXX            MMM         MMM      OOO         OOO      NNN   NNN   NNN      AAA         AAA      000   000   000
      XXX            MMM         MMM      OOO         OOO      NNN   NNN   NNN      AAA         AAA      000   000   000
      XXX            MMM         MMM      OOO         OOO      NNN   NNN   NNN      AAA         AAA      000   000   000
   XXX   XXX         MMM         MMM      OOO         OOO      NNN      NNNNNN      AAAAAAAAAAAAAAA      000000      000
   XXX   XXX         MMM         MMM      OOO         OOO      NNN      NNNNNN      AAAAAAAAAAAAAAA      000000      000
   XXX   XXX         MMM         MMM      OOO         OOO      NNN      NNNNNN      AAAAAAAAAAAAAAA      000000      000
XXX         XXX      MMM         MMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
XXX         XXX      MMM         MMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
XXX         XXX      MMM         MMM      OOO         OOO      NNN         NNN      AAA         AAA      000         000
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN      AAA         AAA         000000000
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN      AAA         AAA         000000000
XXX         XXX      MMM         MMM         OOOOOOOOO         NNN         NNN      AAA         AAA         000000000


   SSSSSSSSSSSS      EEEEEEEEEEEEEEE         QQQQQQQQQ
   SSSSSSSSSSSS      EEEEEEEEEEEEEEE         QQQQQQQQQ
   SSSSSSSSSSSS      EEEEEEEEEEEEEEE         QQQQQQQQQ
SSS                  EEE                  QQQ         QQQ
SSS                  EEE                  QQQ         QQQ
SSS                  EEE                  QQQ         QQQ
SSS                  EEE                  QQQ         QQQ
SSS                  EEE                  QQQ         QQQ
SSS                  EEE                  QQQ         QQQ
   SSSSSSSSS         EEEEEEEEEEEE         QQQ         QQQ
   SSSSSSSSS         EEEEEEEEEEEE         QQQ         QQQ
   SSSSSSSSS         EEEEEEEEEEEE         QQQ         QQQ
            SSS      EEE                  QQQ   QQQ   QQQ
            SSS      EEE                  QQQ   QQQ   QQQ
            SSS      EEE                  QQQ   QQQ   QQQ
            SSS      EEE                  QQQ      QQQ
            SSS      EEE                  QQQ      QQQ
            SSS      EEE                  QQQ      QQQ
SSSSSSSSSSSS         EEEEEEEEEEEEEEE         QQQQQQ   QQQ
SSSSSSSSSSSS         EEEEEEEEEEEEEEE         QQQQQQ   QQQ
SSSSSSSSSSSS         EEEEEEEEEEEEEEE         QQQQQQ   QQQ
```

XERSPL version 102(2263)/3(61) running on MTA061
START User CARPENTER,SA [400,4372] Job XMONA0 Seq. 4205 Date 25-Sep-78 12:27:16 Monitor IPC-F 603 [6B7] *START*
/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133
File: DSKZ:XMONA0.SEQ<057>[400,3341] Created: 21-Sep-78 09:15:00 Printed: 25-Sep-78 12:27:17
QUEUE Switches: /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:4060 /FORMS:NORMAL

IDENTIFICATION

PRODUCT CODE: AC-F080A-MC

PRODUCT NAME: CXMONA0 DEC/X11 MONITOR LIBRARY

PRODUCT DATE: SEPTEMBER 1978

MAINTAINER: DEC/X11 Support Group

```
                        .TITLE MNSPEC - MONITOR SPECIFICATION FILE

                        .SBTTL MONITOR TYPE CODE EQUATES

000001          A       =       000001          ;SMALLEST MONITOR
000002          B       =       000002          ;NO MEMORY MANAGEMENT
000004          C       =       000004          ;MEMORY MANAGEMENT
000010          D       =       000010          ;11/60
000020          E       =       000020          ;11/70
000040          F       =       000040          ;B WITH APT
000100          G       =       000100          ;C WITH APT
000200          H       =       000200          ;D WITH APT
000400          I       =       000400          ;E WITH APT
```

```
                                .SBTTL MONITOR TABLE

000000' 000101                  .ASCIZ  /A/
                                .EVEN
000002' 000001                  .WORD   A

000004' 000102                  .ASCIZ  /B/
                                .EVEN
000006' 000002                  .WORD   B

000010' 000103                  .ASCIZ  /C/
                                .EVEN
000012' 000004                  .WORD   C

000014' 000104                  .ASCIZ  /D/
                                .EVEN
000016' 000010                  .WORD   D

000020' 000105                  .ASCIZ  /E/
                                .EVEN
000022' 000020                  .WORD   E

000024' 000106                  .ASCIZ  /F/
                                .EVEN
000026' 000040                  .WORD   F

000030' 000107                  .ASCIZ  /G/
                                .EVEN
000032' 000100                  .WORD   G

000034' 000110                  .ASCIZ  /H/
                                .EVEN
000036' 000200                  .WORD   H

000040' 000111                  .ASCIZ  /I/
                                .EVEN
000042' 000400                  .WORD   I

000044'     377                 .BYTE -1
```

```
                                .SBTTL MODULE LIST WITH MONITOR TYPE CODES

                                ;+
                                ; THE FOLLOWING MODULES ARE LISTED FIRST BECAUSE THEY MUST
                                ; LIE WITHIN THE LOWEST 4K OF THE RUNTIME EXERCISER.
                                ;-

000045' 104 041505 000130       .ASCIZ  /DECX/
                                .EVEN
000052' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000054' 050101 046524 042117    .ASCIZ  /APTMOD/
000062' 000
        000064'                 .EVEN
000064' 000740                  .WORD   F!G!H!I

000066' 042502 051123 000126    .ASCIZ  /BESRV/
                                .EVEN
000074' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000076' 046102 043517 030066    .ASCIZ  /BLOG60/
000104' 000
        000106'                 .EVEN
000106' 000210                  .WORD   D!H

000110' 046102 043517 030067    .ASCIZ  /BLOG70/
000116' 000
        000120'                 .EVEN
000120' 000420                  .WORD   E!I


000122' 050107 000101           .ASCIZ  /GPA/
                                .EVEN
000126' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000130' 052113 051105 000122    .ASCIZ  /KTERR/
                                .EVEN
000136' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000140' 052113 042523 000124    .ASCIZ  /KTSET/
                                .EVEN
000146' 000734                  .WORD   C!D!E!G!H!I

000150' 040520 042522 051122    .ASCIZ  /PARERR/
000156' 000
        000160'                 .EVEN
000160' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000162' 041520 040504 040524    .ASCIZ  /PCDATA/
000170' 000
        000172'                 .EVEN
000172' 000777                  .WORD   A!B!C!D!E!F!G!H!I

000174' 041520 051124 041514    .ASCIZ  /PCTRLC/
```

```
     000202'    000
615           000204'                 .EVEN
616  000204'  000777                  .WORD   A!B!C!D!E!F!G!H!I
617
618  000206'  043120  044501  000114  .ASCIZ  /PFAIL/
619                                    .EVEN
620  000214'  000777                  .WORD   A!B!C!D!E!F!G!H!I
621
622  000216'  051120  046122  041517  .ASCIZ  /PRRLOC/
     000224'    000
623           000226'                 .EVEN
624  000226'  000734                  .WORD   C!D!E!G!H!I
625
626  000230'  042522  047514  000103  .ASCIZ  /RELOC/
627                                    .EVEN
628  000236'  000734                  .WORD   C!D!E!G!H!I
629
630  000240'  050122  051111  000121  .ASCIZ  /RPIRQ/
631                                    .EVEN
632  000246'  000777                  .WORD   A!B!C!D!E!F!G!H!I
633
634  000250'  051522  051124  054503  .ASCIZ  /RSTRCY/
     000256'    000
635           000260'                 .EVEN
636  000260'  000777                  .WORD   A!B!C!D!E!F!G!H!I
637
638                                    ;+
639                                    ; THE FOLLOWING MODULES MAY LIE ANYWHERE WITHIN THE
640                                    ; RUNTIME EXERCISER.
641                                    ;-
642
643  000262'  041501  044524  000126  .ASCIZ  /ACTIV/
644                                    .EVEN
645  000270'  000777                  .WORD   A!B!C!D!E!F!G!H!I
646
647  000272'  051101  041507  045510  .ASCIZ  /ARGCHK/
     000300'    000
648           000302'                 .EVEN
649  000302'  000777                  .WORD   A!B!C!D!E!F!G!H!I
650
651  000304'  040502  052103  053111  .ASCIZ  /BACTIV/
     000312'    000
652           000314'                 .EVEN
653  000314'  000777                  .WORD   A!B!C!D!E!F!G!H!I
654
655  000316'  040502  046504  046505  .ASCIZ  /BADMEM/
     000324'    000
656           000326'                 .EVEN
657  000326'  000734                  .WORD   C!D!E!G!H!I
658
659  000330'  042102  041501  053116  .ASCIZ  /BDACNV/
     000336'    000
660           000340'                 .EVEN
661  000340'  000777                  .WORD   A!B!C!D!E!F!G!H!I
662
663  000342'  042102  051526  053122  .ASCIZ  /BDVSRV/
```

```
     000350'    000
664           000352'                .EVEN
665  000352'  000776                 .WORD   B!C!D!E!F!G!H!I
666
667  000354'  047502  041501    000  .ASCIZ  /BOAC/
668           000362'                .EVEN
669  000362'  000777                 .WORD   A!B!C!D!E!F!G!H!I
670
671  000364'  047502  030501  000066  .ASCIZ  /BOA16/
672                                   .EVEN
673  000372'  000777                 .WORD   A!B!C!D!E!F!G!H!I
674
675  000374'  046103  042522  050117  .ASCIZ  /CLREOP/
     000402'    000
676           000404'                .EVEN
677  000404'  000777                 .WORD   A!B!C!D!E!F!G!H!I
678
679  000406'  046503  041504  054520  .ASCIZ  /CMDCPY/
     000414'    000
680           000416'                .EVEN
681  000416'  000777                 .WORD   A!B!C!D!E!F!G!H!I
682
683  000420'  046503  042104  042103  .ASCIZ  /CMDDCD/
     000426'    000
684           000430'                .EVEN
685  000430'  000777                 .WORD   A!B!C!D!E!F!G!H!I
686
687  000432'  046503  050104  041522  .ASCIZ  /CMDPRC/
     000440'    000
688           000442'                .EVEN
689  000442'  000777                 .WORD   A!B!C!D!E!F!G!H!I
690
691  000444'  046503  051104  052123  .ASCIZ  /CMDRST/
     000452'    000
692           000454'                .EVEN
693  000454'  000777                 .WORD   A!B!C!D!E!F!G!H!I
694
695  000456'  046503  052104  030502  .ASCIZ  /CMDTB1/
     000464'    000
696           000466'                .EVEN
697  000466'  000734                 .WORD   C!D!E!G!H!I
698
699  000470'  046503  052104  031102  .ASCIZ  /CMDTB2/
     000476'    000
700           000500'                .EVEN
701  000500'  000001                 .WORD   A
702
703  000502'  046503  052104  031502  .ASCIZ  /CMDTB3/
     000510'    000
704           000512'                .EVEN
705  000512'  000042                 .WORD   B!F
706
707  000514'  052103  050514  042525  .ASCIZ  /CTLQUE/
     000522'    000
708           000524'                .EVEN
709  000524'  000777                 .WORD   A!B!C!D!E!F!G!H!I
```

```
                                                .ASCIZ  /CONOF/
                                                .EVEN
000534' 000734                                  .WORD   C!D!E!G!H!I

000536' 050104  047505  000120                   .ASCIZ  /DPEOP/
                                                .EVEN
000544' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000546' 050104  052123  052122                   .ASCIZ  /DPSTRT/
000554'    000
                000556'                         .EVEN
000556' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000560' 051104  046520  042117                   .ASCIZ  /DRPMOD/
000566'    000
                000570'                         .EVEN
000570' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000572' 052504  041516  045510                   .ASCIZ  /DUNCHK/
000600'    000
                000602'                         .EVEN
000602' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000604' 051105  042522  000103                   .ASCIZ  /ERREC/
                                                .EVEN
000612' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000614' 044506  046114  046115                   .ASCIZ  /FILLML/
000622'    000
                000624'                         .EVEN
000624' 000001                                  .WORD   A

000626' 044506  046114  051515                   .ASCIZ  /FILLMS/
000634'    000
                000636'                         .EVEN
000636' 000776                                  .WORD   B!C!D!E!F!G!H!I

000640' 042507  050124  053523                   .ASCIZ  /GETPSW/
000646'    000
                000650'                         .EVEN
000650' 000777                                  .WORD   A!B!C!D!E!F!G!H!I
000652' 051110  040504  051104                   .ASCIZ  /HRDADR/
000660'    000
                000662'                         .EVEN
000662' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000664' 052110  040522  000120                   .ASCIZ  /HTRAP/
                                                .EVEN
000672' 000777                                  .WORD   A!B!C!D!E!F!G!H!I

000674' 041511  051123  047517                   .ASCIZ  /ICSROO/
000702'    000
                000704'                         .EVEN
000704' 000420                                  .WORD   E!I
```

```
758  000706' 041511  051123  055123  .ASCIZ  /ICSRSZ/
     000714'    000
759          000716'                 .EVEN
760  000716' 000420                  .WORD   E!I
761
762  000720' 042513  040530  000115  .ASCIZ  /KEXAM/
763                                  .EVEN
764  000726' 000776                  .WORD   B!C!D!E!F!G!H!I
765
766  000730' 043113  046111  000114  .ASCIZ  /KFILL/
767                                  .EVEN
768  000736' 000777                  .WORD   A!B!C!D!E!F!G!H!I
769
770  000740' 046513  050101     000  .ASCIZ  /KMAP/
771          000746'                 .EVEN
772  000746' 000776                  .WORD   B!C!D!E!F!G!H!I
773
774  000750' 046513  042117     000  .ASCIZ  /KMOD/
775          000756'                 .EVEN
776  000756' 000777                  .WORD   A!B!C!D!E!F!G!H!I
777
778  000760' 046513  047117  043117  .ASCIZ  /KMONOF/
     000766'    000
779          000770'                 .EVEN
780  000770' 000734                  .WORD   C!D!E!G!H!I
781
782  000772' 051113  047125     000  .ASCIZ  /KRUN/
783          001000'                 .EVEN
784  001000' 000734                  .WORD   C!D!E!G!H!I
785
786
787  001002' 051113  047125  000123  .ASCIZ  /KRUNS/
788                                  .EVEN
789  001010' 000043                  .WORD   A!B!F
790
791  001012' 051513  046525     000  .ASCIZ  /KSUM/
792          001020'                 .EVEN
793  001020' 000777                  .WORD   A!B!C!D!E!F!G!H!I
794
795  001022' 051513  051127     000  .ASCIZ  /KSWR/
796          001030'                 .EVEN
797  001030' 000777                  .WORD   A!B!C!D!E!F!G!H!I
798
799  001032' 052113  047117  043117  .ASCIZ  /KTONOF/
     001040'    000
800          001042'                 .EVEN
801  001042' 000734                  .WORD   C!D!E!G!H!I
802
803  001044' 050114  047117  043117  .ASCIZ  /LPONOF/
     001052'    000
804          001054'                 .EVEN
805  001054' 000776                  .WORD   B!C!D!E!F!G!H!I
806
807  001056' 051514  041524  045510  .ASCIZ  /LSTCHK/
     001064'    000
808          001066'                 .EVEN
```

```
809  001066' 000777                    .WORD   A!B!C!D!E!F!G!H!I
810
811  001070' 051515  042107  050505     .ASCIZ  /MSGDEQ/
     001076'    000
812          001100'                    .EVEN
813  001100' 000777                    .WORD   A!B!C!D!E!F!G!H!I
814
815  001102' 051515  042107  042507     .ASCIZ  /MSGDGE/
     001110'    000
816          001112'                    .EVEN
817  001112' 000777                    .WORD   A!B!C!D!E!F!G!H!I
818
819  001114' 051515  042107  047510     .ASCIZ  /MSGDHO/
     001122'    000
820          001124'                    .EVEN
821  001124' 000777                    .WORD   A!B!C!D!E!F!G!H!I
822
823  001126' 046516  041515  045510     .ASCIZ  /NAMCHK/
     001134'    000
824          001136'                    .EVEN
825  001136' 000777                    .WORD   A!B!C!D!E!F!G!H!I
826
827  001140' 042516  041127  000101     .ASCIZ  /NEWBA/
828                                     .EVEN
829  001146' 000734                    .WORD   C!D!E!G!H!I
830
831  001150' 052516  041515  045510     .ASCIZ  /NUMCHK/
     001156'    000
832          001160'                    .EVEN
833  001160' 000777                    .WORD   A!B!C!D!E!F!G!H!I
834
835  001162' 041120  042522  045501     .ASCIZ  /PBREAK/
     001170'    000
836          001172'                    .EVEN
837  001172' 000777                    .WORD   A!B!C!D!E!F!G!H!I
838
839  001174' 042520  042116      000    .ASCIZ  /PEND/
840          001202'                    .EVEN
841  001202' 000777                    .WORD   A!B!C!D!E!F!G!H!I
842
843  001204' 042520  042116  052111     .ASCIZ  /PENDIT/
     001212'    000
844          001214'                    .EVEN
845  001214' 000777                    .WORD   A!B!C!D!E!F!G!H!I
846
847  001216' 042120  052101  051105     .ASCIZ  /PDATER/
     001224'    000
848          001226'                    .EVEN
849  001226' 000777                    .WORD   A!B!C!D!E!F!G!H!I
850
851  001230' 043520  052105  040520     .ASCIZ  /PGETPA/
     001236'    000
852          001240'                    .EVEN
853  001240' 000777                    .WORD   A!B!C!D!E!F!G!H!I
854
855  001242' 046520  050101  031062     .ASCIZ  /PMAP22/
```

```
     001250'    000
856           001252'                  .EVEN
857  001252'  000777                    .WORD   A!B!C!D!E!F!G!H!I
858
859  001254'  047520  047516  000106    .ASCIZ  /PONOF/
860                                      .EVEN
861  001262'  000776                    .WORD   B!C!D!E!F!G!H!I
862
863  001264'  051120  052102  042117    .ASCIZ  /PRBTOD/
     001272'    000
864           001274'                  .EVEN
865  001274'  000777                    .WORD   A!B!C!D!E!F!G!H!I
866
867  001276'  051120  053105  000124    .ASCIZ  /PREVT/
868                                      .EVEN
869  001304'  000777                    .WORD   A!B!C!D!E!F!G!H!I
870
871  001306'  051120  053507  000102    .ASCIZ  /PRGWB/
872                                      .EVEN
873  001314'  000777                    .WORD   A!B!C!D!E!F!G!H!I
874
875  001316'  051120  040510  042122    .ASCIZ  /PRHARD/
     001324'    000
876           001326'                  .EVEN
877  001326'  000777                    .WORD   A!B!C!D!E!F!G!H!I
878
879  001330'  051120  051515  000107    .ASCIZ  /PRMSG/
880                                      .EVEN
881  001336'  000777                    .WORD   A!B!C!D!E!F!G!H!I
882
883  001340'  051120  051515  047107    .ASCIZ  /PRMSGN/
     001346'    000
884           001350'                  .EVEN
885  001350'  000777                    .WORD   A!B!C!D!E!F!G!H!I
886
887  001352'  051120  051515  051507    .ASCIZ  /PRMSGS/
     001360'    000
888           001362'                  .EVEN
889  001362'  000777                    .WORD   A!B!C!D!E!F!G!H!I
890
891  001364'  051120  052117  040517    .ASCIZ  /PROTOA/
     001372'    000
892           001374'                  .EVEN
893  001374'  000777                    .WORD   A!B!C!D!E!F!G!H!I
894
895  001376'  051120  040522  042116    .ASCIZ  /PRRAND/
     001404'    000
896           001406'                  .EVEN
897  001406'  000777                    .WORD   A!B!C!D!E!F!G!H!I
898
899  001410'  051120  047523  052106    .ASCIZ  /PRSOFT/
     001416'    000
900           001420'                  .EVEN
901  001420'  000777                    .WORD   A!B!C!D!E!F!G!H!I
902
903  001422'  040522  042116  046517    .ASCIZ  /RANDOM/
```

```
     001430'    000
304           001432'                  .EVEN
305  001432'  000777                    .WORD   A!B!C!D!E!F!G!H!I
306
307  001434'  042522  041514  046124    .ASCIZ  /RELCTL/
     001442'    000
308           001444'                  .EVEN
309  001444'  000734                    .WORD   C!D!E!G!H!I
310
311  001446'  047522  047516  000106    .ASCIZ  /RONOF/
312                                     .EVEN
313  001454'  000777                    .WORD   A!B!C!D!E!F!G!H!I
314
315  001456'  040523  051126  051505    .ASCIZ  /SAVRES/
     001464'    000
316           001466'                  .EVEN
317  001466'  000777                    .WORD   A!B!C!D!E!F!G!H!I
318
319  001470'  042523  042114  051505    .ASCIZ  /SELDES/
     001476'    000
320           001500'                  .EVEN
321  001500'  000777                    .WORD   A!B!C!D!E!F!G!H!I
322
323  001502'  052123  040522  000120    .ASCIZ  /STRAP/
324                                     .EVEN
325  001510'  000777                    .WORD   A!B!C!D!E!F!G!H!I
326
327  001512'  054523  041523  045514    .ASCIZ  /SYSCLK/
     001520'    000
328           001522'                  .EVEN
329  001522'  000776                    .WORD   B!C!D!E!F!G!H!I
330
331  001524'  054524  050520  042525    .ASCIZ  /TYPQUE/
     001532'    000
332           001534'                  .EVEN
333  001534'  000777                    .WORD   A!B!C!D!E!F!G!H!I
334
335  001536'  047125  046511  050101    .ASCIZ  /UNIMAP/
     001544'    000
336           001546'                  .EVEN
337  001546'  000420                    .WORD   E!I
338
339  001550'  047125  050111  000101    .ASCIZ  /UNIPA/
340                                     .EVEN
341  001556'  000420                    .WORD   E!I
342
343  001560'  041127  046106  046511    .ASCIZ  /WBFLIM/
     001566'    000
344           001570'                  .EVEN
345  001570'  000777                    .WORD   A!B!C!D!E!F!G!H!I
346
347  001572'  051527  041124  051525    .ASCIZ  /WSTBUS/
     001600'    000
348           001602'                  .EVEN
349  001602'  000777                    .WORD   A!B!C!D!E!F!G!H!I
350
```

```
                                        ;+
                                        ; THE FOLLOWING MODULES ARE LISTED LAST BECAUSE THEY MUST NOT
                                        ; LIE WITHIN THE BOTTOM 4K OF THE EXERCISER.
                                        ;-

001604' 041113  051104  000126  .ASCIZ  /KBDRV/
                                .EVEN
001612' 000777                  .WORD   A!B!C!D!E!F!G!H!I

001614' 050114  051104  000126  .ASCIZ  /LPDRV/
                                .EVEN
001622' 000776                  .WORD   B!C!D!E!F!G!H!I

001624' 052124  051104  000126  .ASCIZ  /TTDRV/
                                .EVEN
001632' 000777                  .WORD   A!B!C!D!E!F!G!H!I

001634' 044523  050132  040514  .ASCIZ  /SIZPLA/
001642'    000
        001644'                 .EVEN
001644' 000001                  .WORD   A

001646' 044523  050132  041114  .ASCIZ  /SIZPLB/
001654'    000
        001656'                 .EVEN
001656' 000042                  .WORD   B!F

001660' 044523  050132  041514  .ASCIZ  /SIZPLC/
001666'    000
        001670'                 .EVEN
001670' 000104                  .WORD   C!G

001672' 044523  050132  042114  .ASCIZ  /SIZPLD/
001700'    000
        001702'                 .EVEN
001702' 000210                  .WORD   D!H

001704' 044523  050132  042514  .ASCIZ  /SIZPLE/
001712'    000
        001714'                 .EVEN
001714' 000420                  .WORD   E!I

001716'    377                  .BYTE -1
        000001                  .END
```

```
A      = 000001    CSRA   = 000100    ENBNUL= 000001    LPSTAT= 000001    PSW    = 177776
ACSR   = 000102    CSRC   = 000102    ENDLST= 000000    MAPSTA= 000200    RANNUM= 000054
ACTBIT= 004000    CTRLC  = 000003    EOPBIT= 000001    MED    = 076600    RBUFEA= 000130
ADDR22= 001000    CTRLO  = 000017    ERRTYP= 000106    MEMPAS= 040000    RBUFPA= 000126
ADR    = 000006    CTRLU  = 000025    EVNTBE= 000200    MODEXH= 004000    RBUFSZ= 000132
APTFER= 000004    D      = 000010    EVNTHD= 000200    MODHOL= 002000    RBUFVA= 000124
APTPRE= 000200    DCEVNT= 000011    EVNTKT= 000203    MODSEL= 001000    RDSERV= 000101
ASB    = 000106    DEFRTN= 000400    EVNTPE= 000202    MSGCKD= 000010    RDWHMI= 000022
ASTAT  = 000104    DROPMO= 100000    EVNTRE= 000201    MSGCKS= 000011    RELERR= 000020
AUTO   = 000010    DSEVNT= 000014    F      = 000040    MSGDER= 000005    RELMOD= 020000
AUTOST= 020000    DT.ADD= 000042    FATERR= 100000    MSGDRP= 000017    RELTIM= 010000
AWAS   = 000110    DT.AP  = 000100    G      = 000100    MSGECH= 177777    RES1   = 000056
B      = 000002    DT.APK= 000076    H      = 000200    MSGEOP= 000013    RES2   = 000060
BIT0   = 000001    DT.BLS= 000034    HRDCNT= 000044    MSGHDR= 000004    RICHAR= 001060
BIT00  = 000001    DT.CF0= 000014    HRDPAS= 000050    MSGHNG= 000022    RPTDAT= 002000
BIT01  = 000002    DT.CF1= 000016    I      = 000400    MSGHRD= 000007    RSTRT  = 000112
BIT02  = 000004    DT.ERR= 000020    ICONT  = 000036    MSGMAP= 000021    RUBOUT= 000177
BIT03  = 000010    DT.ESI= 000044    ICOUNT= 000040    MSGNUL= 177775    RUNMOD= 100000
BIT04  = 000020    DT.EVN= 000000    IDNUM  = 000122    MSGPOP= 000002    R5VALU= 001740
BIT05  = 000040    DT.EXS= 000060    IE     = 000100    MSGPRM= 177776    SAM    = 075464
BIT06  = 000100    DT.FCH= 000037    INDPAR= 000040    MSGRES= 000001    SBADR  = 000102
BIT07  = 000200    DT.FCN= 000036    INHDRP= 040000    MSGSFT= 000006    SBKMOD= 000000
BIT08  = 000400    DT.HMX= 000104    INHEPR= 020000    MSGSKE= 000003    SBKSEL= 010000
BIT09  = 001000    DT.KBE= 000024    INHREL= 001000    MSGSMB= 000015    SC.ADR= 000006
BIT1   = 000002    DT.KBP= 000026    INHRRE= 000400    MSGSMH= 000014    SC.ALC= 000014
BIT10  = 002000    DT.KBR= 000022    INIT   = 000030    MSGSMS= 000016    SC.APC= 000016
BIT11  = 004000    DT.KBU= 000030    INTR   = 000120    MSGSTD= 000000    SC.CKL= 000002
BIT12  = 010000    DT.MLS= 000032    IOMOD  = 100000    MSGSYS= 000012    SC.CKP= 000004
BIT13  = 020000    DT.MTI= 000110    IOMODP= 102000    MSGVEC= 000020    SC.CLO= 000000
BIT14  = 040000    DT.OFF= 000070    IOMODR= 112000    NBKMOD= 001000    SC.HLD= 000010
BIT15  = 100000    DT.PAS= 000074    IOMODX= 110000    NCPUOP= 000020    SC.SCA= 000012
BIT2   = 000004    DT.PC  = 000002    JACK   = 035060    NOAPTY= 000002    SENDLS= 177777
BIT3   = 000010    DT.PFL= 000062    KIPAR0= 172340    NULL   = 000000    SOFCNT= 000042
BIT4   = 000020    DT.PSW= 000004    KIPAR1= 172342    OWEN   = 024020    SOFPAS= 000046
BIT5   = 000040    DT.PTA= 000064    KIPAR2= 172344    PAERR  = 000010    SPACE  = 000040
BIT6   = 000100    DT.RCS= 000102    KIPAR3= 172346    PARPRE= 002000    SPOINT= 000032
BIT7   = 000200    DT.REL= 000040    KIPAR4= 172350    PARSTA= 000100    SPVALU= 002200
BIT8   = 000400    DT.SCT= 000066    KIPAR5= 172352    PASCNT= 000034    SR0    = 177572
BIT9   = 001000    DT.SMX= 000106    KIPAR6= 172354    PDPLSI= 020000    SR1    = 177574
BKDEF  = 000002    DT.SP  = 000006    KIPAR7= 172356    PDP60  = 004000    SR2    = 177576
BKMOD  = 000020    DT.SSI= 000046    KIPDR0= 172300    PDP70  = 010000    SR3    = 172516
BKMODE= 040000    DT.ST0= 000010    KIPDR1= 172302    PRI0   = 000000    STAT   = 000026
BKSLSH= 000134    DT.ST1= 000012    KIPDR2= 172304    PRI1   = 000040    STATBI= 064757
C      = 000004    DT.SWR= 000056    KIPDR3= 172306    PRI4   = 000200    STAT1  = 000027
CAPRES= 000004    DT.SYP= 000072    KIPDR4= 172310    PRI5   = 000240    SUSPND= 000001
CASTAT= 000004    DT.WBU= 000050    KIPDR5= 172312    PRI6   = 000300    SVR0   = 000062
CDERCT= 000146    DT.WHL= 000054    KIPDR6= 172314    PRI7   = 000340    SVR1   = 000064
CDWDCT= 000144    DT.WLL= 000052    KIPDR7= 172316    PR0    = 000000    SVR2   = 000066
CKTIM  = 100000    DVID1  = 000014    KTERRO= 000040    PR4    = 000200    SVR3   = 000070
CLKPRE= 000001    E      = 000020    KTPRES= 000400    PR5    = 000240    SVR4   = 000072
CONFIG= 000056    ECCMEM= 000100    KTSTAT= 000020    PR6    = 000300    SVR5   = 000074
CQCVF  = 000001    ECCSTA= 000010    KTXTND= 040000    PR7    = 000340    SVR6   = 000076
CR     = 000015    ENBEOP= 010000    LF     = 000012    PS     = 177776    SYSCNT= 000052
```

```
SYSERR= 000100          UIPAR4= 177650          UIPDR3= 177606          WBUFEA= 000136          WTWHMI= 000222
TMPIO = 000002          UIPAR5= 177652          UIPDR4= 177610          WBUFPA= 000134          XFLAG = 000005
TQOVF = 000002          UIPAR6= 177654          UIPDR5= 177612          WBUFRQ= 000140          XOFF  = 000023
UIPAR0= 177640          UIPAR7= 177656          UIPDR6= 177614          WBUFSZ= 000142          XON   = 000021
UIPAR1= 177642          UIPDR0= 177600          UIPDR7= 177616          WDFR  = 000116          .     = 001717R
UIPAR2= 177644          UIPDR1= 177602          WASADR= 000104          WDTO  = 000114
UIPAR3= 177646          UIPDR2= 177604          WBSTAT= 000040          WTINRE= 000352

. ABS.  000000     000
        001717     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:MNSPEC,DSKZ:MNSPEC=SPMAC/ML,EQUATE,MNSPEC
 RUN-TIME: 1 1 .3 SECONDS
 RUN-TIME RATIO: 9/2=3.2
 CORE USED:  5K  (9 PAGES)
```

```
                                .TITLE  DECX-11  CONTROL MODULE
                                .IDENT  /V0.0/



                                ;++
                                ; MODULE NAME:
                                ;       DECX-11
                                ;
                                ; MODULE DESCRIPTION:
                                ;       THIS MODULE IS THE DEC/X-11 MONITOR CONTROL MODULE AND CONTAINS
                                ;       BOTH THE MONITOR'S DATABASE AND KERNEL ROUTINES
                                ;
                                ;
                                ;
                                ; VERSION:
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;
                                ;--
```

```
                                .SBTTL  DATABASE - MODULE HEADER


                                ;++
                                ; MODULE NAME:
                                ;       MONITOR DATABASE AREA
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THE MONITOR'S DATABASE CONTAINS ALL OF THE PARAMETERS AND VALUES
                                ;       THAT ARE USED BY BOTH THE MONITOR'S KERNEL ROUTINE AND THE
                                ;       SUBORDINATE MONITOR ROUTINES.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       NONE
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       NONE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;--
```

```
                        .SBTTL  DATABASE - TRAP AND VECTOR GLOBAL DEFINITIONS
;
;
;
;
;
                        .GLOBL  HTBUSS
                        .GLOBL  HTREIN
                        .GLOBL  RPIRQ
                        .GLOBL  PWRFL
                        .GLOBL  PWRDOWN
                        .GLOBL  PBDVEC
                        .GLOBL  STINT
                        .GLOBL  KBINT
                        .GLOBL  TTINT
                        .GLOBL  MEMPAR
                        .GLOBL  MEMGMT
                        .GLOBL  AP.PB
                        .GLOBL  $ENDAD
                        .GLOBL  CAST
;
```

```
                                .SBTTL  DATABASE - STUFF TRAP AND VECTOR AREAS


        000000                          .ASECT
        000000                          .=0                     ;START THIS AT ABSOLUTE 0

                                ;+
                                ; LOAD TRAP AND VECTOR AREA
                                ;-

000000  105000                          .WORD   105000          ;(RESERVED)
000002  000000                          .WORD   0
000004  000000G         TIMOUT: .WORD   HTBUSS          ;TIME OUT AND OTHER ERRORS
000006  000340                  .WORD   PRI7
000010  000000G         RESERV: .WORD   HTREIN          ;RESERVED INSTRUCTION VECTOR
000012  000340                  .WORD   PRI7
000014  000016                  .+2                     ;HALT IF T BIT SETS
000016  000000                  HALT
000020  000000G         IOTINS: .WORD   RPIRQ           ;IOT INSTRUCTION VECTOR
000022  000340                  .WORD   PRI7
000024  000000G         PWRFL:  .WORD   PWRDOWN         ;POWER FAILURE VECTOR
000026  000340                  .WORD   PRI7
000030  000000G         EMTINS: .WORD   PBDVEC          ;EMT INSTRUCTION VECTOR
000032  000000                  .WORD   PRI0
000034  000000G         TRPINS: .WORD   STINT           ;TRAP INSTRUCTION
000036  000000                  .WORD   PRI0
000040  000000                  .WORD   0               ;40
000042  000000                  .WORD   0               ;42
000044  000000G         APTVEC: .WORD   AP.PB           ;44 - APT PARAMETER BLOCK
000046  000000G                 .WORD   $ENDAD          ;46
000050  000000                  .WORD   0               ;50
000052  000000                  .WORD   0               ;52
000054  000000G                 .WORD   CAST            ;54
000056  000001                  .WORD   1               ;56

        000060                  .=60
000060  000000G         KBVEC:  .WORD   KBINT           ;KEYBOARD INTERRUPT VECTOR
000062  000040                  .WORD   PRI1
000064  000000G         TTVEC:  .WORD   TTINT           ;TERMINAL INTERRUPT VECTOR
000066  000040                  .WORD   PRI1


                                ;+
                                ; 70 TO 176 AND 204 TO 776 HAVE .+2 AND HALT. THESE VECTOR LOCATIONS
                                ; MAY BE STUFFED BY CERTAIN MONITORS WITH THE ADDRESS OF THE BAD
                                ; VECTOR SERVICE ROUTINE. THE PARITY AND KT VECTORS (114 & 250) MAY
                                ; ALSO BE STUFFED WITH PARITY AND KT TRAP ROUTINES.
                                ;-

        000022                  .REPT   ^D<18>
                                .NLIST
                                .+2
                                HALT
                                .LIST
```

```
                                                .ENDR
                                                .=200
000200  000200  000167  001774          START:  JMP     DX.STRT         ;GO TO START ADDRESS
                                                .REPT   ^D<95>
                                                .NLIST
                                                .+2
                                                HALT
                                                .LIST
                                                .ENDR
                                                .=1000
001000  001000  000167  001174          RESTART:        JMP DX.STRT
```

```
   666                                  .SBTTL  DATABASE - DATA TABLE GLOBAL DEFINITIONS
   667
   668
   669
   670                                  .GLOBL  DTABLE
```

DATABASE - DATA TABLE VALUES GLOBAL DEFINITIONS SEQ 0024

```
                        .SBTTL  DATABASE - DATA TABLE VALUES GLOBAL DEFINITIONS

                        .GLOBL  OV.KBBUF                        ;ADDRESS OF KEYBOARD INPUT BUFFER
                        .GLOBL  AC.MODQ                 ;MODULE-QUEUE-LIST ADDRESS
```

```
                                .SBTTL  DATABASE - DATA TABLE ALLOCATION


                                ;+
                                ; DATA TABLE ALLOCATION
                                ;-


001004                          DTABLE:
001004  000000                          .WORD   0               ;DT.EVNT - EVENT CODE
001006  000000                          .WORD   0               ;DT.PC - PC+ OF TRAP
001010  000000                          .WORD   0               ;DT.PSW - PSW AT TIME OF TRAP
001012  000000                          .WORD   0               ;DT.SP - STACK POINTER AT TIME OF TRAP
001014  000000                          .WORD   0               ;DT.ST0 - STATUS INDICATOR 0
001016  000000                          .WORD   0               ;DT.ST1 - STATUS INDICATOR 1
001020  000000                          .WORD   0               ;DT.CF0 - CONFIGURATION WORD 0
001022  000000                          .WORD   0               ;DT.CF1 - CONFIGURATION WORD 1
001024  000000                          .WORD   0               ;DT.ERR - ERROR WORD
001026  000000                          .WORD   0               ;DT.KBRSP - ADDRESS OF KEYBOARD RESPONSE BUFFER
001030  000000                          .WORD   0               ;DT.KBECH - ADDRESS OF KEYBOARD ECHO BUFFER
001032  000000                          .WORD   0               ;DT.KBPRM - ADDRESS OF KEYBOARD PROMPT
001034  000000G                         .WORD   OV.KBBUFF       ;DT.KBUF - ADDRESS OF KEYBOARD BUFFER POINTER
001036  000000G                         .WORD   AC.MODQ         ;DT.MLST - ADDRESS OF MODULE LIST
001040  000000                          .WORD   0               ;DT.BLST - ADDRESS OF MODULE LIST
001042      004                         .BYTE   4               ;DT.FCNT - FILLER COUNT (# OF NULLS)
001043      015                         .BYTE   <CR>            ;DT.FCHAR - CHARACTER AFTER WHICH NULLS ARE O/P
001044  000200                          .WORD   200             ;DT.REL - RELOCATION CONSTANT
001046  000200                          .WORD   200             ;DT.ADDR - CURRENT ADDRESS
001050  000000                          .WORD   0               ;DT.ESIZ - EXERCISER SIZE
001052  000000                          .WORD   0               ;DT.SSIZ - SYSTEM SIZE IN PAR FORMAT
001054  000000                          .WORD   0               ;DT.WBUF - ADDR OF WRITE BUFFER
001056  000000                          .WORD   0               ;DT.WLLMT - WRITE BUFFER LOW LIMIT
001060  000000                          .WORD   0               ;DT.WHLMT - WRITE BUFFER HIGH LIMIT
001062  000000                          .WORD   0               ;DT.SWR - SOFTWARE SWITCH REG(FRONT PANEL MEANS NOTHING)
001064  000000                          .WORD   0               ;DT.EXS - # OF EXERCISER SYS ERRORS
001066  000000                          .WORD   0               ;DT.PFL - # OF POWER FAILS
001070  001116                          .WORD   PARTAB          ;DT.PTA - ADDRESS OF PARITY TABLE
001072  001160                          .WORD   SCTAB           ;DT.SCT - ADDRESS OF CLOCK TABLE
001074  000000                          .WORD   0               ;DT.OFFSET - OFFSET INTO CURRENT 124K BANK
001076  001100                          .WORD   DTABLE+DT.PAS   ;DT.SYP - SYSTEM END OF PASS COUNT POINTER
001100  000000                          .WORD   0               ;DT.PAS - SYSTEM END OF PASS COUNT IF NOT UNDER APT
001102  001104                          .WORD   DTABLE+DT.AP    ;DT.APK - KEEP ALIVE COUNTER POINTER
001104  000000                          .WORD   0               ;DT.AP - KEEP ALIVE IF NOT UNDER APT
001106  000000                          .WORD   0               ;DT.RCS - FAKE KB ADDRESS FOR APT USE
001110  000024                          .WORD   ^D20            ;DT.HMX - MAX # OF HARD ERRORS ALLOWED BY OPTION MODULE
001112  000050                          .WORD   ^D40            ;DT.SMX - MAX # OF SOFT ERRORS ALLOWED BY OPTION MODULE
001114  001604                          .WORD   ^D900           ;DT.MTIME - MAX END OF PASS TIME(15 MIN.) BY A MODULE
```

```
                                    DATABASE - PARITY TABLE -- PARTAB -- GLOBAL DEFINITION                    SEQ 0026

   727                              .SBTTL  DATABASE - PARITY TABLE -- PARTAB -- GLOBAL DEFINITION
   728
   729
   730                              .GLOBL  PARTAB
   731
   732
   733
   734                              .SBTTL  DATABASE - PARITY TABLE -- PARTAB -- ALLOCATION
   735
   736
   737
   738                              ;+
   739                              ; 16 LOCATIONS FOR PARITY CSRS. THE TABLE IS TERMINATED WITH A 0
   740                              ;-
   741
   742  001116  000000  000000  000000  PARTAB  :       .WORD 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        001124  000000  000000  000000
        001132  000000  000000  000000
        001140  000000  000000  000000
        001146  000000  000000  000000
        001154  000000  000000
   743
```

```
                                .SBTTL  DATABASE - SYSTEM CLOCK TABLE -- SCTAB -- GLOBAL DEFINITIONS


                                ;+
                                ; SYSTEM CLOCK TABLE GLOBAL DEFINITIONS
                                ;-

                                .GLOBL  SCTAB
                                .GLOBL  CLOCK
                                .GLOBL  CLOCKL
                                .GLOBL  CLOCKP

                                ;+
                                ; GLOBALS REFERENCED BY THIS DATABASE
                                ;-

                                .GLOBL  KW11L           ;POINTS TO THE HEADER ADDRESS OF KW11-L
                                                        ;OPTION MODULE
                                .GLOBL  KW11P           ;POINTS TO THE HEADER ADDRESS OF KW11-P
                                                        ;OPTION MODULE
                                .GLOBL  LCLEAR          ;INITIALIZE MODULE TIME TABLE IN KW11-L
                                .GLOBL  PCLEAR          ;INITIALIZE MODULE TIME TABLE IN KW11-P
```

```
                                .SBTTL  DATABASE - SYSTEM CLOCK TABLE AND OFFSET ADDRESSES


                                ;+
                                ; SYSTEM CLOCK TABLE ALLOCATION
                                ;-

001160                          SCTAB:
001160  000000                  CLOCK:  .WORD   0       ;SC.CLO
001162  000000G                 CLOCKL: .WORD   KW11L   ;SC.CKL
001164  000000G                 CLOCKP: .WORD   KW11P   ;SC.CKP
001166  000000                          .WORD   0       ;SC.ADR
001170  000000                          .WORD   0       ;SC.HLD
001172  000000                          .WORD   0       ;SC.SCA
001174  000000G                         .WORD   LCLEAR  ;SC.ALC
001176  000000G                         .WORD   PCLEAR  ;SC.APC
```

```
                                .SBTTL  DATABASE - DEVICE REGISTER GLOBAL DEFINITIONS

                                .GLOBL  RCSR
                                .GLOBL  RBUF
                                .GLOBL  XCSR
                                .GLOBL  XBUF
                                .GLOBL  LPCSR
                                .GLOBL  LPBUF

                                .SBTTL  DATABASE - DEVICE REGISTER DEFINITIONS

001200  177560          RCSR:   .WORD   177560          ;DL-11 RECEIVER CONTROL AND STATUS REGISTER
001202  177562          RBUF:   .WORD   177562          ;DL-11 RECEIVER DATA BUFFER REGISTER
001204  177564          XCSR:   .WORD   177564          ;DL-11 TRANSMITTER CONTROL AND STATUS REGISTER
001206  177566          XBUF:   .WORD   177566          ;DL-11 TRANSMITTER DATA BUFFER REGISTER
001210  177514          LPCSR:  .WORD   177514          ;LP 11 CONTROL STATUS REGISTER
001212  177516          LPBUF:  .WORD   177516          ;LP 11 BUFFER REGISTER
```

```
                                .SBTTL  DATABASE - CPU RELATED REGISTERS GLOBAL DEFINITIONS
                                ;+
                                ;11/70
                                ;-
                                .GLOBL  CPULAR
                                .GLOBL  CPUHAR
                                .GLOBL  CPUMER
                                .GLOBL  CPUMNT
                                .GLOBL  CPUCPE
                                .GLOBL  CPULSZ
                                ;+
                                ; CACHE CONTROL REGISTER AND TEMPORARY STORAGE
                                ;-
                                .GLOBL  CCNTRL
                                .GLOBL  KONTRL
                                .SBTTL  DATABASE - CPU RELATED REGISTER DEFINITIONS
                                ;+
                                ; 11/70
                                ;-
001214  177740                 CPULAR: .WORD   177740          ;LOW ERROR ADDRESS REGISTER
001216  177742                 CPUHAR: .WORD   177742          ;HIGH ERROR ADDRESS REGISTER
001220  177744                 CPUMER: .WORD   177744          ;MEMORY SYSTEM ERROR REGISTER
001222  177750                 CPUMNT: .WORD   177750          ;MAINTENANCE REGISTER
001224  177766                 CPUCPE: .WORD   177766          ;CPU ERROR REGISTER
001226  177760                 CPULSZ: .WORD   177760          ;LOW SIZE REGISTER
                                ;+
                                ; CACHE CONTROL REGISTER AND TEMPORARY STORAGE
                                ;-
001230  177746                 CCNTRL: .WORD   177746          ;CACHE CONTROL REGISTER
001232  000000                 KONTRL: .WORD   0               ;TEMPORARY STORAGE FOR CACHE OPERATIONS
```

```
                                .SBTTL  DATABASE - MISCELLANEOUS STORAGE GLOBAL DEFINITIONS                    SEQ 0031

                                .SBTTL  DATABASE - MISCELLANEOUS STORAGE GLOBAL DEFINITIONS


                                .GLOBL  CTRLOF
```

```
                                .SBTTL  DATABASE - MISCELLANEOUS STORAGE


001234  000000                  CTRLOF: .WORD   0                       ;ENABLE/DISABLE TERMINAL OUTPUT FLAG
```

```
                                .SBTTL  KERNEL - MODULE HEADER


                                ;++
                                ; MODULE NAME:
                                ;       KERNEL ROUTINE
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE CONTROLS THE EXECUTION OF THE DEC/X-11 SYSTEM
                                ;       EXERCISER, I.E., IT IS RESPONSIBLE FOR STARTING THE EXERCISER
                                ;       WHEN FIRST LOADED INTO MEMORY,  INITIALIZING THE ACTUAL RUNNING
                                ;       OF IT, COORDINATING THE RECEIVING AND SERVICING OF THE REAL TIME
                                ;       EVENTS AND PERFORMING SCHEDULING AND ACTIVATING FUNCTIONS OF THE
                                ;       EXERCISER.
                                ;       THIS IS ACCOMPLISHED WITH THE USE OF THE FOLLOWING ROUTINES:
                                ;
                                ;               DX.STRT         THIS ROUTINE PERFORMS THE DUTIES REQUIRED
                                ;                               TO GET THE EXERCISER GOING WHEN FIRST
                                ;                               LOADED INTO MEMORY
                                ;               DX.RSTRT        THIS ROUTINE PERFORMS THOSE FUNCTIONS NEC-
                                ;                               ESSARY TO GET THE EXERCISE RESTARTED
                                ;                               AFTER IT HAS BEEN RUNNING
                                ;               DX.INI          THIS ROUTINE IS RESPONSIBLE FOR INITIAL-
                                ;                               IZATING THE EXERCISER
                                ;               DX.CAPTURE      THIS ROUTINE COORDINATES THE RECEIVING
                                ;                               OF THE EXERCISER'S REALTIME EVENTS AND
                                ;                               THE DISPATCHING OF CONTROL TO THE AP-
                                ;                               PROPRIATE SERVICE ROUTINES
                                ;               DX.DEQUEUE      THIS ROUTINE PERFORMS THE DE-QUEUING AND
                                ;                               ACTIVATING FUNCTIONS OF THE MONITOR
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ;IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       MANY
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       1.
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;--
```

```
                                .SBTTL  KERNEL - COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
001236                                  STRUCT
(1) 001236                      .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN = 1
        000001                          $LSTTAG = 1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  DX.CAP                  ;CAPTURE EVENT ROUTINE ENTRY POINT
                                .GLOBL  DX.DEQ                  ;DE-QUEUING ROUTINE ENTRY POINT
                                .GLOBL  DX.RSTRT                ;RESTART EXERCISER ENTRY POINT
                                .GLOBL  DX.INI                  ;INITIALIZATION ROUTINE
                                .GLOBL  DX.HDR                  ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S HEADER ADDRESS
                                .GLOBL  DX.RET                  ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S RESUME ADDRESS
                                .GLOBL  DX.MON                  ;MONITOR ID
                                .GLOBL  DX.R5
                                .GLOBL  DX.KFL                  ;CR FLAG
                                .GLOBL  DX.SP
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ST.EXT                  ;SOFTWARE TRAP HANDLER EXIT ADDRESS
                                .GLOBL  HT.EXT                  ;HARD ERROR TRAP HANDLER EXIT ADDRESS
                                .GLOBL  APTSLP                  ;APT SIZING AND SLEEP ROUTINE
                                .GLOBL  RSTRCY                  ;RESTART RECOVERY ROUTINE
                                .GLOBL  KBINI                   ;KEYBOARD DRIVER ENTRY POINT
                                .GLOBL  CMDRST                  ;OUTPUT KEYBOARD PROMPT ROUTINE
                                .GLOBL  PREVT                   ;PROCESS EVENT MODULE ENTRY POINT
                                .GLOBL  PWRDOW                  ;POWER DOWN MODULE ENTRY POINT
                                .GLOBL  MSGDEQ                  ;MESSAGE DE-QUEUING MODULE ENTRY POINT
                                .GLOBL  MSGDHOOK                ;MESSAGE HOOK MODULE ENTRY POINT
                                .GLOBL  SAVREG                  ;SAVE REGISTERS ROUTINE
                                .GLOBL  RESREG                  ;RESTORE REGISTERS ROUTINE
                                .GLOBL  LPINT                   ;LP11 INTERRUPT ROUTINE
                                .GLOBL  CQINI                   ;CONTROL QUEUE INITIALIZATOR MODULE ENTTRY POINT
                                .GLOBL  TQINI                   ;TYPE QUEUE INITIALIZATOR MODULE ENTRY POINT
                                .GLOBL  CHKACT                  ;CHECK FOR ACTIVE MODULES
                                .GLOBL  CHKEOP                  ;CHECK FOR END OF PASS TIME
                                .GLOBL  CLREOP                  ;CLEAR EOP INDICATORS
                                .GLOBL  ACTIV                   ;ACTIVATE NEXT OPTION MODULE ROUTINE
                                .GLOBL  AC.TPTR                 ;MODULE-TYPE-LIST POINTER
                                .GLOBL  AC.TYPE                 ;MODULE-TYPE-LIST
                                .GLOBL  AC.MPTR                 ;MODULE-QUEUE-LIST POINTER
                                .GLOBL  BACTIV                  ;BACKGROUND MODULE ACTIVATE ROUTINE
```

```
1002                                    .GLOBL  BA.STAT                 ;BACKGROUND MODULE STATUS WORD
1003                                    .GLOBL  CMDPRC                  ;COMMAND PROCESS
1004                                    .GLOBL  BA.MPTR                 ;BACKGROUND MODULE LIST POINTER
1005                                    .GLOBL  BA.HDR                  ;BACKGROUND NEXT-TO-EXECUTE SLOT
1006                                    .GLOBL  BA.RET                  ;BACKGROUND NEXT-TO-EXECUTE SLOT
1007                                    .GLOBL  BA.STAT                 ;BACKGROUND MECHANISM STATUS WORD
1008                                    .GLOBL  MD.BSY                  ;MESSAGE DE-QUEUE BUSY INDICATOR
1009                                    .GLOBL  DEQCQ                   ;DE-QUEUE NEXT ENTRY IN CONTROL QUEUE
1010                                    .GLOBL  IERRLO                  ;INITIALIZE ERROR LOGGING CPU'S ROUTINE
1011                                    .GLOBL  ERREC                   ;ERROR RECOVERY  ROUTINE
1012                                    .GLOBL  RELCTL                  ;RELOCATION MODULE
1013                                    .GLOBL  SIZPOL                  ;SIZE AND POLL SYSTEM
1014                                    .GLOBL  PRRLOC                  ;PROCESS RELOCATION
1015                                    .GLOBL  BDACNV                  ;BINARY TO DECIMAL ASCII CONVERSION
1016                                    .GLOBL  KTSET                   ;KT APRS SETUP
1017                                    .GLOBL  KCON                    ;CACHE ON KEYBOARD COMMAND
1018                                    .GLOBL  KKTON                   ;KT ON KEYBOARD COMMAND
1019                                    .GLOBL  KMON                    ;MAP BOX ON KEYBOARD COMMAND
1020                                    .GLOBL  KPON                    ;PARITY ON KEYBOARD COMMAND
1021                                    .GLOBL  KRUN                    ;RUN COMMAND
1022                                    .GLOBL  KROTON                  ;WRITE BUFFER ROTATION KEYBOARD COMMAND
1023                                    .GLOBL  DPSTRT                  ;AUTOMATIC MODE START-UP
1024                                    .GLOBL  DPEOP                   ;AUTOMATIC MODE EOP ROUTINE
1025
1026                                    ;
1027                                    ;****************
1028                                    ;
1029                                    ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
1030                                    ;
1031  001236  000000                    DX.KFL: .WORD   0               ;SET WHEN A CR HAS BEEN STRUCK
1032  001240  000000                    DX.HDR: .WORD   0               ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S HEADER ADDRESS
1033  001242  000000                    DX.RET: .WORD   0               ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S RESUME ADDRESS
1034  001244  000000                    DX.QFLG: .WORD  0               ;CONTROL QUEUE UNDERFLOW INDICATOR
1035  001246  001740                    DX.R5:  .WORD   R5VALUE         ;POINTER TO MONITOR'S R5 STACK
1036  001250  002200                    DX.SP:  .WORD   SPVALUE         ;POINTER TO MONITOR'S R6 STACK
1037                                    ;
1038                                    ;****************
1039                                    ;
1040                                    ; LOCAL STORAGE - MESSAGES, ETC.
1041                                    ;
1042  001252  042045  041505  054057    DX.TITLE: .ASCII ?%DEC/X-11 EXERCISER (MONITOR V00.0) MD-ZZ-CXMON-A?
      001260  030455  020061  054105
      001266  051105  044503  042523
      001274  020122  046450  047117
      001302  052111  051117  053040
      001310  030060  030056  020051
      001316  042115  055055  026532
      001324  054103  047515  026516
      001332     101
1043  001333     045  047515  044516              .ASCII        /%MONITOR:  /
      001340  047524  035122  020040
1044  001345  020040  020040     000    DX.MON:    .ASCIZ        /    /
1045  001353     045  054523  052123    DX.SIZ : .ASCII /%SYSTEM SIZE: /
      001360  046505  051440  055111
      001366  035105     040
1046  001371  000005                    DX.SZK : .BLKB  ^D<5>
```

```
 1047  001376  045440  000045                          .ASCIZ / K%/
 1048  001402  047524  052040  051505  DX.MED : .ASCIZ /TO TEST LD MEDIA CLR LOC 40%/
       001410  020124  042114  046440
       001416  042105  040511  041440
       001424  051114  046040  041517
       001432  032040  022460     000
 1049          001440                        .EVEN
 1050
 1051
 1052  001440  001442                  DX.FAK :        .WORD   DX.FK1                  ;FAKE KEYBOARD BUFFER
 1053  001442  000015                  DX.FK1 :        .WORD   CR
 1054
 1055                                  ;
 1056                                  ;****************
 1057                                  ; BOTTOM OF R5 STACK
```

```
                                .SBTTL  KERNEL - START UP ROUTINE


          002200                        .=2200

                                ;+
                                ; THIS ROUTINE IS RESPONSIBLE FOR SETTING UP THE DIFFERENT SOFTWARE COMPONENTS
                                ; OF THE MONITOR WHEN THE EXERCISER IS FIRST LOADED INTO MEMORY OR RESTARTED AT LOC. 100
                                ;-


002200                          DX.STRT:

                                ;+
                                ;  STUFF THE POWERFAILURE VECTOR AND SET UP THE STACKS
                                ;-

002200                          LET PWRFL := #PWRDOWN
 (4) 002200  012767  000000G 175616                                                     MOV     #PWRDOWN,PWRFL
002206                          LET SP := #SPVALUE
 (4) 002206  012706  002200                                                            MOV     #SPVALUE,SP
002212                          LET R5 := #R5VALUE
 (4) 002212  012705  001740                                                            MOV     #R5VALUE,R5

                                ;+
                                ; IF UNDER APT, WE MAY OR MAY NOT HAVE A SYSTEM TTY.  FOR NOW, ASSUME
                                ; WE DO AND STUFF THE TTY ADDRESS.  ALSO, CLEAR THE "NO APT TTY" BIT IN
                                ; CONFIGURATION WORD 0.
                                ;-
002216                          LET RCSR := #177560
 (4) 002216  012767  177560  176754                                                    MOV     #177560,RCSR
002224                          LET DTABLE+DT.CF0 := DTABLE+DT.CF0 CLR.BY #NOAPTY
 (6) 002224  042767  000002  176566                                                    BIC     #NOAPTY,DTABLE+D


002232                           INLINE <RESET>
 (2) 002232  000005                                                                    RESET

                                ;+
                                ; IF FIRST TIME THROUGH DO THE FOLLOWING
                                ;-

002234                          IF DTABLE+DT.ESIZ EQ #0 THEN
 (6) 002234  005767  176610                                                            TST     DTABLE+DT.ESIZ
 (9) 002240  001022                                                                    BNE     50000$

                                ;+
                                ; STUFF THE EXIT ADDRESS FOR THE TWO TRAP HANDLERS: SOFTWARE
                                ; AND HARDWARE
                                ;-

002242                                  LET ST.EXT := #DX.CAP
 (4) 002242  012767  003242  000000G                                                   MOV     #DX.CAP,ST.EXT
002250                                  LET HT.EXT := #DX.CAP
 (4) 002250  012767  003242  000000G                                                   MOV     #DX.CAP,HT.EXT
```

```
1105
1106                                    ;+
1107                                    ; SIZE AND POLL THE SYSTEM AND SORT MODULE Q LIST.  THEN DO ANOTHER RESET.
1108                                    ; CLEAR THE STAT INDICATORS
1109                                    ;-
1110
1111  002256                                    LET DTABLE+DT.ST0 := #0
 (4)  002256  005067  176532                                                    CLR     DTABLE+DT.ST0
1112  002262                                    LET DTABLE+DT.ST1 := #0
 (4)  002262  005067  176530                                                    CLR     DTABLE+DT.ST1
1113  002266                                    CALL SIZPOL IN <#DTABLE>
 (3)  002266  010546                                                            MOV     R5,-(SP)
 (4)  002270  012745  001004                                                    MOV     #DTABLE,-(R5)
 (3)  002274  004767  000000G                                                  JSR     PC,SIZPOL
 (3)  002300  012605                                                            MOV     (SP)+,R5
1114  002302                                    INLINE <RESET>
 (2)  002302  000005                                                            RESET
1115                                    ;+
1116                                    ; ELSE THIS IS NOT THE FIRST TIME THROUGH.
1117                                    ;-
1118  002304                            ELSE
 (4)  002304  000427                                                            BR      50001$
 (3)  002306                                                          50000$:
1119
1120                                    ;+
1121                                    ; GO RECOVER FROM THE RESTART, AND DON'T BE LONG
1122                                    ;-
1123
1124  002306                                    LET DTABLE+DT.ST0 := DTABLE+DT.ST0 CLR.BY #177400
 (6)  002306  042767  177400  176500                                            BIC     #177400,DTABLE+D
1125  002314                                    LET DTABLE+DT.ST1 := #0
 (4)  002314  005067  176476                                                    CLR     DTABLE+DT.ST1
1126  002320                                    CALL RSTRCY IN <#DTABLE>
 (3)  002320  010546                                                            MOV     R5,-(SP)
 (4)  002322  012745  001004                                                    MOV     #DTABLE,-(R5)
 (3)  002326  004767  000000G                                                  JSR     PC,RSTRCY
 (3)  002332  012605                                                            MOV     (SP)+,R5
1127
1128                                    ;+
1129                                    ; IF THE RTE IS RELOCATED, THEN RELOCATE BACK DOWN
1130                                    ;-
1131
1132  002334                                    IF DTABLE+DT.ADDR NE #200 THEN
 (6)  002334  026727  176506  000200                                            CMP     DTABLE+DT.ADDR,#
 (9)  002342  001410                                                            BEQ     50002$
1133  002344                                          CALL PRRLOC IN <#DTABLE,#200>
 (3)  002344  010546                                                            MOV     R5,-(SP)
 (5)  002346  012745  000200                                                    MOV     #200,-(R5)
 (4)  002352  012745  001004                                                    MOV     #DTABLE,-(R5)
 (3)  002356  004767  000000G                                                  JSR     PC,PRRLOC
 (3)  002362  012605                                                            MOV     (SP)+,R5
1134  002364                                    ENDIF
 (4)  002364                                                          50002$:
1135  002364                            ENDIF
 (4)  002364                                                          50001$:
1136
```

```
 1137                                   ;+
 1138                                   ; IF LOADED UNDER APT, DO APT INITIALIZATION.  OTHERWISE, CLEAR THE APT
 1139                                   ; PRESENT BIT.
 1140                                   ;-
 1141
 1142  002364                           IF @#44 NE #0 THEN
  (6)  002364  005737  000044                                                                  TST     @#44
  (9)  002370  001407                                                                          BEQ     50003$
 1143  002372                                   CALL APTSLP IN <#DTABLE>
  (3)  002372  010546                                                                          MOV     R5,-(SP)
  (4)  002374  012745  001004                                                                  MOV     #DTABLE,-(R5)
  (3)  002400  004767  000000G                                                                JSR     PC,APTSLP
  (3)  002404  012605                                                                          MOV     (SP)+,R5
 1144  002406                           ELSE
  (4)  002406  000403                                                                          BR      50004$
  (3)  002410                                                                        50003$:
 1145  002410                                   LET DTABLE+DT.CF0 := DTABLE+DT.CF0 CLR.BY #APTPRES
  (6)  002410  042767  000200  176402                                                         BIC     #APTPRES,DTABLE+
 1146  002416                           ENDIF
  (4)  002416                                                                        50004$:
 1147
 1148                                   ;+
 1149                                   ; LOWER THE PRIORITY
 1150                                   ;-
 1151
 1152  002416                           LET -(SP) := #0
  (4)  002416  005046                                                                          CLR     -(SP)
 1153  002420                           LET -(SP) := #7$
  (4)  002420  012746  002426                                                                 MOV     #7$,-(SP)
 1154  002424                           INLINE <RTI>
  (2)  002424  000002                                                                          RTI
 1155  002426                           INLINE <7$:>
  (2)  002426                                                                                  7$:
 1156
 1157                                   ;+
 1158                                   ; OUTPUT MESSAGE TO IDENTIFY SELF
 1159                                   ;-
 1160
 1161  002426                           CALL MSGDHOOK IN <#DTABLE,#MSGPOP,#DX.TITLE,#2$>
  (3)  002426  010546                                                                          MOV     R5,-(SP)
  (7)  002430  012745  002462                                                                 MOV     #2$,-(R5)
  (6)  002434  012745  001252                                                                 MOV     #DX.TITLE,-(R5)
  (5)  002440  012745  000002                                                                 MOV     #MSGPOP,-(R5)
  (4)  002444  012745  001004                                                                 MOV     #DTABLE,-(R5)
  (3)  002450  004767  000000G                                                                JSR     PC,MSGDHOOK
  (3)  002454  012605                                                                          MOV     (SP)+,R5
 1162  002456                           INLINE <1$:>
  (2)  002456                                                                                  1$:
 1163  002456                           INLINE <NOP>
  (2)  002456  000240                                                                          NOP
 1164  002460                           INLINE <BR 1$>
  (2)  002460  000776                                                                          BR 1$
 1165  002462                           INLINE <2$:>
  (2)  002462                                                                                  2$:
 1166
 1167                                   ;+
```

```
1168                                    ; NOW DETERMINE SYSTEM SIZE IN THOUSANDS
1169                                    ;-
1170
1171  002462                            LET R0 := #0
 (4)  002462  005000                                                    CLR     R0
1172  002464                            LET R1 := DTABLE+DT.SSIZ
 (4)  002464  016701  176362                                            MOV     DTABLE+DT.SSIZ,R
1173  002470                            WHILE R1 GT #0 DO
 (4)  002470                                                    50005$:
 (6)  002470  005701                                                    TST     R1
 (9)  002472  003404                                                    BLE     50006$
1174  002474                                    LET R1 := R1 - #40
 (6)  002474  162701  000040                                            SUB     #40,R1
1175  002500                                    LET R0 := R0 + #1
 (6)  002500  005200                                                    INC     R0
1176  002502                            ENDDO
 (4)  002502  000772                                                    BR      50005$
 (3)  002504                                                    50006$:
1177
1178
1179                                    ;+
1180                                    ; CONVERT SIZE TO DECIMAL ASCIZ
1181                                    ;-
1182
1183  002504                            CALL BDACNV IN <R0,#DX.SZK>
 (3)  002504  010546                                                    MOV     R5,-(SP)
 (5)  002506  012745  001371                                            MOV     #DX.SZK,-(R5)
 (4)  002512  010045                                                    MOV     R0,-(R5)
 (3)  002514  004767  000000G                                          JSR     PC,BDACNV
 (3)  002520  012605                                                    MOV     (SP)+,R5
1184
1185                                    ;+
1186                                    ; OUTPUT SYSTEM SIZE MESSAGE
1187                                    ;-
1188
1189  002522                            CALL MSGDHOOK IN <#DTABLE,#MSGPOP,#DX.SIZ,#4$>
 (3)  002522  010546                                                    MOV     R5,-(SP)
 (7)  002524  012745  002556                                            MOV     #4$,-(R5)
 (6)  002530  012745  001353                                            MOV     #DX.SIZ,-(R5)
 (5)  002534  012745  000002                                            MOV     #MSGPOP,-(R5)
 (4)  002540  012745  001004                                            MOV     #DTABLE,-(R5)
 (3)  002544  004767  000000G                                          JSR     PC,MSGDHOOK
 (3)  002550  012605                                                    MOV     (SP)+,R5
1190  002552                            INLINE <3$:>
 (2)  002552                                                            3$:
1191  002552                            INLINE <NOP>
 (2)  002552  000240                                                    NOP
1192  002554                            INLINE <BR 3$>
 (2)  002554  000776                                                    BR 3$
1193  002556                            INLINE <4$:>
 (2)  002556                                                            4$:
1194
1195                                    ;+
1196                                    ; IF KT PRESENT GO SET UP APRS
1197                                    ;-
1198
```

```
1199  002556                            IF #KTPRES SETIN DTABLE+DT.CF0 THEN
 (6)  002556  032767  000400  176234                                                  BIT     #KTPPES,DTABLE+D
 (9)  002564  001406                                                                 BEQ     50007$
1200  002566                                    CALL KTSET IN <#DTABLE>
 (3)  002566  010546                                                                 MOV     R5,-(SP)
 (4)  002570  012745  001004                                                         MOV     #DTABLE,-(R5)
 (3)  002574  004767  000000G                                                       JSR     PC,KTSET
 (3)  002600  012605                                                                 MOV     (SP)+,R5
1201  002602                            ENDIF
 (4)  002602                                                                 50007$:
1202
1203                                    ;+
1204                                    ; GO TURN ON PROCESSOR OPTIONS ACCORDING TO CONF. WD. 0
1205                                    ; AND UPDATE STATUS INDICATOR WORD.
1206                                    ;-
1207
1208  002602                            CALL DX.TURNON
 (3)  002602  004767  002064                                                        JSR     PC,DX.TURNON
1209
1210                                    ;+
1211                                    ; IF EXERCISER LOADED BY XXDP MEDIUM, OUTPUT MESSAGE.
1212                                    ;-
1213
1214  002606                            IF @#40 NE #0 THEN
 (6)  002606  005737  000040                                                        TST     @#40
 (9)  002612  001416                                                                BEQ     50010$
1215  002614                                    CALL MSGDHOOK IN <#DTABLE,#MSGPOP,#DX.MED,#6$>
 (3)  002614  010546                                                                MOV     R5,-(SP)
 (7)  002616  012745  002650                                                        MOV     #6$,-(R5)
 (6)  002622  012745  001402                                                        MOV     #DX.MED,-(R5)
 (5)  002626  012745  000002                                                        MOV     #MSGPOP,-(R5)
 (4)  002632  012745  001004                                                        MOV     #DTABLE,-(R5)
 (3)  002636  004767  000000G                                                       JSR     PC,MSGDHOOK
 (3)  002642  012605                                                                MOV     (SP)+,R5
1216  002644                            INLINE <5$:>
 (2)  002644                                                                        5$:
1217  002644                                    INLINE <NOP>
 (2)  002644  000240                                                                NOP
1218  002646                                    INLINE <BR 5$>
 (2)  002646  000776                                                                BR 5$
1219  002650                            ENDIF
 (4)  002650                                                                 50010$:
1220  002650                            INLINE <6$:>
 (2)  002650                                                                        6$:
```

```
                                .SBTTL  KERNEL - RESTART ROUTINE


                                ;+
                                ; THIS ROUTINE PERFORMS THOSE FUNCTIONS NEEDED TO GET THE MONITOR GOING AGAIN
                                ; AFTER IT HAS BEEN RUNNING
                                ;-

002650                          DX.RSTRT: LET SP := #SPVALUE
(4) 002650  012706  002200                                                      MOV     #SPVALUE,SP
002654                          LET DX.SP := SP
(4) 002654  010667  176370                                                      MOV     SP,DX.SP
002660                          LET R5 := #R5VALUE
(4) 002660  012705  001740                                                      MOV     #R5VALUE,R5
002664                          LET DX.R5 := R5
(4) 002664  010567  176356                                                      MOV     R5,DX.R5

                                ;+
                                ; CLEAR THE ERROR WORD.
                                ;-

002670                          LET DTABLE+DT.ERR := #0
(4) 002670  005067  176130                                                      CLR     DTABLE+DT.ERR
                                ;+
                                ; RESET THE BACKGROUND POINTER
                                ;-

002674                          LET BA.MPTR := DTABLE+DT.BLST
(4) 002674  016767  176140  000000G                                             MOV     DTABLE+DT.BLST,B


                                ;+
                                ; GO DO THE NECESSARY MONITOR INITIALIZATION, AND HURRY BACK
                                ;-

002702                          CALL DX.INI
(3) 002702  004767  000112                                                      JSR     PC,DX.INI


                                ;+
                                ; IF THIS IS INITIAL START-UP AND WE ARE IN AUTOMATIC MODE (XXDP OR ACT-11),
                                ; THEN MOVE THEXXDP MONITOR (EVEN IF IT DOESN'T EXIST, SINCE WE DON"T KNOW
                                ; WHETHER WE'RE UNDER XXDP OR ACT-11) TO JUST ABOVE THE
                                ; EXERCISER.  THEN FAKE A "RUN" COMMAND TO GET STARTED.
                                ;-

002706                          IF #AUTOST NOTSETIN DTABLE+DT.ST1 THEN
(6) 002706  032767  020000  176102                                              BIT     #AUTOST,DTABLE+D
(9) 002714  001031                                                              BNE     50011$
002716                                  IF #AUTO SETIN DTABLE+DT.CF0 THEN
(6) 002716  032767  000010  176074                                              BIT     #AUTO,DTABLE+DT.
(9) 002724  001406                                                              BEQ     50012$
002726                                          CALL DPSTRT IN <#DTABLE>
(3) 002726  010546                                                              MOV     R5,-(SP)
(4) 002730  012745  001004                                                      MOV     #DTABLE,-(R5)
```

```
  (3)  002734  004767  000000G                                                                  JSR     PC,DPSTRT
  (3)  002740  012605                                                                           MOV     (SP)+,R5
1265   002742                                ENDIF
  (4)  002742                                                                  50012$:
1266
1267                                 ;+
1268                                 ; IF THIS IS INITIAL START-UP WE ARE UNDER APT - GET THIS THING RUNNING RIGHT AWAY !
1269                                 ;-
1270
1271   002742                                IF #APTPRES SETIN DTABLE+DT.CF0 THEN
  (6)  002742  032767  000200  176050                                                           BIT     #APTPRES,DTABLE+
  (9)  002750  001410                                                                           BEQ     50013$
1272   002752                                        CALL KRUN IN <#DTABLE,#0>
  (3)  002752  010546                                                                           MOV     R5,-(SP)
  (5)  002754  012745  000000                                                                   MOV     #0,-(R5)
  (4)  002760  012745  001004                                                                   MOV     #DTABLE,-(R5)
  (3)  002764  004767  000000G                                                                  JSR     PC,KRUN
  (3)  002770  012605                                                                           MOV     (SP)+,R5
1273   002772                                ENDIF
  (4)  002772                                                                  50013$:
1274   002772                                LET DTABLE+DT.ST1 := DTABLE+DT.ST1 SET.BY #AUTOST
  (6)  002772  052767  020000  176016                                                           BIS     #AUTOST,DTABLE+D
1275   003000                         ENDIF
  (4)  003000                                                                  50011$:
1276
1277
1278                                 ;+
1279                                 ; OUTPUT THE KEYBOARD PROMPT AND LET THE OPERATOR KNOW THAT THE
1280                                 ; SYSTEM IS NOW AWAITING KEYBOARD INPUT
1281                                 ;-
1282
1283   003000                        CALL CMDRST IN <#DTABLE>
  (3)  003000  010546                                                                           MOV     R5,-(SP)
  (4)  003002  012745  001004                                                                   MOV     #DTABLE,-(R5)
  (3)  003006  004767  000000G                                                                  JSR     PC,CMDRST
  (3)  003012  012605                                                                           MOV     (SP)+,R5
1284   003014                        INLINE <JMP     @#DX.DEQ>
  (2)  003014  000137  003634                                                                   JMP     @#DX.DEQ
```

```
                                .SBTTL  KERNEL - INITIALIZATION ROUTINE


                                ;+
                                ; THIS ROUTINE PERFORMS THE INITIALIZATION FUNCTIONS THAT ARE REQUIRED
                                ; TO GET THE DEC/X-11 MONITOR GOING. ONCE THE INITIALIZATION IS COMPLETE
                                ; AND THE MONITOR IS RUNNING, CONTROL IS RETURNED TO THE CALLER
                                ;-

003020                          ROUTINE DX.INI
003020                                                                          DX.INI:

                                CALL SAVREG
003020  004767  000000G                                                                 JSR     PC,SAVREG

                                ;+
                                ; INITIALIZE THE QUEUE STRUCTURES - BOTH THE CONTROL QUEUE AND THE
                                ; TYPE QUEUE
                                ;-

003024                          CALL CQINI
003024  004767  000000G                                                                 JSR     PC,CQINI
003030                          CALL TQINI
003030  004767  000000G                                                                 JSR     PC,TQINI

                                ;+
                                ; INITIALIZE THE ERROR LOGGING STUFF
                                ;-

003034                          IF #PDP70!PDP60 SETIN DTABLE+DT.CF0 THEN
003034  032767  014000  175756                                                         BIT     #PDP70!PDP60,DTA
003042  001402                                                                       BEQ     50002$
003044                                  CALL IERRLO
003044  004767  000000G                                                                 JSR     PC,IERRLO
003050                          ENDIF
003050                                                                          50002$:

                                ;+
                                ; INITIALIZE THE OPTION MODULE SERVICING MECHANISM, WHICH CONSISTS OF RESETTING
                                ; THE POINTERS FOR THE MODULE-TYPE-LIST AND THE MODULE-QUEUE-LIST, CLEARING THE
                                ; RELOCATION-MODE FLAGS, RESETTING THE ASSOCIATED BITS IN THE STATUS INDICATOR WORD
                                ; ALSO, CLEAR THE SBKSEL BIT.
                                ;-

003050                          LET AC.TPTR := #AC.TYPE
003050  012767  000000G 000000G                                                         MOV     #AC.TYPE,AC.TPTR
003056                          LET AC.MPTR := #AC.MODQ
003056  012767  000000G 000000G                                                         MOV     #AC.MODQ,AC.MPTR
003064                          LET DTABLE+DT.ST1 := DTABLE+DT.ST1 SET.BY #CKTIM
003064  052767  100000  175724                                                         BIS     #CKTIM,DTABLE+DT
003072                          LET DTABLE+DT.ST0 := DTABLE+DT.ST0 CLR.BY #MODEXH!MODHOLD!BKMODE!TMPIO
003072  042767  046002  175714                                                         BIC     #MODEXH!MODHOLD!
003100                          LET DTABLE+DT.ST1 := DTABLE+DT.ST1 CLR.BY #SBKSEL
003100  042767  010000  175710                                                         BIC     #SBKSEL,DTABLE+D

                                ;+
```

```
 1329                                   ; RESET THE BACKGROUND STATUS WORD
 1330                                   ;-
 1331
 1332  003106                           LET BA.STAT := #0
  (4)  003106  005067  000000G                                                          CLR     BA.STAT
 1333
 1334                                   ;+
 1335                                   ; NOW RESET THE OPTION MODULES, WHICH CONSISTS OF RESETTING THEIR ACTBIT AND EOPBIT BITS
 1336                                   ;-
 1337
 1338  003112                           LET R0 := DTABLE+DT.MLST
  (4)  003112  016700  175720                                                           MOV     DTABLE+DT.MLST,R
 1339  003116                           WHILE (R0) NE #0 DO
  (4)  003116                                                                  50003$:
  (6)  003116  005710                                                                   TST     (R0)
  (9)  003120  001412                                                                   BEQ     50004$
 1340  003122                                   LET R1 := (R0)
  (4)  003122  011001                                                                   MOV     (R0),R1
 1341  003124                                   LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
  (6)  003124  042761  004000  000026                                                  BIC     #ACTBIT,STAT(R1)
 1342  003132                                   LET XFLAG(R1) :B= XFLAG(R1) CLR.BY #EOPBIT
  (6)  003132  142761  000001  000005                                                  BICB    #EOPBIT,XFLAG(R1
 1343  003140                                   LET R0 := R0 + #2
  (6)  003140  062700  000002                                                           ADD     #2,R0
 1344  003144                           ENDDO
  (4)  003144  000764                                                                   BR      50003$
  (3)  003146                                                                  50004$:
 1345
 1346                                   ;+
 1347                                   ; GET BACKGROUND MODULE LIST POINTER AND MAKE SURE THAT THEY ARE
 1348                                   ; ALL BKMODS. DUE TO  THE TMPIO CONDITION.
 1349                                   ;-
 1350
 1351  003146                           LET R0 := DTABLE+DT.BLST
  (4)  003146  016700  175666                                                           MOV     DTABLE+DT.BLST,R
 1352  003152                           WHILE (R0) NE #ENDLST DO
  (4)  003152                                                                  50005$:
  (6)  003152  021027  000000                                                           CMP     (R0),#ENDLST
  (9)  003156  001412                                                                   BEQ     50006$
 1353  003160                                   LET R1 := (R0)
  (4)  003160  011001                                                                   MOV     (R0),R1
 1354  003162                                   LET STAT(R1) := STAT(R1) CLR.BY #IOMOD
  (6)  003162  042761  100000  000026                                                  BIC     #IOMOD,STAT(R1)
 1355  003170                                   LET STAT(R1) := STAT(R1) SET.BY #BKMOD
  (6)  003170  052761  000020  000026                                                  BIS     #BKMOD,STAT(R1)
 1356  003176                                   LET R0 := R0 + #2
  (6)  003176  062700  000002                                                           ADD     #2,R0
 1357  003202                           ENDDO
  (4)  003202  000763                                                                   BR      50005$
  (3)  003204                                                                  50006$:
 1358
 1359                                   ;+
 1360                                   ; INITIALIZE THE MESSAGE DE-QUEUING MECHANISM BY RESETTING IT'S BUSY
 1361                                   ; INDICATOR
 1362                                   ;-
 1363
```

```
 1364  003204                            LET MD.BSY := #0
  (4)  003204  005067  000000G                                                          CLR     MD.BSY
 1365
 1366                                    ;+
 1367                                    ; IF IT IS NOT YET TIME TO RELOCATE,
 1368                                    ; GET THE KEYBOARD GOING, WHICH CONSISTS OF CALLING THE KEYBOARD DRIVER
 1369                                    ; AT ITS I/O INITIALIZATION ENTRY POINT
 1370                                    ;-
 1371
 1372  003210                            IF #RELTIME NOTSETIN DTABLE+DT.STO THEN
  (6)  003210  032767  010000  175576                                                  BIT     #RELTIME,DTABLE+
  (9)  003216  001006                                                                  BNE     50007$
 1373  003220                                    CALL KBINI IN <#DTABLE>
  (3)  003220  010546                                                                  MOV     R5,-(SP)
  (4)  003222  012745  001004                                                          MOV     #DTABLE,-(R5)
  (3)  003226  004767  000000G                                                         JSR     PC,KBINI
  (3)  003232  012605                                                                  MOV     (SP)+,R5
 1374  003234                            ENDIF
  (4)  003234                                                                 50007$:
 1375
 1376  003234                            CALL RESREG
  (3)  003234  004767  000000G                                                         JSR     PC,RESREG
 1377
 1378
 1379  003240                            ENDRTN
  (3)  003240                                                                 50000$:
  (3)  003240                                                                 50001$:
  (2)  003240  000207                                                                  RTS     PC
```

```
                                        .SBTTL  KERNEL - CAPTURE EVENT: DISPATCH TO SERVICE ROUTINE


                                        ;+
                                        ; THIS ROUTINE FIELDS THE PHYSICAL EVENTS THAT ARE RECEIVED BY THE DECX/11
                                        ; MONITOR, I.E., SOFTWARE GENERATED TRAPS AND HARDWARE ERROR TRAPS.
                                        ; THE TYPE OF EVENT IS DETERMINED AND THE CORRESPONDING EVENT
                                        ; CODE IS ENTERED INTO THE MONITOR'S DATA TABLE. THE LINKAGE INFORMATION
                                        ; ON THE STACK IS THEN REMOVED AND ALSO ENTERED INTO THE DATA TABLE. THEN,
                                        ; BASED ON THE TYPE OF EVENT (DIRECT-RETURN OR DEFERRED-RETURN), THE MODULE'S
                                        ; REGISTERS  MAY HAVE TO BE SAVED IN THE MODULE'S HEADER. CONTROL IS THEN
                                        ; PASSED TO THE APPROPRIATE SERVICE ROUTINE.
                                        ;-

003242                                  DX.CAP:

                                        ;+
                                        ; IF THIS EVENT ORIGINATED WITH A BACKGROUND MODULE, RESET THE MONITOR'S BACKGROUND MODE
                                        ; INDICATOR, THUS ALLOWING THE EVENT TO BE SERVICED IN IT'S ENTIRETY
                                        ;-

003242                                  IF #BKMODE SETIN DTABLE+DT.STO THEN
(6) 003242  032767  040000  175544                                                      BIT     #BKMODE,DTABLE+D
(9) 003250  001403                                                                      BEQ     50010$
003252                                          LET DTABLE+DT.STO := DTABLE+DT.STO CLR.BY #BKMODE
(6) 003252  042767  040000  175534                                                      BIC     #BKMODE,DTABLE+D
003260                                  ENDIF
(4) 003260                                                                      50010$:


                                        ;+
                                        ; SAVE THE INFORMATION FOR THE NEW EVENT
                                        ;-

003260                                  POP DTABLE+DT.EVNT
(2) 003260  012667  175520                                                              MOV     (SP)+,DTABLE+DT.
003264                                  POP DTABLE+DT.PC
(2) 003264  012667  175516                                                              MOV     (SP)+,DTABLE+DT.
003270                                  POP DTABLE+DT.PSW
(2) 003270  012667  175514                                                              MOV     (SP)+,DTABLE+DT.
003274                                  LET DTABLE+DT.SP := SP
(4) 003274  010667  175512                                                              MOV     SP,DTABLE+DT.SP


                                        ;+
                                        ; DETERMINE IF THE EVENT ORIGINATED FROM THE MONITOR, IF SO DON'T BOTHER
                                        ; SAVING THE REGISTERS AND SWITCHING STACKS
                                        ;-

003300                                  IF DTABLE+DT.EVNT GE #EVNTHD THEN
(6) 003300  026727  175500  000200                                                      CMP     DTABLE+DT.EVNT,#
(9) 003306  002405                                                                      BLT     50011$
003310                                          IF DTABLE+DT.SP LOS #SPVALUE THEN
(6) 003310  026727  175476  002200                                                      CMP     DTABLE+DT.SP,#SP
(9) 003316  101001                                                                      BHI     50012$
003320                                                  INLINE <BR 1$>
(2) 003320  000423                                                                      BR 1$
```

```
 1424  003322                                ENDIF
  (4)  003322                                                                                  50012$:
 1425  003322                           ENDIF
  (4)  003322                                                                                  50011$:
 1426
 1427                                   ;+
 1428                                   ; THE EVENT WAS EITHER DECLARED EXPLICITLY BY AN OPTION MODULE, OR CAUSED
 1429                                   ; UNINTENTIONALLY BY ONE, SO SAVE THE REGISTERS IN ITS HEADER AREA AND SWITCH
 1430                                   ; TO THE MONITOR'S R6 AND R5 STACKS
 1431                                   ;-
 1432
 1433  003322                           PUSH R0
  (2)  003322  010046                                                                          MOV     R0,-(SP)
 1434  003324                           LET R0 := DX.HDR
  (4)  003324  016700  175710                                                                  MOV     DX.HDR,R0
 1435  003330                           LET STAT(R0) :B= STAT(R0) SET.BY DTABLE+DT.PSW
  (6)  003330  156760  175454  000026                                                         BISB    DTABLE+DT.PSW,ST
 1436  003336                           LET R0 := R0 + #SVR0
  (6)  003336  062700  000062                                                                 ADD     #SVR0,R0
 1437  003342                           POP (R0)+
  (2)  003342  012620                                                                          MOV     (SP)+,(R0)+
 1438  003344                           LET (R0)+ := R1
  (4)  003344  010120                                                                          MOV     R1,(R0)+
 1439  003346                           LET (R0)+ := R2
  (4)  003346  010220                                                                          MOV     R2,(R0)+
 1440  003350                           LET (R0)+ := R3
  (4)  003350  010320                                                                          MOV     R3,(R0)+
 1441  003352                           LET (R0)+ := R4
  (4)  003352  010420                                                                          MOV     R4,(R0)+
 1442  003354                           LET (R0)+ := R5
  (4)  003354  010520                                                                          MOV     R5,(R0)+
 1443  003356                           LET (R0)+ := SP
  (4)  003356  010620                                                                          MOV     SP,(R0)+
 1444  003360                           LET R5 := DX.R5
  (4)  003360  016705  175662                                                                  MOV     DX.R5,R5
 1445  003364                           LET SP := DX.SP
  (4)  003364  016706  175660                                                                  MOV     DX.SP,SP
 1446
 1447  003370                           INLINE  <1$:>
  (2)  003370                                                                                  1$:
 1448
 1449
 1450                                   ;+
 1451                                   ; NOW DETERMINE THE EVENT TYPE AND MARK THE RETURN-TYPE BIT ACCORDINGLY
 1452                                   ; (DIRECT OR DEFERRED RETURN.)
 1453                                   ;-
 1454
 1455  003370                           IF DTABLE+DT.EVNT LT #DSEVNT THEN
  (6)  003370  026727  175410  000014                                                         CMP     DTABLE+DT.EVNT,#
  (9)  003376  002004                                                                          BGE     50013$
 1456  003400                                  LET DTABLE+DT.STO := DTABLE+DT.STO SET.BY #DEFRTN
  (6)  003400  052767  000400  175406                                                         BIS     #DEFRTN,DTABLE+D
 1457  003406                           ELSE
  (4)  003406  000403                                                                          BR      50014$
  (3)  003410                                                                                  50013$:
 1458  003410                                  LET DTABLE+DT.STO := DTABLE+DT.STO CLR.BY #DEFRTN
```

```
   (6)  003410  042767  000400  175376                                                    BIC     #DEFRTN,DTABLE+D
 1459   003416                          ENDIF
   (4)  003416                                                                50014$:
 1460
 1461                                  ;+
 1462                                  ; NOW GO SERVICE THE EVENT
 1463                                  ;-
 1464
 1465   003416                          CALL PREVT IN <#DTABLE>
   (3)  003416  010546                                                                MOV     R5,-(SP)
   (4)  003420  012745  001004                                                        MOV     #DTABLE,-(R5)
   (3)  003424  004767  000000G                                                       JSR     PC,PREVT
   (3)  003430  012605                                                                MOV     (SP)+,R5
```

```
                                .SBTTL  KERNEL - CAPTURE EVENT: RETURN FROM SERVICING EVENT



                                ;+
                                ; WAS THE EVENT AN ERROR, OR DID THE SERVICING OF THE EVENT RESULT
                                ; IN AN ERROR? IF SO PASS CONTROL TO THE ERROR RECOVERY HANDLER AND
                                ; DON'T EXPECT TO GET CONTROL BACK
                                ;-

003432                          IF DTABLE+DT.ERR NE #0 THEN
(6) 003432  005767  175366                                                      TST     DTABLE+DT.ERR
(9) 003436  001406                                                              BEQ     50015$
003440                              CALL ERREC IN <#DTABLE>
(3) 003440  010546                                                              MOV     R5,-(SP)
(4) 003442  012745  001004                                                      MOV     #DTABLE,-(R5)
(3) 003446  004767  000000G                                                     JSR     PC,ERREC
(3) 003452  012605                                                              MOV     (SP)+,R5
003454                          ENDIF
(4) 003454                                                                  50015$:

                                ;+
                                ; IF THE EVENT ORIGINATED WITH A BACKGROUND MODULE, DETERMINE IF DEFERRED
                                ; RETURN.  IF SO, LOAD THE HEADER ADDRESS AND RETURN ADDRESS INTO THE BACKGROUND
                                ; QUEUE, AND SET THE "BK" SUSPEND BIT.
                                ; OTHERWISE, THE MODULE IS WORKING ITS WAY THROUGH THE MONITOR
                                ; QUEUES, SO SET THE BKDEF BIT IN THE BK STATUS WORD.
                                ; IN EITHER CASE, SET THE DEFERRED RETURN BIT IN THE STATUS INDICATOR.
                                ;-

003454                                  LET R0 := DX.HDR
(4) 003454  016700  175560                                                      MOV     DX.HDR,R0
003460                                  IF #BIT04 SETIN STAT(R0) THEN
(6) 003460  032760  000020  000026                                              BIT     #BIT04,STAT(R0)
(9) 003466  001426                                                              BEQ     50016$
003470                                      LET STAT(R0) := STAT(R0) CLR.BY #ACTBIT
(6) 003470  042760  004000  000026                                              BIC     #ACTBIT,STAT(R0)
003476                                      IF #DEFRTN NOTSETIN DTABLE+DT.ST0 THEN
(6) 003476  032767  000400  175310                                              BIT     #DEFRTN,DTABLE+D
(9) 003504  001011                                                              BNE     50017$
003506                                          LET BA.HDR := R0
(4) 003506  010067  000000G                                                     MOV     R0,BA.HDR
003512                                          LET BA.RET := DTABLE+DT.PC
(4) 003512  016767  175270  000000G                                              MOV     DTABLE+DT.PC,BA.
003520                                          LET BA.STATUS := BA.STATUS SET.BY #SUSPND
(6) 003520  052767  000001  000000G                                              BIS     #SUSPND,BA.STATU
003526                                      ELSE
(4) 003526  000403                                                              BR      50020$
(3) 003530                                                                  50017$:
003530                                          LET BA.STAT := BA.STAT SET.BY #BKDEF
(6) 003530  052767  000002  000000G                                              BIS     #BKDEF,BA.STAT
003536                                      ENDIF
(4) 003536                                                                  50020$:
003536                                      LET DTABLE+DT.ST0 := DTABLE+DT.ST0 SET.BY #DEFRTN
(6) 003536  052767  000400  175250                                              BIS     #DEFRTN,DTABLE+D
003544                                  ENDIF
```

```
 (4)  003544                                                                                    50016$:
1502
1503                                    ;+
1504                                    ; WAS THE EVENT JUST SERVICED OF THE 'DEFERRED-RETURN' TYPE WHERE
1505                                    ; CONTROL IS NOT RETURNED IMMEDIATELY TO THE MODULE/PROCESS THAT ISSUED
1506                                    ; THE EVENT. IF IT IS, GO GET THE NEXT MODULE FROM THE CONTROL QUEUE
1507                                    ;-
1508
1509  003544                                    IF #DEFRTN SETIN DTABLE+DT.ST0 THEN
 (6)  003544  032767  000400  175242                                                           BIT     #DEFRTN,DTABLE+D
 (9)  003552  001404                                                                           BEQ     50021$
1510  003554                                          LET DTABLE+DT.ST0 := DTABLE+DT.ST0 CLR.BY #DEFRTN
 (6)  003554  042767  000400  175232                                                           BIC     #DEFRTN,DTABLE+D
1511  003562                                          INLINE <BR DX.DEQ>
 (2)  003562  000424                                                                           BR DX.DEQ
1512  003564                                    ENDIF
 (4)  003564                                                                                    50021$:
1513
1514                                    ;+
1515                                    ; THE EVENT ORIGINATED WITH AN OPTION MODULE, SO RESTORE IT'S REGISTERS AND SWITCH STACK
1516                                    ;-
1517
1518  003564                                LET DX.SP := SP
 (4)  003564  010667  175460                                                                   MOV     SP,DX.SP
1519  003570                                LET DX.R5 := R5
 (4)  003570  010567  175452                                                                   MOV     R5,DX.R5
1520  003574                                LET R0 := DX.HDR + #SVR6
 (4)  003574  016700  175440                                                                   MOV     DX.HDR,R0
 (6)  003600  062700  000076                                                                   ADD     #SVR6,R0
1521  003604                                LET SP := (R0)
 (4)  003604  011006                                                                           MOV     (R0),SP
1522  003606                                LET R5 := -(R0)
 (4)  003606  014005                                                                           MOV     -(R0),R5
1523  003610                                LET R4 := -(R0)
 (4)  003610  014004                                                                           MOV     -(R0),R4
1524  003612                                LET R3 := -(R0)
 (4)  003612  014003                                                                           MOV     -(R0),R3
1525  003614                                LET R2 := -(R0)
 (4)  003614  014002                                                                           MOV     -(R0),R2
1526  003616                                LET R1 := -(R0)
 (4)  003616  014001                                                                           MOV     -(R0),R1
1527  003620                                LET R0 := -(R0)
 (4)  003620  014000                                                                           MOV     -(R0),R0
1528
1529                                    ;+
1530                                    ; NOW RETURN CONTROL
1531                                    ;-
1532
1533  003622                            PUSH DTABLE+DT.PSW
 (2)  003622  016746  175162                                                                   MOV     DTABLE+DT.PSW,-(
1534  003626                            PUSH DTABLE+DT.PC
 (2)  003626  016746  175154                                                                   MOV     DTABLE+DT.PC,-(S
1535  003632                            INLINE <2$:     RTI>
 (2)  003632  000002                                                                           2$:     RTI
```

```
                                .SBTTL  KERNEL - DE-QUEUE NEXT MODULE IN CONTROL QUEUE ROUTINE


                                ;+ SCHEDULER
                                ; THIS ROUTINE IS ENTERED ONLY WHEN NO OTHER MONITOR MODULES OR OPTION
                                ; MODULES ARE EXECUTING. ITS MAIN FUNCTIONS ARE TO DE-QUEUE THE NEXT
                                ; MESSAGE TO BE PRINTED, ACTIVATE THE NEXT MODULE IN THE MODULE-QUEUE-
                                ; LIST, AND TO DE-QUEUE THE NEXT MODULE IN THE CONTROL QUEUE AND ALLOW
                                ; IT TO BEGIN EXECUTION. IF THERE ARE NO MODULES WAITING IN THE MODULE-
                                ; QUEUE-LIST OR IN THE CONTROL QUEUE, THIS ROUTINE SIMPLY KEEPS CONTROL
                                ; BY SITTING IN A CLOSED LOOP.
                                ;-

003634                          DX.DEQ: LET DX.QFLG := #0
 (4) 003634  005067  175404                                                     CLR     DX.QFLG
                                ;       REPEAT
003640                          INLINE <3$:>
 (2) 003640                                                                     3$:
                                ;+
                                ; INCREMENT THE APT "KEEP-ALIVE" COUNTER.
                                ;-

003640                          LET @DTABLE+DT.APK := @DTABLE+DT.APK + #1
 (6) 003640  005277  175236                                                     INC     @DTABLE+DT.APK

                                ;+
                                ; CHECK THE MESSAGE QUEUE FIRST
                                ;-

003644                                  CALL MSGDEQ IN <#DTABLE>
 (3) 003644  010546                                                             MOV     R5,-(SP)
 (4) 003646  012745  001004                                                     MOV     #DTABLE,-(R5)
 (3) 003652  004767  000000G                                                    JSR     PC,MSGDEQ
 (3) 003656  012605                                                             MOV     (SP)+,R5

                                ;+
                                ; IF THE KB FLAG IS SET - WE HAD A CR - SO GO SERVICE
                                ;-

003660                                  IF DX.KFL EQ #1 THEN
 (6) 003660  026727  175352  000001                                           CMP     DX.KFL,#1
 (9) 003666  001010                                                             BNE     50022$
003670                                          CALL CMDPRC IN <#DTABLE>
 (3) 003670  010546                                                             MOV     R5,-(SP)
 (4) 003672  012745  001004                                                     MOV     #DTABLE,-(R5)
 (3) 003676  004767  000000G                                                    JSR     PC,CMDPRC
 (3) 003702  012605                                                             MOV     (SP)+,R5
003704                                          LET DX.KFL := #0
 (4) 003704  005067  175326                                                     CLR     DX.KFL
003710                                  ENDIF
 (4) 003710                                                             50022$:

                                ;+
                                ; IF WE ARE EXECUTING IN RUN MODE, GO THRU THE SCHEDULING ALGORITHM
                                ;-
```

```
1578
1579  003710                                    IF #RUNMODE NOTSETIN DTABLE+DT.ST0 THEN
 (6)  003710  032767  100000  175076                                                    BIT     #RUNMODE,DTABLE+
 (9)  003716  001002                                                                    BNE     50023$
1580  003720                                            INLINE <JMP 6$>
 (2)  003720  000167  000614                                                          JMP 6$
1581  003724                                    ENDIF
 (4)  003724                                                                  50023$:
1582
1583                                       ;+
1584                                       ; START BY LOOKING AT THE RELOCATION STUFF. IF IT'S TIME TO RELOCATE, GO SEE
1585                                       ; IF ALL MODULE ACTIVE BITS ARE CLEAR.  IF SO, RELOCATE.
1586                                       ;-
1587  003724                                    IF #RELTIME SETIN DTABLE+DT.ST0 THEN
 (6)  003724  032767  010000  175062                                                    BIT     #RELTIME,DTABLE+
 (9)  003732  001415                                                                    BEQ     50024$
1588  003734                                            CALL CHKACT IN <#DTABLE>
 (3)  003734  010546                                                                    MOV     R5,-(SP)
 (4)  003736  012745  001004                                                            MOV     #DTABLE,-(R5)
 (3)  003742  004767  000000G                                                          JSR     PC,CHKACT
 (3)  003746  012605                                                                    MOV     (SP)+,R5
1589  003750                                            IF.NO.ERROR THEN
 (6)  003750  103406                                                                    BCS     50025$
1590  003752                                                    CALL RELCTL IN <#DTABLE>
 (3)  003752  010546                                                                    MOV     R5,-(SP)
 (4)  003754  012745  001004                                                            MOV     #DTABLE,-(R5)
 (3)  003760  004767  000000G                                                          JSR     PC,RELCTL
 (3)  003764  012605                                                                    MOV     (SP)+,R5
1591  003766                                            ENDIF
 (4)  003766                                                                  50025$:
1592  003766                                    ENDIF
 (4)  003766                                                                  50024$:
1593
1594                                       ;+
1595                                       ; IF THE LAST SYSTEM EOP WAS IN LOWEST MEMORY (MEMPAS=1), IT IS
1596                                       ; TIME TO CHECK FOR A COUPLE OF THINGS.
1597                                       ;       (1) IF WE ARE RUNNING UNDER XXDP OR ACT-11 (AUTO=1), THEN IT IS TIME
1598                                       ; TO RETURN THE THE XXDP/ACT-11 MONITOR.
1599                                       ;       (2) IF THERE ARE SBK MODULES THAT ARE SELECTED, IT IS TIME TO
1600                                       ; STOP ALL MODULES AND THEN RESTART THEM SO THE
1601                                       ; SBK MODULES WILL RESTART.
1602                                       ;-
1603
1604  003766                                    IF #MEMPAS SETIN DTABLE+DT.ST1 THEN
 (6)  003766  032767  040000  175022                                                    BIT     #MEMPAS,DTABLE+D
 (9)  003774  001434                                                                    BEQ     50026$
1605  003776                                            CALL CHKACT IN <#DTABLE>
 (3)  003776  010546                                                                    MOV     R5,-(SP)
 (4)  004000  012745  001004                                                            MOV     #DTABLE,-(R5)
 (3)  004004  004767  000000G                                                          JSR     PC,CHKACT
 (3)  004010  012605                                                                    MOV     (SP)+,R5
1606  004012                                            IF.NO.ERROR THEN
 (6)  004012  103425                                                                    BCS     50027$
1607  004014                                                    IF #AUTO SETIN DTABLE+DT.CF0 THEN
 (6)  004014  032767  000010  174776                                                    BIT     #AUTO,DTABLE+DT.
 (9)  004022  001406                                                                    BEQ     50030$
```

```
1608  004024                                                  CALL DPEOP IN <#DTABLE>
 (3)  004024  010546                                                              MOV     R5,-(SP)
 (4)  004026  012745  001004                                                      MOV     #DTABLE,-(R5)
 (3)  004032  004767  000000G                                                     JSR     PC,DPEOP
 (3)  004036  012605                                                              MOV     (SP)+,R5
1609  004040                                              ENDIF
 (4)  004040                                                                  50030$:
1610  004040                                              IF #SBKSEL SETIN DTABLE+DT.ST1 THEN
 (6)  004040  032767  010000  174750                                             BIT     #SBKSEL,DTABLE+D
 (9)  004046  001407                                                              BEQ     50031$
1611  004050                                                      CALL DX.INI
 (3)  004050  004767  176744                                                     JSR     PC,DX.INI
1612  004054                                                      LET DTABLE+DT.ST1 := DTABLE+DT.ST1 CLR.BY #MEMPAS
 (6)  004054  042767  040000  174734                                             BIC     #MEMPAS,DTABLE+D
1613  004062                                                      INLINE <JMP DX.DEQ>
 (2)  004062  000167  177546                                                     JMP DX.DEQ
1614  004066                                              ENDIF
 (4)  004066                                                                  50031$:
1615  004066                                          ENDIF
 (4)  004066                                                                  50027$:
1616  004066                                      ENDIF
 (4)  004066                                                                  50026$:
1617
1618
1619                                        ;+
1620                                        ; IF ITS NOT RELOCATION TIME AND NOT TIME FOR XXDP RETURN, THEN
1621                                        ; IF WE'RE NOT IN A 'HOLD' STATE WAITING ON A SBK OR NBK MODULE, SEE IF WE CAN
1622                                        ; GET SOMETHING GOING
1623                                        ;-
1624
1625  004066                                      IF #RELTIME NOTSETIN DTABLE+DT.ST0 AND #MEMPAS NOTSETIN DTABLE+DT.ST1 THEN
 (6)  004066  032767  010000  174720                                             BIT     #RELTIME,DTABLE+
 (9)  004074  001022                                                              BNE     50032$
 (6)  004076  032767  040000  174712                                             BIT     #MEMPAS,DTABLE+D
 (9)  004104  001016                                                              BNE     50032$
1626  004106                                          IF #MODEXH!MODHOLD NOTSETIN DTABLE+DT.ST0 OR #MODEXH!MODSEL SETIN DTABLE
 (6)  004106  032767  006000  174700                                             BIT     #MODEXH!MODHOLD,
 (8)  004114  001404                                                              BEQ     50033$
 (6)  004116  032767  005000  174670                                             BIT     #MODEXH!MODSEL,D
 (9)  004124  001406                                                              BEQ     50034$
 (6)  004126                                                                  50033$:
1627  004126                                                  CALL ACTIV IN <#DTABLE>
 (3)  004126  010546                                                              MOV     R5,-(SP)
 (4)  004130  012745  001004                                                      MOV     #DTABLE,-(R5)
 (3)  004134  004767  000000G                                                     JSR     PC,ACTIV
 (3)  004140  012605                                                              MOV     (SP)+,R5
1628  004142                                          ENDIF
 (4)  004142                                                                  50034$:
1629  004142                                      ENDIF
 (4)  004142                                                                  50032$:
1630
1631                                        ;+
1632                                        ; IS THERE AN OPTION MODULE IN THE CONTROL QUEUE AWAITING EXECUTION?
1633                                        ;-
1634
1635  004142                                      CALL DEQCQ OUT <DX.HDR,DX.RET>
```

```
 (4)  004142  162705  000004                                                    SUB     #2*2,R5
 (3)  004146  004767  000000G                                                   JSR     PC,DEQCQ
 (4)  004152  012567  175062                                                    MOV     (R5)+,DX.HDR
 (4)  004156  012567  175060                                                    MOV     (R5)+,DX.RET
1636  004162                          IF.NO.ERROR THEN
 (6)  004162  103451                                                            BCS     50035$
1637  004164                                LET R0 := DX.HDR
 (4)  004164  016700  175050                                                    MOV     DX.HDR,R0
1638
1639
1640                                ;+
1641                                ; IF THE RETURN ADDRESS IS ZERO OR A MODULE IS DESELECTED OR DROPPED,
1642                                ; SEE IF IT'S A BKMOD AND IF SO, CLEAR THE BKDEF BIT IN BA.STAT SO
1643                                ; WE CAN GET ADDITIONAL BKMODS ON THE AIR
1644                                ;-
1645
1646  004170                                IF DX.RET EQ #0 OR #BIT14 NOTSETIN STAT(R0) OR #BIT13 SETIN STAT(R0) THE
 (6)  004170  005767  175046                                                    TST     DX.RET
 (8)  004174  001410                                                            BEQ     50036$
 (6)  004176  032760  040000  000026                                           BIT     #BIT14,STAT(R0)
 (8)  004204  001404                                                            BEQ     50036$
 (6)  004206  032760  020000  000026                                           BIT     #BIT13,STAT(R0)
 (9)  004214  001410                                                            BEQ     50037$
 (6)  004216                                                          50036$:
1647  004216                                    IF #BKMOD SETIN STAT(R0) THEN
 (6)  004216  032760  000020  000026                                           BIT     #BKMOD,STAT(R0)
 (9)  004224  001403                                                            BEQ     50040$
1648  004226                                        LET BA.STAT := BA.STAT CLR.BY #BKDEF
 (6)  004226  042767  000002  000000G                                          BIC     #BKDEF,BA.STAT
1649  004234                                    ENDIF
 (4)  004234                                                          50040$:
1650  004234                                    INLINE <BR      1$>
 (2)  004234  000470                                                            BR      1$
1651  004236                                ENDIF
 (4)  004236                                                          50037$:
1652
1653                                ;+
1654                                ; DETERMINE IF IT'S A BACKGROUND MODULE, IF SO, ENTER IT INTO
1655                                ; THE BACKGROUND QUEUE AND CONTINUE DE-QUEUING FROM THE CONTROL
1656                                ; QUEUE
1657                                ;-
1658
1659  004236                                IF #BIT04 SETIN STAT(R0) THEN
 (6)  004236  032760  000020  000026                                           BIT     #BIT04,STAT(R0)
 (9)  004244  001415                                                            BEQ     50041$
1660  004246                                    LET BA.HDR := DX.HDR
 (4)  004246  016767  174766  000000G                                          MOV     DX.HDR,BA.HDR
1661  004254                                    LET BA.RET := DX.RET
 (4)  004254  016767  174762  000000G                                          MOV     DX.RET,BA.RET
1662  004262                                    LET BA.STATUS := BA.STATUS SET.BY #SUSPND
 (6)  004262  052767  000001  000000G                                          BIS     #SUSPND,BA.STATU
1663  004270                                    LET BA.STAT := BA.STAT CLR.BY #BKDEF
 (6)  004270  042767  000002  000000G                                          BIC     #BKDEF,BA.STAT
1664  004276                                    INLINE <BR 1$>
 (2)  004276  000447                                                            BR 1$
1665  004300                                ENDIF
```

```
 (4) 004300                                                                                  50041$:
1666                                ;+
1667                                ; THE MODULE IS ELIGIBLE TO RUN, SO SET THE DE-QUEUING FLAG ACCORDINGLY
1668                                ;-
1669
1670 004300                                        LET DX.QFLG := #-1
 (4) 004300 012767 177777 174736                                                            MOV     #-1,DX.QFLG
1671 004306                                    ENDIF
 (4) 004306                                                                                 50035$:
1672
1673                                ;+
1674                                ; IF THE CONTROL QUEUE IS EMPTY GO SEE IF THERE IS A MODULE IN THE BACKGROUND QUEUE
1675                                ; THAT WANTS TO RUN (SELECTED AND NOT DROPPED).
1676                                ; IF SO, SET THE DE-QUEUING FLAG.
1677                                ;-
1678
1679 004306                                    IF #MODHOLD!RELTIME NOTSETIN DTABLE+DT.STO AND DX.QFLG NE #-1 AND #MEMPAS NOTSET
 (6) 004306 032767 012000 174500                                                            BIT     #MODHOLD!RELTIME
 (9) 004314 001040                                                                          BNE     50042$
 (6) 004316 026727 174722 177777                                                            CMP     DX.QFLG,#-1
 (9) 004324 001434                                                                          BEQ     50042$
 (6) 004326 032767 040000 174462                                                            BIT     #MEMPAS,DTABLE+D
 (9) 004334 001030                                                                          BNE     50042$
1680 004336                                        CALL BACTIV IN <#DTABLE>
 (3) 004336 010546                                                                          MOV     R5,-(SP)
 (4) 004340 012745 001004                                                                   MOV     #DTABLE,-(R5)
 (3) 004344 004767 000000G                                                                  JSR     PC,BACTIV
 (3) 004350 012605                                                                          MOV     (SP)+,R5
1681 004352                                        IF.NO.ERROR THEN
 (6) 004352 103421                                                                          BCS     50043$
1682 004354                                            LET R0 := DX.HDR
 (4) 004354 016700 174660                                                                   MOV     DX.HDR,R0
1683 004360                                            IF #BIT14 NOTSETIN STAT(R0) OR #BIT13 SETIN STAT(R0) THEN
 (6) 004360 032760 040000 000026                                                            BIT     #BIT14,STAT(R0)
 (8) 004366 001404                                                                          BEQ     50044$
 (6) 004370 032760 020000 000026                                                            BIT     #BIT13,STAT(R0)
 (9) 004376 001404                                                                          BEQ     50045$
 (6) 004400                                                                                 50044$:
1684 004400                                                LET STAT(R0) := STAT(R0) CLR.BY #ACTBIT
 (6) 004400 042760 004000 000026                                                            BIC     #ACTBIT,STAT(R0)
1685 004406                                            ELSE
 (4) 004406 000403                                                                          BR      50046$
 (3) 004410                                                                                 50045$:
1686 004410                                                LET DX.QFLG := #-1
 (4) 004410 012767 177777 174626                                                            MOV     #-1,DX.QFLG
1687 004416                                            ENDIF
 (4) 004416                                                                                 50046$:
1688 004416                                        ENDIF
 (4) 004416                                                                                 50043$:
1689 004416                                    ENDIF
 (4) 004416                                                                                 50042$:
1690
1691 004416                         INLINE <1$:>
 (2) 004416                                                                                 1$:
1692
1693                                ;+
```

```
1694                                    ; IF IT IS TIME TO SEE IF ALL MODULES HAVE COMPLETED AN EOP, DO IT.
1695                                    ;-
1696
1697  004416                            IF #CKTIM SETIN DTABLE+DT.ST1 THEN
 (6)  004416  032767  100000  174372                                                BIT     #CKTIM,DTABLE+DT
 (9)  004424  001445                                                                BEQ     50047$
1698  004426                                  LET DTABLE+DT.ST1 := DTABLE+DT.ST1 CLR.BY #CKTIM
 (6)  004426  042767  100000  174362                                                BIC     #CKTIM,DTABLE+DT
1699  004434                                  CALL CHKEOP IN <#DTABLE>
 (3)  004434  010546                                                                MOV     R5,-(SP)
 (4)  004436  012745  001004                                                        MOV     #DTABLE,-(R5)
 (3)  004442  004767  000000G                                                       JSR     PC,CHKEOP
 (3)  004446  012605                                                                MOV     (SP)+,R5
1700                                    ;+
1701                                    ; IF ALL MODULES HAVE DONE AN EOP, INCR. SYSTEM PASS COUNT AND SET
1702                                    ; "RELTIME" IF IN RELMODE. IF NOT RELMODE, CLEAR EOP INDICATORS AND
1703                                    ; IF "AUTO" MODE OR "SBKSEL" SET "MEMPAS".
1704                                    ;-
1705  004450                                  IF.NO.ERROR THEN
 (6)  004450  103433                                                                BCS     50050$
1706  004452                                        LET @DTABLE+DT.SYP := @DTABLE+DT.SYP + #1
 (6)  004452  005277  174420                                                        INC     @DTABLE+DT.SYP
1707  004456                                        IF #RELMODE SETIN DTABLE+DT.ST0 THEN
 (6)  004456  032767  020000  174330                                                BIT     #RELMODE,DTABLE+
 (9)  004464  001404                                                                BEQ     50051$
1708  004466                                              LET DTABLE+DT.ST0 := DTABLE+DT.ST0 SET.BY #RELTIME
 (6)  004466  052767  010000  174320                                                BIS     #RELTIME,DTABLE+
1709  004474                                        ELSE
 (4)  004474  000421                                                                BR      50052$
 (3)  004476                                                                  50051$:
1710  004476                                              CALL CLREOP IN <#DTABLE>
 (3)  004476  010546                                                                MOV     R5,-(SP)
 (4)  004500  012745  001004                                                        MOV     #DTABLE,-(R5)
 (3)  004504  004767  000000G                                                       JSR     PC,CLREOP
 (3)  004510  012605                                                                MOV     (SP)+,R5
1711  004512                                              IF #AUTO SETIN DTABLE+DT.CF0 OR #SBKSEL SETIN DTABLE+DT.ST1 THEN
 (6)  004512  032767  000010  174300                                                BIT     #AUTO,DTABLE+DT.
 (8)  004520  001004                                                                BNE     50053$
 (6)  004522  032767  010000  174266                                                BIT     #SBKSEL,DTABLE+D
 (9)  004530  001403                                                                BEQ     50054$
 (6)  004532                                                                  50053$:
1712  004532                                                    LET DTABLE+DT.ST1 := DTABLE+DT.ST1 SET.BY  #MEMPAS
 (6)  004532  052767  040000  174256                                                BIS     #MEMPAS,DTABLE+D
1713  004540                                              ENDIF
 (4)  004540                                                                  50054$:
1714  004540                                        ENDIF
 (4)  004540                                                                  50052$:
1715  004540                                  ENDIF
 (4)  004540                                                                  50050$:
1716  004540                            ENDIF
 (4)  004540                                                                  50047$:
1717                                    ;       UNTIL DX.QFLG NE #0
1718  004540                            INLINE <6$:     TST DX.QFLG>
 (2)  004540  005767  174500                                                  6$:     TST DX.QFLG
1719  004544                            INLINE <BNE 4$>
 (2)  004544  001002                                                                BNE 4$
```

KERNEL - DE-QUEUE NEXT MODULE IN CONTROL QUEUE ROUTINE

SEQ 0057

```
1720  004546                        INLINE <JMP 3$>                    JMP 3$
 (2)  004546  000167  177066
1721  004552                        INLINE <4$:>                       4$:
 (2)  004552
```

```
 1723                                   .SBTTL  KERNEL - DE-QUEUE NEXT MODULE: PASS CPU CONTROL TO MODULE
 1724
 1725
 1726                                   ;+
 1727                                   ; THERE IS A MODULE IN THE MONITOR'S NEXT-TO-EXECUTE SLOT AWAITING EXECUTION,
 1728                                   ; SO LETS GET IT GOING. THIS CONSISTS OF RESTORING ITS REGISTERS AND SWITCHING
 1729                                   ; FROM THE MONITOR'S R5 AND R6 STACKS TO THE MODULE'S R6 STACK, AND THEN PASSING CONTROL
 1730                                   ;-
 1731
 1732  004552                           LET DX.R5 := R5
  (4)  004552  010567  174470                                                          MOV     R5,DX.R5
 1733  004556                           LET DX.SP := SP
  (4)  004556  010667  174466                                                          MOV     SP,DX.SP
 1734  004562                           LET R0 := DX.HDR
  (4)  004562  016700  174452                                                          MOV     DX.HDR,R0
 1735  004566                           LET R0 := R0 + #SVR6
  (6)  004566  062700  000076                                                          ADD     #SVR6,R0
 1736  004572                           LET SP := (R0)
  (4)  004572  011006                                                                  MOV     (R0),SP
 1737  004574                           LET R5 := -(R0)
  (4)  004574  014005                                                                  MOV     -(R0),R5
 1738  004576                           LET R4 := -(R0)
  (4)  004576  014004                                                                  MOV     -(R0),R4
 1739  004600                           LET R3 := -(R0)
  (4)  004600  014003                                                                  MOV     -(R0),R3
 1740  004602                           LET R2 := -(R0)
  (4)  004602  014002                                                                  MOV     -(R0),R2
 1741  004604                           LET R1 := -(R0)
  (4)  004604  014001                                                                  MOV     -(R0),R1
 1742  004606                           LET R0 := -(R0)
  (4)  004606  014000                                                                  MOV     -(R0),R0
 1743
 1744                                   ;+
 1745                                   ;*********************************************************************
 1746                                   ;*************** CONTROL WILL NOW BE PASSED TO THE MODULE ***************
 1747                                   ;*********************************************************************
 1748                                   ;-
 1749
 1750
 1751                                   ;+
 1752                                   ; IF THE OPTION MODULE IS A BACKGROUND MODULE, SET THE MONITOR'S STATUS
 1753                                   ; INDICATOR WORD #0 ACCORDINGLY.  ALSO, SET UP TO RESTORE THE MODULE'S PSW.
 1754                                   ;-
 1755
 1756  004610                           PUSH DX.HDR
  (2)  004610  016746  174424                                                          MOV     DX.HDR,-(SP)
 1757  004614                           LET (SP) := (SP) + #STAT
  (6)  004614  062716  000026                                                          ADD     #STAT,(SP)
 1758  004620                           IF #BIT04 SETIN @(SP) THEN
  (6)  004620  032776  000020  000000                                                  BIT     #BIT04,@(SP)
  (9)  004626  001415                                                                  BEQ     50055$
 1759  004630                                   LET (SP) := @(SP)
  (4)  004630  017616  000000                                                          MOV     @(SP),(SP)
 1760  004634                                   INLINE <BIC #177020,(SP)>
  (2)  004634  042716  177020                                                          BIC #177020,(SP)
 1761
```

KERNEL - DE-QUEUE NEXT MODULE: PASS CPU CONTROL TO MODULE SEQ 0059

```
1762                                    ;+
1763                                    ; RAISE THE PRIORITY TO LEVEL #7 BEFORE SETTING THE BKMODE INDICATOR
1764                                    ; THUS ALLOWING US TIME TO GET THE BKMOD ON THE AIR - BEFORE
1765                                    ; THE BKMOD CAN BE PREEMPTED
1766                                    ;-
1767
1768  004640                                    PUSH #PRI7
 (2)  004640  012746  000340                                                    MOV     #PRI7,-(SP)
1769  004644                                    PUSH #5$
 (2)  004644  012746  004652                                                    MOV     #5$,-(SP)
1770  004650  000002                           RTI
1771
1772  004652                                INLINE <5$:>
 (2)  004652                                                                    5$:
1773  004652                                    LET DTABLE+DT.ST0 := DTABLE+DT.ST0 SET.BY #BKMODE
 (6)  004652  052767  040000  174134                                           BIS     #BKMODE,DTABLE+D
1774  004660                                ELSE
 (4)  004660  000401                                                            BR      50056$
 (3)  004662                                                            50055$:
1775  004662                                    LET (SP) := #0
 (4)  004662  005016                                                            CLR     (SP)
1776  004664                                ENDIF
 (4)  004664                                                            50056$:
1777  004664                                PUSH DX.RET
 (2)  004664  016746  174352                                                    MOV     DX.RET,-(SP)
1778
1779                                    ;+
1780                                    ; DO IT
1781                                    ;-
1782
1783  004670  000002                    RTI
1784
```

```
 1786
 1787                                   ;+
 1788                                   ; THIS ROUTINE PERFORMS THE TURNING ON OF THE APPROPRIATE
 1789                                   ; CPU OPTIONS.
 1790                                   ;-
 1791  004672                           ROUTINE DX.TURNON
  (2)  004672                                                                    DX.TURNON:
 1792
 1793                                   ;+
 1794                                   ; CALL ROTON IF NOT SCRAWNY MONITOR
 1795                                   ; AND OUTPUT APPROPRIATE MESSAGE
 1796                                   ;-
 1797  004672                           IF #NCPUOP NOTSETIN DTABLE+DT.CF0 THEN
  (6)  004672  032767  000020  174120                                           BIT     #NCPUOP,DTABLE+D
  (9)  004700  001012                                                           BNE     50002$
 1798
 1799  004702                                   CALL KROTON IN <#DTABLE,DX.FAK>
  (3)  004702  010546                                                           MOV     R5,-(SP)
  (5)  004704  016745  174530                                                   MOV     DX.FAK,-(R5)
  (4)  004710  012745  001004                                                   MOV     #DTABLE,-(R5)
  (3)  004714  004767  000000G                                                 JSR     PC,KROTON
  (3)  004720  012605                                                           MOV     (SP)+,R5
 1800
 1801  004722                                   CALL DX.TU1
  (3)  004722  004767  000172                                                   JSR     PC,DX.TU1
 1802  004726                           ENDIF
  (4)  004726                                                                    50002$:
 1803
 1804                                   ;+
 1805                                   ; TURN ON KT IF AVAILABLE AND OUTPUT MESSAGE
 1806                                   ;-
 1807
 1808  004726                           IF #KTPRES SETIN DTABLE+DT.CF0 THEN
  (6)  004726  032767  000400  174064                                           BIT     #KTPRES,DTABLE+D
  (9)  004734  001412                                                           BEQ     50003$
 1809  004736                                   CALL KKTON IN <#DTABLE,DX.FAK>
  (3)  004736  010546                                                           MOV     R5,-(SP)
  (5)  004740  016745  174474                                                   MOV     DX.FAK,-(R5)
  (4)  004744  012745  001004                                                   MOV     #DTABLE,-(R5)
  (3)  004750  004767  000000G                                                 JSR     PC,KKTON
  (3)  004754  012605                                                           MOV     (SP)+,R5
 1810  004756                                   CALL DX.TU1
  (3)  004756  004767  000136                                                   JSR     PC,DX.TU1
 1811  004762                           ENDIF
  (4)  004762                                                                    50003$:
 1812
 1813                                   ;+
 1814                                   ; TURN ON "PARITY" OR "ECC" MEMORY IF AVAILABLE, OUTPUT MESSAGE
 1815                                   ;-
 1816
 1817  004762                           IF #ECCMEM SETIN DTABLE+DT.CF0 OR #PARPRES SETIN DTABLE+DT.CF0 THEN
  (6)  004762  032767  000100  174030                                           BIT     #ECCMEM,DTABLE+D
  (8)  004770  001004                                                           BNE     50004$
  (6)  004772  032767  002000  174020                                           BIT     #PARPRES,DTABLE+
  (9)  005000  001412                                                           BEQ     50005$
  (6)  005002                                                                    50004$:
```

```
1818  005002                                  CALL KPON IN <#DTABLE,DX.FAK>
 (3)  005002  010546                                                          MOV     R5,-(SP)
 (5)  005004  016745  174430                                                  MOV     DX.FAK,-(R5)
 (4)  005010  012745  001004                                                  MOV     #DTABLE,-(R5)
 (3)  005014  004767  000000G                                                 JSR     PC,KPON
 (3)  005020  012605                                                          MOV     (SP)+,R5
1819  005022                                  CALL DX.TU1
 (3)  005022  004767  000072                                                  JSR     PC,DX.TU1
1820  005026                              ENDIF
 (4)  005026                                                          50005$:
1821
1822                                        ;+
1823                                        ; TURN ON CACHE IF AVAILABLE AND OUTPUT MESSAGE
1824                                        ;-
1825
1826  005026                              IF #CAPRES SETIN DTABLE+DT.CF0 THEN
 (6)  005026  032767  000004  173764                                          BIT     #CAPRES,DTABLE+D
 (9)  005034  001412                                                          BEQ     50006$
1827  005036                                  CALL KCON IN <#DTABLE,DX.FAK>
 (3)  005036  010546                                                          MOV     R5,-(SP)
 (5)  005040  016745  174374                                                  MOV     DX.FAK,-(R5)
 (4)  005044  012745  001004                                                  MOV     #DTABLE,-(R5)
 (3)  005050  004767  000000G                                                 JSR     PC,KCON
 (3)  005054  012605                                                          MOV     (SP)+,R5
1828  005056                                  CALL DX.TU1
 (3)  005056  004767  000036                                                  JSR     PC,DX.TU1
1829  005062                              ENDIF
 (4)  005062                                                          50006$:
1830
1831                                        ;+
1832                                        ; TURN ON UNIBUS MAP (MAP BOX) IF AVAILABLE AND OUTPUT MESSAGE
1833                                        ;-
1834
1835  005062                              IF #ADDR22 SETIN DTABLE+DT.CF0 THEN
 (6)  005062  032767  001000  173730                                          BIT     #ADDR22,DTABLE+D
 (9)  005070  001412                                                          BEQ     50007$
1836  005072                                  CALL KMON IN <#DTABLE,DX.FAK>
 (3)  005072  010546                                                          MOV     R5,-(SP)
 (5)  005074  016745  174340                                                  MOV     DX.FAK,-(R5)
 (4)  005100  012745  001004                                                  MOV     #DTABLE,-(R5)
 (3)  005104  004767  000000G                                                 JSR     PC,KMON
 (3)  005110  012605                                                          MOV     (SP)+,R5
1837  005112                                  CALL DX.TU1
 (3)  005112  004767  000002                                                  JSR     PC,DX.TU1
1838  005116                              ENDIF
 (4)  005116                                                          50007$:
1839
1840                                        ;+
1841                                        ; RETURN TO CALLER
1842                                        ;-
1843
1844  005116                              ENDRTN
 (3)  005116                                                          50000$:
 (3)  005116                                                          50001$:
 (2)  005116  000207                                                          RTS     PC
1845
```

KERNEL - DE-QUEUE NEXT MODULE: PASS CPU CONTROL TO MODULE SEQ 0062

```
1846
1847  005120                        ROUTINE DX.TU1
 (2)  005120                                                                DX.TU1:
1848
1849                                ;+
1850                                ; RETRIEVE MESSAGE FROM DT.KBRSP WORD IN DTABLE AND OUTPUT
1851                                ; MESSAGE AND WAIT FOR IT TO FINISH OUTPUTTING BEFORE RETURNING
1852                                ; CONTROL
1853                                ;-
1854
1855  005120                        CALL MSGDHOOK IN <#DTABLE,#MSGPOP,DTABLE+DT.KBRSP,#2$>
 (3)  005120  010546                                                        MOV     R5,-(SP)
 (7)  005122  012745  005154                                                MOV     #2$,-(R5)
 (6)  005126  016745  173674                                                MOV     DTABLE+DT.KBRSP,
 (5)  005132  012745  000002                                                MOV     #MSGPOP,-(R5)
 (4)  005136  012745  001004                                                MOV     #DTABLE,-(R5)
 (3)  005142  004767  000000G                                               JSR     PC,MSGDHOOK
 (3)  005146  012605                                                        MOV     (SP)+,R5
1856  005150                        INLINE <1$:>
 (2)  005150                                                                1$:
1857  005150                        INLINE <NOP>
 (2)  005150  000240                                                        NOP
1858  005152                        INLINE <BR 1$>
 (2)  005152  000776                                                        BR 1$
1859  005154                        INLINE <2$:>
 (2)  005154                                                                2$:
1860  005154                        LET DTABLE+DT.KBRSP := #0
 (4)  005154  005067  173646                                                CLR     DTABLE+DT.KBRSP
1861  005160                        ENDRTN
 (3)  005160                                                        50000$:
 (3)  005160                                                        50001$:
 (2)  005160  000207                                                        RTS     PC
1862
1863         002200                 .END    DX.STRT
```

```
ACSR   = 000102
ACTBIT= 004000
ACTIV  = ****** G
AC.MOD= ****** G
AC.MPT= ****** G
AC.TPT= ****** G
AC.TYP= ****** G
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
APTSLP= ****** G
APTVEC  000044
AP.PB  = ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BACTIV= ****** G
BA.HDR= ****** G
BA.MPT= ****** G
BA.RET= ****** G
BA.STA= ****** G
BDACNV= ****** G
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CAST   = ****** G
CASTAT= 000004
CCNTRL  001230 G
CDERCT= 000146
CDWDCT= 000144
CHKACT= ****** G
CHKEOP= ****** G
CKTIM  = 100000
CLKPRE= 000001
CLOCK   001160 G
CLOCKL  001162 G
CLOCKP  001164 G
CLREOP= ****** G
CMDPRC= ****** G
CMDRST= ****** G
CONFIG= 000056
CPUCPE  001224 G
CPUHAR  001216 G
CPULAR  001214 G
CPULSZ  001226 G
CPUMER  001220 G
CPUMNT  001222 G
CQINI  = ****** G
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLOF  001234 G
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DEQCQ  = ****** G
DIAGMC= 000000
DPEOP  = ****** G
DPSTRT= ****** G
DROPMO= 100000
DSEVNT= 000014
DTABLE  001004 G
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MT1= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
DX.CAP  003242 G
DX.DEQ  003634 G
DX.FAK  001440
DX.FK1  001442
DX.HDR  001240 G
DX.INI  003020 G
DX.KFL  001236 G
DX.MED  001402
DX.MON  001346 G
DX.QFL  001244
DX.RET  001242 G
DX.RST  002650 G
DX.R5   001246 G
DX.SIZ  001353
DX.SP   001250 G
DX.STR  002200
DX.SZK  001371
DX.TIT  001252
DX.TUR  004672
DX.TU1  005120
ECCMEM= 000100
ECCSTA= 000010
EMTINS  000030
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERREC  = ****** G
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
HTBUSS= ****** G
HTREIN= ****** G
HT.EXT= ****** G
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
IERRLO= ****** G
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
IOTINS  000020
JACK   = 035060
KBINI  = ****** G
KBINT  = ****** G
KBVEC   000060
KCON   = ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KKTON  = ****** G
KMON   = ****** G
KONTRL  001232 G
KPON   = ****** G
KROTON= ****** G
KRUN   = ****** G
KTERRO= 000040
KTPRES= 000400
KTSET  = ****** G
KTSTAT= 000020
KTXTND= 040000
KW11L  = ****** G
KW11P  = ****** G
LCLEAR= ****** G
LF     = 000012
LPBUF   001212 G
LPCSR   001210 G
LPINT  = ****** G
LPSTAT= 000001
MAPSTA= 000200
MD.BSY= ****** G
MED    = 076600
MEMGMT= ****** G
MEMPAR= ****** G
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
```

```
MSGVEC= 000020           RDWHMI= 000022           SR3    = 172516           WBUFEA= 000136           $ISK3 = 000001
NBKMOD= 001000           RELCTI= ****** G         START    000200           WBUFPA= 000134           $LOCTA= 177777
NCPUOP= 000020           RELERR= 000020           STAT   = 000026           WBUFRQ= 000140           $LSTIN= 000001
NOAPTY= 000002           RELMOD= 020000           STATBI= 064757           WBUFSZ= 000142           $LSTTA= 000001
NULL   = 000000          RELTIM= 010000           STAT1  = 000027           WDFR   = 000116          $NESTL= 177777
OV.KBB= ****** G         RESERV  000010           STINT  = ****** G         WDTO   = 000114          $NSK0 = 000300
OWEN   = 024020          RESREG= ****** G         ST.EXT= ****** G          WTINRE= 000352           $NSK1 = 000110
PAERR  = 000010          RESTAR  001000           SUSPND= 000001            WTWHMI= 000222           $NSK2 = 000110
PARPRE= 002000           RES1   = 000056          SVR0   = 000062           XBUF     001206 G        $NSK3 = 000110
PARSTA= 000100           RES2   = 000060          SVR1   = 000064           XCSR     001204 G        $SAVLE= 177777
PARTAB  001116 G         RICHAR= 031060           SVR2   = 000066           XFLAG  = 000005          $SSK0 = 050006
PASCNT= 000034           RPIRQ  = ****** G         SVR3   = 000070           XOFF   = 000023          $TAGLE= 177777
PBDVEC= ****** G         RPTDAT= 002000           SVR4   = 000072           XON    = 000021          $TAGNU= 050002
PCLEAR= ****** G         RSTRCY= ****** G         SVR5   = 000074           $BGNLE= 177777           $TEMP  = 000300
PDPLSI= 020000           RSTRT  = 000112          SVR6   = 000076           $ENDAD= ****** G         $TSK0 = 050007
PDP50  = 004000          RUBOUT= 000177           SYSCNT= 000052            $ERFLG= 000400           $TSK1 = 050050
PDP70  = 010000          RUNMOD= 100000           SYSERR= 000100            $F$AND= 000310           $TSK2 = 050052
PREVT  = ****** G        R5VALU= 001740           TIMOUT  000004            $F$BAD= 000401           $TSK3 = 050054
PRI0   = 000000          SAM    = 075464          TMPIO  = 000002           $F$BLA= 000170           $$ARGC= 000000
PRI1   = 000040          SAVREG= ****** G         TQINI  = ****** G         $F$CAS= 000150           $$BYTE= 000403
PRI4   = 000200          SBADR  = 000102          TQOVF  = 000002           $F$DEC= 000220           $$CASE= 000000
PRI5   = 000240          SBKMOD= 000000           TRPINS  000034            $F$DO  = 000340          $$DST  = 000000
PRI6   = 000300          SBKSEL= 010000           TTINT  = ****** G         $F$FAL= 000405           $$ELOC= 000402
PRI7   = 000340          SCTAB    001160 G        TTVEC    000064           $F$GOO= 000400           $$ERFL= 000000
PRRLOC= ****** G         SC.ADR= 000006           UIPAR0= 177640            $F$IF  = 000110          $$FLAG= 000001
PR0    = 000000          SC.ALC= 000014           UIPAR1= 177642            $F$INC= 000210           $$FROM= 000000
PR4    = 000200          SC.APC= 000016           UIPAR2= 177644            $F$LOO= 000200           $$LOC  = 005070
PR5    = 000240          SC.CKL= 000002           UIPAR3= 177646            $F$NAM= 000160           $$LOCN= 000000
PR6    = 000300          SC.CKP= 000004           UIPAR4= 177650            $F$NO  = 000403          $$REG  = 177777
PR7    = 000340          SC.CLO= 000000           UIPAR5= 177652            $F$OR  = 000320          $$RETU= 000000
PS     = 177776          SC.HLD= 000010           UIPAR6= 177654            $F$RTI= 000350           $$RTN1= 050000
PSW    = 177776          SC.SCA= 000012           UIPAR7= 177656            $F$RTN= 000300           $$RTN2= 050001
PWRDOW= ****** G         SENDLS= 177777           UIPDR0= 177600            $F$SEL= 000140           $$SRC  = 000000
PWRFL    000024 G        SIZPOL= ****** G         UIPDR1= 177602            $F$THE= 000330           $$TGSV= 000000
RANNUM= 000054           SOFCNT= 000042           UIPDR2= 177604            $F$TRU= 000404           $$TGS1= 000000
RBUF     001202 G        SOFPAS= 000046           UIPDR3= 177606            $F$UNT= 000130           $$TGS2= 000000
RBUFEA= 000130           SPACE  = 000040          UIPDR4= 177610            $F$WHI= 000120           $$TO   = 000004
RBUFPA= 000126           SPOINT= 000032           UIPDR5= 177612            $F$YES= 000402           $$$TAG= 050000
RBUFSZ= 000132           SPVALU= 002200           UIPDR6= 177614            $IFLEV= 177777           .      = 005162
RBUFVA= 000124           SR0    = 177572          UIPDR7= 177616            $ISK0 = 000001
RCSR     001200 G        SR1    = 177574          WASADR= 000104            $ISK1 = 000001
RDSERV= 000101           SR2    = 177576          WBSTAT= 000040            $ISK2 = 000001

. ABS.   005162      000
         000000      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DECX,DSKZ:DECX=SPMAC/ML,EQUATE,DECX
 RUN-TIME: 45 36 .5 SECONDS
 RUN-TIME RATIO: 130/82=1.5
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE APTMOD (APT MODULE PACKAGE)
                        .IDENT /V0.0/
                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       APTMOD
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS MODULE PACKAGE CONTAINS SEVERAL APT ROUTINES:
                        ;               1. APTSLP       APT SLEEP AND CONFIGURATION ROUTINE
                        ;               2. APTSER       APT SOFT ERROR HANDLER
                        ;               3. APTDED       STUFF APT MAILBOX WITH FAILING MODULE NAME
                        ;               4. APTHER       APT HARD ERROR ROUTINE
                        ;               5. APTSEL       SELECT-DESELECT ROUTINE
                        ;               6. APTKIL       STUFF CODE TO MAILBOX IF MONITOR ERROR
                        ;                               (I.E. SYSTEM ERROR) AND HALTS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT                  DATE            BY              REASON
                        ;--
```

```
                            .SBTTL APTMOD (COMMON DEFINITIONS & REFERENCES)

                            .MCALL STRUCT
000000'                     STRUCT
(1) 000000'                 .PRINT  ;SPMAC: VERSION 1.1

        000001              $LSTIN=1
        000001              $LSTTAG=1

                            ;****************
                            ;
                            ; REFERENCED BY OTHER MODULES
                            ;
                            .GLOBL  APTSLP          ;APT SLEEP ROUTINE
                            .GLOBL  APTSER          ;SOFT ERROR HANDLER
                            .GLOBL  APTHER          ;HARD ERROR HANDLER
                            .GLOBL  APTKIL          ;APT HALT ROUTINE
                            .GLOBL  APTDED          ;STUFF FAILING MODULE NAME
                            .GLOBL  APTSEL          ;APT SELECT-DESELECT MODULE ROUTINE
                            .GLOBL  AP.PB           ;APT PARAMETER BLOCK
                            ;****************
                            ; GLOBAL REFERENCES
                            ;
                            .GLOBL  SAVREG          ;
                            .GLOBL  RESREG          ;
                            .GLOBL  RSTRCY          ;RESET RECOVERY ROUTINE
                            .GLOBL  KSUM            ;'SUM' COMMAND ROUTINE
                            .GLOBL  MD.COD          ;MSG CODE
                            .GLOBL  MSGDEQ          ;MESSAGE DEQUER
                            .GLOBL  RCSR            ;KB CSR


                            ;
                            ;****************

                            ; LOCAL STORAGE
                            ;


                            ;******************** APT PARAMETER BLOCK **********************
                            ;
000000'                     AP.PB:
000000' 000000              AP.P00: 0               ;PTABLE WORD 0 - BITS 16 & 17 OF MAILBOX ADDR
000002' 000014'             AP.P02: #AP.MB1         ;LOWER 16 BITS OF MAILBOX ADDRESS
000004' 001604              AP.P04: 900.            ;RUN TIME (IN SECONDS) OF LONGEST TEST
000006' 000170              AP.P06: 120.            ;RUN TIME (IN SECONDS) OF FIRST PASS
                                                    ;INCLUDES THE DELAY REQUIRED BY DEVICES TO
                                                    ;RECOVER FROM POWER FAIL CAUSED BY APT TO
                                                    ;START UP DEC-X11 EXERCISER.
000010' 000000              AP.P10: 0               ;ADDITIONAL RUN TIME (IN SECONDS) OF FIRST
                                                    ;PASS WHEN A SECOND DEVICE IS ADDED.
000012' 000052              AP.P12: <AP.END-AP.MB1>/2       ;LENGTH (IN WORDS) OF E-TABLE + MAILBOX


```

```
                        ;*************** APT MAILBOX ******************
                        ;
000014' 000000          AP.MB1: 0               ;MAILBOX WORD 1 - FATAL ERROR INDICATOR
000016' 020040          AP.MB2: 20040           ;MODULE NAME'S FIRST TWO CHARACTERS
000020' 020040          AP.MB3: 20040           ;MODULE NAME'S THIRD & FIFTH CHARACTERS
000022' 000000          AP.MB4: 0               ;PASS COUNT
000024' 000000          AP.MB5: 0               ;TO BE INCREMENTED EVERY 10 SECONDS
000026' 000000                  0
000030' 000000                  0
000032' 000000                  0

                        ;*************** APT E-TABLE **********************
                        ;
000034'     001         AP.ET1: .BYTE   1               ;E-TABLE - BYTE 1 - SOFT ENVIRONMENT
                                                        ;DEFAULT TO APT MODE
000035'     200                 .BYTE   200             ;BYTE 2 - DEFAULT TO UUT TTY AND DON'T SIZE

000036' 000000          AP.SR1: 0               ;APT SW 1
000040' 000037          AP.SR2: ^D<31>          ;HARD ERROR LIMIT (1) AND SOFT ERROR LIMIT (31.)
000042' 000000          AP.CPU: 0               ;CPU OPTIONS,15-11=TYPE,10=RTC,9=FPP,8=MM
000044' 000000  100000  AP.MM0: 0,100000        ;MEMORY MAP, 100000=16K
000050' 000000  000000  AP.MM1: 0,0             ;ONE BYTE FOR TYPE PLUS THREE BYTES
000054' 000000  000000  AP.MM2: 0,0             ;FOR MAX ADDR OF THAT TYPE
000060' 000000  000000  AP.MM3. 0,0             ;DECX
000064' 000000  000000          0,0             ;DON'T CARE
000070' 000000  000000          0,0             ;ABOUT THESE
000074' 000000  000000          0,0             ;LOCATIONS

000100' 000020          AP.MAP: .BLKW   16.     ;APT MODULE MAP
                                                ;ONE BYTE REQD FOR EACH MODULE
000140'                 AP.END:


                        ;*************** GENERAL LOCAL STORAGE *****************
                        ;
000140' 047515          AP.MC0: .ASCII  /MO/    ;MONITOR ERROR - STUFF "MO" INTO MAILBOX
000142' 050503          AP.MC1: .ASCII  /CQ/    ;CONTROL QUEUE OVERFLOW
000144' 050524                  .ASCII  /TQ/    ;TYPE QUEUE OVERFLOW
000146' 000000                  .WORD   0
000150' 042520                  .ASCII  /PE/    ;PARITY ERRORS
000152' 042515                  .ASCII  /ME/    ;MEMORY ERROR - A LOCATION IS BAD
000154' 052113                  .ASCII  /KT/    ;KT TRAP
000156' 042523                  .ASCII  /SE/    ;TRAP THROUGH LOC. 4 OR 10
                        .EVEN
000160' 000162'         AP.FAK: .WORD   AP.CR   ;POINTER TO NEXT WORD
000162' 000015          AP.CR:  .WORD   CR      ;A FAKE CARRIAGE RETURN FOR 'SUM' COMMAND
                        ;
```

```
                        .SBTTL  APTSLP  APT SLEEP AND CONFIGURE ROUTINE
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       APTSLP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL DETERMINE FROM THE APT ENVIRONMENT WORD 1
                        ;       IF TRULY UNDER APT. IF SO A CHECK IS MADE ON AVAILABILITY OF
                        ;       U.U.T. TERMINAL. IF NONE, THE NOAPTY BIT IS SET IN CONFIG WORD
                        ;       0. IF UNDER APT, A 15 SECOND SLEEP ROUTINE IS ENTERED TO
                        ;       RECOVER FROM POWER LOGIC YANK.
                        ;
                        ; INPUTS:
                        ;       DTABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.CF0
                        ;       DT.APK
                        ;       DT.SYP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL APTSLP IN <DT>
                        ;               WHERE DT = ADDRESS OF DTABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  APTSLP ROUTINE
                                ROUTINE APTSLP <DTADR>
(2)  000164'                                                                    APTSLP:


                                ;+
                                ; SAVE REGISTERS, GET DTABLE ADDR.
                                ;-
                                CALL SAVREG
(3)  000164' 004767  000000G                                                   JSR     PC,SAVREG
                                LET R0 := DTADR(R5)
(4)  000170' 016500  000000                                                    MOV     DTADR(R5),R0
                                ;+
                                ; GET ENVIRONMENT WORD 1 TO R3
                                ;-
                                LET R3 := #AP.ET1
(4)  000174' 012703  000034'                                                   MOV     #AP.ET1,R3

                                ;+
                                ; ZERO FATAL BIT INDICATOR IN CASE IT WAS SET
                                ;-
                                LET AP.MB1 := #0
(4)  000200' 005067  177610                                                    CLR     AP.MB1

                                ;+
                                ; IF BIT0 SET IN ENVIRONMENT WORD 1, SET APTPRES BIT IN DT.CF0, SET UP KEEP ALIVE
                                ; AND SYSTEM PASS COUNT
                                ;-
                                IF #BIT00 SETIN (R3) THEN
(6)  000204' 032713  000001                                                    BIT     #BIT00,(R3)
(9)  000210' 001440                                                            BEQ     50002$
                                        LET DT.CF0(R0) := DT.CF0(R0) SET.BY #APTPRES
(6)  000212' 052760  000200  000014                                           BIS     #APTPRES,DT.CF0(


                                ;+
                                ; **********************************************
                                ; !!!!!! NOTE: KEEP ALIVE COUNTER FOLLOWS - !!!!!!!
                                ; !!!!!! REMOVE SEMI-COLON IF YOU EVER WANT !!!!!!!
                                ; !!!!!! A DEC/X11 KEEP-ALIVE COUNTER. IF   !!!!!!!
                                ; !!!!!! YOU DO, IT WILL BE UPDATED         !!!!!!!
                                ; !!!!!! VERY OFTEN (MANY TIMES A SECOND)   !!!!!!!
                                ; !!!!!! IN DECX .                          !!!!!!!
                                ; **********************************************
                                ;-

                                ;       LET DT.APK(R0) := #AP.MB5


                                        LET DT.SYP(R0) := #AP.MB4
(4)  000220' 012760  000022' 000072                                            MOV     #AP.MB4,DT.SYP(R
```

```
                                ;+
                                ; IF BIT 13 SETIN ENV WORD 1, SET NO APT TTY BIT (NOAPTY)
                                ; IN CONFIG WORD 0 (DT.CF0) AND FAKE OUT THE KB CSR.
                                ;-
000226'                                 IF #BIT13 SETIN (R3) THEN
(6) 000226' 032713  020000                                                      BIT     #BIT13,(R3)
(9) 000232' 001410                                                              BEQ     50003$
000234'                                         LET DT.CF0(R0) := DT.CF0(R0) SET.BY #NOAPTY
(6) 000234' 052760  000002  000014                                              BIS     #NOAPTY,DT.CF0(R
000242'                                         LET RCSR := R0 + #DT.RCS
(4) 000242' 010067  000000G                                                     MOV     R0,RCSR
(6) 000246' 062767  000102  000000G                                             ADD     #DT.RCS,RCSR
000254'                                 ENDIF
(4) 000254'                                                             50003$:

                                ;+
                                ; INIT COUNT FOR SLEEP LOOP
                                ;-

000254'                                 LET R1 := #25
(4) 000254' 012701  000025                                                      MOV     #25,R1
000260'                                 LET R2 := #-1
(4) 000260' 012702  177777                                                      MOV     #-1,R2

                                ;+
                                ; GO TO SLEEP FOR A WHILE TO RECOVER FROM POWER FAIL LINE YANK
                                ;-
000264'                                 WHILE R1 NE #0 DO
(4) 000264'                                                             50004$:
(6) 000264' 005701                                                              TST     R1
(9) 000266' 001410                                                              BEQ     50005$
000270'                                         WHILE R2 NE #0 DO
(4) 000270'                                                             50006$:
(6) 000270' 005702                                                              TST     R2
(9) 000272' 001402                                                              BEQ     50007$
000274'                                                 LET R2 := R2 - #1
(6) 000274' 005302                                                              DEC     R2
000276'                                         ENDDO
(4) 000276' 000774                                                              BR      50006$
(3) 000300'                                                             50007$:
000300'                                         LET R2 := #-1
(4) 000300' 012702  177777                                                      MOV     #-1,R2
000304'                                         LET R1 := R1 - #1
(6) 000304' 005301                                                              DEC     R1
000306'                                 ENDDO
(4) 000306' 000766                                                              BR      50004$
(3) 000310'                                                             50005$:

000310'                         ELSE
(4) 000310' 000403                                                              BR      50010$
(3) 000312'                                                             50002$:


                                ;+
                                ; NOT APT - CLEAR APT PRESENT BIT IN DT.CF0
```

```
  768                                   ;-
  769
  770  000312'                          LET DT.CF0(R0) := DT.CF0(R0) CLR.BY #APTPRES
  (6)  000312' 042760  000200  000014                                    BIC     #APTPRES,DT.CF0(
  771
  772  000320'                          ENDIF
  (4)  000320'                                                    50010$:
  773
  774
  775
  776                                   ;+
  777                                   ; RESTORE REGS
  778                                   ;-
  779
  780  000320'                          CALL RESREG
  (3)  000320' 004767  000000G                                           JSR     PC,RESREG
  781  000324'                          ENDRTN
  (3)  000324'                                                    50000$:
  (3)  000324'                                                    50001$:
  (2)  000324' 000207                                                     RTS     PC
  782
  783
  784
```

```
                        .SBTTL  APTSER (APT SOFT ERROR HANDLER)

                        ;++
                        ; MODULE NAME:
                        ;       APTSER
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS CALLED WHEN THE EXERCISER IS
                        ;       DETERMINED TO BE RUNNING UNDER APT ENVIRONMENT, TO HANDLE
                        ;       A SOFT ERROR.
                        ;
                        ;       THE ERROR COUNT IS COMPARED WITH THE LIMIT SET IN E-TABLE.
                        ;       IF THE COUNT IS LESS THAN THE LIMIT, CONTROL GOES BACK TO
                        ;       THE CALLER.
                        ;
                        ;       IF THE COUNT EQUALS THE LIMIT THE RUN IS OVER (CALL APTDED).
                        ;
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. APTDED
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL APTSER IN <DTA>
                        ;
                        ;           WHERE DTA = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  APTSER CODE

                                ROUTINE APTSER <DTA>
(2)  000326'                                                                    APTSER:

                                ;+
                                ; SAVE REGS; GET DTABLE ADDR, MODULE'S HEADER ADDRESS,
                                ; AND SOFT ERROR LIMIT FROM E-TABLE.
                                ;-

                                CALL SAVREG
(3)  000326' 004767  000000G                                                           JSR     PC,SAVREG

                                LET R0 := DTA(R5)
(4)  000332' 016500  000000                                                            MOV     DTA(R5),R0
                                LET R1 := @DT.PC(R0)
(4)  000336' 017001  000002                                                            MOV     @DT.PC(R0),R1
                                LET R2 := AP.SR2 CLR.BY #177740
(4)  000342' 016702  177472                                                            MOV     AP.SR2,R2
(6)  000346' 042702  177740                                                            BIC     #177740,R2

                                ;+
                                ; IF COUNT IS EQUAL TO THE LIMIT,WE ARE DONE - CALL APTDED
                                ;-

                                IF SOFCNT(R1) HIS R2 THEN
(6)  000352' 026102  000042                                                            CMP     SOFCNT(R1),R2
(9)  000356' 103406                                                                    BLO     50002$
                                        CALL APTDED IN <R0,R1>
(3)  000360' 010546                                                                    MOV     R5,-(SP)
(5)  000362' 010145                                                                    MOV     R1,-(R5)
(4)  000364' 010045                                                                    MOV     R0,-(R5)
(3)  000366' 004767  000010                                                            JSR     PC,APTDED
(3)  000372' 012605                                                                    MOV     (SP)+,R5

                                ENDIF
(4)  000374'                                                                    50002$:

                                ;+
                                ; RESTORE REGS AND RETURN
                                ;-

                                CALL RESREG             ;
(3)  000374' 004767  000000G                                                           JSR     PC,RESREG

                                ENDRTN
(3)  000400'                                                                    50000$:
(3)  000400'                                                                    50001$:
(2)  000400' 000207                                                                    RTS     PC

```

```
                        .SBTTL  APTDED  PUT OPTION MODULE NAME INTO MAILBOX
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       APTDED
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE STUFFS THE 1ST,2ND,3RD AND FIFTH CHARACTERS
                        ;       OF A FAILING OPTION MODULE NAME INTO THE 2ND AND 3RD WORDS
                        ;       OF THE APT MAILBOX. IT RESETS AND RECOVERS. SET THE APT AND
                        ;       FATAL ERROR BITS IN DTABLE'S ERROR WORD.
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;       MODULE HEADER ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ERR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;       RSTRCY
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL APTDED IN <DT,MH>  WHERE:
                        ;               DT=DTABLE ADDRESS
                        ;               MH=MODULE HEADER ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  APTDED ROUTINE

000402'                         ROUTINE APTDED <TB,MD>
(2) 000402'                                                             APTDED:

                                ;+
                                ; SAVE REGISTERS,GET DTABLE ADDR., AND MODULE HEADER ADDR.
                                ;-
000402'                         CALL SAVREG
(3) 000402' 004767  000000G                                             JSR     PC,SAVREG
000406'                         LET R0 := TB(R5)
(4) 000406' 016500  000000                                              MOV     TB(R5),R0
000412'                         LET R1 := MD(R5)
(4) 000412' 016501  000002                                              MOV     MD(R5),R1

                                ;+
                                ; STUFF APT MAILBOX WITH MODULE ASCII NAME,RESET AND RECOVER
                                ; UPDATE ERROR WORD IN DTABLE, AND
                                ; SET APT FATAL BIT IN MB1 AND
                                ; RETURN TO THE CALLER.
                                ;-

000416'                         LET AP.MB2 := (R1)+
(4) 000416' 012167  177374                                              MOV     (R1)+,AP.MB2
000422'                         LET R3 := #AP.MB3
(4) 000422' 012703  000020'                                             MOV     #AP.MB3,R3
000426'                         LET (R3)+ :B= (R1)+
(4) 000426' 112123                                                      MOVB    (R1)+,(R3)+
000430'                         INLINE <TSTB (R1)+>
(2) 000430' 105721                                                      TSTB (R1)+
000432'                         LET (R3) :B= (R1)
(4) 000432' 111113                                                      MOVB    (R1),(R3)

000434'                         INLINE <RESET>
(2) 000434' 000005                                                      RESET
000436'                         CALL RSTRCY IN <R0>
(3) 000436' 010546                                                      MOV     R5,-(SP)
(4) 000440' 010045                                                      MOV     R0,-(R5)
(3) 000442' 004767  000000G                                             JSR     PC,RSTRCY
(3) 000446' 012605                                                      MOV     (SP)+,R5

000450'                         LET DT.ERR(R0) := DT.ERR(R0) SET.BY #APTFER!FATERR
(6) 000450' 052760  100004  000020                                      BIS     #APTFER!FATERR,D
000456'                         LET AP.MB1 := AP.MB1 + #1
(6) 000456' 005267  177332                                              INC     AP.MB1
000462'                         CALL RESREG
(3) 000462' 004767  000000G                                             JSR     PC,RESREG
000466'                         ENDRTN
(3) 000466'                                                     50000$:
(3) 000466'                                                     50001$:
(2) 000466' 000207                                                      RTS     PC
```

```
                        .SBTTL  APTHER (APT HARD ERROR HANDLER)

                        ;++
                        ; MODULE NAME:
                        ;       APTHER
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS CALLED WHEN THE EXERCISER IS
                        ;       DETERMINED TO BE RUNNING UNDER APT ENVIRONMENT, TO HANDLE
                        ;       A HARD ERROR.
                        ;
                        ;       IF THE ERROR COUNT IS LESS THAN THE LIMIT SET IN E-TABLE,
                        ;       CONTROL GOES BACK TO THE CALLER.
                        ;
                        ;       IF THE COUNT EQUALS THE LIMIT THE RUN IS OVER (CALL APTDED).

                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. APTDED
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL APTHER IN <DTA>
                        ;
                        ;           WHERE DTA = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  APTHER CODE

                                ROUTINE APTHER <DTA>
000470'                                                         APTHER:

                                ;+
                                ; SAVE REGS; GET DTABLE ADDR, MODULE'S HEADER ADDRESS,
                                ; AND HARD ERROR LIMIT FROM E-TABLE.
                                ;-

                                CALL SAVREG
000470' 004767  000000G                                                 JSR     PC,SAVREG

                                LET R0 := DTA(R5)
000474' 016500  000000                                                  MOV     DTA(R5),R0
                                LET R1 := @DT.PC(R0)
000500' 017001  000002                                                  MOV     @DT.PC(R0),R1
                                LET R2 := SWAP AP.SR2
000504' 016702  177330                                                  MOV     AP.SR2,R2
000510' 000302                                                          SWAB    R2
                                LET R2 := R2 CLR.BY #177740
000512' 042702  177740                                                  BIC     #177740,R2

                                ;+
                                ; IF COUNT IS EQUAL TO THE LIMIT,THEN WE ARE DONE
                                ;-

                                IF HRDCNT(R1) HIS R2 THEN
000516' 026102  000044                                                  CMP     HRDCNT(R1),R2
000522' 103406                                                          BLO     50002$

                                        CALL APTDED IN <R0,R1>
000524' 010546                                                          MOV     R5,-(SP)
000526' 010145                                                          MOV     R1,-(R5)
000530' 010045                                                          MOV     R0,-(R5)
000532' 004767  177644                                                  JSR     PC,APTDED
000536' 012605                                                          MOV     (SP)+,R5

                                ENDIF
000540'                                                         50002$:

                                ;+
                                ; RESTORE REGS AND RETURN
                                ;-

                                CALL RESREG
000540' 004767  000000G                                                 JSR     PC,RESREG

                                ENDRTN
000544'                                                         50000$:
000544'                                                         50001$:
000544' 000207                                                          RTS     PC

```

```
1043                            .SBTTL  APTSEL - SELECT OPTION MODULES FROM MAP
1044                            .IDENT  /V0.0/
1045
1046                            ;++
1047                            ; MODULE NAME:
1048                            ;       APTSEL
1049                            ;
1050                            ; FUNCTIONAL DESCRIPTION:
1051                            ;       THIS ROUTINE WILL SEARCH THE MODULE LIST AND SELECT MODULES
1052                            ;       IF BITS ARE SET IN MODULE MAP IN APT ETABLE. IF NO BIT IS SET FOR THE
1053                            ;       CORRESPONDING MODULE, THE MODULE IS DESELECTED. THE BITS
1054                            ;       IN THE MAP CORRESPOND TO THE DEV COUNT TO BE USED BY THE MODULE
1055                            ;       (DVID1 GETS THE BITS).
1056                            ;
1057                            ; INPUTS:
1058                            ;       DTABLE ADDRESS
1059                            ;
1060                            ; IMPLICIT INPUTS:
1061                            ;       DT.MLST
1062                            ;
1063                            ; OUTPUTS:
1064                            ;       NONE
1065                            ;
1066                            ; IMPLICIT OUTPUTS:
1067                            ;       NONE
1068                            ;
1069                            ; PATHOLOGICAL CONNECTIONS:
1070                            ;       NONE
1071                            ;
1072                            ; SUBORDINATE ROUTINES CALLED:
1073                            ;       SAVREG
1074                            ;       RESREG
1075                            ;
1076                            ; FUNCTIONAL SIDE EFFECTS:
1077                            ;       NONE
1078                            ;
1079                            ; CALLING SEQUENCE:
1080                            ;       CALL APTSEL IN <DT>
1081                            ;       WHERE DT = ADDRESS OF DTABLE
1082                            ;
1083                            ; VERSION:
1084                            ;       0.0
1085                            ;
1086                            ; EDIT          DATE            BY              REASON
1087                            ;--
1088
```

```
                                .SBTTL  APTSEL ROUTINE

000546'                         ROUTINE APTSEL <DTBL>
(2) 000546'                                                                 APTSEL:

                                ;+
                                ; SAVE REGISTERS,DTABLE AND GET MAP ADDRESS
                                ;-

000546'                         CALL SAVREG
(3) 000546' 004767  000000G                                                 JSR     PC,SAVREG
000552'                         LET R0 := DTBL(R5)
(4) 000552' 016500  000000                                                  MOV     DTBL(R5),R0
000556'                         LET R2 := DT.MLST(R0)
(4) 000556' 016002  000032                                                  MOV     DT.MLST(R0),R2

000562'                         LET R3 := #AP.MAP
(4) 000562' 012703  000100'                                                 MOV     #AP.MAP,R3


                                ;+
                                ; WHILE NOT AT END OF MODULE LIST, GET MODULE HEADER ADDR.; IF BYTE
                                ; IN APT MAP = 0, DESELECT MODULE; IF NOT SELECT MODULE AND SET
                                ; UP DEVICE COUNT
                                ;-
000566'                         WHILE (R2) NE #0 DO
(4) 000566'                                                                 50002$:
(6) 000566' 005712                                                                  TST     (R2)
(9) 000570' 001416                                                                  BEQ     50003$
000572'                                 LET R1 := (R2)+
(4) 000572' 012201                                                                  MOV     (R2)+,R1
000574'                                 IFB (R3) EQ #0 THEN
(6) 000574' 105713                                                                  TSTB    (R3)
(9) 000576' 001004                                                                  BNE     50004$

                                ;+
                                ; IF ZERO IN APT TABLE, DESELECT THE MODULE
                                ;-

000600'                                         LET STAT(R1) := STAT(R1) CLR.BY #BIT14
(6) 000600' 042761  040000  000026                                                 BIC     #BIT14,STAT(R1)
000606'                                 ELSE
(4) 000606' 000405                                                                  BR      50005$
(3) 000610'                                                                 50004$:

000610'                                         LET STAT(R1) := STAT(R1) SET.BY #BIT14
(6) 000610' 052761  040000  000026                                                 BIS     #BIT14,STAT(R1)
000616'                                         LET DVID1(R1) :B= (R3)
(4) 000616' 111361  000014                                                          MOVB    (R3),DVID1(R1)
000622'                                 ENDIF
(4) 000622'                                                                 50005$:

                                ;+
                                ; ADVANCE TO NEXT IN MAP
                                ;-
```

```
 1129  000622'                          INLINE <TSTB    (R3)+>
  (2)  000622' 105723                                          TSTB    (R3)+
 1130  000624'                        ENDDO
  (4)  000624' 000760                                          BR      50002$
  (3)  000626'                                          50003$:
 1131
 1132
 1133                                  ;+
 1134                                  ; RESTORE REGISTERS
 1135                                  ;-
 1136  000626'                        CALL RESREG
  (3)  000626' 004767  000000G                                JSR     PC,RESREG
 1137  000632'                        ENDRTN
  (3)  000632'                                          50000$:
  (3)  000632'                                          50001$:
  (2)  000632' 000207                                          RTS     PC
 1138
```

```
1140                            .SBTTL  APTKIL - KILL APT BY HALTING
1141                            .IDENT  /V0.0/
1142
1143                            ;++
1144                            ; MODULE NAME:
1145                            ;       APTKIL
1146                            ;
1147                            ; FUNCTIONAL DESCRIPTION:
1148                            ;       THIS ROUTINE DETERMINES IF AN ERROR WAS CAUSED BY THE MONITOR
1149                            ;       AND IF SO, STUFFS AN APPROPRIATE 4 LETTER CODE INTO WORDS 2 & 3
1150                            ;       OF APT MAILBOX. A RUN SUMMARY IS OUTPUT.  THE EXERCISER THEN HALTS.
1151                            ;
1152                            ; INPUTS:
1153                            ;       DTABLE ADDRESS
1154                            ;
1155                            ; IMPLICIT INPUTS:
1156                            ;       DT.ERR
1157                            ;
1158                            ; OUTPUTS:
1159                            ;       NONE
1160                            ;
1161                            ; IMPLICIT OUTPUTS:
1162                            ;       NONE
1163                            ;
1164                            ; PATHOLOGICAL CONNECTIONS:
1165                            ;       MD.COD
1166                            ;
1167                            ; SUBORDINATE ROUTINES CALLED:
1168                            ;       SAVREG
1169                            ;       RESREG
1170                            ;       MSGDEQ
1171                            ;       KSUM
1172                            ;
1173                            ; FUNCTIONAL SIDE EFFECTS:
1174                            ;       ***** HALTS THE EXERCISER ******
1175                            ;
1176                            ; CALLING SEQUENCE:
1177                            ;       CALL APTKIL IN <DT> WHERE:
1178                            ;               DT = DTABLE ADDRESS
1179                            ;
1180                            ; VERSION:
1181                            ;       0.0
1182                            ;
1183                            ; EDIT          DATE            BY              REASON
1184                            ;--
1185
```

```
                                .SBTTL  APTKIL ROUTINE

000634'                         ROUTINE APTKIL <TAB>
(2) 000634'                                                             APTKIL:

                                ;+
                                ; SAVE REGISTERS AND DTABLE ADDR.
                                ;-

000634'                         CALL SAVREG
(3) 000634' 004767  000000G                                                     JSR     PC,SAVREG
000640'                         LET R0 := TAB(R5)
(4) 000640' 016500  000000                                                      MOV     TAB(R5),R0

                                ;+
                                ; COPY DT.ERR INTO R1 AND MASK OUT FATAL ERROR BITS
                                ;-

000644'                         LET R1 := DT.ERR(R0) CLR.BY #100004
(4) 000644' 016001  000020                                                      MOV     DT.ERR(R0),R1
(6) 000650' 042701  100004                                                      BIC     #100004,R1

                                ;+
                                ; IF COPY OF DT.ERR IS NON ZERO, MOVE 'MO' TO WORD 2.
                                ; SHIFT DT.ERR COPY UNTIL ZERO; ADVANCE R2 TO POINT TO PROPER
                                ; ADDRESS OF 2 LETTER MSG
                                ;-

000654'                         IF R1 NE #0 THEN
(6) 000654' 005701                                                              TST     R1
(9) 000656' 001416                                                              BEQ     50002$
000660'                                 LET AP.MB2 := AP.MC0
(4) 000660' 016767  177254  177130                                              MOV     AP.MC0,AP.MB2
000666'                                 LET R2 := #AP.MC1
(4) 000666' 012702  000142'                                                     MOV     #AP.MC1,R2
000672'                                 LET R1 := R1 SHIFT -1
(7) 000672' 006201                                                              ASR     R1
000674'                                 WHILE R1 NE #0 DO
(4) 000674'                                                             50003$:
(6) 000674' 005701                                                              TST     R1
(9) 000676' 001404                                                              BEQ     50004$
000700'                                         LET R1 := R1 SHIFT -1
(7) 000700' 006201                                                              ASR     R1
000702'                                         LET R2 := R2 + #2
(6) 000702' 062702  000002                                                      ADD     #2,R2
000706'                                 ENDDO
(4) 000706' 000772                                                              BR      50003$
(3) 000710'                                                             50004$:

                                ;+
                                ; STUFF 2 LETTER CODE TO MB3
                                ;-

000710'                                 LET AP.MB3 := (R2)
(4) 000710' 011267  177104                                                      MOV     (R2),AP.MB3
```

```
 1225  000714'                        ENDIF
  (4)  000714'                                                                    50002$:
 1226                                ;+
 1227                                ; DO A RUN SUMMARY SUPPLYING FAKE CARRIAGE RETURN ARGUMENT. KEEP CALLING
 1228                                ; MSG DEQUER UNTIL IT'S ALL OUTPUTTED.
 1229                                ;-
 1230
 1231  000714'                        CALL KSUM IN <R0,AP.FAK>
  (3)  000714' 010546                                                                     MOV     R5,-(SP)
  (5)  000716' 016745  177236                                                             MOV     AP.FAK,-(R5)
  (4)  000722' 010045                                                                     MOV     R0,-(R5)
  (3)  000724' 004767  000000G                                                            JSR     PC,KSUM
  (3)  000730' 012605                                                                     MOV     (SP)+,R5
 1232  000732'                        REPEAT
  (3)  000732'                                                                    50005$:
 1233  000732'                                CALL MSGDEQ IN <R0>
  (3)  000732' 010546                                                                     MOV     R5,-(SP)
  (4)  000734' 010045                                                                     MOV     R0,-(R5)
  (3)  000736' 004767  000000G                                                            JSR     PC,MSGDEQ
  (3)  000742' 012605                                                                     MOV     (SP)+,R5
 1234  000744'                        UNTIL MD.COD EQ #MSGNUL
  (3)  000744' 026727  000000G 177775                                                     CMP     MD.COD,#MSGNUL
  (6)  000752' 001367                                                                     BNE     50005$
 1235
 1236
 1237                                ;+
 1238                                ; ************************************
 1239                                ; !!!!!!!!!!!!!!!! HALT !!!!!!!!!!!!!
 1240                                ; ************************************
 1241                                ;-
 1242
 1243  000754'                        INLINE <HALT>
  (2)  000754' 000000                                                                     HALT
 1244
 1245                                ;+
 1246                                ; THE END
 1247                                ;-
 1248
 1249                                ;+
 1250                                ;PRESS CONTINUE TO CONTINUE BUT
 1251                                ;ANY FURTHER ERRORS WILL GET U HERE AGAIN
 1252                                ;-
 1253
 1254                                ;+
 1255                                ; RESTORE REGS
 1256                                ;-
 1257
 1258  000756'                        CALL RESREG
  (3)  000756' 004767  000000G                                                            JSR     PC,RESREG
 1259  000762'                        ENDRTN
  (3)  000762'                                                                    50000$:
  (3)  000762'                                                                    50001$:
  (2)  000762' 000207                                                                     RTS     PC
 1260         000001                 .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTDED  000402RG
APTFER= 000004
APTHER  000470RG
APTKIL  000634RG
APTPRE= 000200
APTSEL  000546RC
APTSER  000326RG
APTSLP  000164RG
AP.CPU  000042R
AP.CR   000162R
AP.END  000140R
AP.ET1  000034R
AP.FAK  000160R
AP.MAP  000100R
AP.MB1  000014R
AP.MB2  000016R
AP.MB3  000020R
AP.MB4  000022R
AP.MB5  000024R
AP.MC0  000140R
AP.MC1  000142R
AP.MM0  000044R
AP.MM1  000050R
AP.MM2  000054R
AP.MM3  000060R
AP.PB   000000RG
AP.P00  000000R
AP.P02  000002R
AP.P04  000004R
AP.P06  000006R
AP.P10  000010R
AP.P12  000012R
AP.SR1  000036R
AP.SR2  000040R
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTA    = 000000
DTADR  = 000000
DTBL   = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KSUM   = ****** G
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MD     = 000002
MD.COD= ****** G
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RCSR   = ****** G
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRCY= ****** G
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
```

```
R5VALU= 001740          SVR0  = 000062          UIPDR7= 177616          $F$NO  = 000403          $TSK2  = 050004
SAM   = 075464          SVR1  = 000064          WASADR= 000104          $F$OR  = 000320          $TSK3  = 050006
SAVREG= ****** G        SVR2  = 000066          WBSTAT= 000040          $F$RTI= 000350           $TSK4  = 050007
SBADR = 000102          SVR3  = 000070          WBUFEA= 000136          $F$RTN= 000300           $$ARGC= 000002
SBKMOD= 000000          SVR4  = 000072          WBUFPA= 000134          $F$SEL= 000140           $$BYTE= 000403
SBKSEL= 010000          SVR5  = 000074          WBUFRQ= 000140          $F$THE= 000330           $$CASE= 000000
SC.ADR= 000006          SVR6  = 000076          WBUFSZ= 000142          $F$TRU= 000404           $$DST  = 000000
SC.ALC= 000014          SYSCNT= 000052          WDFR  = 000116          $F$UNT= 000130           $$ELOC= 000402
SC.APC= 000016          SYSERR= 000100          WDTO  = 000114          $F$WHI= 000120           $$ERFL= 000000
SC.CKL= 000002          TAB   = 000000          WTINRE= 000352          $F$YES= 000402           $$FLAG= 000001
SC.CKP= 000004          TB    = 000000          WTWHMI= 000222          $IFLEV= 177777           $$FROM= 000000
SC.CLO= 000000          TMPIO = 000002          XFLAG = 000005          $ISK0  = 000001          $$LOC  = 000752R
SC.HLD= 000010          TQOVF = 000002          XOFF  = 000023          $ISK1  = 000001          $$LOCN= 000000
SC.SCA= 000012          UIPAR0= 177640          XON   = 000021          $LOCTA= 177777           $$REG  = 177777
SENDLS= 177777          UIPAR1= 177642          $BGNLE= 177777          $LSTIN= 000001           $$RETU= 000000
SOFCNT= 000042          UIPAR2= 177644          $ERFLG= 000400          $LSTTA= 000001           $$RTN1= 050000
SOFPAS= 000046          UIPAR3= 177646          $F$AND= 000310          $NESTL= 177777           $$RTN2= 050001
SPACE = 000040          UIPAR4= 177650          $F$BAD= 000401          $NSK0  = 000300          $$SRC  = 000000
SPOINT= 000032          UIPAR5= 177652          $F$BLA= 000170          $NSK1  = 000130          $$TGSV= 000000
SPVALU= 002200          UIPAR6= 177654          $F$CAS= 000150          $NSK2  = 000120          $$TGS1= 000000
SR0   = 177572          UIPAR7= 177656          $F$DEC= 000220          $NSK3  = 000120          $$TGS2= 000000
SR1   = 177574          UIPDR0= 177600          $F$DO  = 000340         $SAVLE= 177777           $$TO   = 000000
SR2   = 177576          UIPDR1= 177602          $F$FAL= 000405          $SSK0  = 050004          $$$TAG= 050000
SR3   = 172516          UIPDR2= 177604          $F$GOO= 000400          $TAGLE= 177777           .      = 000764R
STAT  = 000026          UIPDR3= 177606          $F$IF  = 000110         $TAGNU= 050006
STATBI= 064757          UIPDR4= 177610          $F$INC= 000210          $TEMP  = 000300
STAT1 = 000027          UIPDR5= 177612          $F$LOO= 000200          $TSK0  = 050005
SUSPND= 000001          UIPDR6= 177614          $F$NAM= 000160          $TSK1  = 050003

. ABS.  000000     000
        000764     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:APTMOD,DSKZ:APTMOD=SPMAC/ML,EQUATE,APTMOD
 RUN-TIME: 20 11 .4 SECONDS
 RUN-TIME RATIO: 54/32=1.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  BESRV -BUSS ERROR ROUTINE
                        .IDENT /0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       BESRV
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       IT CONSISTS OF THE FOLLOWING MODULES:
                        ;               1. BUSERR - PROCESS THE PC,PSW,SP & VECTOR AT THE TIME
                        ;                  OF THE TRAP TO 4 OR 10, AND
                        ;               2. MODCHK - INCREMENT THE # OF SYS ERRORS BY AN OPTION
                        ;                  MODULE AND RETURN THE MODULE NAME AND OFFSET PC.
                        ;                  IT WILL ALSO  DROP THE OPTION MODULE IF TOO MANY SYS
                        ;                  ERRORS HAVE BEEN CAUSED BY IT.
                        ;       NOTE:  IT IS ASSUMED THE PROCESSOR IS AT PRIORITY 7
                        ;              WHEN THIS ROUTINE IS ENTERED.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  BUSERR PROCESS BUS ERROR TRAPS AND RESERVED INS. TRAPS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       BUSERR
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS ROUTINE WILL PROCESS A BUS ERROR TRAP AND A RESERVED
                        ;       INSTRUCTION TRAP. IT WILL INCREMENT
                        ;       THE EXERCISER SYSTEM ERROR COUNT AND SET A SYSTEM
                        ;       ERROR BIT IN THE ERROR WORD. IT WILL FORM AND QUEUE
                        ;       A SYSTEM ERROR MESSAGE CONTAINING THE PC, PSW, AND SP AT
                        ;       THE TIME OF THE SYSTEM ERROR. IF THE SYSTEM ERROR WAS
                        ;       CAUSED BY AN OPTION MODULE - THE OPTION MODULE NAME
                        ;       AND OFFSET ADDRESS WHERE SYSTEM ERROR OCCURRED WILL ALSO
                        ;       BE OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.CF0
                        ;       2. DT.EVNT
                        ;       3. DT.MLST
                        ;       4. DT.PSW
                        ;       5. DT.SP
                        ;       6. DT.PC
                        ;       7. DT.ESIZ
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.ERR
                        ;       2. DT.EXS
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. RSTRCY       RECOVER FROM RESET
                        ;       2. ERRLOG       ERROR LOGGING
                        ;       3. MSGDHOOK     HOOK A MESSAGE
                        ;       4. BOA16        16 BIT BINARY TO OCTAL ASCII CONVERSION ROUTINE
                        ;       5. BDACNV       BINARY TO DECIMAL ASCII CONVERSION ROUTINE
                        ;       6. GPA          GET PHYSICAL ADDRESS ROUTINE
                        ;       7. SAVREG       SAVE REGISTERS
                        ;       8. RESREG       RESTORE REGISTERS
                        ;       9. BOAC         BINARY TO OCTAL ASCII CONVERSION
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
```

```
                        ;       CALL BUSERR IN <DTADR>
                        ;
                        ;               WHERE DTADR = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR BUSERR
                        ;
                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1
                 000001 $LSTIN=1
                 000001 $LSTTAG=1


                        ;
                        ;****************
                        ;
                        ; REFERENCED  BY OTHER MODULES
                        ;
                        .GLOBL  BUSERR                  ;ROUTINE'S ENTRY ADDRESS
                        .GLOBL  BE.MCNT                 ;# OF SYS ERRORS BY MONITOR
                        ;
                        ;****************
                        ;
                        ; GLOBAL REFERENCES
                        ;
                        .GLOBL  MSGDHOOK                        ; TT INIT ROUTINE
                        .GLOBL  SAVREG                  ;SAVE REGISTERS
                        .GLOBL  RESREG                  ;RESTORE REGISTERS
                        .GLOBL  GPA                     ;GET PHYSICAL ADDRESS
                        .GLOBL  BOA16                   ;ROUTINE TO CONVERT BINARY TO OCTAL ASCII
                        .GLOBL  BOAC                    ;CONVERT 18 OR 22 BIT PHYS ADDR TO OCTAL ASCII
                        .GLOBL  BDACNV                  ;BINARY TO DECIMAL ASCII CONVERSION RTNE
                        .GLOBL  ERRLOG                  ;ERROR LOGGING
                        .GLOBL  DRPMOD                  ;DROP A MODULE
                        .GLOBL  RSTRCY                  ;RECOVER CPU REGS AFTER RESET
                        ;
                        ;****************
                        ;
                        ; LOCAL STORAGE
                        ;
000000' 000000          BE.VA:  0                       ;VIRTUAL ADDRESS
000002' 000000          BE.PA:  0                       ;LOWER 16 BITS OF PHYSICAL ADDRESS
000004' 000000          BE.EA:  0                       ;EXTENDED ADDRESS WORD
000006' 000000          BE.VPC: .WORD   0               ;STORE PC
000010' 000000          BE.VEC: .WORD   0               ;STORE EITHER A 4 OR 10 (THE VECTOR LOC.)
000012' 000000          BE.OFPC: .WORD  0               ;STORE OFFSET PC HERE
000014' 000000          BE.PSW: .WORD   0               ;STORE PSW
000016' 000000          BE.STK: .WORD   0               ;STORE STACK POINTER
000020' 000012          BE.MX2: .WORD   ^D<10>          ;MAX # OF ALLOWABLE SYS ERRORS BY THE MONITOR
000022' 000000          BE.MCNT: .WORD  0               ;LOC TO HOLD NUMBER OF SYS ERRORS BY MONITOR
000024' 000036          BE.MX3: .WORD   ^D<30>          ;MAX # OF ALLOWABLE SYS ERRORS BY EXERCISER
000026' 000004          BE.MAX: .WORD   ^D<4>           ;MAX NO. OF SYS ERRORS ALLOWED BY AN OPTION MODULE


                        ;+
                        ; THE ORDER OF THE FOLLOWING LABELS SHOULD NOT BE CHANGED
                        ; BECAUSE THEY ARE PART OF THE SYSTEM ERROR MESSAGE
                        ;-

000030' 022445  025052  020040  BE.SEMSG:       .ASCII  /%%**  SYSTEM ERROR  **%/
```

```
        000036' 054523  052123  046505
        000044' 042440  051122  051117
        000052' 020040  025052     045
  660   000057'    045  042526  052103                  .ASCII  /%VECTOR   PC+      ADDR     PSW      SP     ERCT%/
        000064' 051117  020040  050040
        000072' 025503  020040  020040
        000100' 020040  042101  051104
        000106' 020040  020040  050040
        000114' 053523  020040  020040
        000122' 020040  050123  020040
        000130' 020040  042440  041522
        000136' 022524
  661   000140' 000006                  BE.AVEC:        .BLKB   6       ;HOLD ASCII REPRESENTATION OF VECTOR ADR
  662   000146' 020040                                  .ASCII  /  /
  663   000150' 000006                  BE.AVPC:        .BLKB   6       ;HOLD ASCII REPRESENTATION OF VPC
  664   000156' 020040                                  .ASCII  /  /
  665   000160' 000010                  BE.APPC:        .BLKB   8.      ;HOLD ASCII REPRESENTATION OF PPC
  666   000170' 020040                                  .ASCII  /  /
  667   000172' 000006                  BE.APSW:        .BLKB   6       ;HOLD ASCII REPRESENTATION OF PSW
  668   000200' 020040                                  .ASCII  /  /
  669   000202' 000006                  BE.ASTK:        .BLKB   6       ;HOLD ASCII REPRESENTATION OF STACK
  670   000210' 020040                                  .ASCII  /  /
  671   000212' 000006                  BE.AEXS:        .BLKB   6       ;HOLD ASCII REPRESENTATION OF SYS ERR CT
  672   000220' 000045                                  .ASCIZ  /%/
  673                                   ;
  674                                   ;
  675   000222' 040445  020124          BE.ATMSG:       .ASCII  /%AT /  ;"AT"
  676   000226' 000005                  BE.AMOD:        .BLKB   5       ;HOLD ASCII MODULE NAME
  677   000233'    040                                  .ASCII  / /
  678   000234' 000006                  BE.AOFPC:       .BLKB   6       ;HOLD ASCII OF OFFSET PC
  679   000242' 000045                                  .ASCIZ  /%/
  680                                   .EVEN
  681                                   ;****************
  682
  683
```

```
                                .SBTTL  BUSERR ROUTINE

000244'                         ROUTINE BUSERR  <TABADR>
(2) 000244'                                                             BUSERR:

                                ;+
                                ; SAVE REGISTERS AND GET DTABLE
                                ;-

000244'                         CALL SAVREG
(3) 000244' 004767  000000G                                                     JSR     PC,SAVREG
000250'                         LET R0 := TABADR(R5)
(4) 000250' 016500  000000                                                      MOV     TABADR(R5),R0

                                ;+
                                ; GET EVENT CODE AND STORE VECTOR ADDRESS 4 OR 10 IN BE.VEC
                                ; 4 = BUS ERROR AND 10 = RESERVED INSTRUCTION
                                ;-

000254'                         IF DT.EVNT(R0) EQ #EVNTBE THEN
(6) 000254' 026027  000000  000200                                             CMP     DT.EVNT(R0),#EVN
(9) 000262' 001004                                                              BNE     50002$
000264'                                 LET BE.VEC := #4
(4) 000264' 012767  000004  177516                                              MOV     #4,BE.VEC
000272'                         ELSE
(4) 000272' 000403                                                              BR      50003$
(3) 000274'                                                             50002$:
000274'                                 LET BE.VEC := #10
(4) 000274' 012767  000010  177506                                              MOV     #10,BE.VEC
000302'                         ENDIF
(4) 000302'                                                             50003$:

                                ;+
                                ; SAVE PC,PSW, AND STACK POINTER AT TIME OF TRAP
                                ;-

000302'                         LET BE.VPC := DT.PC(R0)
(4) 000302' 016067  000002  177476                                              MOV     DT.PC(R0),BE.VPC
000310'                         LET BE.PSW := DT.PSW(R0)
(4) 000310' 016067  000004  177476                                              MOV     DT.PSW(R0),BE.PS
000316'                         LET BE.STK := DT.SP(R0)
(4) 000316' 016067  000006  177472                                              MOV     DT.SP(R0),BE.STK

                                ;+
                                ; SET SYS ERROR OCCURRED BIT IN ERROR WORD
                                ;-

000324'                         LET DT.ERR(R0) := DT.ERR(R0) SET.BY #SYSERR
(6) 000324' 052760  000100  000020                                              BIS     #SYSERR,DT.ERR(R

                                ;+
                                ; INCREMENT EXERCISER SYS ERROR COUNT
                                ;-
```

```
                                LET DT.EXS(R0) := DT.EXS(R0) + #1
(6)  000332' 005260  000060                                                     INC     DT.EXS(R0)

                                ;+
                                ; IF EXERCISER SYS ERROR COUNT >= MAX,ALSO SET FATAL ERROR BIT IN DT.ERR
                                ;-

                                IF DT.EXS(R0) HIS BE.MX3 THEN
(6)  000336' 026067  000060  177460                                             CMP     DT.EXS(R0),BE.MX
(9)  000344' 103403                                                             BLO     50004$
                                        LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6)  000346' 052760  100000  000020                                             BIS     #FATERR,DT.ERR(R
                                ENDIF
(4)  000354'                                                            50004$:

                                ;+
                                ; GET OPTION MODULE LIST POINTER. IF THE PC IS GREATER THAN
                                ; STARTING ADDRESS OF MODULES,
                                ; BUT LESS THAN THE HIGHEST ADDRESS OF EXERCISER,
                                ; DETERMINE WHICH MODULE CAUSED THE
                                ; ERROR BY CALLING MODCHK. R1 WILL CONTAIN STARTING ADDRESS
                                ; OF OFFENDING MODULE WHEN OPERATION COMPLETE.
                                ; BE.OFPC WILL CONTAIN OFFSET PC INTO A MODULE.
                                ; IF NOT A MODULE
                                ; INCREMENT # OF SYS ERRORS CAUSED BY MONITOR AND IF TOO MANY,SET FATAL
                                ; ERROR BIT IN ERROR WORD, AND SET BE.OFPC TO 0.
                                ;-

                                LET R1 := DT.MLST(R0)
(4)  000354' 016001  000032                                                     MOV     DT.MLST(R0),R1
                                ;+
                                ; GET THE LOWEST ADDRESS IN THE MODULE LIST
                                ;-

                                LET R4 := (R1)
(4)  000360' 011104                                                             MOV     (R1),R4
                                WHILE (R1) NE #0 DO
(4)  000362'                                                            50005$:
(6)  000362' 005711                                                             TST     (R1)
(9)  000364' 001405                                                             BEQ     50006$
                                        IF R4 HI (R1) THEN
(6)  000366' 020411                                                             CMP     R4,(R1)
(9)  000370' 101401                                                             BLOS    50007$
                                                LET R4 := (R1)
(4)  000372' 011104                                                             MOV     (R1),R4
                                        ENDIF
(4)  000374'                                                            50007$:
                                        INLINE <TST     (R1)+>
(2)  000374' 005721                                                             TST     (R1)+
                                ENDDO
(4)  000376' 000771                                                             BR      50005$
(3)  000400'                                                            50006$:
```

```
766
767
768
769  000400'                          IF BE.VPC HIS R4 AND BE.VPC LOS DT.ESIZ(R0) THEN
(6)  000400' 026704  177402                                                        CMP     BE.VPC,R4
(9)  000404' 103430                                                                BLO     50010$
(6)  000406' 026760  177374  000044                                                CMP     BE.VPC,DT.ESIZ(R
(9)  000414' 101024                                                                BHI     50010$
770  000416'                                  CALL MODCHK
(3)  000416' 004767  000500                                                        JSR     PC,MODCHK
771
772                                   ;+
773                                   ; GET THE MODULE ASCII NAME INTO THE MESSAGE
774                                   ;-
775
776  000422'                                  LET R3 := #5
(4)  000422' 012703  000005                                                        MOV     #5,R3
777  000426'                                  LET R2 := #BE.AMOD
(4)  000426' 012702  000226'                                                       MOV     #BE.AMOD,R2
778  000432'                                  WHILE R3 NE #0 DO
(4)  000432'                                                                50011$:
(6)  000432' 005703                                                                TST     R3
(9)  000434' 001403                                                                BEQ     50012$
779  000436'                                          LET (R2)+ :B= (R1)+
(4)  000436' 112122                                                                MOVB    (R1)+,(R2)+
780  000440'                                          LET R3 := R3 - #1
(6)  000440' 005303                                                                DEC     R3
781  000442'                                  ENDDO
(4)  000442' 000773                                                                BR      50011$
(3)  000444'                                                                50012$:
782                                   ;+
783                                   ; CONVERT OFFSET PC TO ASCII
784                                   ;-
785  000444'                                  CALL BOA16 IN <BE.OFPC,#BE.AOFPC>
(3)  000444' 010546                                                                MOV     R5,-(SP)
(5)  000446' 012745  000234'                                                       MOV     #BE.AOFPC,-(R5)
(4)  000452' 016745  177334                                                        MOV     BE.OFPC,-(R5)
(3)  000456' 004767  000000G                                                       JSR     PC,BOA16
(3)  000462' 012605                                                                MOV     (SP)+,R5
786  000464'                          ELSE
(4)  000464' 000413                                                                BR      50013$
(3)  000466'                                                                50010$:
787
788
789                                   ;+
790                                   ; IF NOT A MODULE INCREMENT THE MONITOR SYS ERROR COUNT
791                                   ;-
792
793  000466'                                  LET BE.MCNT := BE.MCNT + #1
(6)  000466' 005267  177330                                                        INC     BE.MCNT
794                                   ;+
795                                   ; SET OFFSET PC TO ZERO (FOR LATER USE)
796                                   ;-
797
798  000472'                                  LET BE.OFPC := #0
(4)  000472' 005067  177314                                                        CLR     BE.OFPC
```

```
                                ;+
                                ; IF TOO MANY SYS ERRORS CAUSED BY MONITOR SET FATAL BIT IN ERROR WORD
                                ;-
000476'                                 IF BE.MCNT HIS BE.MX2 THEN
(6)  000476' 026767  177320  177314                                                    CMP     BE.MCNT,BE.MX2
(9)  000504' 103403                                                                    BLO     50014$
000506'                                         LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6)  000506' 052760  100000  000020                                                    BIS     #FATERR,DT.ERR(R
000514'                                 ENDIF
(4)  000514'                                                                        50014$:
000514'                         ENDIF
(4)  000514'                                                                        50013$:


                                ;+
                                ; CONVERT VECTOR TO ASCII
                                ;-

000514'                         CALL BOA16 IN <BE.VEC,#BE.AVEC>
(3)  000514' 010546                                                                    MOV     R5,-(SP)
(5)  000516' 012745  000140'                                                           MOV     #BE.AVEC,-(R5)
(4)  000522' 016745  177262                                                            MOV     BE.VEC,-(R5)
(3)  000526' 004767  000000G                                                           JSR     PC,BOA16
(3)  000532' 012605                                                                    MOV     (SP)+,R5


                                ;+
                                ; CONVERT VPC TO ASCII
                                ;-

000534'                         CALL BOA16 IN <BE.VPC,#BE.AVPC>
(3)  000534' 010546                                                                    MOV     R5,-(SP)
(5)  000536' 012745  000150'                                                           MOV     #BE.AVPC,-(R5)
(4)  000542' 016745  177240                                                            MOV     BE.VPC,-(R5)
(3)  000546' 004767  000000G                                                           JSR     PC,BOA16
(3)  000552' 012605                                                                    MOV     (SP)+,R5


                                ;+
                                ; CONVERT PHYSICAL ADDRESS TO ASCII
                                ;-

000554'                         LET BE.VA := BE.VPC
(4)  000554' 016767  177226  177216                                                    MOV     BE.VPC,BE.VA
000562'                         CALL GPA IN <R0,#BE.VA>
(3)  000562' 010546                                                                    MOV     R5,-(SP)
(5)  000564' 012745  000000'                                                           MOV     #BE.VA,-(R5)
```

```
   (4)  000570' 010045                                                  MOV     R0,-(R5)
   (3)  000572' 004767  000000G                                         JSR     PC,GPA
   (3)  000576' 012605                                                  MOV     (SP)+,R5
   837  000600'                        CALL BOAC IN <R0,BE.PA,BE.EA,#BE.APPC>
   (3)  000600' 010546                                                  MOV     R5,-(SP)
   (7)  000602' 012745  000160'                                         MOV     #BE.APPC,-(R5)
   (6)  000606' 016745  177172                                          MOV     BE.EA,-(R5)
   (5)  000612' 016745  177164                                          MOV     BE.PA,-(R5)
   (4)  000616' 010045                                                  MOV     R0,-(R5)
   (3)  000620' 004767  000000G                                         JSR     PC,BOAC
   (3)  000624' 012605                                                  MOV     (SP)+,R5
   838
   839                                ;+
   840                                ; CONVERT PSW TO ASCII
   841                                ;-
   842
   843  000626'                        CALL BOA16 IN <BE.PSW,#BE.APSW>
   (3)  000626' 010546                                                  MOV     R5,-(SP)
   (5)  000630' 012745  000172'                                         MOV     #BE.APSW,-(R5)
   (4)  000634' 016745  177154                                          MOV     BE.PSW,-(R5)
   (3)  000640' 004767  000000G                                         JSR     PC,BOA16
   (3)  000644' 012605                                                  MOV     (SP)+,R5
   844
   845
   846
   847                                ;+
   848                                ; CONVERT STACK TO ASCII
   849                                ;-
   850
   851  000646'                        CALL BOA16 IN <BE.STK,#BE.ASTK>
   (3)  000646' 010546                                                  MOV     R5,-(SP)
   (5)  000650' 012745  000202'                                         MOV     #BE.ASTK,-(R5)
   (4)  000654' 016745  177136                                          MOV     BE.STK,-(R5)
   (3)  000660' 004767  000000G                                         JSR     PC,BOA16
   (3)  000664' 012605                                                  MOV     (SP)+,R5
   852
   853                                ;+
   854                                ; CONVERT SYS ERROR COUNT TO ASCII
   855                                ;-
   856
   857  000666'                        CALL BDACNV IN <DT.EXS(R0),#BE.AEXS>
   (3)  000666' 010546                                                  MOV     R5,-(SP)
   (5)  000670' 012745  000212'                                         MOV     #BE.AEXS,-(R5)
   (4)  000674' 016045  000060                                          MOV     DT.EXS(R0),-(R5)
   (3)  000700' 004767  000000G                                         JSR     PC,BDACNV
   (3)  000704' 012605                                                  MOV     (SP)+,R5
   858
   859
   860                                ;+
   861                                ; RESET THE WORLD TO STOP ANY INTERRUPTS
   862                                ;-
   863
   864  000706' 000005                RESET
   865
   866
   867
```

```
                                ;+
                                ; RECOVER THE BAD THINGS THAT THE RESET DID
                                ; AND GET THE PRIORITY BACK DOWN TO ZERO SO WE CAN TYPE
                                ;-


000710'                         CALL RSTRCY IN <R0>
(3) 000710' 010546                                                  MOV     R5,-(SP)
(4) 000712' 010045                                                  MOV     R0,-(R5)
(3) 000714' 004767  000000G                                         JSR     PC,RSTRCY
(3) 000720' 012605                                                  MOV     (SP)+,R5
000722'                         LET -(SP) := #0
(4) 000722' 005046                                                  CLR     -(SP)
000724'                         LET -(SP) := #456$
(4) 000724' 012746  000732'                                         MOV     #456$,-(SP)

000730'                         INLINE <RTI>
(2) 000730' 000002                                                  RTI
000732'                         INLINE <456$:>
(2) 000732'                                                         456$:


                                ;+
                                ; QUEUE THE SYSTEM ERROR MESSAGE (STANDARD SYSTEM ERROR)
                                ;-

000732'                         CALL MSGDHOOK IN <R0,#MSGPOP,#BE.SEMSG,#2$>
(3) 000732' 010546                                                  MOV     R5,-(SP)
(7) 000734' 012745  000764'                                         MOV     #2$,-(R5)
(6) 000740' 012745  000030'                                         MOV     #BE.SEMSG,-(R5)
(5) 000744' 012745  000002                                          MOV     #MSGPOP,-(R5)
(4) 000750' 010045                                                  MOV     R0,-(R5)
(3) 000752' 004767  000000G                                         JSR     PC,MSGDHOOK
(3) 000756' 012605                                                  MOV     (SP)+,R5

000760'                         INLINE <1$:     NOP>
(2) 000760' 000240                                                  1$:     NOP
000762'                         INLINE <BR 1$>
(2) 000762' 000776                                                  BR 1$


                                ;+
                                ; IF ERROR WAS CAUSED BY OPTION MODULE, ALSO QUEUE THE OPTION
                                ; MODULE NAME AND APC
                                ;-

000764'                         2$: IF BE.OFPC NE #0 THEN
(6) 000764' 005767  177022                                          TST     BE.OFPC
(9) 000770' 001415                                                  BEQ     50015$
000772'                                 CALL MSGDHOOK IN <R0,#MSGPOP,#BE.ATMSG,#4$>
(3) 000772' 010546                                                  MOV     R5,-(SP)
(7) 000774' 012745  001024'                                         MOV     #4$,-(R5)
(6) 001000' 012745  000222'                                         MOV     #BE.ATMSG,-(R5)
(5) 001004' 012745  000002                                          MOV     #MSGPOP,-(R5)
```

```
   (4)  001010' 010045                                                                  MOV     R0,-(R5)
   (3)  001012' 004767  000000G                                                         JSR     PC,MSGDHOOK
   (3)  001016' 012605                                                                  MOV     (SP)+,R5
   901  001020'                          INLINE <3$:     NOP>
   (2)  001020' 000240                                                                  3$:     NOP
   902  001022'                          INLINE <BR 3$>
   (2)  001022' 000776                                                                  BR 3$
   903  001024'                      4$:
   904  001024'                      ENDIF
   (4)  001024'                                                                 50015$:
   905
   906                               ;+
   907                               ; IF AN ERROR LOGGING CPU, CALL THE LOG ROUTINE
   908                               ;-
   909
   910  001024'                      IF #PDP60 SETIN DT.CF0(R0) OR #PDP70 SETIN DT.CF0(R0) THEN
   (6)  001024' 032760  004000  000014                                                 BIT     #PDP60,DT.CF0(R0
   (8)  001032' 001004                                                                 BNE     50016$
   (6)  001034' 032760  010000  000014                                                 BIT     #PDP70,DT.CF0(R0
   (9)  001042' 001405                                                                 BEQ     50017$
   (6)  001044'                                                                 50016$:
   911  001044'                          CALL ERRLOG IN <R0>
   (3)  001044' 010546                                                                  MOV     R5,-(SP)
   (4)  001046' 010045                                                                  MOV     R0,-(R5)
   (3)  001050' 004767  000000G                                                         JSR     PC,ERRLOG
   (3)  001054' 012605                                                                  MOV     (SP)+,R5
   912  001056'                      ENDIF
   (4)  001056'                                                                 50017$:
   913
   914
   915                               ;+
   916                               ; BACK UP R1 TO POINT TO MODULE HEADER ADDRESS
   917                               ;-
   918
   919  001056'                      LET R1 := R1 - #5
   (6)  001056' 162701  000005                                                          SUB     #5,R1
   920
   921
   922                               ;+
   923                               ; IF ERROR CAUSED BY OPTION MODULE AND LIMIT REACHED, DROP THE MODULE
   924                               ;-
   925
   926  001062'                      IF BE.OFPC NE #0 AND SYSCNT(R1) HIS BE.MAX THEN
   (6)  001062' 005767  176724                                                          TST     BE.OFPC
   (9)  001066' 001412                                                                 BEQ     50020$
   (6)  001070' 026167  000052  176730                                                 CMP     SYSCNT(R1),BE.MA
   (9)  001076' 103406                                                                 BLO     50020$
   927  001100'                          CALL DRPMOD IN <R0,R1>
   (3)  001100' 010546                                                                  MOV     R5,-(SP)
   (5)  001102' 010145                                                                  MOV     R1,-(R5)
   (4)  001104' 010045                                                                  MOV     R0,-(R5)
   (3)  001106' 004767  000000G                                                         JSR     PC,DRPMOD
   (3)  001112' 012605                                                                  MOV     (SP)+,R5
   928  001114'                      ENDIF
   (4)  001114'                                                                 50020$:
   929
```

```
  930
  931
  932                                   ;+
  933                                   ; RESTORE REGISTERS
  934                                   ;-
  935
  936  001114'                          CALL RESREG
  (3)  001114' 004767  000000G                                          JSR     PC,RESREG
  937
  938
  939  001120'                          ENDRTN
  (3)  001120'                                                  50000$:
  (3)  001120'                                                  50001$:
  (2)  001120' 000207                                           RTS     PC
```

```
941                             .SBTTL  MODCHK GET OPTION MODULE HEADER ADDRESS AND OFFSET
942                             .IDENT /V0.0/
943
944                             ; FUNCTIONAL DESCRIPTION:
945                             ;
946                             ;       THIS ROUTINE IS ENTERED TO DETERMINE THE OPTION MODULE
947                             ;       WHICH CONTAINS THE ADDRESS AT WHICH A BUS ERROR OR RESERVED
948                             ;       INSTRUCTION OCCURRED. IT WILL RETURN THE HEADER ADDRESS
949                             ;       OF THE OPTION MODULE, INCREMENT THE SYS ERROR COUNT FOR THE
950                             ;       MODULE AND IF TOO MANY SYS ERRORS HAVE OCCURRED,DROP THE MODULE
951                             ;       FROM AN EXERCISER. IT WILL ALSO RETURN THE ADDRESS
952                             ;       AT WHICH THE SYS ERROR OCCURRED AS AN OFFSET INTO THE MODULE.
953                             ;       THIS OFFSET IS ALSO KNOWN AS THE ASSEMBLED PC OR APC.
954                             ;
955
956
```

```
                                .SBTTL  MODCHK ROUTINE

001122'                         ROUTINE MODCHK
(2) 001122'                                                             MODCHK:

                                ;+
                                ; GET ADDRESS OF OPTION MODULE LIST,PUT IN R2
                                ; GET ADDRESS AT WHICH SYS ERROR OCCURRED INTO R3
                                ;-

001122'                         LET R2 := DT.MLST(R0)
(4) 001122' 016002  000032                                                      MOV     DT.MLST(R0),R2
001126'                         LET R3 := DT.PC(R0)
(4) 001126' 016003  000002                                                      MOV     DT.PC(R0),R3

                                ;+
                                ; INITIALIZATION COMPLETE
                                ;-


                                ;+
                                ; FIND THE SMALLEST ADDRESS THAT DOES NOT EXCEED THE
                                ; ADDRESS OF SYS ERROR
                                ;-

001132'                         LET R1 := #0
(4) 001132' 005001                                                              CLR     R1
001134'                         WHILE (R2) NE #0 DO
(4) 001134'                                                             50002$:
(6) 001134' 005712                                                              TST     (R2)
(9) 001136' 001407                                                              BEQ     50003$
001140'                                 IF (R2) LOS R3 THEN
(6) 001140' 021203                                                              CMP     (R2),R3
(9) 001142' 101003                                                              BHI     50004$
001144'                                         IF (R2) HI R1 THEN
(6) 001144' 021201                                                              CMP     (R2),R1
(9) 001146' 101401                                                              BLOS    50005$
001150'                                                 LET R1 := (R2)
(4) 001150' 011201                                                              MOV     (R2),R1
001152'                                         ENDIF
(4) 001152'                                                             50005$:
001152'                                 ENDIF
(4) 001152'                                                             50004$:
001152'                                 INLINE <TST     (R2)+>
(2) 001152' 005722                                                              TST     (R2)+
001154'                         ENDDO
(4) 001154' 000767                                                              BR      50002$
(3) 001156'                                                             50003$:


                                ;+
                                ; CALCULATE THE OFFSET ADDRESS
                                ;-
```

```
  997  001156'                        LET BE.OFPC := R3 - R1
  (4)  001156' 010367  176630                                                  MOV     R3,BE.OFPC
  (6)  001162' 160167  176624                                                  SUB     R1,BE.OFPC
  998
  999                                 ;+
 1000                                 ; ADD ONE TO OPTION MODULE'S SYS ERROR COUNT
 1001                                 ;-
 1002
 1003  001166'                        LET SYSCNT(R1) := SYSCNT(R1) + #1
  (6)  001166' 005261  000052                                                  INC     SYSCNT(R1)
 1004
 1005
 1006
 1007
 1008
 1009
 1010  001172'                        ENDRTN
  (3)  001172'                                                        50000$:
  (3)  001172'                                                        50001$:
  (2)  001172' 000207                                                          RTS     PC
 1011          000001                 .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BDACNV= ****** G
BE.AEX  000212R
BE.AMO  000226R
BE.AOF  000234R
BE.APP  000160R
BE.APS  000172R
BE.AST  000202R
BE.ATM  000222R
BE.AVE  000140R
BE.AVP  000150R
BE.EA   000004R
BE.MAX  000026R
BE.MCN  000022RG
BE.MX2  000020R
BE.MX3  000024R
BE.OFP  000012R
BE.PA   000002R
BE.PSW  000014R
BE.SEM  000030R
BE.STK  000016R
BE.VA   000000R
BE.VEC  000010R
BE.VPC  000006R
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BOAC   = ****** G
BOA16  = ****** G
BUSERR  000244RG
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRLOG= ****** G
ERRTYP= 000105
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GPA    = ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODCHK  001122R
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPDP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRCY= ****** G
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
```

```
SC.CKL= 000002  SVR6  = 000076  WBUFPA= 000134  $F$OR = 000320  $TSK0 = 050002
SC.CKP= 000004  SYSCNT= 000052  WBUFRQ= 000140  $F$RTI= 000350  $TSK1 = 050003
SC.CLO= 000000  SYSERR= 000100  WBUFSZ= 000142  $F$RTN= 000300  $TSK2 = 050004
SC.HLD= 000010  TABADR= 000000  WDFR  = 000116  $F$SEL= 000140  $TSK3 = 050005
SC.SCA= 000012  TMPIO = 000002  WDTO  = 000114  $F$THE= 000330  $$ARGC= 000000
SENDLS= 177777  TQOVF = 000002  WTINRE= 000352  $F$TRU= 000404  $$BYTE= 000403
SOFCNT= 000042  UIPAR0= 177640  WTWHMI= 000222  $F$UNT= 000130  $$CASE= 000000
SOFPAS= 000046  UIPAR1= 177642  XFLAG = 000005  $F$WHI= 000120  $$DST = 000000
SPACE = 000040  UIPAR2= 177644  XOFF  = 000023  $F$YES= 000402  $$ELOC= 000402
SPOINT= 000032  UIPAR3= 177646  XON   = 000021  $IFLEV= 177777  $$ERFL= 000000
SPVALU= 002200  UIPAR4= 177650  $BGNLE= 177777  $ISK0 = 000001  $$FLAG= 000001
SR0   = 177572  UIPAR5= 177652  $ERFLG= 000400  $ISK1 = 000001  $$FROM= 000000
SR1   = 177574  UIPAR6= 177654  $F$AND= 000310  $LOCTA= 177777  $$LOC = 001146R
SR2   = 177576  UIPAR7= 177656  $F$BAD= 000401  $LSTIN= 000001  $$LOCN= 000000
SR3   = 172516  UIPDR0= 177600  $F$BLA= 000170  $LSTTA= 000001  $$REG = 177777
STAT  = 000026  UIPDR1= 177602  $F$CAS= 000150  $NESTL= 177777  $$RETU= 000000
STATBI= 064757  UIPDR2= 177604  $F$DEC= 000220  $NSK0 = 000300  $$RTN1= 050000
STAT1 = 000027  UIPDR3= 177606  $F$DO  = 000340  $NSK1 = 000120  $$RTN2= 050001
SUSPND= 000001  UIPDR4= 177610  $F$FAL= 000405  $NSK2 = 000110  $$SRC = 000000
SVR0  = 000062  UIPDR5= 177612  $F$GOO= 000400  $NSK3 = 000110  $$TGSV= 000000
SVR1  = 000064  UIPDR6= 177614  $F$IF  = 000110  $SAVLE= 177777  $$TGS1= 000000
SVR2  = 000066  UIPDR7= 177616  $F$INC= 000210  $SSK0 = 050003  $$TGS2= 000000
SVR3  = 000070  WASADR= 000104  $F$LOO= 000200  $TAGLE= 177777  $$TO  = 000000
SVR4  = 000072  WBSTAT= 000040  $F$NAM= 000160  $TAGNU= 050006  $$$TAG= 050000
SVR5  = 000074  WBUFEA= 000136  $F$NO  = 000403  $TEMP = 000300  .     = 001174R

. ABS.  000000  000
        001174  001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BESRV,DSKZ:BESRV=SPMAC/ML,EQUATE,BESRV
 RUN-TIME: 20 10 .4 SECONDS
 RUN-TIME RATIO: 54/32=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE  BLOG60 BINDER FOR 11/60 ERROR LOG ROUTINES

                                .IDENT /V0.0/

                                ;++
                                ; MODULE PACKAGE NAME:
                                ;       BLOG60
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS IS A BINDER TO KEEP 11/60 ERROR LOG MODULES PACKAGED
                                ;       TOGETHER.
                                ;       CONTAINS THE FOLLOWING MODULES:
                                ;                 IERRLO - INITIALIZE 11/60 ERROR LOG
                                ;                 ERRLOG  - 11/60 ERROR LOG
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
          000001                $LSTIN = 1
          000001                $LSTTAG = 1


                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  IERRLO                  ;MODULE ENTRY POINT
                                .GLOBL  ERRLOG                  ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SAVE REGISTER ROUTINE
                                .GLOBL  BOA16                   ;BINARY TO OCTAL ASCII CONVERSION (16 BIT)
                                .GLOBL  MSGDHOOK                ;MESSAGE HOOK ROUTINE
                                .GLOBL  RESREG                  ;RESTORE REGISTERS

                                ;****************
                                ; LOCAL EQUATES
                                ;
          000100                L6.OFF  = 100                   ;OFFSET INTO ERROR LOG REGISTERS
          100001                L6.ENB  = 100001                ;ENABLE AND FREEZE ON FIRST ERROR
          000010                L6.STO  = ^D<8>                 ;SIZE OF LOG BUFFER

                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 000010                  L6.BUF: .BLKW   L6.STO          ;LOG BUFFER FOR 11/60 ERROR LOG
000020' 000036'                 L6.TAB: .WORD   L6.JAM          ;TABLE POINTING TO LOCAL STORAGE FOR
000022' 000104'                         .WORD   L6.SRV          ;MESSAGE
000024' 000122'                         .WORD   L6.PBA
000026' 000140'                         .WORD   L6.CUA
000030' 000163'                         .WORD   L6.FLG
000032' 000203'                         .WORD   L6.WHA
000034' 000223'                         .WORD   L6.CDA
000036' 000246'                         .WORD   L6.CTA

                                ;****************
                                ; LOCAL MESSAGES
                                ;
000040'                         L6.MSG:
000040' 030445  027461  030066          .ASCII  '%11/60 ERROR LOG%'
000046' 042440  051122  051117
000054' 046040  043517     045
000061'    112  046501  020057          .ASCII  'JAM/ '
000066' 000006                  L6.JAM: .BLKB   ^D<6>
000074' 020040  051440  053122          .ASCII  '   SRV/ '
000102' 020057
```

```
581  000104' 000006                      L6.SRV: .BLKB   ^D<6>
582  000112' 020040  050040  040502              .ASCII  '   PBA/ '
     000120' 020057
583  000122' 000006                      L6.PBA: .BLKB   ^D<6>
584  000130' 020040  041440  040525              .ASCII  '   CUA/ '
     000136' 020057
585  000140' 000006                      L6.CUA: .BLKB   ^D<6>
586  000146' 020045  020040  046106              .ASCII  '%   FLG-INT/ '
     000154' 026507  047111  027524
     000162'    040
587  000163' 000006                      L6.FLG: .BLKB   ^D<6>
588  000171'    040  020040  044127              .ASCII  '   WHAMI/ '
     000176' 046501  027511     040
589  000203' 000006                      L6.WHA: .BLKB   ^D<6>
590  000211'    040  020040  042103              .ASCII  '   CDATA/ '
     000216' 052101  027501     040
591  000223' 000006                      L6.CDA: .BLKB   ^D<6>
592  000231'    040  020040  052103              .ASCII  '   CTAG-CPU/ '
     000236' 043501  041455  052520
     000244' 020057
593  000246' 000006                      L6.CTA: .BLKB   ^D<6>
594  000254' 000045                              .ASCIZ  '%'
595                                              .EVEN
```

```
                        .SBTTL IERRLO INITIALIZE 11/60 ERROR LOG
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       IERRLO
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL INITIALIZE THE 11/60 ERROR LOG TO THE ENABLE
                        ;       AND FREEZE ON FIRST ERROR STATES.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL IERRLO
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL INITIALIZE 11/60 ERROR LOG ROUTINE

000256'                         ROUTINE IERRLO
(2) 000256'                                                             IERRLO:

                                ;+
                                ; SAVE REGISTER R0, WHEN EXECUTING A "MED" INSTRUCTION R0 IS
                                ; USED TO TRANSFER DATA TOC AND FROM WHAMI REGISTER.
                                ;-

000256'                         PUSH R0
(2) 000256' 010046                                                              MOV     R0,-(SP)

                                ;+
                                ; READ WHAMI REGISTER THEN SET THE ENABLE BIT INTO
                                ; R0 AND THEN WRITE BACK TO WHAMI REGISTER
                                ;-

000260' 076600                  MED
000262' 000022                  RDWHMI

000264'                         LET R0 := R0 SET.BY #L6.ENB
(6) 000264' 052700  100001                                                      BIS     #L6.ENB,R0

000270' 076600                  MED
000272' 000222                  WTWHMI

                                ;+
                                ; RESTORE REGISTER AND RETURN
                                ;-

000274'                         POP R0
(2) 000274' 012600                                                              MOV     (SP)+,R0

000276'                         ENDRTN
(3) 000276'                                                             50000$:
(3) 000276'                                                             50001$:
(2) 000276' 000207                                                              RTS     PC
```

```
                        .SBTTL ERRLOG 11/60 ERROR LOGGING ROUTINE
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       ERRLOG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL READ THE 11/60 PROCESSOR ERROR LOG SCRATCHPAD AND
                        ;       TRANSFER THE ERROR LOG TO PHYSICAL CORE AND THEN GENERATE A
                        ;       MESSAGE THAT WILL OUTPUT 8 OF THE SCRATCHPAD REGISTERS (100-107).
                        ;       THESE REGISTERS ARE IDENTIFIED AS FOLLOWS:
                        ;                 JAM - JAM REGISTER STATUS
                        ;                 SRV - SERVICE REGISTER OF STATUS
                        ;                 PBA - PHYSICAL BUS ADDRESS REGISTER (BIT 17,16)
                        ;                 CUA - MICROPROGRAM ADDRESS
                        ;                 FLG/INT - FLAG REQUEST REGISTER OF STATUS/LAST VECTOR SERVICED
                        ;                 WHAMI - VARIETY OF PROCESSOR OPTION STATUS BITS, PROCESSOR
                        ;                    OPTION STATUS
                        ;                 CDATA - CACHE DATA
                        ;                 CTAG/CPU- CACHE TAG DATA/HIT REGISTER
                        ;
                        ; INPUTS:
                        ;       DTABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       BOA16 - BINARY TO OCTAL ASCII CONVERSION (16 BITS)
                        ;       MSGDHOOK - MESSAGE HOOK ROUTINE
                        ;       RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ERRLOG IN <A>
                        ;                 A -- ADDRESS OF DTABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL ERRLOG 11/60 ERROR LOG ROUTINE

000300'                         ROUTINE ERRLOG <DTABLE>
(2) 000300'                                                             ERRLOG:


                                ;+
                                ; SAVE REGISTERS AND GET ADDRESS OF LOG BUFFER ALONG WITH INITIALIZING
                                ; LOG COMMAND AND ADDRESS TO ERROR LOG REGISTER 100
                                ;-

000300'                         CALL SAVREG
(3) 000300' 004767  000000G                                             JSR     PC,SAVREG
000304'                         LET R1 := #L6.BUF
(4) 000304' 012701  000000'                                             MOV     #L6.BUF,R1
000310'                         LET R2 := #L6.OFF
(4) 000310' 012702  000100                                              MOV     #L6.OFF,R2

                                ;+
                                ; NOW READ ERROR LOG WORD AND LOAD IT INTO LOG BUFFER UNTIL
                                ; ERROR LOG IS SAVED.
                                ;-

000314'                         WHILE R2 LT #L6.OFF+#L6.STO DO
(4) 000314'                                                             50002$:
(6) 000314' 020227  000110                                              CMP     R2,#L6.OFF+#L6.S
(9) 000320' 002007                                                      BGE     50003$

000322'                                 LET 1$ := R2
(4) 000322' 010267  000002                                              MOV     R2,1$
000326'                         INLINE  <MED>
(2) 000326' 076600                                                      MED
000330'                         INLINE <1$:     .WORD   0>
(2) 000330' 000000                                                      1$:     .WORD   0

                                ;+
                                ; LOAD R0  (DATA FROM LOG) INTO LOG BUFFER AND POINT LOG BUFFER
                                ; TO NEXT ENTRY AND R2 TO NEXT COMMAND AND ADDRESS
                                ;-

000332'                                 LET (R1)+ := R0
(4) 000332' 010021                                                      MOV     R0,(R1)+
000334'                                 LET R2 := R2 + #1
(6) 000334' 005202                                                      INC     R2

000336'                         ENDDO
(4) 000336' 000766                                                      BR      50002$
(3) 000340'                                                             50003$:

                                ;+
                                ; POINT TO "FP.LOG JAM" ENTRY IN LOG BUFFER AND POINT TO INDEX TABLE, CONVERT TO OCTAL
                                ; ASCII AND LOAD MESSAGE ALSO DO THE SAME TO FOLLOWING 7 ENTRIES IN LOG BUFFER.
                                ;-
```

```
  773  000340'                        LET R2 := #0
  (4)  000340' 005002                                                    CLR     R2
  774  000342'                        LET R0 := #L6.BUF
  (4)  000342' 012700  000000'                                          MOV     #L6.BUF,R0
  775  000346'                        LET R1 := #L6.TAB
  (4)  000346' 012701  000020'                                          MOV     #L6.TAB,R1
  776  000352'                        WHILE R2 NE #L6.STO DO
  (4)  000352'                                                  50004$:
  (6)  000352' 020227  000010                                           CMP     R2,#L6.STO
  (9)  000356' 001410                                                   BEQ     50005$
  777  000360'                                CALL BOA16 IN <(R0)+,(R1)+>
  (3)  000360' 010546                                                   MOV     R5,-(SP)
  (5)  000362' 012145                                                   MOV     (R1)+,-(R5)
  (4)  000364' 012045                                                   MOV     (R0)+,-(R5)
  (3)  000366' 004767  000000G                                         JSR     PC,BOA16
  (3)  000372' 012605                                                   MOV     (SP)+,R5
  778  000374'                                INLINE <TSTB (R2)+>
  (2)  000374' 105722                                                   TSTB (R2)+
  779  000376'                        ENDDO
  (4)  000376' 000765                                                   BR      50004$
  (3)  000400'                                                  50005$:
  780
  781                                 ;+
  782                                 ; OUTPUT MESSAGE
  783                                 ;-
  784
  785  000400'                        LET R0 := DTABLE(R5)
  (4)  000400' 016500  000000                                           MOV     DTABLE(R5),R0
  786  000404'                        CALL MSGDHOOK IN <R0,#MSGPOP,#L6.MSG,#3$>
  (3)  000404' 010546                                                   MOV     R5,-(SP)
  (7)  000406' 012745  000436'                                          MOV     #3$,-(R5)
  (6)  000412' 012745  000040'                                          MOV     #L6.MSG,-(R5)
  (5)  000416' 012745  000002                                           MOV     #MSGPOP,-(R5)
  (4)  000422' 010045                                                   MOV     R0,-(R5)
  (3)  000424' 004767  000000G                                         JSR     PC,MSGDHOOK
  (3)  000430' 012605                                                   MOV     (SP)+,R5
  787  000432'                        INLINE <2$:     NOP>
  (2)  000432' 000240                                           2$:     NOP
  788  000434'                        INLINE <BR      2$>
  (2)  000434' 000776                                                   BR      2$
  789
  790                                 ;+
  791                                 ; RESTORE REGISTERS AND RETURN
  792                                 ;-
  793
  794  000436'                        INLINE <3$:>
  (2)  000436'                                                  3$:
  795  000436'                        CALL RESREG
  (3)  000436' 004767  000000G                                         JSR     PC,RESREG
  796
  797  000442'                        ENDRTN
  (3)  000442'                                                  50000$:
  (3)  000442'                                                  50001$:
  (2)  000442' 000207                                                   RTS     PC
  798          000000                 .END
```

```
ACSR   = 000102          CSRC   = 000102          ENDLST= 000000          L6.CTA  000246R          PRI0   = 000000
ACTBIT= 004000          CTRLC  = 000003          EOPBIT= 000001          L6.CUA  000140R          PRI1   = 000040
ADDR22= 001000          CTRLO  = 000017          ERRLOG  000300RG        L6.ENB= 100001          PRI4   = 000200
ADR    = 000006          CTRLU  = 000025          ERRTYP= 000105          L6.FLG  000163R          PRI5   = 000240
APTER= 000004          DCEVNT= 000011          EVNTBE= 000200          L6.JAM  000066R          PRI6   = 000300
APTPRE= 000200          DEFRTN= 000400          EVNTHD= 000200          L6.MSG  000040R          PRI7   = 000340
ASB    = 000106          DIAGMC= 000000          EVNTKT= 000203          L6.OFF= 000100          PR0    = 000000
ASSEMB= 000010          DROPMO= 100000          EVNTPE= 000202          L6.PBA  000122R          PR4    = 000200
ASTAT  = 000104          DSEVNT= 000014          EVNTRE= 000201          L6.SRV  000104R          PR5    = 000240
AUTO   = 000010          DTABLE= 000000          FATERR= 100000          L6.STO= 000010          PR6    = 000300
AUTOST= 020000          DT.ADD= 000042          HRDCNT= 000044          L6.TAB  000020R          PR7    = 000340
AWAS   = 000110          DT.AP  = 000100          HRDPAS= 000050          L6.WHA  000203R          PS     = 177776
BIT0   = 000001          DT.APK= 000076          ICONT  = 000036          MAPSTA= 000200          PSW    = 177776
BIT00  = 000001          DT.BLS= 000034          ICOUNT= 000040          MED    = 076600          RANNUM= 000054
BIT01  = 000002          DT.CF0= 000014          IDNUM  = 000122          MEMPAS= 040000          RBUFEA= 000130
BIT02  = 000004          DT.CF1= 000016          IE     = 000100          MODEXH= 004000          RBUFPA= 000126
BIT03  = 000010          DT.ERR= 000020          IERRLO  000256RG        MODHOL= 002000          RBUFSZ= 000132
BIT04  = 000020          DT.ESI= 000044          INDPAR= 000040          MODSEL= 001000          RBUFVA= 000124
BIT05  = 000040          DT.EVN= 000000          INHDRP= 040000          MSGCKD= 000010          RDSERV= 000101
BIT06  = 000100          DT.EXS= 000060          INHEPR= 020000          MSGCKS= 000011          RDWHMI= 000022
BIT07  = 000200          DT.FCH= 000037          INHREL= 001000          MSGDER= 000005          RELERR= 000020
BIT08  = 000400          DT.FCN= 000036          INHRRE= 000400          MSGDHO= ****** G        RELMOD= 020000
BIT09  = 001000          DT.HMX= 000104          INIT   = 000030          MSGDRP= 000017          RELTIM= 010000
BIT1   = 000002          DT.KBE= 000024          INTR   = 000120          MSGECH= 177777          RESREG= ****** G
BIT10  = 002000          DT.KBP= 000026          IOMOD  = 100000          MSGEOP= 000013          RES1   = 000056
BIT11  = 004000          DT.KBR= 000022          IOMODP= 102000          MSGHDR= 000004          RES2   = 000060
BIT12  = 010000          DT.KBU= 000030          IOMODR= 112000          MSGHNG= 000022          RICHAR= 031060
BIT13  = 020000          DT.MLS= 000032          IOMODX= 110000          MSGHRD= 000007          RPTDAT= 002000
BIT14  = 040000          DT.MTI= 000110          JACK   = 035060          MSGMAP= 000021          RSTRT  = 000112
BIT15  = 100000          DT.OFF= 000070          KIPAR0= 172340          MSGNUL= 177775          RUBOUT= 000177
BIT2   = 000004          DT.PAS= 000074          KIPAR1= 172342          MSGPOP= 000002          RUNMOD= 100000
BIT3   = 000010          DT.PC  = 000002          KIPAR2= 172344          MSGPRM= 177776          R5VALU= 001740
BIT4   = 000020          DT.PFL= 000062          KIPAR3= 172346          MSGRES= 000001          SAM    = 075464
BIT5   = 000040          DT.PSW= 000004          KIPAR4= 172350          MSGSFT= 000006          SAVREG= ****** G
BIT6   = 000100          DT.PTA= 000064          KIPAR5= 172352          MSGSKE= 000003          SBADR  = 000102
BIT7   = 000200          DT.RCS= 000102          KIPAR6= 172354          MSGSMB= 000015          SBKMOD= 000000
BIT8   = 000400          DT.REL= 000040          KIPAR7= 172356          MSGSMH= 000014          SBKSEL= 010000
BIT9   = 001000          DT.SCT= 000066          KIPDR0= 172300          MSGSMS= 000016          SC.ADR= 000006
BKDEF  = 000002          DT.SMX= 000106          KIPDR1= 172302          MSGSTD= 000000          SC.ALC= 000014
BKMOD  = 000020          DT.SP  = 000006          KIPDR2= 172304          MSGSYS= 000012          SC.APC= 000016
BKMODE= 040000          DT.SSI= 000046          KIPDR3= 172306          MSGVEC= 000020          SC.CKL= 000002
BKSLSH= 000134          DT.ST0= 000010          KIPDR4= 172310          NBKMOD= 001000          SC.CKP= 000004
BOA16  = ****** G        DT.ST1= 000012          KIPDR5= 172312          NCPUOP= 000020          SC.CLO= 000000
CAPRES= 000004          DT.SWR= 000056          KIPDR6= 172314          NOAPTY= 000002          SC.HLD= 000010
CASTAT= 000004          DT.SYP= 000072          KIPDR7= 172316          NULL   = 000000          SC.SCA= 000012
CDERCT= 000146          DT.WBU= 000050          KTERRO= 000040          OWEN   = 024020          SENDLS= 177777
CDWDCT= 000144          DT.WHL= 000054          KTPRES= 000400          PAERR  = 000010          SOFCNT= 000042
CKTIM  = 100000          DT.WLL= 000052          KTSTAT= 000020          PARPRE= 002000          SOFPAS= 000046
CLKPRE= 000001          DVID1  = 000014          KTXTND= 040000          PARSTA= 000100          SPACE  = 000040
CONFIG= 000056          ECCMEM= 000100          LF     = 000012          PASCNT= 000034          SPOINT= 000032
CQOVF  = 000001          ECCSTA= 000010          LPSTAT= 000001          PDPLSI= 020000          SPVALU= 002200
CR     = 000015          ENBEOP= 010000          L6.BUF  000000R         PDP60  = 004000          SR0    = 177572
CSRA   = 000100          ENBNUL= 000001          L6.CDA  000223R         PDP70  = 010000          SR1    = 177574
```

```
SR2    = 177576          UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $$ARGC= 000002
SR3    = 172516          UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $$BYTE= 000403
STAT   = 000026          UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $$CASE= 000000
STATBI= 064757           UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $$DST = 000000
STAT1  = 000027          UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $$ELOC= 000000
SUSPND= 000001           UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $$ERFL= 000000
SVR0   = 000062          UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$FLAG= 000340
SVR1   = 000064          UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$FROM= 000000
SVR2   = 000066          UIPDR4= 177610          $F$BLA= 000170          $LOCTA= 177777          $$LOC = 000356R
SVR3   = 000070          UIPDR5= 177612          $F$CAS= 000150          $LSTIN= 000001          $$LOCN= 000000
SVR4   = 000072          UIPDR6= 177614          $F$DEC= 000220          $LSTTA= 000001          $$REG = 177777
SVR5   = 000074          UIPDR7= 177616          $F$DO = 000340          $NESTL= 177777          $$RETU= 000000
SVR6   = 000076          WASADR= 000104          $F$FAL= 000405          $NSK0 = 000300          $$RTN1= 050000
SYSCNT= 000052           WBSTAT= 000040          $F$GOO= 000400          $NSK1 = 000120          $$RTN2= 050001
SYSERR= 000100           WBUFEA= 000136          $F$IF = 000110          $SAVLE= 177777          $$SRC = 000000
TMPIO  = 000002          WBUFPA= 000134          $F$INC= 000210          $SSK0 = 050005          $$TGSV= 000000
TQOVF  = 000002          WBUFRQ= 000140          $F$LOO= 000200          $TAGLE= 177777          $$TGS1= 000000
UIPAR0= 177640           WBUFSZ= 000142          $F$NAM= 000160          $TAGNU= 050006          $$TGS2= 000000
UIPAR1= 177642           WDFR   = 000116         $F$NO = 000403          $TEMP = 000300          $$TO  = 000000
UIPAR2= 177644           WDTO   = 000114         $F$OR = 000320          $TSK0 = 050004          $$$TAG= 050000
UIPAR3= 177646           WTINRE= 000352          $F$RTI= 000350          $TSK1 = 050005          .     = 000444R

. ABS.  000000    000
        000444    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BLOG60,DSKZ:BLOG60=SPMAC/ML,EQUATE,BLOG60
 RUN-TIME: 13 3 .3 SECONDS
 RUN-TIME RATIO: 36/16=2.1
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  BLOG70 BINDER FOR ERRLOG MODULES
                        .IDENT /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       BLOG70
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS IS A BINDER TO KEEP ERRLOG MODULE TOGETHER.
                        ;       CONTAINS FOLLOWING MODULES:
                        ;                 IERRLO - INITIALIZE 11/70 ERROR LOG
                        ;                 ERRLOG  - 11/70 ERROR LOG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN = 1
                000001          $LSTTAG = 1


                                ;******************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  IERRLO                  ;MODULE ENTRY POINT
                                .GLOBL  ERRLOG                  ;MODULE ENNTRY POIUNT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  BOA16                   ;CONVERT TO OCTAL ASCII (16 BITS)
                                .GLOBL  BOAC                    ;CONVERT TO OCTAL ASCII (22 BITS)
                                .GLOBL  MSGDHOOK                ;MESSAGE HOOK ROUTINE
                                .GLOBL  CCNTRL                  ;CONTROL REGISTER (CACHE)
                                .GLOBL  KONTRL                  ;TEMP STORAGE FOR CACHE OPERATION
                                .GLOBL  CPULAR                  ;LOW ERROR ADDRESS REGISTER
                                .GLOBL  CPUHAR                  ;HIGH ERROR ADDRESS REGISTER
                                .GLOBL  CPUMER                  ;MEMORY SYSTEM ERROR REGISTER
                                .GLOBL  CPUMNT                  ;MAINTENANCE REGISTER
                                .GLOBL  CPUCPE                  ;CPU ERROR REGISTER

                                ;****************
                                ; LOCAL MESSAGES
                                ;
000000'                         L7.MSG:
000000' 030445  027461  030067          .ASCII  '%11/70 ERROR LOG'
000006' 042440  051122  051117
000014' 046040  043517
000020' 046445  046505  051105          .ASCII  '%MEMERREG/ '
000026' 042522  027507     040
000033' 000006                  L7.MED: .BLKB   ^D<6>
000041'    040  020040  050103          .ASCII  '   CPUERREG/ '
000046' 042525  051122  043505
000054' 020057
000056' 000006                  L7.CED: .BLKB   ^D<6>
000064' 000045                          .ASCIZ  /%/

000066' 042101  051104  020057  L7.MS1: .ASCII  'ADDR/ '
000074' 000010                  L7.ADR: .BLKB   ^D<8>
000104' 000045                          .ASCIZ  /%/
                                        .EVEN
```

```
574
575                     .SBTTL IERRLO INITIALIZE 11/70 REGISTERS AND ERROR LOG
576                     .IDENT /V0.0/
577
578                     ;++
579                     ; MODULE NAME:
580                     ;       IERRLO
581                     ;
582                     ; FUNCTIONAL DESCRIPTION:
583                     ;       THIS ROUTINE WILL INITIALIZE THE FOLLOWING REGISTERS:
584                     ;                   MEMORY SYSTEM ERROR REGISTER
585                     ;                   CPU ERROR REGISTER
586                     ;                   LOW ERROR ADDRESS REGISTER
587                     ;                   HIGH ERROR ADDRESS REGISTER
588                     ;                   MAINTENANCE REGISTER
589                     ;                   CONTROL REGISTER (CACHE)
590                     ;
591                     ; INPUTS:
592                     ;       NONE
593                     ;
594                     ; IMPLICIT INPUTS:
595                     ;       NONE
596                     ;
597                     ; OUTPUTS:
598                     ;       NONE
599                     ;
600                     ; IMPLICIT OUTPUTS:
601                     ;       NONE
602                     ;
603                     ; PATHOLOGICAL CONNECTIONS:
604                     ;       NONE
605                     ;
606                     ; SUBORDINATE ROUTINES CALLED:
607                     ;       NONE
608                     ;
609                     ; FUNCTIONAL SIDE EFFECTS:
610                     ;       NONE
611                     ;
612                     ; CALLING SEQUENCE:
613                     ;       CALL IERRLO
614                     ;
615                     ; VERSION:
616                     ;       0.0
617                     ;
618                     ;       EDIT            DATE            BY              REASON
619                     ;--
```

```
  621
  622                                   .SBTTL INITIALIZE 11/70 REGISTERS ROUTINE
  623
  624  000106'                          ROUTINE IERRLO
  (2)  000106'                                                                  IERRLO:
  625
  626                                   ;+
  627                                   ; INITIALIZE THE 11/70 REGISTERS; MEMORY SYSTEM ERROR, LOW ERROR ADDRESS,
  628                                   ; HIGH ADDRESS, CONTROL, MAINTENANCE AND CPU ERROR
  629                                   ;-
  630
  631  000106'                          LET @CPUMER := @CPUMER
  (4)  000106' 017777  000000G 000000G                                          MOV     @CPUMER,@CPUMER
  632  000114'                          LET @CPULAR := #0
  (4)  000114' 005077  000000G                                                  CLR     @CPULAR
  633  000120'                          LET @CPUHAR := #0
  (4)  000120' 005077  000000G                                                  CLR     @CPUHAR
  634  000124'                          LET @CCNTRL := KONTRL
  (4)  000124' 016777  000000G 000000G                                          MOV     KONTRL,@CCNTRL
  635  000132'                          LET @CPUCPE := #0
  (4)  000132' 005077  000000G                                                  CLR     @CPUCPE
  636  000136'                          LET @CPUMNT := #0
  (4)  000136' 005077  000000G                                                  CLR     @CPUMNT
  637
  638  000142'                          ENDRTN
  (3)  000142'                                                                  50000$:
  (3)  000142'                                                                  50001$:
  (2)  000142' 000207                                                           RTS     PC
  639
```

```
641
642                             .SBTTL ERRLOG  11/70 ERROR LOGGING
643                             .IDENT /V0.0/
644
645                             ;++
646                             ; MODULE NAME:
647                             ;       ERRLOG
648                             ;
649                             ; FUNCTIONAL DESCRIPTION:
650                             ;       THIS MODULE WILL OUTPUT THE CONTENTS OF THE MEMORY SYSTEM ERROR
651                             ;       REGISTER(17777744) AND THE CPU ERROR REGISTER(17777766).
652                             ;       IF A PARITY ERROR OCCURRED THEN A 22 BIT ADDRESS IS ALSO OUTPUTTED,
653                             ;       THE ADDRESS IS GENERATED FROM THE LOW ERROR ADDRESS REGISTER (17777740)
654                             ;       AND THE HIGH ERROR ADDRESS REGISTER (17777742)
655                             ;
656                             ; INPUTS:
657                             ;       DTABLE
658                             ;
659                             ; IMPLICIT INPUTS:
660                             ;       DT.EVNT - EVENT WORD OF DTABLE
661                             ;
662                             ; OUTPUTS:
663                             ;       NONE
664                             ;
665                             ; IMPLICIT OUTPUTS:
666                             ;       NONE
667                             ;
668                             ; PATHOLOGICAL CONNECTIONS:
669                             ;       NONE
670                             ;
671                             ; SUBORDINATE ROUTINES CALLED:
672                             ;       BOA16   - OCTAL TO ASCII CONVERSION (16 BITS)
673                             ;       BOAC    - OCTAL TO ASCII CONVERSION (22 BITS)
674                             ;       MSGDHOOK- MESSAGE HOOK ROUTINE
675                             ;
676                             ; FUNCTIONAL SIDE EFFECTS:
677                             ;       NONE
678                             ;
679                             ; CALLING SEQUENCE:
680                             ;       CALL ERRLOG IN <A>
681                             ;                   A -ADDRESS OF DTABLE
682                             ;
683                             ; VERSION:
684                             ;       0.0
685                             ;
686                             ;       EDIT            DATE            BY              REASON
687                             ;--
```

```
                                .SBTTL 11/70 ERROR LOG ROUTINE

000144'                         ROUTINE ERRLOG <DTABLE>
(2) 000144'                                                             ERRLOG:


                                ;+
                                ; SAVE REGISTERS, GET DTABLE AND EVENT CODE
                                ;-

000144'                         PUSH R0,R1
(2) 000144' 010046                                                      MOV     R0,-(SP)
(3) 000146' 010146                                                      MOV     R1,-(SP)

000150'                         LET R0 := DTABLE(R5)
(4) 000150' 016500  000000                                              MOV     DTABLE(R5),R0
000154'                         LET R1 := DT.EVNT(R0)
(4) 000154' 016001  000000                                              MOV     DT.EVNT(R0),R1

                                ;+
                                ; CONVERT MEMORY SYSTEM ERROR REGISTER CONTENTS TO OCTAL ASCII AND
                                ; LOAD INTO MESSAGE
                                ;-

000160'                         CALL BOA16 IN <@CPUMER,#L7.MED>
(3) 000160' 010546                                                      MOV     R5,-(SP)
(5) 000162' 012745  000033'                                             MOV     #L7.MED,-(R5)
(4) 000166' 017745  000000G                                             MOV     @CPUMER,-(R5)
(3) 000172' 004767  000000G                                             JSR     PC,BOA16
(3) 000176' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; CONVERT CPU ERROR REGISTER CONTENTS TO OCTAL ASCII AND LOAD
                                ; INTO MESSAGE
                                ;-

000200'                         CALL BOA16 IN <@CPUCPE,#L7.CED>
(3) 000200' 010546                                                      MOV     R5,-(SP)
(5) 000202' 012745  000056'                                             MOV     #L7.CED,-(R5)
(4) 000206' 017745  000000G                                             MOV     @CPUCPE,-(R5)
(3) 000212' 004767  000000G                                             JSR     PC,BOA16
(3) 000216' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; OUTPUT MESSAGE
                                ;-

000220'                         CALL MSGDHOOK IN <R0,#MSGPOP,#L7.MSG,#2$>
(3) 000220' 010546                                                      MOV     R5,-(SP)
(7) 000222' 012745  000252'                                             MOV     #2$,-(R5)
(6) 000226' 012745  000000'                                             MOV     #L7.MSG,-(R5)
(5) 000232' 012745  000002                                              MOV     #MSGPOP,-(R5)
(4) 000236' 010045                                                      MOV     R0,-(R5)
(3) 000240' 004767  000000G                                             JSR     PC,MSGDHOOK
(3) 000244' 012605                                                      MOV     (SP)+,R5
```

```
  723  000246'                        INLINE <1$:     NOP>
  (2)  000246' 000240                                                    1$:     NOP
  724  000250'                        INLINE <BR      1$>
  (2)  000250' 000776                                                    BR      1$
  725
  726                                 ;+
  727                                 ; DETERMINE IF PARITY ERROR OCCURED, IF SO, CONVERT THE 22 BIT ADDRESS CONTAINED
  728                                 ; IN LOW AND HIGH ERROR ADDRESS REGISTERS TO OCTAL ASCII AND OUTPUT MESSAGE
  729                                 ;-
  730
  731  000252'                        INLINE <2$:>
  (2)  000252'                                                           2$:
  732  000252'                        IF #EVNTPE EQ R1 THEN
  (6)  000252' 022701  000202                                            CMP     #EVNTPE,R1
  (9)  000256' 001030                                                    BNE     50002$
  733  000260'                                CALL BOAC IN <R0,@CPULAR,@CPUHAR,#L7.ADR>
  (3)  000260' 010546                                                    MOV     R5,-(SP)
  (7)  000262' 012745  000074'                                           MOV     #L7.ADR,-(R5)
  (6)  000266' 017745  000000G                                           MOV     @CPUHAR,-(R5)
  (5)  000272' 017745  000000G                                           MOV     @CPULAR,-(R5)
  (4)  000276' 010045                                                    MOV     R0,-(R5)
  (3)  000300' 004767  000000G                                           JSR     PC,BOAC
  (3)  000304' 012605                                                    MOV     (SP)+,R5
  734
  735  000306'                                CALL MSGDHOOK IN <R0,#MSGPOP,#L7.MS1,#4$>
  (3)  000306' 010546                                                    MOV     R5,-(SP)
  (7)  000310' 012745  000340'                                           MOV     #4$,-(R5)
  (6)  000314' 012745  000066'                                           MOV     #L7.MS1,-(R5)
  (5)  000320' 012745  000002                                            MOV     #MSGPOP,-(R5)
  (4)  000324' 010045                                                    MOV     R0,-(R5)
  (3)  000326' 004767  000000G                                           JSR     PC,MSGDHOOK
  (3)  000332' 012605                                                    MOV     (SP)+,R5
  736  000334'                        INLINE <3$:     NOP>
  (2)  000334' 000240                                                    3$:     NOP
  737  000336'                        INLINE <        BR      3$>
  (2)  000336' 000776                                                            BR      3$
  738
  739  000340'                        ENDIF
  (4)  000340'                                                   50002$:
  740
  741                                 ;+
  742                                 ; RESTORE REGISTERS AND RETURN
  743                                 ;-
  744
  745  000340'                        INLINE <4$:>
  (2)  000340'                                                           4$:
  746  000340'                        POP R1,R0
  (2)  000340' 012601                                                    MOV     (SP)+,R1
  (3)  000342' 012600                                                    MOV     (SP)+,R0
  747
  748  000344'                        ENDRTN
  (3)  000344'                                                   50000$:
  (3)  000344'                                                   50001$:
  (2)  000344' 000207                                                    RTS     PC
  749
  750          000001                .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BOAC   = ****** G
BOA16  = ****** G
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CPUCPE= ****** G
CPUHAR= ****** G
CPULAR= ****** G
CPUMER= ****** G
CPUMNT= ****** G
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRLOG  000144RG
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
IERRLO  000106RG
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
L7.ADR  000074R
L7.CED  000056R
L7.MED  000033R
L7.MSG  000000R
L7.MS1  000066R
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKO= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRIO   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
RSVALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
```

```
STATBI= 064757          UIPAR7= 177656          XON    = 000021          $F$TRU= 000404          $$ELOC= 000402
STAT1 = 000027          UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $$ERFL= 000000
SUSPND= 000001          UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $$FLAG= 000001
SVR0  = 000062          UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$FROM= 000000
SVR1  = 000064          UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$LOC = 000256R
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$LOCN= 000000
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $LOCTA= 177777          $$REG = 177777
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $LSTIN= 000001          $$RETU= 000000
SVR5  = 000074          UIPDR7= 177616          $F$DO  = 000340          $LSTTA= 000001          $$RTN1= 050000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $NESTL= 177777          $$RTN2= 050001
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $NSK0 = 000300          $$SRC = 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF  = 000110          $NSK1 = 000110          $$TGSV= 000000
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $SAVLE= 177777          $$TGS1= 000000
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $TAGLE= 177777          $$TGS2= 000000
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $TAGNU= 050003          $$TO  = 000004
UIPAR1= 177642          WDFR   = 000116          $F$NO  = 000403          $TEMP  = 000300          $$$TAG= 050000
UIPAR2= 177644          WDTO   = 000114          $F$OR  = 000320          $TSK0  = 050002          .      = 000346R
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $$ARGC= 000002
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $$BYTE= 000403
UIPAR5= 177652          XFLAG  = 000005          $F$SEL= 000140          $$CASE= 000000
UIPAR6= 177654          XOFF   = 000023          $F$THE= 000330          $$DST  = 000000

. ABS.  000000      000
        000346      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BLOG70,DSKZ:BLOG70=SPMAC/ML,EQUATE,BLOG70
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 39/16=2.3
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE GPA (GET PHYSICAL ADDRESS)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       GPA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE DETERMINES THE 16-BIT, 18-BIT OR 22-BIT
                        ;       PHYSICAL ADDRESS FROM THE 16-BIT VIRTUAL ADDRESS GIVEN.
                        ;
                        ;       IF 18-BIT ADDRESS, THE TOP 2 BITS ARE ALLIGNED TO POSITIONS
                        ;       4 & 5 OF THE EA WORD. IF 22-BIT ADDRESS SYSTEM,
                        ;       THE TOP 6 BITS ARE RIGHT JUSTIFIED IN THE EA WORD.
                        ;
                        ; INPUTS:
                        ;       1. DTABLE
                        ;       2. TABLE ADDRESS CONTAINING VA,PA,EA WORDS
                        ;               WHERE VA = VIRTUAL ADDRESS
                        ;                     PA = PHYSICAL ADDRESS
                        ;                     EA = EXTENDED ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.CF0
                        ;       2. DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. PHYSICAL ADDRESS WORD
                        ;       2. EXTENDED ADDRESS BITS WORD
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL GPA IN <DT,TBL>
                        ;           WHERE DT = DTABLE ADDRESS
                        ;                 TBL = TABLE ADDRESS (CONTAINS VA,PA,EA WORDS)
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL GPA (COMMON DEFINITIONS & REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL GPA              ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
```

```
                                .SBTTL GPA (CODE)

000000'                         ROUTINE GPA <DT,TBL>
(2) 000000'                                                             GPA:

                                ;+
                                ; SAVE REGS, GET DTABLE ADDRESS AND TABLE ADDRESS
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                             JSR     PC,SAVREG
000004'                         LET R0 := DT(R5)
(4) 000004' 016500  000000                                              MOV     DT(R5),R0
000010'                         LET R1 := TBL(R5)
(4) 000010' 016501  000002                                              MOV     TBL(R5),R1

                                ;+
                                ; FETCH VIRTUAL ADDRESS FROM FIRST WORD OF USER SUPPLIED TABLE
                                ; AND LOAD INTO THE PHYSICAL ADDRESS WORD (2ND WORD OF USER SUPPLIED
                                ; TABLE).
                                ;-

000014'                         LET (R1) := (R1)+
(4) 000014' 012111                                                      MOV     (R1)+,(R1)

                                ;+
                                ; IF KT IS NOT ON, CLEAR EA WORD AND RETURN
                                ;-

000016'                         IF #KTSTAT NOTSETIN DT.STO(R0) THEN
(6) 000016' 032760  000020  000010                                      BIT     #KTSTAT,DT.STO(R
(9) 000024' 001003                                                      BNE     50002$
000026'                                 LET 2(R1) := #0
(4) 000026' 005061  000002                                              CLR     2(R1)

000032'                         ELSE
(4) 000032' 000454                                                      BR      50003$
(3) 000034'                                                         50002$:

                                ;+
                                ; KT IS ON. GENERATE 18 OR 22 BIT ADDRESS.
                                ; BRING PAGE BITS 13-15 OF PA IN POS.1-3, ADD TO KIPAR0
                                ; ADDRESS, AND GET THE CONTENTS OF THE FORMED KIPAR REG.
                                ;-

000034'                                 LET (R1) := SWAP (R1)
(6) 000034' 000311                                                      SWAB    (R1)
000036'                                 LET (R1) := (R1) SHIFT -4
(7) 000036' 006211                                                      ASR     (R1)
(7) 000040' 006211                                                      ASR     (R1)
(7) 000042' 006211                                                      ASR     (R1)
(7) 000044' 006211                                                      ASR     (R1)
000046'                                 LET (R1) := (R1) CLR.BY #177761
(6) 000046' 042711  177761                                              BIC     #177761,(R1)
000052'                                 LET (R1) := (R1) + #KIPAR0
(6) 000052' 062711  172340                                              ADD     #KIPAR0,(R1)
```

```
623  000056'                                    LET (R1) := @(R1)
(4)  000056' 017111  000000                                                  MOV     @(R1),(R1)
624                                             ;+
625                                             ; BITS 15-10 ARE EA BITS. BRING THEM IN POSITION 5-0
626                                             ; AND STORE IN EA WORD.
627                                             ;-
628  000062'                                    LET R2 := SWAP (R1)
(6)  000062' 011102                                                          MOV     (R1),R2
(6)  000064' 000302                                                          SWAB    R2
629  000066'                                    LET 2(R1) := R2 SHIFT -2
(4)  000066' 010261  000002                                                  MOV     R2,2(R1)
(7)  000072' 006261  000002                                                  ASR     2(R1)
(7)  000076' 006261  000002                                                  ASR     2(R1)
630
631                                             ;+
632                                             ; SHIFT THE CONTENTS OF PAR
633                                             ; 6 PLACES TO THE LEFT.ADD LOW 12 BITS OF V.A.
634                                             ; RESULT IS STORED IN PA
635                                             ; AND ANY CARRY IS ADDED TO THE EA WORD TO GET THE FINAL EA WORD.
636                                             ;-
637  000102'                                    LET (R1) := (R1) SHIFT 6
(7)  000102' 006311                                                          ASL     (R1)
(7)  000104' 006311                                                          ASL     (R1)
(7)  000106' 006311                                                          ASL     (R1)
(7)  000110' 006311                                                          ASL     (R1)
(7)  000112' 006311                                                          ASL     (R1)
(7)  000114' 006311                                                          ASL     (R1)
638  000116'                                    LET R2 := -2(R1) CLR.BY #160000
(4)  000116' 016102  177776                                                  MOV     -2(R1),R2
(6)  000122' 042702  160000                                                  BIC     #160000,R2
639  000126'                                    INLINE <ADD R2, (R1)+>
(2)  000126' 060221                                                          ADD R2, (R1)+
640  000130'                                    LET (R1) := (R1) + CARRY
(6)  000130' 005511                                                          ADC     (R1)
641                                             ;+
642                                             ; NOW CHECK IF 22-BIT ADDRESS SYSTEM. IF IT IS, GENERATE A
643                                             ; 6-BIT EA WORD RIGHT JUSTIFIED, OTHERWISE GENERATE
644                                             ; A 2-BIT EA WORD AND ALLIGN THE BITS IN POSITION 5-4.
645                                             ;-
646
647  000132'                                    IF #ADDR22 SETIN DT.CF0(R0) THEN
(6)  000132' 032760  001000  000014                                          BIT     #ADDR22,DT.CF0(R
(9)  000140' 001403                                                          BEQ     50004$
648  000142'                                           LET (R1) := (R1) CLR.BY #177700
(6)  000142' 042711  177700                                                  BIC     #177700,(R1)
649  000146'                                    ELSE
(4)  000146' 000406                                                          BR      50005$
(3)  000150'                                                          50004$:
650  000150'                                           LET (R1) := (R1) CLR.BY #177774
(6)  000150' 042711  177774                                                  BIC     #177774,(R1)
651  000154'                                           LET (R1) := (R1) SHIFT 4
(7)  000154' 006311                                                          ASL     (R1)
(7)  000156' 006311                                                          ASL     (R1)
(7)  000160' 006311                                                          ASL     (R1)
(7)  000162' 006311                                                          ASL     (R1)
652  000164'                                    ENDIF
```

```
   (4)  000164'                                                                50005$:
   653
   654  000164'                        ENDIF
   (4)  000164'                                                                50003$:
   655                                 ;+
   656                                 ; RESTORE REGS AND RETURN
   657                                 ;-
   658  000164'                        CALL RESREG
   (3)  000164' 004767  000000G                                                        JSR     PC,RESREG
   659  000170'                        ENDRTN
   (3)  000170'                                                                50000$:
   (3)  000170'                                                                50001$:
   (2)  000170' 000207                                                                 RTS     PC
   660
   661          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTFRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT    = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GPA     000000RG
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TBL   = 000002
TMPIO = 000002
```

```
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LOCTA= 177777          $$FLAG= 000001
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LSTIN= 000001          $$FROM= 000000
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $LSTTA= 000001          $$LOC = 000140R
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NESTL= 177777          $$LOCN= 000000
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK0 = 000300          $$REG = 177777
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK1 = 000110          $$RETU= 000000
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $NSK2 = 000110          $$RTN1= 050000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $SAVLE= 177777          $$RTN2= 050001
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $TAGLE= 177777          $$SRC = 000000
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $TAGNU= 050006          $$TGSV= 000000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TEMP = 000300          $$TGS1= 000000
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TSK0 = 050003          $$TGS2= 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TSK1 = 050005          $$TO  = 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $$ARGC= 000004          $$$TAG= 050000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$BYTE= 000403          .     = 000172R
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$CASE= 000000
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$DST = 000000
WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$ELOC= 000402
WBSTAT= 000040          $F$DO = 000340          $ISK1 = 000001          $$ERFL= 000000

. ABS.  000000     000
        000172     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:GPA,DSKZ:GPA=SPMAC/ML,EQUATE,GPA
 RUN-TIME: 14 4 .3 SECONDS
 RUN-TIME RATIO: 40/18=2.1
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KTERR  PROCESS MEMORY MANAGEMENT TRAPS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KTERR
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES MEMORY MANAGEMENT TRAPS (THROUGH
                        ;       LOC 250). IT WILL FETCH THE CONTENTS OF THE STATUS
                        ;       REGISTERS, CONVERT TO ASCII, AND ENQUEUE A MESSAGE
                        ;       CONTAINING THE CONTENTS. THE KT ERROR BIT WILL BE SET
                        ;       IN THE DTABLE ERROR WORD.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0
                        ;       DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ERR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       MSGDHOOK -  PRINT A MESSAGE
                        ;       RSTRCY - RESET RECOVERY
                        ;
                        ;       ERRLOG
                        ;       BOA16
                        ;       SAVREG
                        ;       RESREG
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KTERR IN <DTADR>
                        ;        WHERE DTADR = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KTERR

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KTERR           ;MODULE ENTRY POINT
                                ;
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  ERRLOG          ;ERROR LOGGING
                                .GLOBL  RSTRCY          ;RESET RECOVERY
                                .GLOBL  MSGDHOOK        ;PRINT A MESSAGE
                                .GLOBL  BOA16           ;OCTAL TO ASCII CONVERSION ROUTINE
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ;
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 044445  046114  020056  KT.BADKT:       .ASCIZ  /%ILL. TRAP THRU KT VCT.%/
000006' 051124  050101  052040
000014' 051110  020125  052113
000022' 053040  052103  022456
000030'     000
000031'     045 025052  045440  KT.MG1: .ASCII  /%** KT TRAP **%/
000036' 020124  051124  050101
000044' 025040  022452
000050' 051445  030122  020040          .ASCII  /%SR0    SR2%/
000056' 020040  051123  022462
000064' 000006                  KT.SS0MG:       .BLKB   6
000072' 020040                          .ASCII  /  /
000074' 000006                  KT.SS2MG:       .BLKB   6
000102' 000045                          .ASCIZ  /%/
                                ;
000104' 051445  030522  020040  KT.MG2: .ASCII  /%SR1    SR3%/
000112' 020040  051123  022463
000120' 000006                  KT.SS1MG:       .BLKB   6
000126' 020040                          .ASCII  /  /
000130' 000006                  KT.SS3MG:       .BLKB   6
000136' 000045                          .ASCIZ  /%/
                                .EVEN
                                ;
```

```
                                    ;****************
```

```
                                .SBTTL  KTERR ROUTINE

000140'                         ROUTINE KTERR <DTADR>
(2) 000140'                                                                     KTERR:

                                ;+
                                ; INIT AND SAVE DTABLE
                                ;-

000140'                         CALL SAVREG
(3) 000140' 004767  000000G                                                     JSR     PC,SAVREG
000144'                         LET R0 := DTADR(R5)
(4) 000144' 016500  000000                                                      MOV     DTADR(R5),R0


                                ;+
                                ; SET KT ERROR BIT IN ERROR WORD IN DTABLE
                                ;-

000150'                         LET DT.ERR(R0) := DT.ERR(R0) SET.BY #KTERROR
(6) 000150' 052760  000040  000020                                              BIS     #KTERROR,DT.ERR(


                                ;+
                                ; IF KT ENABLED, SET FATAL BIT IN ERROR WORD
                                ;-

000156'                         IF #KTSTAT SETIN DT.ST0(R0) THEN
(6) 000156' 032760  000020  000010                                              BIT     #KTSTAT,DT.ST0(R
(9) 000164' 001535                                                              BEQ     50002$

                                ;+
                                ;  SET FATAL BIT, GET THE CONTENTS OF SR0, SR2, CONVERT
                                ; TO ASCII, AND ENQUEUE MESSAGE
                                ;-
000166'                                 LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6) 000166' 052760  100000  000020                                              BIS     #FATERR,DT.ERR(R

000174'                                 CALL BOA16 IN <@#SR0,#KT.SS0MG>
(3) 000174' 010546                                                              MOV     R5,-(SP)
(5) 000176' 012745  000064'                                                     MOV     #KT.SS0MG,-(R5)
(4) 000202' 013745  177572                                                      MOV     @#SR0,-(R5)
(3) 000206' 004767  000000G                                                     JSR     PC,BOA16
(3) 000212' 012605                                                              MOV     (SP)+,R5
000214'                                 CALL BOA16 IN <@#SR2,#KT.SS2MG>
(3) 000214' 010546                                                              MOV     R5,-(SP)
(5) 000216' 012745  000074'                                                     MOV     #KT.SS2MG,-(R5)
(4) 000222' 013745  177576                                                      MOV     @#SR2,-(R5)
(3) 000226' 004767  000000G                                                     JSR     PC,BOA16
(3) 000232' 012605                                                              MOV     (SP)+,R5

                                ;+
                                ; SEE IF EXTENDED KT
                                ;-
```

```
650
651  000234'                                    IF #KTXTND SETIN DT.CF0(R0) THEN                                BIT     #KTXTND,DT.CF0(R
(6)  000234' 032760  040000  000014                                                                            BEQ     50003$
(9)  000242' 001420
652
653                                          ;+
654                                          ; ITS AN EXTENDED KT, SO PUMP OUT SR1,SR3
655                                          ;-
656
657  000244'                                            CALL BOA16 IN <@#SR1,#KT.SS1MG>
(3)  000244' 010546                                                                                            MOV     R5,-(SP)
(5)  000246' 012745  000120'                                                                                   MOV     #KT.SS1MG,-(R5)
(4)  000252' 013745  177574                                                                                    MOV     @#SR1,-(R5)
(3)  000256' 004767  000000G                                                                                   JSR     PC,BOA16
(3)  000262' 012605                                                                                            MOV     (SP)+,R5
658  000264'                                            CALL BOA16 IN <@#SR3,#KT.SS3MG>
(3)  000264' 010546                                                                                            MOV     R5,-(SP)
(5)  000266' 012745  000130'                                                                                   MOV     #KT.SS3MG,-(R5)
(4)  000272' 013745  172516                                                                                    MOV     @#SR3,-(R5)
(3)  000276' 004767  000000G                                                                                   JSR     PC,BOA16
(3)  000302' 012605                                                                                            MOV     (SP)+,R5
659  000304'                                    ENDIF
(4)  000304'                                                                                        50003$:
660                                          ;+
661                                          ; RESET AND RESTORE, LOWER PRIORITY
662                                          ;-
663  000304'                                    INLINE <RESET>
(2)  000304' 000005                                                                                            RESET
664  000306'                                    CALL RSTRCY IN <R0>
(3)  000306' 010546                                                                                            MOV     R5,-(SP)
(4)  000310' 010045                                                                                            MOV     R0,-(R5)
(3)  000312' 004767  000000G                                                                                   JSR     PC,RSTRCY
(3)  000316' 012605                                                                                            MOV     (SP)+,R5
665
666
667  000320'                                    LET -(SP) := #0
(4)  000320' 005046                                                                                            CLR     -(SP)
668  000322'                                    LET -(SP) := #443$
(4)  000322' 012746  000330'                                                                                   MOV     #443$,-(SP)
669  000326'                                    INLINE <RTI>
(2)  000326' 000002                                                                                            RTI
670
671  000330'                                    INLINE <443$:>
(2)  000330'                                                                                                   443$:
672  000330'                                    CALL MSGDHOOK IN <R0,#MSGPOP,#KT.MG1,#1$>
(3)  000330' 010546                                                                                            MOV     R5,-(SP)
(7)  000332' 012745  000362'                                                                                   MOV     #1$,-(R5)
(6)  000336' 012745  000031'                                                                                   MOV     #KT.MG1,-(R5)
(5)  000342' 012745  000002                                                                                    MOV     #MSGPOP,-(R5)
(4)  000346' 010045                                                                                            MOV     R0,-(R5)
(3)  000350' 004767  000000G                                                                                   JSR     PC,MSGDHOOK
(3)  000354' 012605                                                                                            MOV     (SP)+,R5
673
674  000356'                                    INLINE <4$:     NOP>
(2)  000356' 000240                                                                                            4$:     NOP
675  000360'                                    INLINE <BR 4$>
```

```
(2)  000360' 000776                                                                    BR 4$
676
677  000362'                                INLINE <1$:>                               1$:
(2)  000362'
678
679
680  000362'                                IF #KTXTND SETIN DT.CF0(R0) THEN
(6)  000362' 032760  040000  000014                                                    BIT     #KTXTND,DT.CF0(R
(9)  000370' 001415                                                                    BEQ     50004$
681  000372'                                        CALL MSGDHCOK IN <R0,#MSGPOP,#KT.MG2,#2$>
(3)  000372' 010546                                                                    MOV     R5,-(SP)
(7)  000374' 012745  000424'                                                           MOV     #2$,-(R5)
(6)  000400' 012745  000104'                                                           MOV     #KT.MG2,-(R5)
(5)  000404' 012745  000002                                                            MOV     #MSGPOP,-(R5)
(4)  000410' 010045                                                                    MOV     R0,-(R5)
(3)  000412' 004767  000000G                                                           JSR     PC,MSGDHOOK
(3)  000416' 012605                                                                    MOV     (SP)+,R5
682  000420'                                        INLINE <6$:     NOP>               6$:     NOP
(2)  000420' 000240
683  000422'                                        INLINE <BR      6$>                BR      6$
(2)  000422' 000776
684
685  000424'                                        INLINE <2$:>                       2$:
(2)  000424'
686  000424'                                IF
(4)  000424'                                                                                   50004$:
687                                     ;+
688                                     ; DO ERROR LOGGING IF NECESSARY
689                                     ;-
690
691  000424'                                IF #PDP60 SETIN DT.CF0(R0) OR #PDP70 SETIN DT.CF0(R0) THEN
(6)  000424' 032760  004000  000014                                                    BIT     #PDP60,DT.CF0(R0
(8)  000432' 001004                                                                    BNE     50005$
(6)  000434' 032760  010000  000014                                                    BIT     #PDP70,DT.CF0(R0
(9)  000442' 001405                                                                    BEQ     50006$
(6)  000444'                                                                                   50005$:
692
693  000444'                                        CALL ERRLOG IN <R0>
(3)  000444' 010546                                                                    MOV     R5,-(SP)
(4)  000446' 010045                                                                    MOV     R0,-(R5)
(3)  000450' 004767  000000G                                                           JSR     PC,ERRLOG
(3)  000454' 012605                                                                    MOV     (SP)+,R5
694
695  000456'                                ENDIF
(4)  000456'                                                                                   50006$:
696
697  000456'                                ELSE                                       BR      50007$
(4)  000456' 000427
(3)  000460'                                                                                   50002$:
698                                     ;+
699                                     ; RESET AND RESTORE, LOWER PRIORITY
700                                     ;-
701  000460'                                        INLINE <RESET>                     RESET
(2)  000460' 000005
702  000462'                                        CALL RSTRCY IN <R0>                MOV     R5,-(SP)
(3)  000462' 010546
```

```
(4)  000464' 010045                                                          MOV     R0,-(R5)
(3)  000466' 004767  000000G                                                 JSR     PC,RSTRCY
(3)  000472' 012605                                                          MOV     (SP)+,R5
703
704
705  000474'                         LET -(SP) := #0
(4)  000474' 005046                                                          CLR     -(SP)
706  000476'                         LET -(SP) := #444$
(4)  000476' 012746  000504'                                                 MOV     #444$,-(SP)
707  000502'                         INLINE <RTI>
(2)  000502' 000002                                                          RTI
708
709  000504'                         INLINE <444$:>
(2)  000504'                                                                 444$:
710
711
712                              ;+
713                              ; NO, KT IS NOT ENABLED, SO ENQUEUE THE BAD MESSAGE
714                              ;-
715  000504'                         CALL MSGDHOOK IN <R0,#MSGPOP,#KT.BADKT,#3$>
(3)  000504' 010546                                                          MOV     R5,-(SP)
(7)  000506' 012745  000536'                                                 MOV     #3$,-(R5)
(6)  000512' 012745  000000'                                                 MOV     #KT.BADKT,-(R5)
(5)  000516' 012745  000002                                                  MOV     #MSGPOP,-(R5)
(4)  000522' 010045                                                          MOV     R0,-(R5)
(3)  000524' 004767  000000G                                                 JSR     PC,MSGDHOOK
(3)  000530' 012605                                                          MOV     (SP)+,R5
716  000532'                         INLINE <8$:     NOP>
(2)  000532' 000240                                                          8$:     NOP
717  000534'                         INLINE <BR      8$>
(2)  000534' 000776                                                          BR      8$
718
719  000536'                         INLINE <3$:>
(2)  000536'                                                                 3$:
720  000536'                     ENDIF
(4)  000536'                                                         50007$:
721
722                              ;+
723                              ; CLEAN UP AND GET OUT
724                              ;-
725
726  000536'                     CALL RESREG
(3)  000536' 004767  000000G                                                 JSR     PC,RESREG
727
728  000542'                     ENDRTN
(3)  000542'                                                         50000$:
(3)  000542'                                                         50001$:
(2)  000542' 000207                                                          RTS     PC
729
730          000001             .END
```

```
ACSR  = 000102          CSRC  = 000102          ENDLST= 000000          KT.SS1  000120R         PR4   = 000200
ACTBIT= 004000          CTRLC = 000003          EOPBIT= 000001          KT.SS2  000074R         PR5   = 000240
ADDR22= 001000          CTRLO = 000017          ERRLOG= ****** G        KT.SS3  000130R         PR6   = 000300
ADR   = 000006          CTRLU = 000025          ERRTYP= 000106          LF    = 000012          PR7   = 000340
APTFER= 000004          DCEVNT= 000011          EVNTBE= 000200          LPSTAT= 000001          PS    = 177776
APTPRE= 000200          DEFRTN= 000400          EVNTHD= 000200          MAPSTA= 000200          PSW   = 177776
ASB   = 000106          DIAGMC= 000000          EVNTKT= 000203          MED   = 076600          RANNUM= 000054
ASSEMB= 000010          DROPMO= 100000          EVNTPE= 000202          MEMPAS= 040000          RBUFEA= 000130
ASTAT = 000104          DSEVNT= 000014          EVNTRE= 000201          MODEXH= 004000          RBUFPA= 000126
AUTO  = 000010          DTADR = 000000          FATERR= 100000          MODHOL= 002000          RBUFSZ= 000132
AUTOST= 020000          DT.ADD= 000042          HRDCNT= 000044          MODSEL= 001000          RBUFVA= 000124
AWAS  = 000110          DT.AP = 000100          HRDPAS= 000050          MSGCKD= 000010          RDSERV= 000101
BIT0  = 000001          DT.APK= 000076          ICONT = 000036          MSGCKS= 000011          RDWHMI= 000022
BIT00 = 000001          DT.BLS= 000034          ICOUNT= 000040          MSGDER= 000005          RELERR= 000020
BIT01 = 000002          DT.CF0= 000014          IDNUM = 000122          MSGDHO= ****** G        RELMOD= 020000
BIT02 = 000004          DT.CF1= 000016          IE    = 000100          MSGDRP= 000017          RELTIM= 010000
BIT03 = 000010          DT.ERR= 000020          INDPAR= 000040          MSGECH= 177777          RESREG= ****** G
BIT04 = 000020          DT.ESI= 000044          INHDRP= 040000          MSGEOP= 000013          RES1  = 000056
BIT05 = 000040          DT.EVN= 000000          INHEPR= 020000          MSGHDR= 000004          RES2  = 000060
BIT06 = 000100          DT.EXS= 000060          INHREL= 001000          MSGHNG= 000022          RICHAR= 031060
BIT07 = 000200          DT.FCH= 000037          INHRRE= 000400          MSGHRD= 000007          RPTDAT= 002000
BIT08 = 000400          DT.FCN= 000036          INIT  = 000030          MSGMAP= 000021          RSTRCY= ****** G
BIT09 = 001000          DT.HMX= 000104          INTR  = 000120          MSGNUL= 177775          RSTRT = 000112
BIT1  = 000002          DT.KBE= 000024          IOMOD = 100000          MSGPOP= 000002          RUBOUT= 000177
BIT10 = 002000          DT.KBP= 000026          IOMODP= 102000          MSGPRM= 177776          RUNMOD= 100000
BIT11 = 004000          DT.KBR= 000022          IOMODR= 112000          MSGRES= 000001          R5VALU= 001740
BIT12 = 010000          DT.KBU= 000030          IOMODX= 110000          MSGSFT= 000006          SAM   = 075464
BIT13 = 020000          DT.MLS= 000032          JACK  = 035060          MSGSKE= 000003          SAVREG= ****** G
BIT14 = 040000          DT.MTI= 000110          KIPAR0= 172340          MSGSMB= 000015          SBADR = 000102
BIT15 = 100000          DT.OFF= 000070          KIPAR1= 172342          MSGSMH= 000014          SBKMOD= 000000
BIT2  = 000004          DT.PAS= 000074          KIPAR2= 172344          MSGSMS= 000016          SBKSEL= 010000
BIT3  = 000010          DT.PC = 000002          KIPAR3= 172346          MSGSTD= 000000          SC.ADR= 000006
BIT4  = 000020          DT.PFL= 000062          KIPAR4= 172350          MSGSYS= 000012          SC.ALC= 000014
BIT5  = 000040          DT.PSW= 000004          KIPAR5= 172352          MSGVEC= 000020          SC.APC= 000016
BIT6  = 000100          DT.PTA= 000064          KIPAR6= 172354          NBKMOD= 001000          SC.CKL= 000002
BIT7  = 000200          DT.RCS= 000102          KIPAR7= 172356          NCPUOP= 000020          SC.CKP= 000004
BIT8  = 000400          DT.REL= 000040          KIPDR0= 172300          NOAPTY= 000002          SC.CLO= 000000
BIT9  = 001000          DT.SCT= 000066          KIPDR1= 172302          NULL  = 000000          SC.HLD= 000010
BKDEF = 000002          DT.SMX= 000106          KIPDR2= 172304          OWEN  = 024020          SC.SCA= 000012
BKMOD = 000020          DT.SP = 000006          KIPDR3= 172306          PAERR = 000010          SENDLS= 177777
BKMODE= 040000          DT.SSI= 000046          KIPDR4= 172310          PARPRE= 002000          SOFCNT= 000042
BKSLSH= 000134          DT.ST0= 000010          KIPDR5= 172312          PARSTA= 000100          SOFPAS= 000046
BOA16 = ****** G        DT.ST1= 000012          KIPDR6= 172314          PASCNT= 000034          SPACE = 000040
CAPRES= 000004          DT.SWR= 000C56          KIPDR7= 172316          PDPLSI= 020000          SPOINT= 000032
CASTAT= 000004          DT.SYP= 000072          KTERR   000140RG        PDP60 = 004000          SPVALU= 002200
CDERCT= 000146          DT.WBU= 000050          KTERRO= 000040          PDP70 = 010000          SR0   = 177572
CDWDCT= 000144          DT.WHL= 000054          KTPRES= 000400          PRI0  = 000000          SR1   = 177574
CKTIM = 100000          DT.WLL= 000052          KTSTAT= 000020          PRI1  = 000040          SR2   = 177576
CLKPRE= 000001          DVID1 = 000014          KTXTND= 040000          PRI4  = 000200          SR3   = 172516
CONFIG= 000056          ECCMEM= 000100          KT.BAD  000000R         PRI5  = 000240          STAT  = 000026
CQOVF = 000001          ECCSTA= 000010          KT.MG1  000001R         PRI6  = 000300          STATBI= 064757
CR    = 000015          ENBEOP= 010000          KT.MG2  000104R         PRI7  = 000340          STAT1 = 000027
CSRA  = 000100          ENBNUL= 000001          KT.SS0  000064R         PR0   = 000000          SUSPND= 000001
```

```
SVR0  = 000062          UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$ELOC= 000402
SVR1  = 000064          UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$ERFL= 000000
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$FLAG= 000001
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$FROM= 000000
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $LOCTA= 177777          $$LOC = 000442R
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $LSTIN= 000001          $$LOCN= 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LSTTA= 000001          $$REG = 177777
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $NESTL= 177777          $$RETU= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $NSK0 = 000300          $$RTN1= 050000
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK1 = 000110          $$RTN2= 050001
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK2 = 000110          $$SRC = 000000
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $SAVLE= 177777          $$TGSV= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $TAGLE= 177777          $$TGS1= 000000
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $TAGNU= 050010          $$TGS2= 000000
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $TEMP = 000300          $$TO  = 000000
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $TSK0 = 050007          $$$TAG= 050000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TSK1 = 050006          .     = 000544R
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $$ARGC= 000002
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $$BYTE= 000403
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $$CASE= 000000
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $$DST = 000000

. ABS.  000000     000
        000544     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KTERR,DSKZ:KTERR=SPMAC/ML,EQUATE,KTERR
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 36/19=1.8
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                                .TITLE  KTSET  MAP MEMORY MANAGEMENT REGISTERS
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       KTSET
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       ROUTINE WILL SET UP THE PDRS (USER AND KERNEL) TO THE FOLLOWING:
                                ;                 READ/WRITE PROTECTION (NO SYSTEM TRAP/ABORT ACTION),
                                ;                 UPWARD PAGE ADDRESSING,
                                ;                 MAXIMUM BLOCK COUNT (4K).
                                ;       THE PARS WILL BE SET UP WITH PAR0 SET TO BANK0, PAR1-PAR5 SET UP TO
                                ;       DEFINED OFFSET, AND PAR7 TO THE I/O PAGE.
                                ;
                                ; INPUTS:
                                ;       DTABLE ADDRESS
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.ADDR - CURRENT BASE OFFSET
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       NONE
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL KTSET IN <A>
                                ;                 A - ADDRESS OF DTABLE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN = 1
        000001                  $LSTTAG = 1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  KTSET                   ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS

                                ;****************
                                ; LOCAL EQUATES
                                ;
        077406                  KS.PDR  = 77406                 ;PDR VALUE
```

```
                                .SBTTL KTSET ROUTINE

000000'                         ROUTINE KTSET <DTABLE>
(2) 000000'                                                     KTSET:

                                ;+
                                ; SAVE REGISTERS, RETRIEVE ADDRESS OF DTABLE
                                ; INITIALIZE POINTER TO BANK0, SETUP R2 TO 77406 VALUE
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                             JSR     PC,SAVREG
000004'                         LET R0 := DTABLE(R5)
(4) 000004' 016500  000000                                              MOV     DTABLE(R5),R0
000010'                         LET R2 := #KS.PDR
(4) 000010' 012702  077406                                              MOV     #KS.PDR,R2

                                ;+
                                ; CALL KTSET0, PASS ADDRESS OF UIPDR0
                                ;-

000014'                         LET R3 := #UIPDR0
(4) 000014' 012703  177600                                              MOV     #UIPDR0,R3
000020'                         CALL KTSET0
(3) 000020' 004767  000016                                              JSR     PC,KTSET0

                                ;+
                                ; CALL KTSET0, PASS ADDRESS OF KIPDR0
                                ;-

000024'                         LET R3 := #KIPDR0
(4) 000024' 012703  172300                                              MOV     #KIPDR0,R3
000030'                         CALL KTSET0
(3) 000030' 004767  000006                                              JSR     PC,KTSET0

                                ;+
                                ; RESTORE REGISTERS AND RETURN
                                ;-
000034'                         CALL RESREG
(3) 000034' 004767  000000G                                             JSR     PC,RESREG
000040'                         ENDRTN
(3) 000040'                                                     50000$:
(3) 000040'                                                     50001$:
(2) 000040' 000207                                                      RTS     PC

000042'                         ROUTINE KTSET0
(2) 000042'                                                     KTSET0:

                                ;+
                                ; LOAD APR0 BY MAPPING TO BANK0, SET UP R4 TO POINT TO APR7
                                ;-

000042'                         LET R4 := R3 + #16
(4) 000042' 010304                                                      MOV     R3,R4
```

```
   (6)  000044' 062704  000016                                                  ADD     #16,R4
621  000050'                        LET (R3) := R2
   (4)  000050' 010213                                                          MOV     R2,(R3)
622
623                                 ;+
624                                 ; LOAD PAR0 TO MAP TO BANK 0
625                                 ;-
626
627  000052'                        LET 40(R3) := #0
   (4)  000052' 005063  000040                                                  CLR     40(R3)
628  000056'                        INLINE <TST (R3)+>
   (2)  000056' 005723                                                          TST (R3)+
629
630                                 ;+
631                                 ; LOAD DT.ADDR INTO R1 AND LOAD APR0 - APR6
632                                 ;-
633
634  000060'                        LET R1 := DT.ADDR(R0)
   (4)  000060' 016001  000042                                                  MOV     DT.ADDR(R0),R1
635  000064'                        WHILE R3 NE R4 DO
   (4)  000064'                                                         50002$:
   (6)  000064' 020304                                                          CMP     R3,R4
   (9)  000066' 001407                                                          BEQ     50003$
636  000070'                                LET (R3) := R2
   (4)  000070' 010213                                                          MOV     R2,(R3)
637  000072'                                LET 40(R3) := R1
   (4)  000072' 010163  000040                                                  MOV     R1,40(R3)
638  000076'                                INLINE <TST (R3)+>
   (2)  000076' 005723                                                          TST (R3)+
639  000100'                                LET R1 := R1 + #200
   (6)  000100' 062701  000200                                                  ADD     #200,R1
640  000104'                        ENDDO
   (4)  000104' 000767                                                          BR      50002$
   (3)  000106'                                                         50003$:
641
642                                 ;+
643                                 ; NOW LOAD APR7
644                                 ;-
645
646  000106'                        LET (R3) := R2
   (4)  000106' 010213                                                          MOV     R2,(R3)
647  000110'                        LET 40(R3) := #177600
   (4)  000110' 012763  177600  000040                                          MOV     #177600,40(R3)
648
649                                 ;+
650                                 ; RETURN
651                                 ;-
652
653  000116'                        ENDRTN
   (3)  000116'                                                         50000$:
   (3)  000116'                                                         50001$:
   (2)  000116' 000207                                                          RTS     PC
654
655          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KS.PDR= 077406
KTERR0= 000040
KTPRES= 000400
KTSET   000000RG
KTSET0  000042R
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SGFCNT= 000042
SGFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
```

```
TMPIO = 000002          WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$DST = 000000
TQOVF = 000002          WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$ELOC= 000000
UIPAR0= 177640          WBUFEA= 000136          $F$DO = 000340          $LOCTA= 177777          $$ERFL= 000000
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000340
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
UIPAR3= 177646          WBUFSZ= 000142          $F$IF = 000110          $NESTL= 177777          $$LOC = 000066R
UIPAR4= 177650          WDFR  = 000116          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
UIPAR5= 177652          WDTO  = 000114          $F$LOO= 000200          $NSK1 = 000120          $$REG = 177777
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $SAVLE= 177777          $$RETU= 000000
UIPAR7= 177656          WTWHMI= 000222          $F$NO = 000403          $SSK0 = 050003          $$RTN1= 050000
UIPDR0= 177600          XFLAG = 000005          $F$OR = 000320          $TAGLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XOFF  = 000023          $F$RTI= 000350          $TAGNU= 050004          $$SRC = 000000
UIPDR2= 177604          XON   = 000021          $F$RTN= 000300          $TEMP = 000300          $$TGSV= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TSK0 = 050002          $$TGS1= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $TSK1 = 050003          $$TGS2= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $$ARGC= 000000          $$TO  = 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$BYTE= 000403          $$$TAG= 050000
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$CASE= 000000          .     = 000120R

. ABS.  000000      000
        000120      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KTSET,DSKZ:KTSET=SPMAC/ML,EQUATE,KTSET
 RUN-TIME: 12 2 .3 SECONDS
 RUN-TIME RATIO: 29/15=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PARERR PROCESS PARITY ERROR TRAPS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PARERR
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PROCESSES PARITY ERROR TRAPS. WHEN ENTERED
                        ;       IT WILL TEST FOR PARITY ENABLED. IF NOT ENABLED, AN APPROPRIATE
                        ;       MESSAGE IS ENQUEUED.
                        ;       A CHECK IS PERFORMED TO DETERMINE WHICH PARITY CSR
                        ;       HAS THE ERROR BIT SET. THE CSR AND ITS CONTENTS ARE ENQUEUED
                        ;       IN A PARITY MESSAGE(STANDARD).
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0
                        ;       DT.CF0
                        ;       DT.PTA
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ENQTQ   -       QUEUE A MESSAGE
                        ;       BOA16 -         BINARY TO OCTAL ASCII CONVERSION ROUTINE
                        ;       KTSET  -       SETUP KT REGISTERS
                        ;       SAVREG
                        ;       RESREG
                        ;       MSGDHOOK
                        ;       ERRLOG
                        ;       RSTRCY
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PARERR IN <DTADR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR PARERR

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

            000001              $LSTIN=1
            000001              $LSTTAG=1

                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  PARERR          ;MODULE ENTRY POINT
                                .GLOBL  PA.CNT          ;NUMBER OF PARITY ERRORS
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  MSGDHOOK        ;ENQUEUE A MESSAGE
                                .GLOBL  RSTRCY          ;RESET RECOVERY ROUTINE
                                .GLOBL  ERRLOG          ;1160, 1170 ERROR LOGGING
                                .GLOBL  CCNTRL          ;CACHE CONTROL REGISTER
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONVERSION ROUTINE
                                .GLOBL  KTSET           ;SETUP KT APRS'
                                ;
                                ;****************
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 025045  025052  052040  PA.MSG:         .ASCII  /%*** TRAP THRU VEC. 114 ***%/
000006' 040522  020120  044124
000014' 052522  053040  041505
000022' 020056  030461  020064
000030' 025052  022452
000034'    000                  PA.SP:  .BYTE   0
000035'    040  051503  020122          .ASCII  / CSR    CONTENTS%/
000042' 020040  041440  047117
000050' 042524  052116  022523
000056' 000006                  PA.CSR:         .BLKB   6
000064' 020040                          .ASCII  /  /
000066' 000006                  PA.CSC:         .BLKB   6
000074' 000045                          .ASCIZ  /%/

000076' 000000                  PA.CNT: .WORD   0       ;NUMBER OF PARITY ERRORS
000100' 000010                  PA.MAX: .WORD   10      ;MAX NUMBER OF ALLOWABLE PARITY ERRORS
                                .EVEN
                                ;****************
                                ; LOCAL EQUATES
                                ;
            170370              PA.M36 = 170370         ;MAP REGISTER 36 ADDRESS
            170372              PA.M37 = 170372         ;MAP REGISTER 36 UPPER 6 BITS
```

```
                        ;+
                        ; PARITY TABLE - FILLED WITH PARITY CSR ADDRESSES
                        ;-
```

```
                                .SBTTL  PARERR ROUTINE

000102'                         ROUTINE PARERR <DTABLE>
(2) 000102'                                                             PARERR:

                                ;+
                                ; INITIALIZE AND SAVE DTABLE ADDRESS
                                ;-

000102'                         CALL SAVREG
(3) 000102' 004767  000000G                                                     JSR     PC,SAVREG
000106'                         LET R0 := DTABLE(R5)
(4) 000106' 016500  000000                                                      MOV     DTABLE(R5),R0

                                ;+
                                ; SET ERROR BIT IN DTABLE'S ERROR WORD
                                ;-

000112'                         LET DT.ERR(R0) := DT.ERR(R0) SET.BY #PAERR
(6) 000112' 052760  000010  000020                                              BIS     #PAERR,DT.ERR(R0


                                ;+
                                ; RESET, RECOVER FROM RESET, AND LOWER PRIORITY
                                ;-
000120'                         INLINE <RESET>
(2) 000120' 000005                                                              RESET
000122'                         CALL RSTRCY IN <R0>
(3) 000122' 010546                                                              MOV     R5,-(SP)
(4) 000124' 010045                                                              MOV     R0,-(R5)
(3) 000126' 004767  000000G                                                     JSR     PC,RSTRCY
(3) 000132' 012605                                                              MOV     (SP)+,R5
000134'                         LET -(SP) := #0
(4) 000134' 005046                                                              CLR     -(SP)
000136'                         LET -(SP) := #443$
(4) 000136' 012746  000144'                                                     MOV     #443$,-(SP)
000142'                         INLINE <RTI>
(2) 000142' 000002                                                              RTI

000144'                         INLINE <443$:>
(2) 000144'                                                             443$:

                                ;+
                                ; IS PARITY OR ECC ENABLED?
                                ;-

000144'                         IF #PARSTAT SETIN DT.STO(R0) OR #ECCSTAT SETIN DT.STO(R0) THEN
(6) 000144' 032760  000100  000010                                              BIT     #PARSTAT,DT.STO(
(8) 000152' 001004                                                              BNE     50002$
(6) 000154' 032760  000010  000010                                              BIT     #ECCSTAT,DT.STO(
(9) 000162' 001552                                                              BEQ     50003$
(6) 000164'                                                             50002$:

```

```
                                ;+
                                ; IF 11/60 OR 11/70, CALL ERRLOG
                                ;-

000164'                                 IF #PDP70 SETIN DT.CF0(R0) OR #PDP60 SETIN DT.CF0(R0) THEN
(6) 000164' 032760 010000 000014                                                BIT     #PDP70,DT.CF0(R0
(8) 000172' 001004                                                              BNE     50004$
(6) 000174' 032760 004000 000014                                                BIT     #PDP60,DT.CF0(R0
(9) 000202' 001405                                                              BEQ     50005$
(6) 000204'                                                             50004$:
000204'                                         CALL ERRLOG IN <R0>
(3) 000204' 010546                                                              MOV     R5,-(SP)
(4) 000206' 010045                                                              MOV     R0,-(R5)
(3) 000210' 004767 000000G                                                      JSR     PC,ERRLOG
(3) 000214' 012605                                                              MOV     (SP)+,R5
000216'                                 ENDIF
(4) 000216'                                                             50005$:

                                ;+
                                ; GET PTABLE ADDRESS
                                ;-

000216'                                 LET R1 := DT.PTA(R0)
(4) 000216' 016001 000064                                                       MOV     DT.PTA(R0),R1

                                ;+
                                ; IF INDIRECT CSRS THEN SAVE APPROPRIATE REGISTERS (CACHE, KT AND MAP)
                                ;-

000222'                                 IF #INDPAR SETIN DT.CF0(R0) THEN
(6) 000222' 032760 000040 000014                                                BIT     #INDPAR,DT.CF0(R
(9) 000230' 001443                                                              BEQ     50006$
000232'                                         LET -(SP) := @CCNTRL
(4) 000232' 017746 000000G                                                      MOV     @CCNTRL,-(SP)
000236'                                         LET -(SP) := @#SR0
(4) 000236' 013746 177572                                                       MOV     @#SR0,-(SP)
000242'                                         LET -(SP) := @#SR3
(4) 000242' 013746 172516                                                       MOV     @#SR3,-(SP)
000246'                                         LET -(SP) := @#PA.M36
(4) 000246' 013746 170370                                                       MOV     @#PA.M36,-(SP)
000252'                                         LET -(SP) := @#PA.M37
(4) 000252' 013746 170372                                                       MOV     @#PA.M37,-(SP)
000256'                                         LET -(SP) := @#KIPAR6
(4) 000256' 013746 172354                                                       MOV     @#KIPAR6,-(SP)

                                ;+
                                ; NOW SETUP APPROPRIATE REGISTERS TO GO AFTER INDIRECT CSRS
                                ;-

000262'                                         LET @CCNTRL := @CCNTRL SET.BY #14
(6) 000262' 052777 000014 000000G                                               BIS     #14,@CCNTRL
000270'                                         CALL KTSET IN <R0>
(3) 000270' 010546                                                              MOV     R5,-(SP)
(4) 000272' 010045                                                              MOV     R0,-(R5)
(3) 000274' 004767 000000G                                                      JSR     PC,KTSET
```

```
(3)  000300' 012605                                                                    MOV     (SP)+,R5
692  000302'                                LET @#KIPAR6 := #177400
(4)  000302' 012737  177400  172354                                                   MOV     #177400,@#KIPAR6
693  000310'                                LET @#PA.M36 := #160000
(4)  000310' 012737  160000  170370                                                   MOV     #160000,@#PA.M36
694  000316'                                LET @#PA.M37 := #77
(4)  000316' 012737  000077  170372                                                   MOV     #77,@#PA.M37
695  000324'                                LET @#SR0 := @#SR0 SET.BY #BIT00
(6)  000324' 052737  000001  177572                                                   BIS     #BIT00,@#SR0
696  000332'                                LET @#SR3 := @#SR3 SET.BY #60
(6)  000332' 052737  000060  172516                                                   BIS     #60,@#SR3
697  000340'                        ENDIF
(4)  000340'                                                                  50006$:
698
699
700                                 ;+
701                                 ; SEARCH PARITY TABLE UNTIL ERROR IS FOUND AND O/P MESSAGE...
702                                 ;-
703
704  000340'                        WHILE (R1) NE #0 DO
(4)  000340'                                                                  50007$:
(6)  000340' 005711                                                                   TST     (R1)
(9)  000342' 001462                                                                   BEQ     50010$
705  000344'                                IF #BIT15 SETIN @(R1) OR #BIT04 SETIN @(R1) THEN
(6)  000344' 032771  100000  000000                                                   BIT     #BIT15,@(R1)
(8)  000352' 001004                                                                   BNE     50011$
(6)  000354' 032771  000020  000000                                                   BIT     #BIT04,@(R1)
(9)  000362' 001451                                                                   BEQ     50012$
(6)  000364'                                                                  50011$:
706  000364'                                    LET R3 := (R1)+
(4)  000364' 012103                                                                   MOV     (R1)+,R3
707  000366'                                    CALL BOA16 IN <R3,#PA.CSR>
(3)  000366' 010546                                                                   MOV     R5,-(SP)
(5)  000370' 012745  000056'                                                          MOV     #PA.CSR,-(R5)
(4)  000374' 010345                                                                   MOV     R3,-(R5)
(3)  000376' 004767  000000G                                                          JSR     PC,BOA16
(3)  000402' 012605                                                                   MOV     (SP)+,R5
708  000404'                                    CALL BOA16 IN <(R3),#PA.CSC>
(3)  000404' 010546                                                                   MOV     R5,-(SP)
(5)  000406' 012745  000066'                                                          MOV     #PA.CSC,-(R5)
(4)  000412' 011345                                                                   MOV     (R3),-(R5)
(3)  000414' 004767  000000G                                                          JSR     PC,BOA16
(3)  000420' 012605                                                                   MOV     (SP)+,R5
709  000422'                                    LET PA.SP :B= #SPACE
(4)  000422' 112767  000040  177404                                                   MOVB    #SPACE,PA.SP
710  000430'                                    CALL MSGDHOOK IN <R0,#MSGPOP,#PA.MSG,#20$>
(3)  000430' 010546                                                                   MOV     R5,-(SP)
(7)  000432' 012745  000462'                                                          MOV     #20$,-(R5)
(6)  000436' 012745  000000'                                                          MOV     #PA.MSG,-(R5)
(5)  000442' 012745  000002                                                           MOV     #MSGPOP,-(R5)
(4)  000446' 010045                                                                   MOV     R0,-(R5)
(3)  000450' 004767  000000G                                                          JSR     PC,MSGDHOOK
(3)  000454' 012605                                                                   MOV     (SP)+,R5
711  000456'                                    INLINE <60$:    NOP>
(2)  000456' 000240                                                           60$:    NOP
712  000460'                                    INLINE <BR      60$>
```

```
(2)  000460' 000776                                                                    BR      60$
713  000462'                                  INLINE <20$:>
(2)  000462'                                                                           20$:
714  000462'                                  LET PA.CNT := PA.CNT + #1
(6)  000462' 005267  177410                                                            INC     PA.CNT
715  000466'                                  IF PA.CNT GE PA.MAX THEN
(6)  000466' 026767  177404  177404                                                    CMP     PA.CNT,PA.MAX
(9)  000474' 002403                                                                    BLT     50013$
716  000476'                                          LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6)  000476' 052760  100000  000020                                                    BIS     #FATERR,DT.ERR(R
717  000504'                                  ENDIF
(4)  000504'                                                                 50013$:
718  000504'                                  INLINE <BR 100$>
(2)  000504' 000424                                                                    BR 100$
719
720  000506'                                      ENDIF
(4)  000506'                                                                 50012$:
721  000506'                                 ENDDO
(4)  000506' 000714                                                                    BR      50007$
(3)  000510'                                                                 50010$:
722  000510'                             ENDIF
(4)  000510'                                                                 50003$:
723
724                                      ;+
725                                      ; NO PARITY OR ECC ENABLED SO ISSUE ERROR MSG -
726                                      ; OR NO ERROR WAS FOUND IN THE PARITY CSRS.
727                                      ; DO NOT PRINT THE MSG IF THIS IS AN 11/70 BECAUSE
728                                      ; IT DOES NOT HAVE PARITY CSRS. IT MAY OR MAY NOT
729                                      ; HAVE ECC CSRS IN THE TABLE
730                                      ;-
731
732
733
734  000510'                             IF #PDP70 NOTSETIN DT.CF0(R0) THEN
(6)  000510' 032760  010000  000014                                                    BIT     #PDP70,DT.CF0(R0
(9)  000516' 001017                                                                    BNE     50014$
735
736  000520'                                     LET PA.SP :B= #0
(4)  000520' 105067  177310                                                            CLRB    PA.SP
737  000524'                                     CALL MSGDHOOK IN <R0,#MSGPOP,#PA.MSG,#9$>
(3)  000524' 010546                                                                    MOV     R5,-(SP)
(7)  000526' 012745  000556'                                                           MOV     #9$,-(R5)
(6)  000532' 012745  000000'                                                           MOV     #PA.MSG,-(R5)
(5)  000536' 012745  000002                                                            MOV     #MSGPOP,-(R5)
(4)  000542' 010045                                                                    MOV     R0,-(R5)
(3)  000544' 004767  000000G                                                           JSR     PC,MSGDHOOK
(3)  000550' 012605                                                                    MOV     (SP)+,R5
738  000552'                                     INLINE <4$:     NOP>
(2)  000552' 000240                                                                    4$:     NOP
739  000554'                                     INLINE <BR      4$>
(2)  000554' 000776                                                                    BR      4$
740  000556'                                     INLINE <9$:>
(2)  000556'                                                                           9$:
741
742  000556'                             ENDIF
(4)  000556'                                                                 50014$:
```

```
                                ;+
                                ; IF INDIRECT CSRS THEN RESTORE CACHE, KT AND MAP REGISTERS
                                ;-

000556'                         INLINE <100$:>
000556'                                                                 100$:
000556'                         IF #INDPAR SETIN DT.CF0(R0) THEN
000556' 032760  000040  000014                                          BIT     #INDPAR,DT.CF0(R
000564' 001414                                                          BEQ     50015$
000566'                                 LET @#KIPAR6 := (SP)+
000566' 012637  172354                                                  MOV     (SP)+,@#KIPAR6
000572'                                 LET @#PA.M37 := (SP)+
000572' 012637  170372                                                  MOV     (SP)+,@#PA.M37
000576'                                 LET @#PA.M36 := (SP)+
000576' 012637  170370                                                  MOV     (SP)+,@#PA.M36
000602'                                 LET @#SR3 := (SP)+
000602' 012637  172516                                                  MOV     (SP)+,@#SR3
000606'                                 LET @#SR0 := (SP)+
000606' 012637  177572                                                  MOV     (SP)+,@#SR0
000612'                                 LET @CCNTRL := (SP)+
000612' 012677  000000G                                                 MOV     (SP)+,@CCNTRL

000616'                         ENDIF
000616'                                                         50015$:
                                ;+
                                ; RESTORE REGISTERS AND RETURN
                                ;-

000616'                         CALL RESREG
000616' 004767  000000G                                                 JSR     PC,RESREG


000622'                         ENDRTN
000622'                                                         50000$:
000622'                                                         50001$:
000622' 000207                                                          RTS     PC
        000001                  .END
```

```
ACSR  = 000102        CSRA  = 000100        ENBNUL= 000001        MED   = 076600        PRI6  = 000300
ACTBIT= 004000        CSRC  = 000102        ENDLST= 000000        MEMPAS= 040000        PRI7  = 000340
ADDR22= 001000        CTRLC = 000003        EOPBIT= 000001        MODEXH= 004000        PR0   = 000000
ADR   = 000006        CTRLO = 000017        ERRLOG= ****** G      MODHOL= 002000        PR4   = 000200
APTFER= 000004        CTRLU = 000025        ERRTYP= 000106        MODSEL= 001000        PR5   = 000240
APTPRE= 000200        DCEVNT= 000011        EVNTBE= 000200        MSGCKD= 000010        PR6   = 000300
ASB   = 000106        DEFRTN= 000400        EVNTHD= 000200        MSGCKS= 000011        PR7   = 000340
ASSEMB= 000010        DIAGMC= 000000        EVNTKT= 000203        MSGDER= 000005        PS    = 177776
ASTAT = 000104        DROPMO= 100000        EVNTPE= 000202        MSGDHO= ****** G      PSW   = 177776
AUTO  = 000010        DSEVNT= 000014        EVNTRE= 000201        MSGDRP= 000017        RANNUM= 000054
AUTOST= 020000        DTABLE= 000000        FATERR= 100000        MSGECH= 177777        RBUFEA= 000130
AWAS  = 000110        DT.ADD= 000042        HRDCNT= 000044        MSGEOP= 000013        RBUFPA= 000126
BIT0  = 000001        DT.AP = 000100        HRDPAS= 000050        MSGHDR= 000004        RBUFSZ= 000132
BIT00 = 000001        DT.APK= 000076        ICONT = 000036        MSGHNG= 000022        RBUFVA= 000124
BIT01 = 000002        DT.BLS= 000034        ICOUNT= 000040        MSGHRD= 000007        RDSERV= 000101
BIT02 = 000004        DT.CF0= 000014        IDNUM = 000122        MSGMAP= 000021        RDWHMI= 000022
BIT03 = 000010        DT.CF1= 000016        IE    = 000100        MSGNUL= 177775        RELERR= 000020
BIT04 = 000020        DT.ERR= 000020        INDPAR= 000040        MSGPOP= 000002        RELMOD= 020000
BIT05 = 000040        DT.ESI= 000044        INHDRP= 040000        MSGPRM= 177776        RELTIM= 010000
BIT06 = 000100        DT.EVN= 000000        INHEPR= 020000        MSGRES= 000001        RESREG= ****** G
BIT07 = 000200        DT.EXS= 000060        INHREL= 001000        MSGSFT= 000006        RES1  = 000056
BIT08 = 000400        DT.FCH= 000037        INHRRE= 000400        MSGSKE= 000003        RES2  = 000060
BIT09 = 001000        DT.FCN= 000036        INIT  = 000030        MSGSMB= 000015        RICHAR= 031060
BIT1  = 000002        DT.HMX= 000104        INTR  = 000120        MSGSMH= 000014        RPTDAT= 002000
BIT10 = 002000        DT.KBE= 000024        IOMOD = 100000        MSGSMS= 000016        RSTRCY= ****** G
BIT11 = 004000        DT.KBP= 000026        IOMODP= 102000        MSGSTD= 000000        RSTRT = 000112
BIT12 = 010000        DT.KBR= 000022        IOMODR= 112000        MSGSYS= 000012        RUBOUT= 000177
BIT13 = 020000        DT.KBU= 000030        IOMODX= 110000        MSGVEC= 000020        RUNMOD= 100000
BIT14 = 040000        DT.MLS= 000032        JACK  = 035060        NBKMOD= 001000        R5VALU= 001740
BIT15 = 100000        DT.MTI= 000110        KIPAR0= 172340        NCPUOP= 000020        SAM   = 075464
BIT2  = 000004        DT.OFF= 000070        KIPAR1= 172342        NOAPTY= 000002        SAVREG= ****** G
BIT3  = 000010        DT.PAS= 000074        KIPAR2= 172344        NULL  = 000000        SBADR = 000102
BIT4  = 000020        DT.PC = 000002        KIPAR3= 172346        OWEN  = 024020        SBKMOD= 000000
BIT5  = 000040        DT.PFL= 000062        KIPAR4= 172350        PAERR = 000010        SBKSEL= 010000
BIT6  = 000100        DT.PSW= 000004        KIPAR5= 172352        PARERR  000102RG      SC.ADR= 000006
BIT7  = 000200        DT.PTA= 000064        KIPAR6= 172354        PARPRE= 002000        SC.ALC= 000014
BIT8  = 000400        DT.RCS= 000102        KIPAR7= 172356        PARSTA= 000100        SC.APC= 000016
BIT9  = 001000        DT.REL= 000040        KIPDR0= 172300        PASCNT= 000034        SC.CKL= 000002
BKDEF = 000002        DT.SCT= 000066        KIPDR1= 172302        PA.CNT  000076RG      SC.CKP= 000004
BKMOD = 000020        DT.SMX= 000106        KIPDR2= 172304        PA.CSC  000066R       SC.CLO= 000000
BKMODE= 040000        DT.SP = 000006        KIPDR3= 172306        PA.CSR  000056R       SC.HLD= 000010
BKSLSH= 000134        DT.SSI= 000046        KIPDR4= 172310        PA.MAX  000100R       SC.SCA= 000012
BOA16 = ****** G      DT.ST0= 000010        KIPDR5= 172312        PA.MSG  000000R       SENDLS= 177777
CAPRES= 000004        DT.ST1= 000012        KIPDR6= 172314        PA.M36= 170370        SOFCNT= 000042
CASTAT= 000004        DT.SWR= 000056        KIPDR7= 172316        PA.M37= 170372        SOFPAS= 000046
CCNTRL= ****** G      DT.SYP= 000072        KTERRO= 000040        PA.SP   000034R       SPACE = 000040
CDERCT= 000146        DT.WBU= 000050        KTPRES= 000400        PDPLSI= 020000        SPOINT= 000032
CDWDCT= 000144        DT.WHL= 000054        KTSET = ****** G      PDP60 = 004000        SPVALU= 002200
CKTIM = 100000        DT.WLL= 000052        KTSTAT= 000020        PDP70 = 010000        SR0   = 177572
CLKPRE= 000001        DVID1 = 000014        KTXTND= 040000        PRI0  = 000000        SR1   = 177574
CONFIG= 000056        ECCMEM= 000100        LF    = 000012        PRI1  = 000040        SR2   = 177576
CQCVF = 000001        ECCSTA= 000010        LPSTAT= 000001        PRI4  = 000200        SR3   = 172516
CR    = 000015        ENBEOP= 010000        MAPSTA= 000200        PRI5  = 000240        STAT  = 000026
```

```
STATBI= 064757          UIPDR1= 177602          $F$BAD= 000401          $ISK1 = 000001          $$CASE= 000000
STAT1 = 000027          UIPDR2= 177604          $F$BLA= 000170          $ISK2 = 000001          $$DST = 000000
SUSPND= 000001          UIPDR3= 177606          $F$CAS= 000150          $LOCTA= 177777          $$ELOC= 000402
SVR0  = 000062          UIPDR4= 177610          $F$DEC= 000220          $LSTIN= 000001          $$ERFL= 000000
SVR1  = 000064          UIPDR5= 177612          $F$DO = 000340          $LSTTA= 000001          $$FLAG= 000001
SVR2  = 000066          UIPDR6= 177614          $F$FAL= 000405          $NESTL= 177777          $$FROM= 000000
SVR3  = 000070          UIPDR7= 177616          $F$GOO= 000400          $NSK0 = 000000          $$LOC = 000564R
SVR4  = 000072          WASADR= 000104          $F$IF = 000110          $NSK1 = 000110          $$LOCN= 000000
SVR5  = 000074          WBSTAT= 000040          $F$INC= 000210          $NSK2 = 000120          $$REG = 177777
SVR6  = 000076          WBUFEA= 000136          $F$LOO= 000200          $NSK3 = 000110          $$RETU= 000000
SYSCNT= 000052          WBUFPA= 000134          $F$NAM= 000160          $NSK4 = 000110          $$RTN1= 050000
SYSERR= 000100          WBUFRQ= 000140          $F$NO = 000403          $SAVLE= 177777          $$RTN2= 050001
TMPIO = 000002          WBUFSZ= 000142          $F$OR = 000320          $SSK0 = 050010          $$SRC = 000000
TQOVF = 000002          WDFR  = 000116          $F$RTI= 000350          $TAGLE= 177777          $$TGSV= 000000
UIPAR0= 177640          WDTO  = 000114          $F$RTN= 000300          $TAGNU= 050016          $$TGS1= 000000
UIPAR1= 177642          WTINRE= 000352          $F$SEL= 000140          $TEMP = 000300          $$TGS2= 000000
UIPAR2= 177644          WTWHMI= 000222          $F$THE= 000330          $TSK0 = 050015          $$TO  = 000000
UIPAR3= 177646          XFLAG = 000005          $F$TRU= 000404          $TSK1 = 050007          $$$TAG= 050000
UIPAR4= 177650          XOFF  = 000023          $F$UNT= 000130          $TSK2 = 050010          .     = 000624R
UIPAR5= 177652          XON   = 000021          $F$WHI= 000120          $TSK3 = 050012
UIPAR6= 177654          $BGNLE= 177777          $F$YES= 000402          $TSK4 = 050013
UIPAR7= 177656          $ERFLG= 000400          $IFLEV= 177777          $$ARGC= 000002
UIPDR0= 177600          $F$AND= 000310          $ISK0 = 000001          $$BYTE= 000403

. ABS.  000000    000
        000624    001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:PARERR,DSKZ:PARERR=SPMAC/ML,EQUATE,PARERR
RUN-TIME: 17 7 .4 SECONDS
RUN-TIME RATIO: 41/25=1.6
CORE USED:  14K  (27 PAGES)
```

SEQ 0243

```
TABL  = 000000          WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$ELOC= 000402
TMPIO = 000002          WBSTAT= 000040          $F$DO = 000340          $ISK1 = 000001          $$ERFL= 000000
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LOCTA= 177777          $$FLAG= 000001
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LSTIN= 000001          $$FROM= 000000
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $LSTTA= 000001          $$LOC = 000172R
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NESTL= 177777          $$LOCN= 000000
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK0 = 000300          $$REG = 177777
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK1 = 000110          $$RETU= 000000
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $NSK2 = 000110          $$RTN1= 050000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $SAVLE= 177777          $$RTN2= 050001
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $TAGLE= 177777          $$SRC = 000000
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $TAGNU= 050012          $$TGSV= 000000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TEMP = 000300          $$TGS1= 000000
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TSK0 = 050007          $$TGS2= 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TSK1 = 050011          $$TO  = 000002
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $$ARGC= 000002          $$$TAG= 050000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$BYTE= 000403          .     = 000216R
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$CASE= 000000
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$DST = 000000

. ABS.  000000     000
        000216     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RSTRCY,DSKZ:RSTRCY=SPMAC/ML,EQUATE,RSTRCY
 RUN-TIME: 14 4 .3 SECONDS
 RUN-TIME RATIO: 37/19=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PCDATA (PROCESS CDATA & DATCK TRAP CALLS)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PCDATA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS IS A TWO ENTRY MODULE (PCDATA & PDATCK BEING ENTRY POINTS).
                        ;       THIS MODULE IS CALLED AS A RESULT OF CDATA OR DATCK TRAP CALL
                        ;       EXECUTED BY AN OPTION MODULE, TO CHECK A BLOCK OF DATA
                        ;       TRANSFERRED BY A DEVICE.
                        ;
                        ;       IF THERE ARE NO ERRORS IN THE BLOCK, CONTROL RETURNS IMME-
                        ;       DIATELY TO THE INSTRUCTION FOLLOWING THE TRAP CALL IN THE MODULE.
                        ;
                        ;       IF THERE ARE ERRORS, HOWEVER, AN ERROR MESSAGE IS ENQUEUED (WITH
                        ;       TRAP ADDRESS FILLED AS THE MODULE'S RETURN ADDRESS), WORD
                        ;       COUNT AND ERROR COUNT ARE SAVED IN THE MODULE HEADER.
                        ;
                        ;       IF 'PRINT ALL DATA ERRORS' SWITCH IS SET, ALL ERRORS ARE
                        ;       ENQUEUED, OTHERWISE ONLY THE FIRST THREE ERRORS ARE ENQUEUED AND
                        ;       THE REST ARE LOGGED. THE SOFT ERROR COUNT IS INCREMENTED EVERY
                        ;       TIME AN ERROR OCCURS AND AT THE END OF THE BLOCK CHECK, A SUMMARY MESSAGE
                        ;       IS PRINTED FOR CDATA CALLS. SOFT
                        ;       ERROR WORD AND SOFT ERRORS PER PASS WORD ARE INCREMENTED IN THE MODULE'S
                        ;       HEADER.
                        ;
                        ;       IF THE TRAP CALL IS DATCK, THE TOTAL NUMBER OF ERRORS IS PASSED
                        ;       TO A LOCATION SPECIFIED IN THE TRAP CALL.
                        ;
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC
                        ;       2. DT.ST0
                        ;       3. DT.SWR
                        ;       4. DT.EVNT
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.ST0
                        ;       2. DT.PC
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ENQTQ        ;ENQUEUES AN ERROR MESSAGE
                        ;       2. UNIPA        ;MAP22 ROUTINE
                        ;       3. SAVREG       ;
                        ;       4. RESREG       ;
```

```
                        ;       5. GETPSW       ;GETS PS WORD
                        ;       6. APTSER       ;APT SOFT ERROR ROUTINE
                        ;       7. DRPMOD       ;DROP A MODULE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PCDATA IN <AA>
                        ;       CALL PDATCK IN <AA>
                        ;
                        ;                       WHERE AA = DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL PCDATA (COMMON DEFINITIONS AND REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN=1
            000001              $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL PCDATA           ;MODULE ENTRY POINT
                                .GLOBL PDATCK           ;MODULE ENTRY POINT
                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL ENQTQ            ;ENQUE A MESSAGE ROUTINE
                                .GLOBL UNIPA            ;MAP22 ROUTINE
                                .GLOBL  DRPMOD          ;DROP A MODULE
                                .GLOBL SAVREG
                                .GLOBL RESREG
                                .GLOBL GETPSW           ;GET PS WORD ROUTINE
                                .GLOBL  APTSER


                                ;****************
                                ; LOCAL STORAGE
                                ;
                                ; NOTE: DO NOT DISTURB THE ORDER OF THE NEXT 10 WORDS AS THEY FORM
                                ;       A TABLE.

000000' 000000                  PC.RPA: 0               ;RBUFPA WORD
000002' 000000                  PC.REA: 0               ;RBUFEA WORD
000004' 000000                  PC.RLO: 0               ;RBUF LOWER 16 BITS
000006' 000000                  PC.RHI: 0               ;RBUF UPPER 2 BITS/6 BITS
000010' 000000                  PC.RTP: 0               ;RBUF TOP 12 BITS/16 BITS
000012' 000000                  PC.WPA: 0               ;WBUFPA WORD
000014' 000000                  PC.WEA: 0               ;WBUFEA WORD
000016' 000000                  PC.WLO: 0               ;WBUF LOWER 16 BITS
000020' 000000                  PC.WHI: 0               ;WBUF UPPER 2 BITS/6 BITS
000022' 000000                  PC.WTP: 0               ;WBUF TOP 12 BITS/16 BITS

000024' 000000                  PC.RSZ: 0               ;READ BUFFER SIZE
000026' 000000                  PC.TPC: 0               ;PC OF CALL
000030' 000000                  PC.TMP: 0               ;TEMP STORAGE
000032' 000000                  PC.DCF: 0               ;DATCK FLAG
000034' 000000                          .WORD   0       ;USER MODE STACK
000036' 000000                          .WORD   0
000040' 000000                          .WORD   0
000042' 000000                          .WORD   0

```

```
                                .SBTTL PCDATA (CODE)

000044'                         ROUTINE PCDATA <DT>
(2) 000044'                                                             PCDATA:

                                LET PC.DCF := #0
(4) 000044' 005067  177762                                                      CLR     PC.DCF
000050'                         INLINE <BR COM>
(2) 000050' 000403                                                              BR COM

000052'                         ROUTINE PDATCK <DT>
(2) 000052'                                                             PDATCK:

000052'                         LET PC.DCF := #1
(4) 000052' 012767  000001  177752                                              MOV     #1,PC.DCF

                                ;+
                                ; SAVE REGISTERS
                                ;-

000060'                         INLINE <COM:>
(2) 000060'                                                                     COM:
                                CALL SAVREG
(3) 000060' 004767  000000G                                                     JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO DTABLE ADDRESS, R1 TO HEADER ADDRESS AND R2 TO TABLE ADDRESS
                                ; POINTING TO RBUF PA,EA & SIZE.  R3 AND R4 WILL BE POINTERS TO READ AND
                                ; WRITE BUFFERS RESPECTIVELY. R2 ALSO SERVES AS A WHILE-DO LOOP CONTROL.
                                ;-

000064'                         LET R0 := DT(R5)
(4) 000064' 016500  000000                                                      MOV     DT(R5),R0
000070'                         LET R1 := @DT.PC(R0)
(4) 000070' 017001  000002                                                      MOV     @DT.PC(R0),R1

                                ;+
                                ; DEVELOP RETURN ADDRESS FOR MESSAGE
                                ;-
000074'                         LET PC.TPC := DT.PC(R0) - #2
(4) 000074' 016067  000002  177724                                              MOV     DT.PC(R0),PC.TPC
(6) 000102' 162767  000002  177716                                              SUB     #2,PC.TPC

                                ;+
                                ; GET ADDRESS OF RBUF,PA,EA
                                ;-

000110'                         LET R2 := PC.TPC + #4
(4) 000110' 016702  177712                                                      MOV     PC.TPC,R2
(6) 000114' 062702  000004                                                      ADD     #4,R2

                                ;+
                                ; R3 GETS PA WORD AND SAVE ALSO IN PC.RPA
                                ;-

000120'                         LET R2 := (R2)
```

```
   (4)  000120' 011202                                                                  MOV     (R2),R2
577  000122'                        LET R3 := (R2)+
   (4)  000122' 012203                                                                  MOV     (R2)+,R3
578  000124'                        LET PC.RPA := R3
   (4)  000124' 010367  177650                                                          MOV     R3,PC.RPA
579
580                                 ;+
581                                 ; SAVE EA BITS AND SIZE
582                                 ;-
583
584  000130'                        LET PC.REA := (R2)+
   (4)  000130' 012267  177646                                                          MOV     (R2)+,PC.REA
585  000134'                        LET PC.RSZ := (R2)
   (4)  000134' 011267  177664                                                          MOV     (R2),PC.RSZ
586                                 ;
587                                 ;+
588                                 ; GET WBUFPA INTO R4
589                                 ;-
590
591  000140'                        LET R4 := WBUFPA(R1)
   (4)  000140' 016104  000134                                                          MOV     WBUFPA(R1),R4
592
593                                 ;+
594                                 ; FINALLY GET WBUFEA BITS INTO LOCAL STORAGE
595                                 ;-
596
597  000144'                        LET PC.WPA := R4
   (4)  000144' 010467  177642                                                          MOV     R4,PC.WPA
598  000150'                        LET PC.WEA := WBUFEA(R1)
   (4)  000150' 016167  000136  177636                                                  MOV     WBUFEA(R1),PC.WE
599
700                                 ;+
701                                 ; IF BIT15 OF WORD COUNT WORD OF MODULE HEADER IS NOT SET THEN YOU
702                                 ; ARE COMING TO THIS MODULE FOR THE FIRST TIME  -  CLEAR THE WORD
703                                 ; COUNT AND ERROR COUNT WORDS IN THE MODULE HEADER.
704                                 ;-
705
706  000156'                        LET R2 := CDWDCT(R1)
   (4)  000156' 016102  000144                                                          MOV     CDWDCT(R1),R2
707  000162'                        IFCOND PL THEN
   (6)  000162  100406                                                                  BMI     50002$
708  000164'                            LET R2 := #0
   (4)  000164' 005002                                                                  CLR     R2
709  000166'                            LET CDWDCT(R1) := #0
   (4)  000166' 005061  000144                                                          CLR     CDWDCT(R1)
710  000172'                            LET CDERCT(R1) := #0
   (4)  000172' 005061  000146                                                          CLR     CDERCT(R1)
711  000176'                        ELSE
   (4)  000176' 000421                                                                  BR      50003$
   (3)  000200'                                                                 50002$:
712
713                                 ;+
714                                 ; ELSE YOU WERE HERE
715                                 ; BEFORE FROM THE SAME TRAP  -  MODIFY THE WORD COUNT, THE WRITE
716                                 ; BUFFER AND THE READ BUFFER POINTERS ACCORDINGLY.
717                                 ;-
```

```
                                    LET R2 := R2 CLR.BY #BIT15
(6)  000200' 042702  100000                                                 BIC     #BIT15,R2
                                    LET R2 := R2 + #1           ;UPDATE WORD COUNT
(6)  000204' 005202                                                         INC     R2

                                ;+
                                ; UPDATE READ BUFFER POINTER AND EA BITS
                                ;-

                                    LET R2 := R2 SHIFT 1        ;FORM OFFSET
(7)  000206' 006302                                                         ASL     R2
                                    LET R3 := R3 + R2           ;NEW RBUF PTR
(6)  000210' 060203                                                         ADD     R2,R3
                                    LET PC.RPA := R3
(4)  000212' 010367  177562                                                 MOV     R3,PC.RPA
                                    IFCOND CS THEN
(6)  000216' 103003                                                         BCC     50004$
                                        LET PC.REA := PC.REA + #20
(6)  000220' 062767  000020  177554                                         ADD     #20,PC.REA
                                    ENDIF
(4)  000226'                                                        50004$:

                                ;+
                                ; UPDATE WRITE BUFFER POINTER AND IT'S EA BITS
                                ;-

                                    LET R4 := R4 + R2           ;NEW WBUFF PTR
(6)  000226' 060204                                                         ADD     R2,R4
                                    IFCOND CS THEN
(6)  000230' 103003                                                         BCC     50005$
                                        LET PC.WEA := PC.WEA + #20
(6)  000232' 062767  000020  177554                                         ADD     #20,PC.WEA
                                    ENDIF
(4)  000240'                                                        50005$:

                                ;+
                                ; RESTORE WORD COUNT
                                ;-

                                    LET R2 := R2 SHIFT -1
(7)  000240' 006202                                                         ASR     R2
                                ENDIF
(4)  000242'                                                        50003$:

                                ;+
                                ; NOW LOAD LOCAL STORAGE WITH WRITE BUFFER PA IN R4
                                ;-

                                LET PC.WPA := R4
(4)  000242' 010467  177544                                                 MOV     R4,PC.WPA

                                ;+
                                ; IF KT IS ON THEN DETERMINE PHYSICAL ADDRESS
                                ; IN PAR FORMAT.....
                                ;-
```

```
  759  000246'                          IF #KTSTAT SETIN DT.STO(R0) THEN
  (6)  000246' 032760  000020  000010                                                BIT     #KTSTAT,DT.STO(R
  (9)  000254' 001515                                                                BEQ     50006$
  760
  761                                   ;+
  762                                   ; IF 22 BIT ADDRESSING ENABLED, GET 22 BIT PHYSICAL ADDRESS
  763                                   ;-
  764
  765  000256'                              IF #MAPSTAT SETIN DT.STO(R0) THEN
  (6)  000256' 032760  000200  000010                                                BIT     #MAPSTAT,DT.STO(
  (9)  000264' 001431                                                                BEQ     50007$
  766  000266'                                  CALL UNIPA IN <#PC.RPA>
  (3)  000266' 010546                                                                MOV     R5,-(SP)
  (4)  000270' 012745  000000'                                                       MOV     #PC.RPA,-(R5)
  (3)  000274' 004767  000000G                                                       JSR     PC,UNIPA
  (3)  000300' 012605                                                                MOV     (SP)+,R5
  767  000302'                                  CALL UNIPA IN <#PC.WPA>
  (3)  000302' 010546                                                                MOV     R5,-(SP)
  (4)  000304' 012745  000012'                                                       MOV     #PC.WPA,-(R5)
  (3)  000310' 004767  000000G                                                       JSR     PC,UNIPA
  (3)  000314' 012605                                                                MOV     (SP)+,R5
  768  000316'                                  CALL TOP16 IN <#PC.RLO>
  (3)  000316' 010546                                                                MOV     R5,-(SP)
  (4)  000320' 012745  000004'                                                       MOV     #PC.RLO,-(R5)
  (3)  000324' 004767  000730                                                        JSR     PC,TOP16
  (3)  000330' 012605                                                                MOV     (SP)+,R5
  769  000332'                                  CALL TOP16 IN <#PC.WLO>
  (3)  000332' 010546                                                                MOV     R5,-(SP)
  (4)  000334' 012745  000016'                                                       MOV     #PC.WLO,-(R5)
  (3)  000340' 004767  000714                                                        JSR     PC,TOP16
  (3)  000344' 012605                                                                MOV     (SP)+,R5
  770  000345'                              ELSE
  (4)  000346' 000414                                                                BR      50010$
  (3)  000350'                                                                50007$:
  771  000350'                                  CALL TOP12 IN <#PC.RPA>
  (3)  000350' 010546                                                                MOV     R5,-(SP)
  (4)  000352' 012745  000000'                                                       MOV     #PC.RPA,-(R5)
  (3)  000356' 004767  000752                                                        JSR     PC,TOP12
  (3)  000362' 012605                                                                MOV     (SP)+,R5
  772  000364'                                  CALL TOP12 IN <#PC.WPA>
  (3)  000364' 010546                                                                MOV     R5,-(SP)
  (4)  000366' 012745  000012'                                                       MOV     #PC.WPA,-(R5)
  (3)  000372' 004767  000736                                                        JSR     PC,TOP12
  (3)  000376' 012605                                                                MOV     (SP)+,R5
  773  000400'                              ENDIF
  (4)  000400'                                                                50010$:
  774
  775                                   ;+
  776                                   ; LOAD PARS WITH VALUES PREVIOUSLY DETERMINED
  777                                   ;-
  778
  779  000400'                              LET @#UIPAR0 := @#KIPAR0
  (4)  000400' 013737  172340  177640                                                MOV     @#KIPAR0,@#UIPAR
  780  000406'                              LET @#UIPAR5 := PC.WTP
  (4)  000406' 016737  177410  177652                                                MOV     PC.WTP,@#UIPAR5
  781  000414'                              LET @#UIPAR6 := PC.RTP
```

```
   (4)  000414' 016737  177370  177654                                                    MOV     PC.RTP,@#UIPAR6
   782  000422'                        LET @#UIPAR7 := #177600
   (4)  000422' 012737  177600  177656                                                    MOV     #177600,@#UIPAR7
   783
   784                                ;+
   785                                ; SET R3 TO SELECT PAR6 AND R4 TO SELECT PAR5 AND ALSO
   786                                ; LOAD R3 AND R4 WITH LOWER 6 BITS OF RBUF AND WBUF ADDRESSES
   787                                ;-
   788
   789  000430'                           LET R3 := R3 CLR.BY #177700
   (6)  000430' 042703  177700                                                            BIC     #177700,R3
   790  000434'                           LET R3 := R3 SET.BY #140000
   (6)  000434' 052703  140000                                                            BIS     #140000,R3
   791  000440'                           LET R4 := R4 CLR.BY #177700
   (6)  000440' 042704  177700                                                            BIC     #177700,R4
   792  000444'                           LET R4 := R4 SET.BY #120000
   (6)  000444' 052704  120000                                                            BIS     #120000,R4
   793
   794                                ;+
   795                                ; GET THE PSW AND GO TO USER MODE
   796                                ;-
   797
   798  000450'                           CALL GETPSW OUT <PC.TMP>
   (4)  000450' 162705  000002                                                            SUB     #1*2,R5
   (3)  000454' 004767  000000G                                                          JSR     PC,GETPSW
   (4)  000460' 012567  177344                                                            MOV     (R5)+,PC.TMP
   799  000464'                           LET PC.TMP := PC.TMP SET.BY #140000
   (6)  000464' 052767  140000  177336                                                    BIS     #140000,PC.TMP
   800  000472'                           PUSH PC.TMP,#33$
   (2)  000472' 016746  177332                                                            MOV     PC.TMP,-(SP)
   (3)  000476' 012746  000504'                                                          MOV     #33$,-(SP)
   801  000502'                           INLINE <RTI>
   (2)  000502' 000002                                                                    RTI
   802  000504'                          INLINE <33$:>
   (2)  000504'                                                                           33$:
   803                                ;+
   804                                ; SET UP USER MODE STACK
   805                                ;-
   806
   807  000504'                           LET SP := #PCDATA
   (4)  000504' 012706  000044'                                                          MOV     #PCDATA,SP
   808
   809  000510'                        ENDIF
   (4)  000510'                                                                                   50006$:
   810
   811  000510'                        INLINE <1$:>
   (2)  000510'                                                                           1$:
   812                                ;+
   813                                ; NOW WE ARE READY TO COMPARE DATA WORDS
   814                                ;-
   815
   816  000510'                        WHILE R2 LT PC.RSZ DO
   (4)  000510'                                                                                   50011$:
   (6)  000510' 020267  177310                                                                    CMP     R2,PC.RSZ
   (9)  000514' 002113                                                                            BGE     50012$
   817  000516'                              LET R2 := R2 + #1
```

```
(6)  000516' 005202                                                                    INC     R2
818  000520'                          IF (R3) NE (R4) THEN
(6)  000520' 021314                                                                    CMP     (R3),(R4)
(9)  000522' 001461                                                                    BEQ     50013$
819
820                                ;+
821                                ; WORDS DID NOT COMPARE.... UPDATE ERROR COUNTS AND SEE IF
822                                ; MESSAGE SHOULD BE OUTPUTTED.
823                                ;-
824
825  000524'                             LET CDERCT(R1) := CDERCT(R1) + #1
(6)  000524' 005261  000146                                                            INC     CDERCT(R1)
826  000530'                             IF CDERCT(R1) LE #3 OR   #BIT10 SETIN DT.SWR(R0) THEN
(6)  000530' 026127  000146  000003                                                    CMP     CDERCT(R1),#3
(8)  000536' 003404                                                                    BLE     50014$
(6)  000540' 032760  002000  000056                                                    BIT     #BIT10,DT.SWR(R0
(9)  000546' 001446                                                                    BEQ     50015$
(6)  000550'                                                                  50014$:
827  000550'                                 LET PC.WPA := (R3)
(4)  000550' 011367  177236                                                            MOV     (R3),PC.WPA
828  000554'                                 LET PC.WEA := (R4)
(4)  000554' 011467  177234                                                            MOV     (R4),PC.WEA
829  000560'                                 IF #KTSTAT SETIN DT.ST0(R0) THEN
(6)  000560' 032760  000020  000010                                                    BIT     #KTSTAT,DT.ST0(R
(9)  000566' 001406                                                                    BEQ     50016$
830  000570'                                     LET PC.TMP := PC.TMP CLR.BY #140000
(6)  000570' 042767  140000  177232                                                    BIC     #140000,PC.TMP
831                                ;+
832                                ; ************************************************************
833                                ; NOTE !!!!! DIRECT REFERENCE TO PSW FOLLOWS  !!!!!   IN AN LSI
834                                ; MACHINE WITH KT ( SHOULD THIS EVER EXIST ), DIRECT
835                                ; ACCESS TO PSW MAY NOT BE ALLOWED !!! THE USE OF SOME TRAP MAY
836                                ; BE NECESSARY TO SWITCH TO KERNEL !!!!
837                                ; ************************************************************
838                                ;-
839
840  000576'                                 LET @#PSW := PC.TMP
(4)  000576' 016737  177226  177776                                                    MOV     PC.TMP,@#PSW
841  000604'                                 ENDIF
(4)  000604'                                                                  50016$:
842
843                                ;+
844                                ; LOAD EXPECTED DATA WORD AND SHOULD BE DATA WORD IN MODULE HEADER, SET
845                                ; BIT15 IN WORD COUNT WORD IN MODULE HEADER AND ENQUE THE ERROR MESSAGE.
846                                ;-
847
848  000604'                                 LET ASB(R1) := PC.WEA
(4)  000604' 016761  177204  000106                                                    MOV     PC.WEA,ASB(R1)
849  000612'                                 LET AWAS(R1) := PC.WPA
(4)  000612' 016761  177174  000110                                                    MOV     PC.WPA,AWAS(R1)
850
851                                ;+
852                                ; LOAD THE "BEEN HERE BEFORE ON THIS TRAP FLAG"
853                                ;-
854
855  000620'                                 LET CDWDCT(R1) := R2 SET.BY #BIT15
```

```
(4)  000620' 010261  000144                                                    MOV     R2,CDWDCT(R1)
(6)  000624' 052761  100000  000144                                            BIS     #BIT15,CDWDCT(R1
856  000632'                          CALL ENQTQ IN <R0,#MSGCKD,PC.TPC,R1,PC.TPC>
(3)  000632' 010546                                                            MOV     R5,-(SP)
(8)  000634' 016745  177166                                                    MOV     PC.TPC,-(R5)
(7)  000640' 010145                                                            MOV     R1,-(R5)
(6)  000642' 016745  177160                                                    MOV     PC.TPC,-(R5)
(5)  000646' 012745  000010                                                    MOV     #MSGCKD,-(R5)
(4)  000652' 010045                                                            MOV     R0,-(R5)
(3)  000654' 004767  000000G                                                   JSR     PC,ENQTQ
(3)  000660' 012605                                                            MOV     (SP)+,R5
857  000662'                          INLINE <BR 3$>
(2)  000662' 000530                                                            BR 3$
858  000664'                      ENDIF
(4)  000664'                                                           50015$:
859  000664'                  ELSE
(4)  000664' 000426                                                            BR      50017$
(3)  000666'                                                           50013$:
860
861                           ;+
862                           ; NO ERROR IN DATA WORDS, INCREMENT POINTERS.
863                           ; UPDATE PAR REGISTERS IF NECESSARY
864                           ;-
865
866  000666'                      INLINE <TST (R3)+>
(2)  000666' 005723                                                            TST (R3)+
867  000670'                      INLINE <TST (R4)+>
(2)  000670' 005724                                                            TST (R4)+
868  000672'                      IF #KTSTAT SETIN DT.ST0(R0) THEN
(6)  000672' 032760  000020  000010                                            BIT     #KTSTAT,DT.ST0(R
(9)  000700' 001420                                                            BEQ     50020$
869  000702'                          IF R3 EQ #160000 THEN
(6)  000702' 020327  160000                                                    CMP     R3,#160000
(9)  000706' 001005                                                            BNE     50021$
870  000710'                              LET R3 := #140000
(4)  000710' 012703  140000                                                    MOV     #140000,R3
871  000714'                              LET @#UIPAR6 := @#UIPAR6 + #200
(6)  000714' 062737  000200  177654                                            ADD     #200,@#UIPAR6
872  000722'                          ENDIF
(4)  000722'                                                           50021$:
873  000722'                          IF R4 EQ #140000 THEN
(6)  000722' 020427  140000                                                    CMP     R4,#140000
(9)  000726' 001005                                                            BNE     50022$
874  000730'                              LET R4 := #120000
(4)  000730' 012704  120000                                                    MOV     #120000,R4
875  000734'                              LET @#UIPAR5 := @#UIPAR5 + #200
(6)  000734' 062737  000200  177652                                            ADD     #200,@#UIPAR5
876  000742'                          ENDIF
(4)  000742'                                                           50022$:
877  000742'                      ENDIF
(4)  000742'                                                           50020$:
878  000742'                  ENDIF
(4)  000742'                                                           50017$:
879  000742'              ENDDO
(4)  000742' 000662                                                            BR      50011$
(3)  000744'                                                           50012$:
```

```
                                ;+
                                ; NOW YOU GO TO KERNEL MODE, FOR SURE. CHECK IF THERE WERE ANY ERRORS.
                                ; IF NONE, UPDATE DT.PC FOR PROPER RETURN AND CLEAR THE DEFERRED SERVICE
                                ; BIT IN DT.ST0, AND THAT'S IT !!!
                                ;-

000744'                         IF #KTSTAT SETIN DT.ST0(R0) THEN
(6) 000744' 032760  000020  000010                                             BIT     #KTSTAT,DT.ST0(R
(9) 000752' 001406                                                              BEQ     50023$
000754'                             LET PC.TMP := PC.TMP CLR.BY #140000
(6) 000754' 042767  140000  177046                                             BIC     #140000,PC.TMP
                                ;+
                                ; **************************************************************
                                ; NOTE !!!!! DIRECT REFERENCE TO PSW FOLLOWS  !!!!!   IN AN LSI
                                ; MACHINE WITH KT ( SHOULD THIS EVER EXIST ), DIRECT
                                ; ACCESS TO PSW MAY NOT BE ALLOWED !!! THE USE OF SOME TRAP MAY
                                ; BE NECESSARY TO SWITCH TO KERNEL !!!!
                                ; **************************************************************
                                ;-

000762'                             LET @#PSW := PC.TMP
(4) 000762' 016737  177042  177776                                             MOV     PC.TMP,@#PSW
000770'                         ENDIF
(4) 000770'                                                             50023$:

                                ;+
                                ; IF ERRORS, UPDATE ERRORS COUNTS AND DO APT STUFF IF NECESSARY

000770'                         IF CDERCT(R1) NE #0 THEN
(6) 000770' 005761  000146                                                     TST     CDERCT(R1)
(9) 000774' 001447                                                             BEQ     50024$
000776'                         LET SOFCNT(R1) := SOFCNT(R1) + #1
(6) 000776' 005261  000042                                                     INC     SOFCNT(R1)
001002'                         LET SOFPAS(R1) := SOFPAS(R1) + #1
(6) 001002' 005261  000046                                                     INC     SOFPAS(R1)
                                ;+
                                ; IF UNDER APT, CALL APT SOFT ERROR ROUTINE
                                ;-

001006'                         IF #APTPRES SETIN DT.CF0(R0) THEN
(6) 001006' 032760  000200  000014                                             BIT     #APTPRES,DT.CF0(
(9) 001014' 001406                                                             BEQ     50025$
001016'                                 CALL APTSER IN <R0>
(3) 001016' 010546                                                             MOV     R5,-(SP)
(4) 001020' 010045                                                             MOV     R0,-(R5)
(3) 001022' 004767  000000G                                                    JSR     PC,APTSER
(3) 001026' 012605                                                             MOV     (SP)+,R5
001030'                         ELSE
(4) 001030' 000431                                                             BR      50026$
(3) 001032'                                                             50025$:

                                ;+
                                ; NOT APT.....SEE IF TOO MANY ERRORS
```

```
                                ;-
                                ;+
                                ; IF BIT 15 SET IN SWITCH REGISTER... OR IF TOO MANY SOFT ERRORS AND
                                ; BIT 14 SETIN SWITCH REGISTER... ALSO DUMP MODULE....
                                ;-
001032'                                 IF #BIT15 SETIN DT.SWR(R0) THEN
(6) 001032' 032760 100000 000056                                                BIT     #BIT15,DT.SWR(R0
(9) 001040' 001407                                                              BEQ     50027$
001042'                                         CALL DRPMOD IN <R0,R1>
(3) 001042' 010546                                                              MOV     R5,-(SP)
(5) 001044' 010145                                                              MOV     R1,-(R5)
(4) 001046' 010045                                                              MOV     R0,-(R5)
(3) 001050' 004767 000000G                                                      JSR     PC,DRPMOD
(3) 001054' 012605                                                              MOV     (SP)+,R5
001056'                                 ELSE
(4) 001056' 000416                                                              BR      50030$
(3) 001060'                                                                  50027$:
001060'                                         IF SOFCNT(R1) HI DT.SMX(R0) AND #BIT14 NOTSETIN DT.SWR(R0) THEN
(6) 001060' 026160 000042 000106                                                CMP     SOFCNT(R1),DT.SM
(9) 001066' 101412                                                              BLOS    50031$
(6) 001070' 032760 040000 000056                                                BIT     #BIT14,DT.SWR(R0
(9) 001076' 001003                                                              BNE     50031$
001100'                                                 CALL DRPMOD IN <R0,R1>
(3) 001100' 010546                                                              MOV     R5,-(SP)
(5) 001102' 010145                                                              MOV     R1,-(R5)
(4) 001104' 010045                                                              MOV     R0,-(R5)
(3) 001106' 004767 000000G                                                      JSR     PC,DRPMOD
(3) 001112' 012605                                                              MOV     (SP)+,R5
001114'                                         ENDIF
(4) 001114'                                                                  50031$:
001114'                                 ENDIF
(4) 001114'                                                                  50030$:
001114'                         ENDIF
(4) 001114'                                                                  50026$:

001114'                         ENDIF
(4) 001114'                                                                  50024$:

                                ;+
                                ; UPDATE RETURN P.C.
                                ;-
001114'                         LET DT.PC(R0) := DT.PC(R0) + #6
(6) 001114' 062760 000006 000002                                                ADD     #6,DT.PC(R0)

                                ;+
                                ; IF DATACK, UPDATE AGAIN
                                ;-

001122'                             IF PC.DCF NE #0 THEN
(6) 001122' 005767 176704                                                       TST     PC.DCF
(9) 001126' 001403                                                              BEQ     50032$
001130'                                     LET DT.PC(R0) := DT.PC(R0) + #2
(6) 001130' 062760 000002 000002                                                ADD     #2,DT.PC(R0)
```

```
                                    ENDIF
                                                                                    50032$:

                                ;+
                                ; MARK IT AS DEFFERRED RETURN
                                ;-

                                    LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #DEFRTN
001136' 042760  000400  000010                                                      BIC     #DEFRTN,DT.ST0(R

                                ;+
                                ; IF THERE ARE SOME ERRORS, IF COUNT IS DONE, CHECK
                                ; IF THE TRAP CALL WAS A CDATA CALL.  IF SO, ENQUE THE ERROR
                                ; SUMMARY MESSAGE AND CHANGE THE RETURN ADDRESS TO LOCATION
                                ; IN THE TRAP CALL.  FOR DATCK TRAP, CLEAR THE DEFERRED SERVICE BIT
                                ; IN DT.ST0 AND LOAD ERROR COUNT INTO LOCATION IN THE TRAP CALL.
                                ; IN CASE YOU FORGOT R2 CONTAINED WORD COUNT SO FAR.
                                ;-

001144'                         INLINE <3$:>
001144'                                                                             3$:
001144'                           IF CDERCT(R1) NE #0 THEN
001144' 005761  000146                                                              TST     CDERCT(R1)
001150' 001440                                                                      BEQ     50033$
001152'                             IF R2 GE PC.RSZ THEN
001152' 020267  176646                                                              CMP     R2,PC.RSZ
001156' 002435                                                                      BLT     50034$



                                ;+
                                ; UPDATE WORD COUNT AND SET R2 TO RETURN ADDRESS
                                ;-
001160'                                 LET CDWDCT(R1) := R2
001160' 010261  000144                                                              MOV     R2,CDWDCT(R1)
001164'                                 LET R2 := PC.TPC + #6
001164' 016702  176636                                                              MOV     PC.TPC,R2
001170' 062702  000006                                                              ADD     #6,R2

                                ;+
                                ; IF CDATA, ENQUEUE SUMMARY MESSAGE
                                ;-
001174'                                 IF PC.DCF EQ #0 THEN
001174' 005767  176632                                                              TST     PC.DCF
001200' 001014                                                                      BNE     50035$
001202'                                     CALL ENQTQ IN <R0,#MSGCKS,PC.TPC,R1,(R2)>
001202' 010546                                                                      MOV     R5,-(SP)
001204' 011245                                                                      MOV     (R2),-(R5)
001206' 010145                                                                      MOV     R1,-(R5)
001210' 016745  176612                                                              MOV     PC.TPC,-(R5)
001214' 012745  000011                                                              MOV     #MSGCKS,-(R5)
001220' 010045                                                                      MOV     R0,-(R5)
001222' 004767  000000G                                                             JSR     PC,ENQTQ
001226' 012605                                                                      MOV     (SP)+,R5
001230'                                 ELSE
```

```
   (4)  001230' 000410                                                                  BR      50036$
   (3)  001232'                                                                 50035$:
    985
    986                                 ;+
    987                                 ; DATACK......R2 CONTAINS RETURN ADDRESS
    988                                 ;-
    989
    990  001232'                                LET 2(R2) := CDERCT(R1)
   (4)  001232' 016162  000146  000002                                           MOV     CDERCT(R1),2(R2)
    991
    992                                 ;+
    993                                 ; UPDATE RETURN P.C.
    994                                 ;-
    995  001240'                                LET DT.PC(R0) := (R2)
   (4)  001240' 011260  000002                                                   MOV     (R2),DT.PC(R0)
    996  001244'                                LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #DEFRTN
   (6)  001244' 042760  000400  000010                                           BIC     #DEFRTN,DT.ST0(R
    997  001252'                                    ENDIF
   (4)  001252'                                                                 50036$:
    998  001252'                                ENDIF
   (4)  001252'                                                                 50034$:
    999  001252'                            ENDIF
   (4)  001252'                                                                 50033$:
   1000
   1001
   1002
   1003
   1004                                 ;+
   1005                                 ; CLEAN UP AND GOODBYE
   1006                                 ;-
   1007
   1008  001252'                        CALL RESREG
   (3)  001252' 004767  000000G                                                 JSR     PC,RESREG
   1009  001256'                        ENDRTN
   (3)  001256'                                                                 50000$:
   (3)  001256'                                                                 50001$:
   (2)  001256' 000207                                                          RTS     PC
   1010
```

```
                                .SBTTL TOP16 SUBROUTINE

001260'                         ROUTINE TOP16 <AA>
(2) 001260'                                                             TOP16:

                                ;+
                                ; THIS ROUTINE COMBINES THE 6 EA BITS WITH THE TOP 10 BITS OF
                                ; THE PA WORD AND RETURNS THE RESULT IN PC.RTP/PC.WTP WORD.
                                ; AA IS THE TABLE ADDRESS AND THE TABLE IS OF THE FORM:
                                ;                               LO  -  LOWER 16 BITS FROM MAP22
                                ;                               HI  -  UPPER 6 BITS FROM MAP22
                                ;                               (   )  -  RESULT GOES HERE
                                ;-

001260'                         CALL SAVREG
(3) 001260' 004767  000000G                                             JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO TABLE ADDRESS, R2 GETS 16-BIT PA AND R1 GETS 6-BIT
                                ; EA. SHIFT PA 6 PLACES TO THE RIGHT AND BRING EA BITS IN POSITION
                                ; 15-10.  ADD EA AND PA TO GET TOP 16-BIT WORD.  RETURN THE
                                ; RESULT.
                                ;-

001264'                         LET R0 := AA(R5)
(4) 001264' 016500  000000                                              MOV     AA(R5),R0
001270'                         LET R2 := (R0)+
(4) 001270' 012002                                                      MOV     (R0)+,R2
001272'                         LET R1 := (R0)+
(4) 001272' 012001                                                      MOV     (R0)+,R1
001274'                         LET R2 := R2 SHIFT -6
(7) 001274' 006202                                                      ASR     R2
(7) 001276' 006202                                                      ASR     R2
(7) 001300' 006202                                                      ASR     R2
(7) 001302' 006202                                                      ASR     R2
(7) 001304' 006202                                                      ASR     R2
(7) 001306' 006202                                                      ASR     R2
001310'                         LET R2 := R2 CLR.BY #176000
(6) 001310' 042702  176000                                              BIC     #176000,R2
001314'                         LET R1 := SWAP R1
(6) 001314' 000301                                                      SWAB    R1
001316'                         LET R1 := R1 SHIFT 2
(7) 001316' 006301                                                      ASL     R1
(7) 001320' 006301                                                      ASL     R1
001322'                         LET (R0) := R2 + R1
(4) 001322' 010210                                                      MOV     R2,(R0)
(6) 001324' 060110                                                      ADD     R1,(R0)


                                ;+
                                ; RESTORE REGISTERS
                                ;-
001326'                         CALL RESREG
(3) 001326' 004767  000000G                                             JSR     PC,RESREG

001332'                         ENDRTN
```

```
   (3)  001332'                                                    50000$:
   (3)  001332'                                                    50001$:
   (2)  001332' 000207                                                     RTS     PC
  1050
```

```
                                .SBTTL TOP12 SUBROUTINE

001334'                         ROUTINE TOP12 <BB>
(2) 001334'                                                                 TOP12:

                                ;+
                                ; THIS ROUTINE COMBINES THE 2 EA BITS WITH THE TOP 10 BITS OF
                                ; THE PA WORD AND RETURNS THE RESULT INTO LOCATION PC.RTP/PC.WTP.
                                ; BB IS THE TABLE ADDRESS AND THE TABLE IS OF THE FORM:
                                ;                               PA
                                ;                               EA
                                ;                               LO
                                ;                               HI
                                ;                               (       )  -  RESULT GOES HERE
                                ;-

001334'                         CALL SAVREG
(3) 001334' 004767  000000G                                                 JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO TABLE ADDRESS. R2 GETS PA AND R1 GETS EA.
                                ; BRING EA BITS IN POSITION 9-10 AND SHIFT PA 6 PLACES
                                ; TO THE RIGHT.  ADD EA AND PA WORDS TO GET TOP 12 BIT WORD.
                                ; RETURN THE RESULT.
                                ;-

001340'                         LET R0 := BB(R5)
(4) 001340' 016500  000000                                                  MOV     BB(R5),R0
001344'                         LET R2 := (R0)+
(4) 001344' 012002                                                          MOV     (R0)+,R2
001346'                         LET R1 := (R0)+
(4) 001346' 012001                                                          MOV     (R0)+,R1
001350'                         LET R1 := R1 SHIFT 6
(7) 001350' 006301                                                          ASL     R1
(7) 001352' 006301                                                          ASL     R1
(7) 001354' 006301                                                          ASL     R1
(7) 001356' 006301                                                          ASL     R1
(7) 001360' 006301                                                          ASL     R1
(7) 001362' 006301                                                          ASL     R1
001364'                         LET R2 := R2 SHIFT -6
(7) 001364' 006202                                                          ASR     R2
(7) 001366' 006202                                                          ASR     R2
(7) 001370' 006202                                                          ASR     R2
(7) 001372' 006202                                                          ASR     R2
(7) 001374' 006202                                                          ASR     R2
(7) 001376' 006202                                                          ASR     R2
001400'                         LET R2 := R2 CLR.BY #176000
(6) 001400' 042702  176000                                                  BIC     #176000,R2
001404'                         LET 4(R0) := R2 + R1
(4) 001404' 010260  000004                                                  MOV     R2,4(R0)
(6) 001410' 060160  000004                                                  ADD     R1,4(R0)

001414'                         CALL RESREG
(3) 001414' 004767  000000G                                                 JSR     PC,RESREG

001420'                         ENDRTN
```

```
   (3)  001420'                                                          50000$:
   (3)  001420'                                                          50001$:
   (2)  001420' 000207                                                           RTS     PC
  1087
  1083          000001                  .END
```

```
AA     = 000000
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
APTSER= ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BB     = 000000
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
COM      000060R
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DT     = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GETPSW= ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PCDATA   000044RG
PC.DCF   000032R
PC.REA   000002R
PC.RHI   000006R
PC.RLO   000004R
PC.RPA   000000R
PC.RSZ   000024R
PC.RTP   000010R
PC.TMP   000030R
PC.TPC   000026R
PC.WEA   000014R
PC.WHI   000020R
PC.WLO   000016R
PC.WPA   000012R
PC.WTP   000022R
PDATCK   000052RG
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
```

```
SPACE = 000040          UIPAR1= 177642          WTWHMI= 000222          $F$UNT= 000130          $TSK2 = 050036
SPOINT= 000032          UIPAR2= 177644          XFLAG = 000005          $F$WHI= 000120          $TSK3 = 050031
SPVALU= 002200          UIPAR3= 177646          XOFF  = 000023          $F$YES= 000402          $TSK4 = 050022
SR0   = 177572          UIPAR4= 177650          XON   = 000021          $IFLEV= 177777          $$ARGC= 000002
SR1   = 177574          UIPAR5= 177652          $BGNLE= 177777          $ISK0 = 000001          $$BYTE= 000403
SR2   = 177576          UIPAR6= 177654          $ERFLG= 000400          $ISK1 = 000001          $$CASE= 000000
SR3   = 172516          UIPAR7= 177656          $F$AND= 000310          $ISK2 = 000001          $$DST = 000000
STAT  = 000026          UIPDR0= 177600          $F$BAD= 000401          $ISK3 = 000001          $$ELOC= 000402
STATBI= 064757          UIPDR1= 177602          $F$BLA= 000170          $LOCTA= 177777          $$ERFL= 000000
STAT1 = 000027          UIPDR2= 177604          $F$CAS= 000150          $LSTIN= 000001          $$FLAG= 000001
SUSPND= 000001          UIPDR3= 177606          $F$DEC= 000220          $LSTTA= 000001          $$FROM= 000000
SVR0  = 000062          UIPDR4= 177610          $F$DO = 000340          $NESTL= 000000          $$LOC = 001200R
SVR1  = 000064          UIPDR5= 177612          $F$FAL= 000405          $NSK0 = 000300          $$LOCN= 000000
SVR2  = 000066          UIPDR6= 177614          $F$GOO= 000400          $NSK1 = 000300          $$REG = 177777
SVR3  = 000070          UIPDR7= 177616          $F$IF = 000110          $NSK2 = 000110          $$RETU= 000000
SVR4  = 000072          UNIPA = ****** G        $F$INC= 000210          $NSK3 = 000110          $$RTN1= 050000
SVR5  = 000074          WASADR= 000104          $F$LOU= 000200          $NSK4 = 000110          $$RTN2= 050001
SVR6  = 000076          WBSTAT= 000040          $F$NAM= 000160          $NSK5 = 000110          $$SRC = 000000
SYSCNT= 000052          WBUFEA= 000136          $F$NO = 000403          $SAVLE= 177777          $$TGSV= 000000
SYSERR= 000100          WBUFPA= 000134          $F$OR = 000320          $SSK0 = 050012          $$TGS1= 000000
TMPIO = 000002          WBUFRQ= 000140          $F$RTI= 000350          $TAGLE= 177777          $$TGS2= 000000
TOP12   001334R         WBUFSZ= 000142          $F$RTN= 000300          $TAGNU= 050002          $$TO  = 000000
TOP16   001260R         WDFR  = 000116          $F$SEL= 000140          $TEMP = 000300          $$$TAG= 050000
TQOVF = 000002          WDTO  = 000114          $F$THE= 000330          $TSK0 = 050033          .     = 001422R
UIPAR0= 177640          WTINRE= 000352          $F$TRU= 000404          $TSK1 = 050034

. ABS.  000000     000
        001422     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PCDATA,DSKZ:PCDATA=SPMAC/ML,EQUATE,PCDATA
 RUN-TIME: 28 19 .4 SECONDS
 RUN-TIME RATIO: 67/47=1.4
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                                .TITLE PCTRLC - PROCESS CTRL C
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       PCTRLC
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       CONTROLS THE PROCESSING OF A CTRL C COMMAND FROM
                                ;       THE OPERATOR.
                                ;
                                ; INPUTS:
                                ;       DATA TABLE ADDRESS
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.ADDR, DT.ST0, DT.CF0
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       DT.ST0
                                ;
                                ; PATOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE MODULES CALLED:
                                ;       MSGDHOOK                        ;HOOK MESSAGE TO PROPER DRIVER
                                ;       PRRLOC                          ;PROCESS RELOCATION
                                ;       WBFLIM                          ;DETERMINE WRITE BUFFER LIMITS
                                ;       UNIMAP                          ;LOAD UNIBUS MAP
                                ;       RSTRCY                          ;RECOVER FROM RESET
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL PCTRLC IN <A>
                                ;               A=DATA TABLE ADDRESS
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN=1
            000001              $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PCTRLC                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  MSGDHOOK                        ;HOOK MESSAGE TO PROPER DRIVER
                                .GLOBL  PRRLOC                          ;PROCESS RELOCATION
                                .GLOBL  WBFLIM                          ;DETERMINE WRITE BUFFER LIMITS
                                .GLOBL  UNIMAP                          ;LOAD UNIBUS MAP
                                .GLOBL  RSTRCY                          ;RECOVER FROM RESET
                                .GLOBL  DX.RSTRT                        ;MONITOR RESTART ADDRESS
                                .GLOBL  XCSR                            ;KEYBOARD PRINTER CSR
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 041536  000045          PC.MSG: .ASCIZ /^C%/
                                .EVEN
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL PCTRLC ROUTINE


000004'                         ROUTINE PCTRLC <TABL>
(2) 000004'                                                                     PCTRLC:


                                ;+
                                ; SAVE REGISTERS
                                ;-

000004'                         PUSH R0
(2) 000004' 010046                                                                      MOV     R0,-(SP)


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

000006'                         LET R0 := TABL(R5)
(4) 000006' 016500  000000                                                              MOV     TABL(R5),R0


                                ;+
                                ; RESET THE UNIBUS
                                ;-

000012'                         INLINE <RESET>
(2) 000012' 000005                                                                      RESET


                                ;+
                                ; CLEAR THE PSW. (LOWER THE PRIORITY)
                                ;-

000014'                         LET -(SP) := #0
(4) 000014' 005046                                                                      CLR     -(SP)
000016'                         LET -(SP) := #1$
(4) 000016' 012746  000024'                                                             MOV     #1$,-(SP)
000022'                         INLINE <RTI>
(2) 000022' 000002                                                                      RTI

000024'                         INLINE <1$:>
(2) 000024'                                                                             1$:


                                ;+
                                ; RESTORE THE PROCESSOR HARDWARE TO THE STATE IT WAS IN BEFORE THE RESET.
                                ;-

000024'                         CALL RSTRCY IN <R0>
(3) 000024' 010546                                                                      MOV     R5,-(SP)
(4) 000026' 010045                                                                      MOV     R0,-(R5)
(3) 000030' 004767  000000G                                                             JSR     PC,RSTRCY
(3) 000034' 012605                                                                      MOV     (SP)+,R5
```

```
                                ;+
                                ; OUTPUT "^C".
                                ;-

000036'                         CALL MSGDHOOK IN <R0,#MSGPOP,#PC.MSG,#3$>
(3) 000036' 010546                                                          MOV     R5,-(SP)
(7) 000040' 012745  000070'                                                 MOV     #3$,-(R5)
(6) 000044' 012745  000000'                                                 MOV     #PC.MSG,-(R5)
(5) 000050' 012745  000002                                                  MOV     #MSGPOP,-(R5)
(4) 000054' 010045                                                          MOV     R0,-(R5)
(3) 000056' 004767  000000G                                                 JSR     PC,MSGDHOOK
(3) 000062' 012605                                                          MOV     (SP)+,R5
000064'                         INLINE <2$: NOP>
(2) 000064' 000240                                                          2$: NOP
000066'                         INLINE <BR 2$>
(2) 000066' 000776                                                          BR 2$

000070'                         INLINE <3$:>
(2) 000070'                                                                 3$:


                                ;+
                                ; IF THE EXERCISER IS NOT IN LOWEST MEMORY MOVE IT THERE, THEN RESET THE WRITE
                                ; BUFFER LIMITS, AND RELOAD THE UNIBUS MAP IF IT EXISTS.
                                ;-

000070'                         IF DT.ADDR(R0) NE #200 THEN
(6) 000070' 026027  000042  000200                                          CMP     DT.ADDR(R0),#200
(9) 000076' 001425                                                          BEQ     50002$
000100'                                 CALL PRRLOC IN <R0,#200>
(3) 000100' 010546                                                          MOV     R5,-(SP)
(5) 000102' 012745  000200                                                  MOV     #200,-(R5)
(4) 000106' 010045                                                          MOV     R0,-(R5)
(3) 000110' 004767  000000G                                                 JSR     PC,PRRLOC
(3) 000114' 012605                                                          MOV     (SP)+,R5
000116'                                 CALL WBFLIM IN <R0>
(3) 000116' 010546                                                          MOV     R5,-(SP)
(4) 000120' 010045                                                          MOV     R0,-(R5)
(3) 000122' 004767  000C00G                                                 JSR     PC,WBFLIM
(3) 000126' 012605                                                          MOV     (SP)+,R5
000130'                                 IF #ADDR22 SETIN DT.CF0(R0) THEN
(6) 000130' 032760  001000  000014                                          BIT     #ADDR22,DT.CF0(R
(9) 000136' 001405                                                          BEQ     50003$
000140'                                         CALL UNIMAP IN <R0>
(3) 000140' 010546                                                          MOV     R5,-(SP)
(4) 000142' 010045                                                          MOV     R0,-(R5)
(3) 000144' 004767  000000G                                                 JSR     PC,UNIMAP
(3) 000150' 012605                                                          MOV     (SP)+,R5
000152'                                 ENDIF
(4) 000152'                                                         50003$:
000152'                         ENDIF
(4) 000152'                                                         50002$:


                                ;+
```

```
  662                                   ; CLEAR THE RUN MODE INDICATOR AND THE RELOCATION TIME INDICATOR.
  663                                   ;-
  664
  665  000152'                          LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RUNMODE!RELTIME
  (6)  000152' 042760  110000  000010                                                  BIC     #RUNMODE!RELTIME
  666
  667
  668                                   ;+
  669                                   ; RESTORE REGISTERS
  670                                   ;-
  671
  672  000160'                          POP R0
  (2)  000160' 012600                                                                  MOV     (SP)+,R0
  673
  674
  675                                   ;+
  676                                   ; RESTART THE MONITOR
  677                                   ;-
  678
  679  000162'                          INLINE <JMP DX.RSTRT>
  (2)  000162' 000167  000000G                                                        JMP DX.RSTRT
  680
  681
  682  000166'                          ENDRTN
  (3)  000166'                                                                 50000$:
  (3)  000166'                                                                 50001$:
  (2)  000166' 000207                                                                  RTS     PC
  683          000001                   .END
```

```
ACSR   = 000102        CTRLC  = 000003        EOPBIT= 000001        MODSEL= 001000        RBUFEA= 000130
ACTBIT= 004000        CTRLO  = 000017        ERRTYP= 000106        MSGCKD= 000010        RBUFPA= 000126
ADDR22= 001000        CTRLU  = 000025        EVNTBE= 000200        MSGCKS= 000011        RBUFSZ= 000132
ADR    = 000006        DCEVNT= 000011        EVNTHD= 000200        MSGDER= 000005        RBUFVA= 000124
APTFER= 000004        DEFRTN= 000400        EVNTKT= 000203        MSGDHO= ****** G      RDSERV= 000101
APTPRE= 000200        DIAGMC= 000000        EVNTPE= 000202        MSGDRP= 000017        RDWHMI= 000022
ASB    = 000106        DROPMO= 100000        EVNTRE= 000201        MSGECH= 177777        RELERR= 000020
ASSEMB= 000010        DSEVNT= 000014        FATERR= 100000        MSGEOP= 000013        RELMOD= 020000
ASTAT  = 000104        DT.ADD= 000042        HRDCNT= 000044        MSGHDR= 000004        RELTIM= 010000
AUTO   = 000010        DT.AP  = 000100        HRDPAS= 000050        MSGHNG= 000022        RES1   = 000056
AUTOST= 020000        DT.APK= 000076        ICONT  = 000036        MSGHRD= 000007        RES2   = 000060
AWAS   = 000110        DT.BLS= 000034        ICOUNT= 000040        MSGMAP= 000021        RICHAR= 031060
BIT0   = 000001        DT.CF0= 000014        IDNUM  = 000122        MSGNUL= 177775        RPTDAT= 002000
BIT00  = 000001        DT.CF1= 000016        IE     = 000100        MSGPOP= 000002        RSTRCY= ****** G
BIT01  = 000002        DT.ERR= 000020        INDPAR= 000040        MSGPRM= 177776        RSTRT  = 000112
BIT02  = 000004        DT.ESI= 000044        INHDRP= 040000        MSGRES= 000001        RUBOUT= 000177
BIT03  = 000010        DT.EVN= 000000        INHEPR= 020000        MSGSFT= 000006        RUNMOD= 100000
BIT04  = 000020        DT.EXS= 000060        INHREL= 001000        MSGSKE= 000003        R5VALU= 001740
BIT05  = 000040        DT.FCH= 000037        INHRRE= 000400        MSGSMB= 000015        SAM    = 075464
BIT06  = 000100        DT.FCN= 000036        INIT   = 000030        MSGSMH= 000014        SBADR  = 000102
BIT07  = 000200        DT.HMX= 000104        INTR   = 000120        MSGSMS= 000016        SBKMOD= 000000
BIT08  = 000400        DT.KBE= 000024        IOMOD  = 100000        MSGSTD= 000000        SBKSEL= 010000
BIT09  = 001000        DT.KBP= 000026        IOMODP= 102000        MSGSYS= 000012        SC.ADR= 000006
BIT1   = 000002        DT.KBR= 000022        IOMODR= 112000        MSGVEC= 000020        SC.ALC= 000014
BIT10  = 002000        DT.KBU= 000030        IOMODX= 110000        NBKMOD= 001000        SC.APC= 000016
BIT11  = 004000        DT.MLS= 000032        JACK   = 035060        NCPUOP= 000020        SC.CKL= 000002
BIT12  = 010000        DT.MTI= 000110        KIPAR0= 172340        NOAPTY= 000002        SC.CKP= 000004
BIT13  = 020000        DT.OFF= 000070        KIPAR1= 172342        NULL   = 000000        SC.CLO= 000000
BIT14  = 040000        DT.PAS= 000074        KIPAR2= 172344        OWEN   = 024020        SC.HLD= 000010
BIT15  = 100000        DT.PC  = 000002        KIPAR3= 172346        PAERR  = 000010        SC.SCA= 000012
BIT2   = 000004        DT.PFL= 000062        KIPAR4= 172350        PARPRE= 002000        SENDLS= 177777
BIT3   = 000010        DT.PSW= 000004        KIPAR5= 172352        PARSTA= 000100        SOFCNT= 000042
BIT4   = 000020        DT.PTA= 000064        KIPAR6= 172354        PASCNT= 000034        SOFPAS= 000046
BIT5   = 000040        DT.RCS= 000102        KIPAR7= 172356        PCTRLC  000004RG      SPACE  = 000040
BIT6   = 000100        DT.REL= 000040        KIPDR0= 172300        PC.MSG  000000R       SPOINT= 000032
BIT7   = 000200        DT.SCT= 000066        KIPDR1= 172302        PDPLSI= 020000        SPVALU= 002200
BIT8   = 000400        DT.SMX= 000106        KIPDR2= 172304        PDP60  = 004000        SR0    = 177572
BIT9   = 001000        DT.SP  = 000036        KIPDR3= 172306        PDP70  = 010000        SR1    = 177574
BKDEF  = 000002        DT.SSI= 000046        KIPDR4= 172310        PRI0   = 000000        SR2    = 177576
BKMOD  = 000020        DT.ST0= 000010        KIPDR5= 172312        PRI1   = 000040        SR3    = 172516
BKMODE= 040000        DT.ST1= 000012        KIPDR6= 172314        PRI4   = 000200        STAT   = 000026
BKSLSH= 000134        DT.SWR= 000056        KIPDR7= 172316        PRI5   = 000240        STATBI= 064757
CAPRES= 000004        DT.SYP= 000072        KTERRO= 000040        PRI6   = 000300        STAT1  = 000027
CASTAT= 000004        DT.WBU= 000050        KTPRES= 000400        PRI7   = 000340        SUSPND= 000001
CDERCT= 000146        DT.WHL= 000054        KTSTAT= 000020        PRRLOC= ****** G       SVR0   = 000062
CDWDCT= 000144        DT.WLL= 000052        KTXTND= 040000        PR0    = 000000        SVR1   = 000064
CKTIM  = 100000        DVID1  = 000014        LF     = 000012        PR4    = 000200        SVR2   = 000066
CLKPRE= 000001        DX.RST= ****** G       LPSTAT= 000001        PR5    = 000240        SVR3   = 000070
CONFIG= 000056        ECCMEM= 000100        MAPSTA= 000200        PR6    = 000300        SVR4   = 000072
CQOVF  = 000001        ECCSTA= 000010        MED    = 076600        PR7    = 000340        SVR5   = 000074
CR     = 000015        ENBEOP= 010000        MEMPAS= 040000        PS     = 177776        SVR6   = 000076
CSRA   = 000100        ENBNUL= 000001        MODEXH= 004000        PSW    = 177776        SYSCNT= 000052
CSRC   = 000102        ENDLST= 000000        MODHOL= 002000        RANNUM= 000054        SYSERR= 000100
```

```
TABL  = 000000          WASADR= 000104          $F$CAS= 000150          $ISK0 = 000001          $$ERFL= 000000
TMPIO = 000002          WBFLIM= ****** G        $F$DEC= 000220          $ISK1 = 000001          $$FLAG= 000001
TQOVF = 000002          WBSTAT= 000040          $F$DO = 000340          $LOCTA= 177777          $$FROM= 000000
UIPAR0= 177640          WBUFEA= 000136          $F$FAL= 000405          $LSTIN= 000001          $$LOC = 000136R
UIPAR1= 177642          WBUFPA= 000134          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
UIPAR2= 177644          WBUFRQ= 000140          $F$IF = 000110          $NESTL= 177777          $$REG = 177777
UIPAR3= 177646          WBUFSZ= 000142          $F$INC= 000210          $NSK0 = 000300          $$RETU= 000000
UIPAR4= 177650          WDFR  = 000116          $F$LOO= 000200          $NSK1 = 000110          $$RTN1= 050000
UIPAR5= 177652          WDTO  = 000114          $F$NAM= 000160          $NSK2 = 000110          $$RTN2= 050001
UIPAR6= 177654          WTINRE= 000352          $F$NO = 000403          $SAVLE= 177777          $$SRC = 000000
UIPAR7= 177656          WTWHMI= 000222          $F$OR = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPDR0= 177600          XCSR  = ****** G        $F$RTI= 000350          $TAGNU= 050004          $$TGS1= 000000
UIPDR1= 177602          XFLAG = 000005          $F$RTN= 000300          $TEMP = 000300          $$TGS2= 000000
UIPDR2= 177604          XOFF  = 000023          $F$SEL= 000140          $TSK0 = 050002          $$TO  = 000001
UIPDR3= 177606          XON   = 000021          $F$THE= 000330          $TSK1 = 050003          $$$TAG= 050000
UIPDR4= 177610          $BGNLE= 177777          $F$TRU= 000404          $$ARGC= 000002          .     = 000170R
UIPDR5= 177612          $ERFLG= 000400          $F$UNT= 000130          $$BYTE= 000403
UIPDR6= 177614          $F$AND= 000310          $F$WHI= 000120          $$CASE= 000000
UIPDR7= 177616          $F$BAD= 000401          $F$YES= 000402          $$DST = 000000
UNIMAP= ****** G        $F$BLA= 000170          $IFLEV= 177777          $$ELOC= 000402

. ABS.  000000     000
        000170     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PCTRLC,DSKZ:PCTRLC=SPMAC/ML,EQUATE,PCTRLC
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 29/15=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PFAIL (PROCESS POWER FAIL/POWER UP)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PFAIL
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE CONTAINS THE FOLLOWING TWO ROUTINES:
                        ;
                        ;           1. PWRDOWN          ;HANDLES POWER DOWN SEQUENCE.
                        ;           2. PWRUP            ;HANDLES POWER UP SEQUENCE.
                        ;
                        ;
                        ;       ON POWER FAIL, IF THE SYSTEM CLOCK IS AVAILABLE IT IS TURNED
                        ;       OFF, POWER-UP VECTOR IS SET UP, AND THE CPU HALTS.
                        ;
                        ;       ON POWER UP, THE FOLLOWING IS DONE:
                        ;
                        ;                   1. SET UP POWER-DOWN VECTOR.
                        ;                   2. REINITIALIZE STACKS (R5 AND R6).
                        ;                   3. INCREMENT POWER FAIL COUNT IN DTABLE.
                        ;                   4. INITIALIZE APARS IF REQUIRED.
                        ;                   5. TURN KT ON IF IT WAS ON BEFORE POWER FAIL.
                        ;                   6. TURN SYSTEM CLOCK ON IF AVAILABLE.
                        ;                   7. PRINT POWER FAIL MESSAGE.
                        ;                   8. JUMP TO BOX00 (DX.RSTRT).
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1 .DT.ST0       ;STATUS WORD #0
                        ;       2. DT.CF0       ;CONFIGURATION WORD #0
                        ;       3. DT.PFL       ;POWER FAIL COUNT WORD
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.ST0
                        ;       2. DT.PFL
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. DTABLE ADDRESS
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. CLKON        ;CLOCK ON ROUTINE
                        ;       2. CLKOFF       ;CLOCK OFF ROUTINE
                        ;       3. MSGDHOOK     ;OUTPUTS A MESSAGE
                        ;       4. KTSET        ;STUFFS THE KERNEL & USER MODE APARS
                        ;       5. RSTRCY       ;RESET RECOVERY ROUTINE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
```

```
                        ; CALLING SEQUENCE:
                        ;       THE MODULES ARE ENTERED DIRECTLY AS A RESULT OF POWER FAIL/
                        ;       POWER UP.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL PFAIL (COMMON DEFINITIONS AND REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL PWRFL            ;POWER FAIL VECTOR - LOC. 24
                                .GLOBL PWRDOWN          ;POWER FAIL ENTRY POINT
                                .GLOBL PWRUP            ;POWER UP ENTRY POINT
                                .GLOBL RSTRCY           ;RESET RECOVERY ROUTINE
                                .GLOBL KTSET            ;STUFF APARS
                                .GLOBL CLKON            ;TURN SYSTEM CLOCK ON
                                .GLOBL CLKOFF           ;TURN SYSTEM CLOCK OFF
                                .GLOBL MSGDHOOK         ;TYPE A MESSAGE ROUTINE
                                .GLOBL DX.RSTRT         ;JUMP TO BOX00
                                .GLOBL DTABLE           ;DATA TABLE

                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 000000G                 PF.EXT: #DX.RSTRT       ;RESTART ROUTINE
000002' 050045  053517  051105  PF.MSG: .ASCIZ  /%POWER FAIL OCCURRED%/
000010' 043040  044501  020114
000016' 041517  052503  051122
000024' 042105  000045

                                .EVEN
```

```
                                .SBTTL PFAIL (CODE)

                                ;+
                                ; POWER DOWN ROUTINE
                                ;-

                                INLINE <PWRDOWN:>                       PWRDOWN:


                                ;+
                                ; SET R0 TO DTABLE ADDRESS. SET UP POWER-UP VECTOR,
                                ; AND TURN OFF SYSTEM CLOCK IF AVAILABLE.  THEN HALT.
                                ;-

                                LET SP := #SPVALUE                      MOV     #SPVALUE,SP

                                LET R5 := #R5VALUE                      MOV     #R5VALUE,R5

                                LET R0 := #DTABLE                       MOV     #DTABLE,R0

                                LET PWRFL := #PWRUP                     MOV     #PWRUP,PWRFL

                                IF #CLKPRES SETIN DT.CF0(R0) THEN       BIT     #CLKPRES,DT.CF0(
                                                                        BEQ     50000$

                                    CALL CLKOFF IN <R0>                 MOV     R5,-(SP)
                                                                        MOV     R0,-(R5)
                                                                        JSR     PC,CLKOFF
                                                                        MOV     (SP)+,R5

                                ENDIF
                                                                50000$:
                                INLINE <HALT>                           HALT
```

```
  629
  630                                   ;+
  631                                   ; POWER UP ROUTINE
  632                                   ;-
  633
  634  000076'                          INLINE <PWRUP:>                                  PWRUP:
(2)  000076'
  635
  636                                   ;+
  637                                   ; SET UP POWER-DOWN VECTOR, INITIALIZE STACKS
  638                                   ; , SET R0 TO DTABLE ADDRESS, AND INCREMENT
  639                                   ; POWER-FAIL COUNT WORD IN DTABLE.
  640                                   ;-
  641
  642  000076'                          LET PWRFL := #PWRDOWN                            MOV     #PWRDOWN,PWRFL
(4)  000076'  012767  000030' 000000G
  643  000104'                          INLINE <RESET>                                   RESET
(2)  000104'  000005
  644  000106'                          LET SP := #SPVALUE                               MOV     #SPVALUE,SP
(4)  000106'  012706  002200
  645  000112'                          LET R5 := #R5VALUE                               MOV     #R5VALUE,R5
(4)  000112'  012705  001740
  646  000116'                          LET R0 := #DTABLE                                MOV     #DTABLE,R0
(4)  000116'  012700  000000G
  647  000122'                          LET DT.PFL(R0) := DT.PFL(R0) + #1                INC     DT.PFL(R0)
(6)  000122'  005260  000062
  648
  649                                   ;+
  650                                   ; IF KT PRESENT THEN MAP APRS AND RESTORE HARDWARE TO PRE-POWER FAIL STATUS
  651                                   ;-
  652
  653  000126'                          IF #KTPRES SETIN DT.CF0(R0) THEN                 BIT     #KTPRES,DT.CF0(R
(6)  000126'  032760  000400  000014                                                   BEQ     50001$
(9)  000134'  001412
  654  000136'                              CALL KTSET IN <R0>                           MOV     R5,-(SP)
(3)  000136'  010546                                                                   MOV     R0,-(R5)
(4)  000140'  010045                                                                   JSR     PC,KTSET
(3)  000142'  004767  000000G                                                          MOV     (SP)+,R5
(3)  000146'  012605
  655  000150'                              CALL RSTRCY IN <R0>                          MOV     R5,-(SP)
(3)  000150'  010546                                                                   MOV     R0,-(R5)
(4)  000152'  010045                                                                   JSR     PC,RSTRCY
(3)  000154'  004767  000000G                                                          MOV     (SP)+,R5
(3)  000160'  012605
  656  000162'                          ENDIF                                    50001$:
(4)  000162'
  657
  658                                   ;+
  659                                   ; IF SYSTEM CLOCK AVAILABLE, TURN IT ON
  660                                   ;-
  661
  662  000162'                          IF #CLKPRES SETIN DT.CF0(R0) THEN                BIT     #CLKPRES,DT.CF0(
(6)  000162'  032760  000001  000014                                                   BEQ     50002$
(9)  000170'  001405
  663  000172'                              CALL CLKON IN <R0>                           MOV     R5,-(SP)
(3)  000172'  010546
```

```
   (4)  000174' 010045                                                          MOV     R0,-(R5)
   (3)  000176' 004767  000000G                                                 JSR     PC,CLKON
   (3)  000202' 012605                                                          MOV     (SP)+,R5
   664  000204'                        ENDIF
   (4)  000204'                                                         50002$:
   665
   666                                ;+
   667                                ; DO A TIMING LOOP TO LET EVERYBODY SETTLE DOWN
   668                                ;-
   669
   670  000204'                        LET R2 := #^D<10>
   (4)  000204' 012702  000012                                                  MOV     #^D<10>,R2
   671  000210'                        WHILE R2 NE #0 DO
   (4)  000210'                                                         50003$:
   (6)  000210' 005702                                                          TST     R2
   (9)  000212' 001410                                                          BEQ     50004$
   672  000214'                                LET R1 := #-1
   (4)  000214' 012701  177777                                                  MOV     #-1,R1
   673  000220'                                WHILE R1 NE #0 DO
   (4)  000220'                                                         50005$:
   (6)  000220' 005701                                                          TST     R1
   (9)  000222' 001402                                                          BEQ     50006$
   674  000224'                                        LET R1 := R1 - #1
   (6)  000224' 005301                                                          DEC     R1
   675  000226'                                ENDDO
   (4)  000226' 000774                                                          BR      50005$
   (3)  000230'                                                         50006$:
   676  000230'                                LET R2 := R2 - #1
   (6)  000230' 005302                                                          DEC     R2
   677  000232'                        ENDDO
   (4)  000232' 000766                                                          BR      50003$
   (3)  000234'                                                         50004$:
   678
   679                                ;+
   680                                ; NOW LOWER PROIRITY TO 0
   681                                ;-
   682
   683  000234'                        LET -(SP) := #0
   (4)  000234' 005046                                                          CLR     -(SP)
   684  000236'                        LET -(SP) := #1$
   (4)  000236' 012746  000244'                                                 MOV     #1$,-(SP)
   685  000242'                        INLINE  <RTI>
   (2)  000242' 000002                                                          RTI
   686
   687                                ;+
   688                                ; PRINT THE POWER FAIL MESSAGE AND JUMP TO DX.RSTRT.
   689                                ;-
   690
   691  000244'                        INLINE <1$:>
   (2)  000244'                                                         1$:
   692  000244'                        CALL MSGDHOOK IN <R0,#MSGPOP,#PF.MSG,#3$>
   (3)  000244' 010546                                                          MOV     R5,-(SP)
   (7)  000246' 012745  000276'                                                 MOV     #3$,-(R5)
   (6)  000252' 012745  000002'                                                 MOV     #PF.MSG,-(R5)
   (5)  000256' 012745  000002                                                  MOV     #MSGPOP,-(R5)
   (4)  000262' 010045                                                          MOV     R0,-(R5)
```

```
   (3)  000264' 004767  000000G                                    JSR     PC,MSGDHOOK
   (3)  000270' 012605                                             MOV     (SP)+,R5
   693  000272'                      INLINE <2$:>
   (2)  000272'                                                    2$:
   694  000272'                      INLINE <NOP>
   (2)  000272' 000240                                             NOP
   695  000274'                      INLINE <BR 2$>
   (2)  000274' 000776                                             BR 2$
   696  000276'                      INLINE <3$:>
   (2)  000276'                                                    3$:
   697  000276'                      INLINE <JMP @PF.EXT>
   (2)  000276' 000177  177476                                     JMP @PF.EXT
   698
   699          000001               .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKOFF= ****** G
CLKON  = ****** G
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= ****** G
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
DX.RST= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSET  = ****** G
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PF.EXT   000000R
PF.MSG   000002R
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
PWRDOW  000030RG
PWRFL  = ****** G
PWRUP   000076RG
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRCY= ****** G
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
```

```
SVR3   = 000070          UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$CASE= 000000
SVR4   = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$DST = 000000
SVR5   = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$ELOC= 000402
SVR6   = 000076          UIPDR7= 177616          $F$DEC= 000220          $LOCTA= 177777          $$ERFL= 000000
SYSCNT= 000052           WASADR= 000104          $F$DO  = 000340          $LSTIN= 000001          $$FLAG= 000340
SYSERR= 000100           WBSTAT= 000040          $F$FAL= 000405          $LSTTA= 000001          $$FROM= 000000
TMPID  = 000002          WBUFEA= 000136          $F$GOO= 000400          $NESTL= 177777          $$LOC = 000222R
TQOVF  = 000002          WBUFPA= 000134          $F$IF  = 000110          $NSK0 = 000120          $$LOCN= 000000
UIPAR0= 177640           WBUFRQ= 000140          $F$INC= 000210          $NSK1 = 000120          $$REG = 177777
UIPAR1= 177642           WBUFSZ= 000142          $F$LOO= 000200          $SAVLE= 177777          $$RETU= 000000
UIPAR2= 177644           WDFR   = 000116          $F$NAM= 000160          $SSK0 = 050004          $$RTN1= 000000
UIPAR3= 177646           WDTO   = 000114          $F$NO  = 000403          $TAGLE= 177777          $$RTN2= 000000
UIPAR4= 177650           WTINRE= 000352          $F$OR  = 000320          $TAGNU= 050007          $$SRC = 000000
UIPAR5= 177652           WTWHMI= 000222          $F$RTI= 000350          $TEMP  = 000402          $$TGSV= 000000
UIPAR6= 177654           XFLAG  = 000005          $F$RTN= 000300          $TSK0  = 050003          $$TGS1= 000000
UIPAR7= 177656           XOFF   = 000023          $F$SEL= 000140          $TSK1  = 050004          $$TGS2= 000000
UIPDR0= 177600           XON    = 000021          $F$THE= 000330          $TSK2  = 050005          $$TO  = 000004
UIPDR1= 177602           $BGNLE= 177777          $F$TRU= 000404          $TSK3  = 050006          $$$TAG= 050000
UIPDR2= 177604           $ERFLG= 000400          $F$UNT= 000130          $$ARGC= 000000          .      = 000302R
UIPDR3= 177606           $F$AND= 000310          $F$WHI= 000120          $$BYTE= 000403

. ABS.  000000     000
        000302     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PFAIL,DSKZ:PFAIL=SPMAC/ML,EQUATE,PFAIL
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 34/19=1.7
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                                .SBTTL BDACNV ROUTINE

                                ROUTINE BDACNV <NUM,ADR>
000012'                                                         BDACNV:
(2)  000012'

                                ;+
                                ; SAVE REGISTERS.
                                ;-

                                CALL SAVREG
000012'                                                                 JSR     PC,SAVREG
(3)  000012' 004767  000000G

                                ;+
                                ; SET R0 TO THE START OF THE STORAGE AREA
                                ; OF THE RESULT.
                                ;-

                                LET R0 := ADR(R5)
000016'                                                                 MOV     ADR(R5),R0
(4)  000016' 016500  000002

                                ;+
                                ; SET R1 TO THE ADDRESS OF THE POWERS OF TEN TABLE.
                                ;-

                                LET R1 := #BD.TEN
000022'                                                                 MOV     #BD.TEN,R1
(4)  000022' 012701  000000'

                                ;+
                                ; SET R4 TO COUNT 5 CHARACTERS.
                                ;-

                                LET R4 := R0 + #5
000026'                                                                 MOV     R0,R4
(4)  000026' 010004                                                     ADD     #5,R4
(6)  000030' 062704  000005

                                ;+
                                ; PERFORM THE CONVERSION.
                                ;       1. GET A POWER OF TEN FROM THE TABLE.
                                ;       2. SUBTRACT THIS VALUE FROM THE NUMBER WE ARE
                                ;          CONVERTING AS MANY TIMES AS POSSIBLE, KEEPING A COUNT
                                ;          OF THE NO. OF TIMES WE SUBTRACTED.
                                ;       3. CONVERT THIS COUNT TO ASCII AND STORE IT IN THE
                                ;          STORAGE TABLE.
                                ;       4. GET THE NEXT POWER OF TEN AND REPEAT.
                                ;-

                                WHILE R0 LO R4 DO
000034'                                                         50002$:
(4)  000034'                                                            CMP     R0,R4
(6)  000034' 020004                                                     BHIS    50003$
(9)  000036' 103016

                                        LET R3 := #0
000040'
```

```
                        .TITLE PRRLOC - PROCESS RELOCATION
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRRLOC
                        ;
                        ; FUNCTIONAL DESCRIPTION
                        ;       CONTROLS THE RELOCATING OF THE RUNTIME EXERCISER FROM
                        ;       ITS CURRENT LOCATION IN MEMORY TO A NEW LOCATION.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ADDR, DT.ERR, DT.STO, DT.SWR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ADDR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NEWBA                   ;ESTABLISH NEW BASE ADDRESS
                        ;       RELOC                   ;RELOCATE CODE
                        ;       MSGDHOOK                ;HOOK MSG TO PROPER OUTPUT DRIVER
                        ;       BOAC                    ;BINARY TO OCTAL ASCII CONVERSION
                        ;       SAVREG                  ;SAVE REGISTERS
                        ;       RESREG                  ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       QUEUEING MECHANISMS ARE INACTIVE DURING RELOCATION
                        ;       PROCESSING.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRRLOC IN <A,B>
                        ;               A=DATA TABLE ADDRESS
                        ;               B=NEW BASE OFFSET (IF SET TO #0, NEWBA WILL CALCULATE
                        ;                 A NEW BASE ADDRESS.)
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTTAG=1
                        000001  $LSTIN=1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PRRLOC                  ;MODULE ENTRY POINT
                                .GLOBL  PR.RL1                  ;1ST RELOCATION FLAG
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  MSGDHOOK                ;HOOK MSG TO PROPER OUTPUT DRIVER
                                .GLOBL  NEWBA                   ;ROUTINE TO FIND NEW BASE OFFSET
                                .GLOBL  RELOC                   ;ROUTINE TO MOVE CODE
                                .GLOBL  BOAC                    ;BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  PR.RL1: .WORD 0                 ;FIRST RELOCATION FLAG - PATHOLOGICALLY
                                                                ;   SET AT INITIALIZATION
000002' 000002                  PR.TBL: .BLKW 2                 ;TABLE FOR BOAC ROUTINE
000006' 051045  046105  041517  PR.MSG: .ASCII /%RELOCATED TO /  ;RELOCATION MESSAGE
000014' 052101  042105  052040
000022' 020117
000024' 000010                  PR.NBR:         .BLKB 10        ;STORAGE FOR ASCII BASE ADDRESS
000034' 000045                                  .ASCIZ /%/
000036' 051045  046105  041517  PR.MS1: .ASCIZ /%RELOCATION ABORTED%/  ;RELOCATION ABORTED MESSAGE
000044' 052101  047511  020116
000052' 041101  051117  042524
000060' 022504     000
                        000064' .EVEN
                                ;
                                ;***************
                                ;
```

```
                                .SBTTL PRRLOC ROUTINE

                                ROUTINE PRRLOC <TABL,BASE>
(2)  000064'                                                    PRRLOC:

                                ;+
                                ; SAVE REGISTERS
                                ;-

000064'                         CALL SAVREG
(3)  000064' 004767  000000G                                    JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO START OF DATA TABLE
                                ;-

000070'                         LET R0 := TABL(R5)
(4)  000070' 016500  000000                                     MOV     TABL(R5),R0

                                ;+
                                ; GET THE CURRENT BASE ADDRESS OFFSET
                                ;-

000074'                         LET R3 := DT.ADDR(R0)
(4)  000074' 016003  000042                                     MOV     DT.ADDR(R0),R3

                                ;+
                                ; CALL NEWBA TO GENERATE A NEW BASE OFFSET.  (THIS MUST BE DONE, EVEN IF A NEW
                                ; BASE WAS PASSED TO THIS MODULE, TO ASSURE THAT NEWBA IS PROPERLY
                                ; INITIALIZED.)  AFTER CALLING NEWBA, IF A NEW BASE ADDRESS WAS PASSED
                                ; TO THE MODULE, USE IT INSTEAD OF THE ONE NEWBA CALCULATED.
                                ;-

000100'                         CALL NEWBA IN <R0,R3,PR.RL1> OUT <R4>
(4)  000100' 162705  000002                                     SUB     #1*2,R5
(3)  000104' 010546                                             MOV     R5,-(SP)
(6)  000106' 016745  177666                                     MOV     PR.RL1,-(R5)
(5)  000112' 010345                                             MOV     R3,-(R5)
(4)  000114' 010045                                             MOV     R0,-(R5)
(3)  000116' 004767  000000G                                    JSR     PC,NEWBA
(3)  000122' 012605                                             MOV     (SP)+,R5
(4)  000124' 012504                                             MOV     (R5)+,R4
000126'                         IF BASE(R5) NE #0 THEN
(6)  000126' 005765  000002                                     TST     BASE(R5)
(9)  000132' 001402                                             BEQ     50002$
000134'                                 LET R4 := BASE(R5)
(4)  000134' 016504  000002                                     MOV     BASE(R5),R4
000140'                         ENDIF
(4)  000140'                                                    50002$:
```

```
                        ;+
                        ; CALL THE ROUTINE WHICH WILL MOVE THE CODE
                        ;-

000140'                 CALL RELOC IN <R0,R3,R4,PR.RL1>
(3)  000140' 010546                                             MOV     R5,-(SP)
(7)  000142' 016745  177632                                     MOV     PR.RL1,-(R5)
(6)  000146' 010445                                             MOV     R4,-(R5)
(5)  000150' 010345                                             MOV     R3,-(R5)
(4)  000152' 010045                                             MOV     R0,-(R5)
(3)  000154' 004767  000000G                                    JSR     PC,RELOC
(3)  000160' 012605                                             MOV     (SP)+,R5


                        ;+
                        ; IF THERE WAS A RELOCATION ERROR, THE EXERCISER WAS RETURNED
                        ; TO ITS STARTING POINT, SO THE NEW BASE EQUALS THE OLD.
                        ;-

000162'                 IF #RELERR SETIN DT.ERR(R0) THEN
(6)  000162' 032760  000020  000020                            BIT     #RELERR,DT.ERR(R
(9)  000170' 001401                                             BEQ     50003$
000172'                         LET R4 := R3
(4)  000172' 010304                                             MOV     R3,R4
000174'                 ENDIF
(4)  000174'                                                50003$:


                        ;+
                        ; IF THE RELOCATION ERROR BIT IS NOT SET
                        ; IN THE ERROR WORD OF THE DATA TABLE, WE HAD A SUCCESSFUL
                        ; RELOCATION, SO FORM AND OUTPUT A MESSAGE, UNLESS INHIBITTED BY THE
                        ; SOFTWARE SWITCH REGISTER.  OTHERWISE, OUTPUT AN ERROR MESSAGE.
                        ;-

000174'                 IF #RELERR NOTSETIN  DT.ERR(R0) THEN
(6)  000174' 032760  000020  000020                            BIT     #RELERR,DT.ERR(R
(9)  000202' 001061                                             BNE     50004$
000204'                         IF #INHREL NOTSETIN DT.SWR(R0) THEN
(6)  000204' 032760  001000  000056                            BIT     #INHREL,DT.SWR(R
(9)  000212' 001054                                             BNE     50005$


                        ;+
                        ; CONVERT THE NEW BASE OFFSET TO A PHYSICAL ADDRESS AND PUT IT
                        ; IN THE PROPER FORMAT FOR BOAC TO USE.
                        ;-

000214'                                 LET R1 := R4 SHIFT #+6
(4)  000214' 010401                                             MOV     R4,R1
(7)  000216' 006301                                             ASL     R1
(7)  000220' 006301                                             ASL     R1
(7)  000222' 006301                                             ASL     R1
(7)  000224' 006301                                             ASL     R1
(7)  000226' 006301                                             ASL     R1
```

```
(7)  000230' 006301                                                                 ASL     R1
679  000232'                                LET R2 := SWAP R4
(6)  000232' 010402                                                                 MOV     R4,R2
(6)  000234' 000302                                                                 SWAB    R2
680
681                                 ;+
682                                 ; IF THE UNIBUS MAP IS ENABLED, SET UP FOR 22-BIT ADDRESS CALCULATION,
683                                 ; OTHERWISE SET UP FOR 18-BIT CALCULATION.
684                                 ;-
685
686  000236'                        IF #MAPSTAT SETIN DT.ST0(R0) THEN
(6)  000236' 032760  000200  000010                                                BIT     #MAPSTAT,DT.ST0(
(9)  000244' 001405                                                                 BEQ     50006$
687  000246'                               LET R2 := R2 SHIFT #-2
(7)  000246' 006202                                                                 ASR     R2
(7)  000250' 006202                                                                 ASR     R2
688  000252'                               LET R2 := R2 CLR.BY #177400
(6)  000252' 042702  177400                                                         BIC     #177400,R2
689  000256'                        ELSE
(4)  000256' 000404                                                                 BR      50007$
(3)  000260'                                                                50006$:
690  000260'                               LET R2 := R2 SHIFT #+2
(7)  000260' 006302                                                                 ASL     R2
(7)  000262' 006302                                                                 ASL     R2
691  000264'                               LET R2 := R2 CLR.BY #177717
(6)  000264' 042702  177717                                                         BIC     #177717,R2
692  000270'                        ENDIF
(4)  000270'                                                                50007$:
693
694  000270'                        CALL BOAC IN <R0,R1,R2,#PR.NBR>
(3)  000270' 010546                                                                 MOV     R5,-(SP)
(7)  000272' 012745  000024'                                                        MOV     #PR.NBR,-(R5)
(6)  000276' 010245                                                                 MOV     R2,-(R5)
(5)  000300' 010145                                                                 MOV     R1,-(R5)
(4)  000302' 010045                                                                 MOV     R0,-(R5)
(3)  000304' 004767  000000G                                                        JSR     PC,BOAC
(3)  000310' 012605                                                                 MOV     (SP)+,R5
695  000312'                        CALL MSGDHOOK IN <R0,#MSGPOP,#PR.MSG,#2$>
(3)  000312' 010546                                                                 MOV     R5,-(SP)
(7)  000314' 012745  000344'                                                        MOV     #2$,-(R5)
(6)  000320' 012745  000006'                                                        MOV     #PR.MSG,-(R5)
(5)  000324' 012745  000002                                                         MOV     #MSGPOP,-(R5)
(4)  000330' 010045                                                                 MOV     R0,-(R5)
(3)  000332' 004767  000000G                                                        JSR     PC,MSGDHOOK
(3)  000336' 012605                                                                 MOV     (SP)+,R5
696  000340'                        INLINE< 1$: NOP>                                1$: NOP
(2)  000340' 000240
697  000342'                        INLINE< BR 1$>                                  BR 1$
(2)  000342' 000776
698
699  000344'                   INLINE <2$:>                                   2$:
(2)  000344'
700  000344'                          ENDIF                                  50005$:
(4)  000344'
701
702                                 ;+
```

```
                                ; OUTPUT AN ERROR MESSAGE.
                                ;-

000344'                         ELSE
(4) 000344' 000415                                                      BR      50010$
(3) 000346'                                                     50004$:

000346'                                 CALL MSGDHOOK IN <R0,#MSGPOP,#PR.MS1,#10$>
(3) 000346' 010546                                                      MOV     R5,-(SP)
(7) 000350' 012745  000400'                                             MOV     #10$,-(R5)
(6) 000354' 012745  000036'                                             MOV     #PR.MS1,-(R5)
(5) 000360' 012745  000002                                              MOV     #MSGPOP,-(R5)
(4) 000364' 010045                                                      MOV     R0,-(R5)
(3) 000366' 004767  000000G                                             JSR     PC,MSGDHOOK
(3) 000372' 012605                                                      MOV     (SP)+,R5

000374'                                 INLINE<9$:      NOP>
(2) 000374' 000240                                                      9$:     NOP
000376'                                 INLINE <BR      9$>
(2) 000376' 000776                                                      BR      9$
000400'                         INLINE <10$:>
(2) 000400'                                                             10$:

000400'                         ENDIF
(4) 000400'                                                     50010$:

                                ;+
                                ; CLEAR THE FIRST RELOCATION FLAG
                                ;-

000400'                         LET PR.RL1 := #0
(4) 000400' 005067  177374                                              CLR     PR.RL1

                                ;+
                                ; RESTORE REGISTERS
                                ;-

000404'                         CALL RESREG
(3) 000404' 004767  000000G                                             JSR     PC,RESREG

000410'                         ENDRTN
(3) 000410'                                                     50000$:
(3) 000410'                                                     50001$:
(2) 000410' 000207                                                      RTS     PC

        000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BASE  = 000002
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOAC  = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NEWBA = ****** G
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRRLOC  000064RG
PR.MSG  000006R
PR.MS1  000036R
PR.NBR  000024R
PR.RL1  000000RG
PR.TBL  000002R
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELOC = ****** G
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
```

```
SVR2  = 000066        UIPDR3= 177606        $F$BAD= 000401        $IFLEV= 177777        $$CASE= 000000
SVR3  = 000070        UIPDR4= 177610        $F$BLA= 000170        $ISK0 = 000001        $$DST = 000000
SVR4  = 000072        UIPDR5= 177612        $F$CAS= 000150        $ISK1 = 000001        $$ELOC= 000402
SVR5  = 000074        UIPDR6= 177614        $F$DEC= 000220        $ISK2 = 000001        $$ERFL= 000000
SVR6  = 000076        UIPDR7= 177616        $F$DO = 000340        $LOCTA= 177777        $$FLAG= 000001
SYSCNT= 000052        WASADR= 000104        $F$FAL= 000405        $LSTIN= 000001        $$FROM= 000000
SYSERR= 000100        WBSTAT= 000040        $F$GOO= 000400        $LSTTA= 000001        $$LOC = 000244R
TABL  = 000000        WBUFEA= 000136        $F$IF = 000110        SNESTL= 177777        $$LOCN= 000000
TMPIO = 000002        WBUFPA= 000134        $F$INC= 000210        $NSK0 = 000300        $$REG = 177777
TQOVF = 000002        WBUFRQ= 000140        $F$LOO= 000200        $NSK1 = 000110        $$RETU= 000000
UIPAR0= 177640        WBUFSZ= 000142        $F$NAM= 000160        $NSK2 = 000110        $$RTN1= 050000
UIPAR1= 177642        WDFR  = 000116        $F$NO = 000403        $NSK3 = 000110        $$RTN2= 050001
UIPAR2= 177644        WDTO  = 000114        $F$OR = 000320        $SAVLE= 177777        $$SRC = 000000
UIPAR3= 177646        WTINRE= 000352        $F$RTI= 000350        $TAGLE= 177777        $$TGSV= 000000
UIPAR4= 177650        WTWHMI= 000222        $F$RTN= 000300        $TAGNU= 050011        $$TGS1= 000000
UIPAR5= 177652        XFLAG = 000005        $F$SEL= 000140        $TEMP = 000300        $$TGS2= 000000
UIPAR6= 177654        XOFF  = 000023        $F$THE= 000330        $TSK0 = 050010        $$TO  = 000000
UIPAR7= 177656        XON   = 000021        $F$TRU= 000404        $TSK1 = 050005        $$$TAG= 050000
UIPDR0= 177600        $BGNLE= 177777        $F$UNT= 000130        $TSK2 = 050007        .     = 000412R
UIPDR1= 177602        $ERFLG= 000400        $F$WHI= 000120        $$ARGC= 000004
UIPDR2= 177604        $F$AND= 000310        $F$YES= 000402        $$BYTE= 000403

. ABS.  000000    000
        000412    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRRLOC,DSKZ:PRRLOC=SPMAC/ML,EQUATE,PRRLOC
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 37/20=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                                .SBTTL BADVEC ROUTINE

                                ;+
                                ; EXECUTE EMT INSTRUCTION TO INVOKE ROUTINE TO DETERMINE A BAD VECTOR
                                ; THE EMT WILL ALSO PUSH THE PSW ONTO THE STACK SO IT
                                ; CAN BE READ. BADVC1 IS FOR VECTORS LESS THAN 400,
                                ; BADVEC IS FOR VECTORS 400 AND UP.
                                ;-

000014'                         IROUTINE BADVC1
000014'                                                                         BADVC1:
000014'                         INLINE <EMT>
                                                                                        EMT
000014' 104000
000016'                         ENDRTI
                                                                                50000$:
000016'                                                                         50001$:
000016'                                                                                 RTI
000016' 000002

000020'                         IROUTINE BADVEC
000020'                                                                         BADVEC:
000020'                         INLINE  <EMT>
                                                                                        EMT
000020' 104000
000022'                         ENDRTI
                                                                                50000$:
000022'                                                                         50001$:
000022'                                                                                 RTI
000022' 000002
```

```
                        .TITLE RELOC - RELOCATE CODE
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       RELOC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE RELOCATES THE MOVABLE PORTION OF THE RUNTIME
                        ;       EXERCISER FROM ITS CURRENT LOCATION IN MEMORY TO A NEW AREA.
                        ;       IF A BAD MEMORY LOCATION IS ENCOUNTERED DURING THE MOVE,
                        ;       THE EXERCISER IS RETURNED TO THE CURRENT LOCATION AND
                        ;       AN ERROR INDICATOR IS SET.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS, OLD ADDRESS OFFSET, NEW ADDRESS OFFSET,
                        ;       FIRST RELOCATION FLAG
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ADDR,DT.ESIZ,DT.ERR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ERR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;       BADMEM                  ;REPORT BAD MEMORY TRANSFER
                        ;       GPA                     ;GET PHYSICAL ADDRESS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       CURRENT COPY OF CODE MAY BE OVERLAYED
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL RELOC IN <A,B,C,D>
                        ;               A=ADDRESS OF DATA TABLE
                        ;               B=OLD ADDRESS OFFSET
                        ;               C=NEW ADDRESS OFFSET
                        ;               D=FIRST RELOCATION FLAG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES


                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  RELOC                           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  GPA                             ;ROUTINE TO GET A PHYSICAL ADDRESS
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                .GLOBL  BADMEM                          ;REPORT BAD MEMORY TRANSFER
                                ;
                                ;****************
                                ;
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  RL.REG: .WORD 0                         ;REGISTER VALUE NEEDED TO POINT
                                                                        ; TO TOP OF EXERCISER
000002' 000000                  RL.PAR: .WORD 0                         ;PAR VALUE NEEDED TO POINT
                                                                        ; TO TOP OF EXERCISER
000004' 000000                  RL.ERRCHK:      .WORD 0                 ;ERROR-SERVICING FLAG. WHEN SET,
                                                                        ; SERVICE MOVE ERRORS.  WHEN NOT SET,
                                                                        ; IGNORE ERRORS.  SET AT BEGINNING
                                                                        ; OF ROUTINE, CLEARED AFTER 1ST ERROR
000006' 000000                  RL.RVAL:        .WORD 0                 ;TEMPORARY STORAGE FOR RL.DN SUBROUTINE
000010' 000003                  RL.TBL: .BLKW 3                         ;STORAGE FOR GPA ROUTINE
000016' 000000                  RL.DBD: .WORD 0                         ;BAD DATA OF FAILING LOCATION
000020' 000000                  RL.DGD: .WORD 0                 ;GOOD DATA OF FAILING LOCATION
```

```
                                .SBTTL RELOC ROUTINE


                                ROUTINE RELOC <TABL,OLDPAR,NEWPAR,RL1>
(2)  000022'                                                                    RELOC:


                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
(3)  000022' 004767  000000G                                                   JSR     PC,SAVREG


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

000026'                         LET R0 := TABL(R5)
(4)  000026' 016500  000000                                                    MOV     TABL(R5),R0


                                ;+
                                ; RAISE THE PROCESSOR'S PRIORITY TO 7.
                                ;-

000032'                         PUSH #PRI7
(2)  000032' 012746  000340                                                    MOV     #PRI7,-(SP)
000036'                         PUSH #10$
(2)  000036' 012746  000044'                                                   MOV     #10$,-(SP)
000042'                         INLINE <RTI>
(2)  000042' 000002                                                            RTI

000044'                         INLINE <10$:>
(2)  000044'                                                                   10$:
                                ;+
                                ; IS THIS THE FIRST RELOCATION?
                                ;-

000044'                         IF RL1(R5) EQ #1 THEN
(6)  000044' 026527  000006  000001                                            CMP     RL1(R5),#1
(9)  000052' 001031                                                            BNE     50002$


                                ;+
                                ; YES.  FIND THE REGISTER AND PAR VALUES USED AS OFFSETS TO POINT TO
                                ; THE TOP OF THE EXERCISER.  THE REGISTER VALUE (RL.REG) IS BITS 0-12 OF THE EXERCISER
                                ; SIZE (DT.ESIZ).  THE PAR VALUE (RL.PAR) IS 200 MULTIPLIED BY ONE LESS
                                ; THAN THE VALUE IN BITS 13-15 OF DT.ESIZ.
                                ;-

000054'                                 LET R3 := DT.ESIZ(R0)
(4)  000054' 016003  000044                                                    MOV     DT.ESIZ(R0),R3
000060'                                 LET RL.REG := R3 CLR.BY #160000
```

```
   (4)  000060' 010367  177714                                                  MOV     R3,RL.REG
   (6)  000064' 042767  160000  177706                                          BIC     #160000,RL.REG
   651
   652  000072'                              LET R3 := SWAP R3
   (6)  000072' 000303                                                          SWAB    R3
   653  000074'                              LET R3 := R3 SHIFT #-5
   (7)  000074' 006203                                                          ASR     R3
   (7)  000076' 006203                                                          ASR     R3
   (7)  000100' 006203                                                          ASR     R3
   (7)  000102' 006203                                                          ASR     R3
   (7)  000104' 006203                                                          ASR     R3
   654  000106'                              LET R3 := R3 CLR.BY #177770
   (6)  000106' 042703  177770                                                  BIC     #177770,R3
   655  000112'                              LET RL.PAR := #0
   (4)  000112' 005067  177664                                                  CLR     RL.PAR
   656  000116'                              LET R3 := R3 - #1
   (6)  000116' 005303                                                          DEC     R3
   657  000120'                              WHILE R3 GT #0 DO
   (4)  000120'                                                         50003$:
   (6)  000120' 005703                                                          TST     R3
   (9)  000122' 003405                                                          BLE     50004$
   658  000124'                                      LET RL.PAR := RL.PAR + #200
   (6)  000124' 062767  000200  177650                                          ADD     #200,RL.PAR
   659  000132'                                      LET R3 := R3 - #1
   (6)  000132' 005303                                                          DEC     R3
   660  000134'                              ENDDO
   (4)  000134' 000771                                                          BR      50003$
   (3)  000136'                                                         50004$:
   661  000136'                         ENDIF
   (4)  000136'                                                         50002$:
   662
   663
   664                                  ;+
   665                                  ; SET THE ERROR-SERVICING FLAG
   666                                  ;-
   667
   668  000136'                         LET RL.ERRCHK := #1
   (4)  000136' 012767  000001  177640                                          MOV     #1,RL.ERRCHK
   669
   670
   671                                  ;+
   672                                  ; ARE WE MOVING UP OR DOWN?
   673                                  ;-
   674
   675  000144'                         IF NEWPAR(R5) HI OLDPAR(R5) THEN
   (6)  000144' 026565  000004  000002                                          CMP     NEWPAR(R5),OLDPA
   (9)  000152' 101427                                                          BLOS    50005$
   676
   677
   678                                  ;+
   679                                  ; MOVING UP
   680                                  ;-
   681
   682
   683                                  ;+
   684                                  ; SET THE OLD ADDRESS POINTER TO THE TOP ADDRESS OF THE OLD AREA+2
```

```
                                ;-
                                
000154'                                 LET R1 := RL.REG + #20002
(4) 000154' 016701  177620                                              MOV     RL.REG,R1
(6) 000160' 062701  020002                                              ADD     #20002,R1
                                
000164'                                 LET @#KIPAR1 := OLDPAR(R5) + RL.PAR
(4) 000164' 016537  000002  172342                                      MOV     OLDPAR(R5),@#KIP
(6) 000172' 066737  177604  172342                                      ADD     RL.PAR,@#KIPAR1
                                
                                
                                ;+
                                ; SET THE NEW ADDRESS POINTER TO THE TOP ADDRESS OF THE NEW AREA+2
                                ;-
                                
000200'                                 LET R2 := RL.REG + #40002
(4) 000200' 016702  177574                                              MOV     RL.REG,R2
(6) 000204' 062702  040002                                              ADD     #40002,R2
000210'                                 LET @#KIPAR2 := NEWPAR(R5) + RL.PAR
(4) 000210' 016537  000004  172344                                      MOV     NEWPAR(R5),@#KIP
(6) 000216' 066737  177560  172344                                      ADD     RL.PAR,@#KIPAR2
000224'                                 CALL RL.UP
(3) 000224' 004767  000146                                              JSR     PC,RL.UP
                                
                                
000230'                         ELSE
(4) 000230' 000414                                                      BR      50006$
(3) 000232'                                                     50005$:
                                
                                
                                ;+
                                ; MOVING DOWN
                                ;-
                                
                                ;+
                                ; SET THE OLD ADDRESS POINTER TO THE BOTTOM OF THE OLD AREA
                                ;-
                                
000232'                                 LET R1 := #20000
(4) 000232' 012701  020000                                              MOV     #20000,R1
000236'                                 LET @#KIPAR1 := OLDPAR(R5)
(4) 000236' 016537  000002  172342                                      MOV     OLDPAR(R5),@#KIP
                                
                                
                                ;+
                                ; SET THE NEW ADDRESS POINTER TO THE BOTTOM OF THE NEW AREA
                                ;-
                                
000244'                                 LET R2 := #40000
(4) 000244' 012702  040000                                              MOV     #40000,R2
000250'                                 LET @#KIPAR2 := NEWPAR(R5)
(4) 000250' 016537  000004  172344                                      MOV     NEWPAR(R5),@#KIP
000256'                                 CALL RL.DN
(3) 000256' 004767  000224                                              JSR     PC,RL.DN
000262'                         ENDIF
(4) 000262'                                                     50006$:
```

```
                                ;+
                                ; NOW RELOAD THE MEMORY MANAGEMENT PAR'S SO THE EXERCISER
                                ; CALL RUN OUT OF THE NEW AREA.  PAR0 IS NOT CHANGED SINCE IT POINTS
                                ; TO THE LOWEST 4K OF THE EXERCISER, WHICH ALWAYS STAYS IN LOWEST MEMORY.
                                ;-

000262'                         LET R1 := #KIPAR1
(4) 000262' 012701  172342                                          MOV     #KIPAR1,R1
000266'                         LET R2 := NEWPAR(R5)
(4) 000266' 016502  000004                                          MOV     NEWPAR(R5),R2
000272'                         LET R3 := #6
(4) 000272' 012703  000006                                          MOV     #6,R3
000276'                         REPEAT
(3) 000276'                                                 50007$:
000276'                                 LET R3 := R3 - #1
(6) 000276' 005303                                                  DEC     R3
000300'                                 LET (R1)+ := R2
(4) 000300' 010221                                                  MOV     R2,(R1)+
000302'                                 LET R2 := R2 + #200
(6) 000302' 062702  000200                                          ADD     #200,R2
000306'                         UNTIL R3 EQ #0
(3) 000306' 005703                                                  TST     R3
(6) 000310' 001372                                                  BNE     50007$

                                ;+
                                ; UPDATE THE CURRENT BASE ADDRESS.
                                ;-

000312'                         LET DT.ADDR(R0) := NEWPAR(R5)
(4) 000312' 016560  000004  000042                                  MOV     NEWPAR(R5),DT.AD

                                ;+
                                ; LOWER THE PROCESSOR'S PRIORITY.
                                ;-

000320'                         PUSH #PRI0
(2) 000320' 012746  000000                                          MOV     #PRI0,-(SP)
000324'                         PUSH #20$
(2) 000324' 012746  000332'                                         MOV     #20$,-(SP)
000330'                         INLINE <RTI>
(2) 000330' 000002                                                  RTI

000332'                         INLINE <20$:>
(2) 000332'                                                 20$:

                                ;+
                                ; IF THE RELOCATION WAS UNSUCCESSFUL BECAUSE OF A BAD MEMORY
                                ; LOCATION, OUTPUT A MESSAGE INDICATING THAT LOCATION.
                                ;-
```

```
  766  000332'                          IF #RELERR SETIN DT.ERR(R0) THEN
  (6)  000332' 032760  000020  000020                                        BIT     #RELERR,DT.ERR(R
  (9)  000340' 001413                                                        BEQ     50010$
  767  000342'                                  CALL BADMEM IN <R0,#RL.TB1+2,RL.DBD,RL.DGD>
  (3)  000342' 010546                                                        MOV     R5,-(SP)
  (7)  000344' 016745  177450                                                MOV     RL.DGD,-(R5)
  (6)  000350' 016745  177442                                                MOV     RL.DBD,-(R5)
  (5)  000354' 012745  000012'                                               MOV     #RL.TB1+2,-(R5)
  (4)  000360' 010045                                                        MOV     R0,-(R5)
  (3)  000362' 004767  000000G                                               JSR     PC,BADMEM
  (3)  000366' 012605                                                        MOV     (SP)+,R5
  768  000370'                          ENDIF
  (4)  000370'                                                       50010$:
  769
  770                                   ;+
  771                                   ; RESTORE REGISTERS AND RETURN
  772                                   ;-
  773
  774  000370'                          CALL RESREG
  (3)  000370' 004767  000000G                                               JSR     PC,RESREG
  775
  776  000374'                          ENDRTN
  (3)  000374'                                                       50000$:
  (3)  000374'                                                       50001$:
  (2)  000374' 000207                                                        RTS     PC
  777
  778
```

```
                                .SBTTL  ROUTINE RL.UP - MOVE CODE UP

                                ;+
                                ; THIS SUBROUTINE MOVES CODE TO AN AREA HIGHER IN MEMORY THAN THE CURRENT
                                ; AREA.  THE MOVE BEGINS WITH THE HIGHEST ADDRESS TO BE RELOCATED AND WORKS
                                ; DOWNWARD TO THE LOWEST ADDRESS.  EACH TIME THE PAR'S ARE ADJUSTED TO THE
                                ; NEXT LOWEST 4K AREA, THE REGISTERS ARE SET TO THE TOP OF THE 4K AREA.
                                ;-

000376'                         ROUTINE RL.UP
000376'                                                                 RL.UP:

                                WHILE @#KIPAR1 HIS OLDPAR(R5) DO
000376'                                                                 50002$:
000376' 023765  172342  000002                                                  CMP     @#KIPAR1,OLDPAR(
000404' 103437                                                                  BLO     50003$
000406'                                 WHILE R1 GT #20000 DO
000406'                                                                 50004$:
000406' 020127  020000                                                          CMP     R1,#20000
000412' 003421                                                                  BLE     50005$

                                ;+
                                ; DECREMENT THE POINTERS AND MOVE A WORD
                                ;-

000414'                                         LET -(R2) := -(R1)
000414' 014142                                                                  MOV     -(R1),-(R2)

                                ;+
                                ; COMPARE THE CONTENTS OF THE OLD AND NEW ADDRESSES.  IF THEY MATCH, KEEP
                                ; GOING.  IF THEY DON'T MATCH, THEN SEE IF THE ERROR-SERVICING FLAG IS SET.  IF
                                ; IT IS, CALL THE ERROR ROUTINE AND THEN MOVE THE CODE BACK TO
                                ; WHERE IT WAS AND RETURN TO THE CALLER.  IF THE
                                ; ERROR-SERVICING FLAG IS NOT SET, THEN WE HAVE ALREADY DETECTED AN ERROR
                                ; AND DO NOT WISH TO CHECK FOR ANY MORE, SO JUST CONTINUE MOVING CODE.
                                ;-

000416'                                         IF (R1) NE (R2) THEN
000416' 021112                                                                  CMP     (R1),(R2)
000420' 001415                                                                  BEQ     50006$
000422'                                                 IF RL.ERRCHK EQ #1 THEN
000422' 026727  177356  000001                                                  CMP     RL.ERRCHK,#1
000430' 001011                                                                  BNE     50007$
000432'                                                         CALL RL.GOBACK
000432' 004767  000210                                                          JSR     PC,RL.GOBACK
000436'                                                         LET R1 := R1 + #2
000436' 062701  000002                                                          ADD     #2,R1
000442'                                                         LET R2 := R2 + #2
000442' 062702  000002                                                          ADD     #2,R2
000446'                                                         CALL RL.DN
000446' 004767  000034                                                          JSR     PC,RL.DN
000452'                                                         RETURN
000452' 000414                                                                  BR      50000$
```

```
                                        ENDIF
                                                                50007$:
                                    ENDIF
                                                                50006$:
                                ENDDO
                                                                        BR      50004$
                                                                50005$:

                        ;+
                        ; POSITION THE POINTERS TO THE TOP OF THE NEXT LOWER 4K PIECE.
                        ;-

                                LET R1 := #40000
                                                                        MOV     #40000,R1
                                LET R2 := #60000
                                                                        MOV     #60000,R2
                                LET @#KIPAR1 := @#KIPAR1 - #200
                                                                        SUB     #200,@#KIPAR1
                                LET @#KIPAR2 := @#KIPAR2 - #200
                                                                        SUB     #200,@#KIPAR2
                        ENDDO
                                                                        BR      50002$
                                                                50003$:
                        ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC
```

```
                                .SBTTL  ROUTINE RL.DN - MOVE CODE DOWN

                                ;+
                                ; THIS SUBROUTINE MOVES CODE TO AN AREA LOWER IN MEMORY THAN THE CURRENT
                                ; AREA.  THE MOVE BEGINS WITH THE LOWEST ADDRESS TO BE RELOCATED AND WORKS
                                ; UPWARD TO THE HIGHEST ADDRESS.  EACH TIME THE PAR'S ARE ADJUSTED TO POINT TO
                                ; THE NEXT HIGHER 4K AREA OF MEMORY, THE REGISTERS ARE SET TO POINT TO
                                ; THE BOTTOM OF THE 4K AREA. AND ARE INCREMENTED UNTIL THEY POINT TO THE
                                ; TOP ADDRESS OF THAT 4K AREA.  WHEN THE HIGHEST 4K SEGMENT OF THE EXERCISER IS
                                ; REACHED, HOWEVER, THE REGISTERS MUST BE INCREMENTED ONLY TO THE HIGHEST EXERCISER
                                ; ADDRESS AND NOT TO THE TOP OF THE 4K AREA.
                                ;-


000506'                         ROUTINE RL.DN
(2) 000506'                                                             RL.DN:

                                LET R3 := OLDPAR(R5) + RL.PAR
(4) 000506' 016503  000002                                                      MOV     OLDPAR(R5),R3
(6) 000512' 066703  177264                                                      ADD     RL.PAR,R3
                                WHILE @#KIPAR1 LOS R3 DO
(4) 000516'                                                             50002$:
(6) 000516' 023703  172342                                                      CMP     @#KIPAR1,R3
(9) 000522' 101050                                                              BHI     50003$
                                        IF @#KIPAR1 LO R3 THEN
(6) 000524' 023703  172342                                                      CMP     @#KIPAR1,R3
(9) 000530' 103004                                                              BHIS    50004$
                                                LET RL.RVAL := #37776
(4) 000532' 012767  037776  177246                                              MOV     #37776,RL.RVAL
                                        ELSE
(4) 000540' 000406                                                              BR      50005$
(3) 000542'                                                             50004$:
                                                LET RL.RVAL := RL.REG + #20000
(4) 000542' 016767  177232  177236                                              MOV     RL.REG,RL.RVAL
(6) 000550' 062767  020000  177230                                              ADD     #20000,RL.RVAL
                                        ENDIF
(4) 000556'                                                             50005$:
                                        WHILE R1 LOS RL.RVAL DO
(4) 000556'                                                             50006$:
(6) 000556' 020167  177224                                                      CMP     R1,RL.RVAL
(9) 000562' 101015                                                              BHI     50007$

                                ;+
                                ; MOVE A WORD
                                ;-

                                        LET (R2) := (R1)
000564'                                                                 MOV     (R1),(R2)
(4) 000564' 011112

                                ;+
                                ; COMPARE THE CONTENTS OF THE OLD AND NEW ADDRESSES AND THEN INCREMENT
                                ; THE POINTERS. IF THE CONTENTS MATCH, KEEP GOING.  IF THEY DON'T MATCH,
                                ; SEE IF THE ERROR-SERVICING FLAG IS SET.  IF IT IS,
```

```
                        ; CALL THE ERROR ROUTINE, THEN MOVE THE CODE BACK TO WHERE IT WAS
                        ; AND RETURN TO THE CALLER.  IF THE ERROR-SERVICING FLAG IS NOT
                        ; SET, THEN WE HAVE ALREADY ENCOUNTERED AN ERROR AND DO NOT WISH TO
                        ; CHECK FOR ANY MORE, SO JUST CONTINUE MOVING CODE.
                        ;-

000566'                         IF (R1)+ NE (R2)+ THEN
(6) 000566' 022122                                                      CMP     (R1)+,(R2)+
(9) 000570' 001411                                                      BEQ     50010$
000572'                                 IF RL.ERRCHK EQ #1 THEN
(6) 000572' 026727  177206  000001                                      CMP     RL.ERRCHK,#1
(9) 000600' 001005                                                      BNE     50011$
000602'                                         CALL RL.GOBACK
(3) 000602' 004767  000040                                              JSR     PC,RL.GOBACK
000606'                                         CALL RL.UP
(3) 000606' 004767  177564                                              JSR     PC,RL.UP
000612'                                         RETURN
(4) 000612' 000414                                                      BR      50000$
000614'                                 ENDIF
(4) 000614'                                                     50011$:
000614'                         ENDIF
(4) 000614'                                                     50010$:
000614'                 ENDDO
(4) 000614' 000760                                                      BR      50006$
(3) 000616'                                                     50007$:


                        ;+
                        ; POSITION THE POINTERS TO THE BOTTOM OF THE NEXT 4K SEGMENT.
                        ;-

000616'                 LET R1 := #20000
(4) 000616' 012701  020000                                              MOV     #20000,R1
000622'                 LET R2 := #40000
(4) 000622' 012702  040000                                              MOV     #40000,R2
000626'                 LET @#KIPAR1 := @#KIPAR1 + #200
(6) 000626' 062737  000200  172342                                      ADD     #200,@#KIPAR1
000634'                 LET @#KIPAR2 := @#KIPAR2 + #200
(6) 000634' 062737  000200  172344                                      ADD     #200,@#KIPAR2
000642'         ENDDO
(4) 000642' 000725                                                      BR      50002$
(3) 000644'                                                     50003$:
000644'         ENDRTN
(3) 000644'                                                     50000$:
(3) 000644'                                                     50001$:
(2) 000644' 000207                                                      RTS     PC
```

```
                                .SBTTL  ROUTINE RL.GOBACK - RETURN CODE TO ORIGINAL LOCATION


                                ;+
                                ; THIS SUBROUTINE IS CALLED UPON THE DETECTION OF A RELOCATION ERROR.
                                ; IT PERFORMS THE FOLLOWING FUNCTIONS:
                                ; (1) SAVES THE ADDRESS OF THE FAILING LOCATION, ALONG WITH ITS
                                ;     ACTUAL AND SHOULD-BE CONTENTS.
                                ; (2) CLEARS THE ERROR-SERVICING FLAG.
                                ; (3) SETS THE RELOCATION ERROR BIT OF THE ERROR WORD IN THE
                                ;     DATA TABLE.
                                ; (4) REVERSES THE POINTERS.  I.E., MOVES THE CONTENTS OF THE OLD ADDRESS
                                ;     POINTER INTO THE NEW ADDRESS POINTER, AND VICE VERSA, SO WE
                                ;     CAN MOVE THE CODE BACK.
                                ; (5) REVERSES THE VALUES IN OLDPAR AND NEWPAR.  THIS ALSO IS
                                ;     NECESSARY FOR MOVING THE CODE BACK.
                                ;-


000646'                         ROUTINE RL.GOBACK
(2) 000646'                                                     RL.GOBACK:

                                LET RL.TB1 := R2
(4) 000646' 010267  177136                                      MOV     R2,RL.TB1
                                LET RL.DBD := (R2)
(4) 000652' 011267  177140                                      MOV     (R2),RL.DBD
                                LET RL.DGD := (R1)
(4) 000656' 011167  177136                                      MOV     (R1),RL.DGD
                                CALL GPA IN <R0,#RL.TB1>
(3) 000662' 010546                                              MOV     R5,-(SP)
(5) 000664' 012745  000010'                                     MOV     #RL.TB1,-(R5)
(4) 000670' 010045                                              MOV     R0,-(R5)
(3) 000672' 004767  000000G                                     JSR     PC,GPA
(3) 000676' 012605                                              MOV     (SP)+,R5

                                LET RL.ERRCHK := #0
(4) 000700' 005067  177100                                      CLR     RL.ERRCHK
                                LET DT.ERR(R0) := DT.ERR(R0) SET.BY #RELERR
(6) 000704' 052760  000020  000020                              BIS     #RELERR,DT.ERR(R

                                LET R4 := @#KIPAR1
(4) 000712' 013704  172342                                      MOV     @#KIPAR1,R4
                                LET @#KIPAR1 := @#KIPAR2
(4) 000716' 013737  172344  172342                              MOV     @#KIPAR2,@#KIPAR
                                LET @#KIPAR2 := R4
(4) 000724' 010437  172344                                      MOV     R4,@#KIPAR2

                                LET R4 := NEWPAR(R5)
(4) 000730' 016504  000004                                      MOV     NEWPAR(R5),R4
                                LET NEWPAR(R5) := OLDPAR(R5)
(4) 000734' 016565  000002  000004                              MOV     OLDPAR(R5),NEWPA
                                LET OLDPAR(R5) := R4
(4) 000742' 010465  000002                                      MOV     R4,OLDPAR(R5)
000746'                         ENDRTN
```

```
   (3)  000746'                                                          50000$:
   (3)  000746'                                                          50001$:
   (2)  000746' 000207                                                           RTS     PC
   943          000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BADMEM= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GPA   = ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NEWPAR= 000004
NOAPTY= 000002
NULL  = 000000
OLDPAR= 000002
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELOC   000022RG
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RL.DBD  000016R
RL.DGD  000020R
RL.DN   000506R
RL.ERR  000004R
RL.GDB  000646R
RL.PAR  000002R
RL.REG  000000R
RL.RVA  000006R
RL.TB1  000010R
RL.UP   000376R
RL1   = 000006
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
RSVALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
```

```
STATBI= 064757          UIPDR0= 177600          $F$AND= 000310          $ISK0 = 000001          $$BYTE= 000403
STAT1 = 000027          UIPDR1= 177602          $F$BAD= 000401          $ISK1 = 000001          $$CASE= 000000
SUSPND= 000001          UIPDR2= 177604          $F$BLA= 000170          $LOCTA= 177777          $$DST = 000000
SVR0  = 000062          UIPDR3= 177606          $F$CAS= 000150          $LSTIN= 000001          $$ELOC= 000402
SVR1  = 000064          UIPDR4= 177610          $F$DEC= 000220          $LSTTA= 000001          $$ERFL= 000000
SVR2  = 000066          UIPDR5= 177612          $F$DO = 000340          $NESTL= 177777          $$FLAG= 000001
SVR3  = 000070          UIPDR6= 177614          $F$FAL= 000405          $NSK0 = 000300          $$FROM= 000000
SVR4  = 000072          UIPDR7= 177616          $F$GOO= 000400          $NSK1 = 000120          $$LOC = 000600R
SVR5  = 000074          WASADR= 000104          $F$IF = 000110          $NSK2 = 000120          $$LOCN= 000000
SVR6  = 000076          WBSTAT= 000040          $F$INC= 000210          $NSK3 = 000110          $$REG = 177777
SYSCNT= 000052          WBUFEA= 000136          $F$LOO= 000200          $NSK4 = 000110          $$RETU= 000000
SYSERR= 000100          WBUFPA= 000134          $F$NAM= 000160          $SAVLE= 177777          $$RTN1= 050000
TABL  = 000000          WBUFRQ= 000140          $F$NO = 000403          $SSK0 = 050003          $$RTN2= 050001
TMPIO = 000002          WBUFSZ= 000142          $F$OR = 000320          $TAGLE= 177777          $$SRC = 000000
TQOVF = 000002          WDFR  = 000116          $F$RTI= 000350          $TAGNU= 050002          $$TGSV= 000000
UIPAR0= 177640          WDTO  = 000114          $F$RTN= 000300          $TEMP = 000300          $$TGS1= 000000
UIPAR1= 177642          WTINRE= 000352          $F$SEL= 000140          $TSK0 = 050002          $$TGS2= 000000
UIPAR2= 177644          WTWHMI= 000222          $F$THE= 000330          $TSK1 = 050003          $$TO  = 000002
UIPAR3= 177646          XFLAG = 000005          $F$TRU= 000404          $TSK2 = 050006          $$$TAG= 050000
UIPAR4= 177650          XOFF  = 000023          $F$UNT= 000130          $TSK3 = 050007          .     = 000750R
UIPAR5= 177652          XON   = 000021          $F$WHI= 000120          $TSK4 = 050010
UIPAR6= 177654          $BGNLE= 177777          $F$YES= 000402          $TSK5 = 050011
UIPAR7= 177656          $ERFLG= 000400          $IFLEV= 177777          $$ARGC= 000000

. ABS.  000000      000
        000750      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RELOC,DSKZ:RELOC=SPMAC/ML,EQUATE,RELOC
 RUN-TIME: 22 12 .4 SECONDS
 RUN-TIME RATIO: 57/35=1.6
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE RPIRQ (PIRQ SERVICE ROUTINE)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       RPIRQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE SERVICES THE PIRQ TRAP CALL (WHICH IS AN IOT).
                        ;       IT ENQUEUES HEADER ADDR. AND RETURN PC OF THE MODULE
                        ;       INTO THE CONTROL QUEUE, UPDATES THE SYSTEM STACK AND RETURNS TO
                        ;       THE INTERRUPTED ROUTINE.  TO ENSURE THAT THE ENQUEUEING ROUTINE
                        ;       IS NOT INTERRUPTED, THIS MODULE IS HANDLED AT PRIORITY 7.
                        ;       IF A BACKGROUND MODULE IS THE INTERRUPTED MODULE, IT WILL
                        ;       BE TAKEN OFF THE AIR.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       DATA TABLE ADDRESS
                        ;       BACKGROUND HEADER STORAGE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ENQCQ           ;PUT ITEMS ON THE CONTROL QUEUE.
                        ;       ERREC           ;ERROR RECOVERY ROUTINE
                        ;       RSTRCY          ;RESTORE STATUS ROUTINE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       THIS ROUTINE IS ENTERED BY EXECUTING A 'PIRQ' INSTRUCTION.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       -----   --------    ----------  ----------------
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS & REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  RPIRQ                   ;MODULE ENTRY POINT

                                ;***************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL ENQCQ                    ;ROUTINE TO ENQUE ITEMS ON CONTROL QUEUE
                                .GLOBL ERREC                    ;ERROR RECOVERY ROUTINE
                                .GLOBL RSTRCY                   ;RESTORE STATUS ROUTINE

                                .GLOBL DTABLE                   ;SYSTEM DATA TABLE
                                .GLOBL DX.R5                    ;SYSTEM LINK REGISTER
                                .GLOBL DX.SP                    ;SYSTEM STACK
                                .GLOBL DX.DEQ                   ;RET ADDR IN BOX00

                                .GLOBL BA.HDR                   ;BKMOD HEADER ADDRESS
                                .GLOBL BA.RET                   ;BKMOD RETURN ADDRESS
                                .GLOBL BA.STAT                  ;BKMOD STATUS WORD

                                ; LOCAL STORAGE
                                ;
000000' 000000G                 RP.EXT: DX.DEQ                  ;GOTO ADDRESS IF INTERRUPTED MODULE IS A BK
        000005                          .REPT   ^D<5>           ;R5 STORAGE
                                        .WORD   0
                                        .ENDR
000014'                         RP.R5S:                         ;R5 STACK ALLOCATION FOR THIS MODULE
```

```
                                .SBTTL RPIRQ (CODE)

                                ;****************
                                ; PIRQ SERVICE ROUTINE STARTS HERE
                                ;

000014'                         INLINE <RPIRQ:>
(2) 000014'                                                     RPIRQ:

                                ;+
                                ; SAVE R1 AND LOAD IT WITH MODULE'S BEGIN ADDRESS
                                ;-

000014'                         PUSH R1
(2) 000014' 010146                                              MOV     R1,-(SP)
000016'                         LET R1 := 2(SP)
(4) 000016' 016601  000002                                      MOV     2(SP),R1

                                ;+
                                ; SAVE R5 AND ALLOCATE LOCAL R5 STACK
                                ;-

000022'                         PUSH R5
(2) 000022' 010546                                              MOV     R5,-(SP)
000024'                         LET R5 := #RP.R5S
(4) 000024' 012705  000014'                                     MOV     #RP.R5S,R5

                                ;+
                                ; ENQUEUE THE MODULE INTO THE CONTROL QUEUE
                                ;-

000030'                         CALL ENQCQ IN <#DTABLE,(R1),2(R1)>
(3) 000030' 010546                                              MOV     R5,-(SP)
(6) 000032' 016145  000002                                      MOV     2(R1),-(R5)
(5) 000036' 011145                                              MOV     (R1),-(R5)
(4) 000040' 012745  000000G                                     MOV     #DTABLE,-(R5)
(3) 000044' 004767  000000G                                     JSR     PC,ENQCQ
(3) 000050' 012605                                              MOV     (SP)+,R5

                                ;+
                                ; IF NO ERROR RESTORE R5 STACK
                                ;-

000052'                         IF.NO.ERROR THEN
(6) 000052' 103500                                              BCS     50000$
000054'                                 POP R5
(2) 000054' 012605                                              MOV     (SP)+,R5

                                ;+
                                ; NOW GET ADDRESS OF DTABLE AND DETERMINE IF BACKGROUND
                                ; MODULE WAS INTERRUPTED.
                                ;-

000056'                                 LET R1 := #DTABLE
(4) 000056' 012701  000000G                                     MOV     #DTABLE,R1
000062'                                 IF #BKMODE SETIN DT.ST0(R1) THEN
```

```
(6)  000062' 032761  040000  000010                                                    BIT     #BKMODE,DT.ST0(R
(9)  000070' 001464                                                                    BEQ     50001$
639
640                                 ;*************************************************************
641                                 ;+
642                                 ; TRY TO DETERMINE IF INTERRUPTED MODULE WAS TRULY A BACKGROUND
643                                 ; MODULE, THIS IS DONE BY CHECKING THE STACK TO SEE IF THE INTERRUPTED MODULE
644                                 ; HAS A PRIORITY "0".
645                                 ;-
646
647  000072'                                    IF #BIT07!BIT06!BIT05 NOTSETIN 10(SP) THEN
(6)  000072' 032766  000340  000010                                                    BIT     #BIT07!BIT06!BIT
(9)  000100' 001060                                                                    BNE     50002$
648
649                                 ;*************************************************************
650
651                                 ;+
652                                 ; IF IT WAS A BACKGROUND MODULE THEN GET IT'S HEADER ADDRESS FROM
653                                 ; "BA.HDR" WHICH CONTAINS THE HEADER ADDRESS OF CURRENT RUNNING
654                                 ; BACKGROUND MODULE.
655                                 ;-
656
657  000102'                                          PUSH R0
(2)  000102' 010046                                                                    MOV     R0,-(SP)
658  000104'                                          LET R0 := BA.HDR
(4)  000104' 016700  000000G                                                           MOV     BA.HDR,R0
659
660                                 ;+
661                                 ; BE SURE TO SAVE THE BKMOD'S COND CODES IN IT'S STAT WORD
662                                 ;-
663
664  000110'                                          LET STAT(R0) :B= 12(SP)
(4)  000110' 116660  000012  000026                                                    MOVB    12(SP),STAT(R0)
665  000116'                                          LET STAT(R0) := STAT(R0) SET.BY #BKMOD
(6)  000116' 052760  000020  000026                                                    BIS     #BKMOD,STAT(R0)
666                                 ;+
667                                 ; NOW TAKE MODULE OFF THE AIR BY CLEARING ACTBIT.
668                                 ;-
669
670  000124'                                          LET STAT(R0) := STAT(R0) CLR.BY #ACTBIT
(6)  000124' 042760  004000  000026                                                    BIC     #ACTBIT,STAT(R0)
671
672                                 ;+
673                                 ; NOW START SAVING MODULES REGISTERS
674                                 ;-
675
676  000132'                                          POP SVR0(R0)
(2)  000132' 012660  000062                                                            MOV     (SP)+,SVR0(R0)
677  000136'                                          POP SVR1(R0)
(2)  000136' 012660  000064                                                            MOV     (SP)+,SVR1(R0)
678  000142'                                          LET SVR2(R0) := R2
(4)  000142' 010260  000066                                                            MOV     R2,SVR2(R0)
679  000146'                                          LET SVR3(R0) := R3
(4)  000146' 010360  000070                                                            MOV     R3,SVR3(R0)
680  000152'                                          LET SVR4(R0) := R4
(4)  000152' 010460  000072                                                            MOV     R4,SVR4(R0)
```

```
681  000156'                                        LET SVR5(R0) := R5
(4)  000156' 010560  000074                                                 MOV     R5,SVR5(R0)
682
683                                 ;+
684                                 ; UPDATE STACK TO LOOK AT INTERRUPTED MODULE
685                                 ;-
686
687  000162'                                        LET SP := SP + #4
(6)  000162' 062706  000004                                                 ADD     #4,SP
688
689                                 ;+
690                                 ; BEFORE SAVING R6 DETERMINE RETURN ADDRESS OF BACKGROUND
691                                 ; MODULE BY LOOKING AT STACK
692                                 ;-
693
694  000166'                                        POP BA.RET
(2)  000166' 012667  000000G                                                MOV     (SP)+,BA.RET
695  000172'                                        LET SVR6(R0) := SP + #2
(4)  000172' 010660  000076                                                 MOV     SP,SVR6(R0)
(6)  000176' 062760  000002  000076                                         ADD     #2,SVR6(R0)
696
697                                 ;+
698                                 ; IT IS NOW TIME TO SET THE BACKGROUND SUSPEND BIT IN "BA.STAT
699                                 ; AND ALSO CLEAR OUT THE "BKMODE" BIT IN DT.ST0.
700                                 ;-
701
702  000204'                                        LET BA.STAT := BA.STAT SET.BY #SUSPND
(6)  000204' 052767  000001  000000G                                        BIS     #SUSPND,BA.STAT
703  000212'                                        LET DT.ST0(R1) := DT.ST0(R1) CLR.BY #BKMODE
(6)  000212' 042761  040000  000010                                         BIC     #BKMODE,DT.ST0(R
704
705                                 ;+
706                                 ; GET THE INTERRUPTED MODULE PSW INTO R0
707                                 ;-
708
709  000220'                                        LET R0 := (SP)+
(4)  000220' 012600                                                         MOV     (SP)+,R0
710
711                                 ;+
712                                 ; NOW SETUP MONITOR R5 AND R6 STACKS
713                                 ;-
714
715  000222'                                        LET R5 := DX.R5
(4)  000222' 016705  000000G                                                MOV     DX.R5,R5
716  000226'                                        LET SP := DX.SP
(4)  000226' 016706  000000G                                                MOV     DX.SP,SP
717
718                                 ;+
719                                 ; LOAD UP PSW ON NEW R6 STACK AND A RETURN ADDRESS AND DO AN RTI TO THAT ADDRESS
720                                 ;-
721
722  000232'                                        LET -(SP) := R0
(4)  000232' 010046                                                         MOV     R0,-(SP)
723  000234'                                        LET -(SP) := RP.EXT
(4)  000234' 016746  177540                                                 MOV     RP.EXT,-(SP)
724  000240'                                        INLINE <RTI>
```

```
   (2)  000240' 000002                                                                  RTI
   725  000242'                                ENDIF
   (4)  000242'                                                                50002$:
   726  000242'                        ENDIF
   (4)  000242'                                                                50001$:
   727
   728                                ;+
   729                                ; WAS NOT BACKGROUND MODULE THEN RESTORE R1 AND ADJUST THE STACK
   730                                ; AND RETURN TO INTERRUPTED MODULE.
   731                                ;-
   732
   733  000242'                               POP R1
   (2)  000242' 012601                                                          MOV     (SP)+,R1
   734  000244'                               LET SP := SP + #4
   (6)  000244' 062706  000004                                                  ADD     #4,SP
   735  000250'                               INLINE <RTI>
   (2)  000250' 000002                                                          RTI
   736
   737  000252'                        ELSE
   (4)  000252' 000421                                                          BR      50003$
   (3)  000254'                                                                50000$:
   738
   739                                ;+
   740                                ; ERROR, CONTROL QUEUE OVERFLOW DO A RESET AND SET UP R5 AND R6 STACKS.
   741                                ;-
   742
   743  000254'                               INLINE <RESET>
   (2)  000254' 000005                                                          RESET
   744  000256'                               LET R5 := DX.R5
   (4)  000256' 016705  000000G                                                MOV     DX.R5,R5
   745  000262'                               LET SP := DX.SP
   (4)  000262' 016706  000000G                                                MOV     DX.SP,SP
   746  000266'                               CALL RSTRCY IN <#DTABLE>
   (3)  000266' 010546                                                          MOV     R5,-(SP)
   (4)  000270' 012745  000000G                                                MOV     #DTABLE,-(R5)
   (3)  000274' 004767  000000G                                                JSR     PC,RSTRCY
   (3)  000300' 012605                                                          MOV     (SP)+,R5
   747
   748                                ;+
   749                                ; NOW CALL ERROR RECOVERY ROUTINE
   750                                ;-
   751
   752  000302'                               CALL ERREC IN <#DTABLE>
   (3)  000302' 010546                                                          MOV     R5,-(SP)
   (4)  000304' 012745  000000G                                                MOV     #DTABLE,-(R5)
   (3)  000310' 004767  000000G                                                JSR     PC,ERREC
   (3)  000314' 012605                                                          MOV     (SP)+,R5
   753  000316'                        ENDIF
   (4)  000316'                                                                50003$:
   754          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BA.HDR= ****** G
BA.RET= ****** G
BA.STA= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= ****** G
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.DEQ= ****** G
DX.R5 = ****** G
DX.SP = ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQCQ = ****** G
EOPBIT= 000001
ERREC = ****** G
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPIRQ   000014RG
RPTDAT= 002000
RP.EXT  000000R
RP.R5S  000014R
RSTRCY= ****** G
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
```

```
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$ELOC= 000402
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$ERFL= 000000
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $ISK2 = 000001          $$FLAG= 000001
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $LOCTA= 177777          $$FROM= 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LSTIN= 000001          $$LOC = 000100R
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $NESTL= 177777          $$REG = 177777
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK0 = 000110          $$RETU= 000000
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK1 = 000110          $$RTN1= 000000
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $NSK2 = 000110          $$RTN2= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $SAVLE= 177777          $$SRC = 000000
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $TAGNU= 050004          $$TGS1= 000000
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $TEMP = 050003          $$TGS2= 000000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TSK0 = 050003          $$TO  = 000001
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $TSK1 = 050001          $$$TAG= 050000
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $TSK2 = 050002          .     = 000316R
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $$ARGC= 000000
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $$BYTE= 000403
UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$CASE= 000000
UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$DST = 000000

. ABS.  000000     000
        000316     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RPIRQ,DSKZ:RPIRQ=SPMAC/ML,EQUATE,RPIRQ
 RUN-TIME: 14 4 .3 SECONDS
 RUN-TIME RATIO: 39/19=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE RSTRCY - RESET RECOVERY
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       RSTRCY
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       RESTORES PROCESSOR HARDWARE OPTIONS TO THE STATES THEY WERE
                                ;       IN BEFORE THE "RESET" WAS ISSUED.
                                ;
                                ; INPUTS:
                                ;       DATA TABLE ADDRESS
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.ST0
                                ;       DT.CF0
                                ;       DT.PTA
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       DT.KBRSP
                                ;
                                ; SUBORDINATE MODULES CALLED:
                                ;       KPON - PARITY ON ROUTINE
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL RSTRCY IN <A>
                                ;               A=DATA TABLE ADDRESS
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  RSTRCY                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  KONTRL                          ;BUFFER TO CACHE REGISTER
                                .GLOBL  CCNTRL                          ;CACHE REGISTER
                                .GLOBL  KPON                            ;PARITY ON ROUTINE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000002'                 RST.PTR: .WORD RST.BUF                  ;FAKE BUFFER POINTER
000002' 000015                  RST.BUF: .WORD <CR>                     ;FAKE BUFFER
```

```
                                .SBTTL RSTRCY ROUTINE


000004'                         ROUTINE RSTRCY <TABL>
(2) 000004'                                                                     RSTRCY:


                                ;+
                                ; SAVE REGISTERS
                                ;-

000004'                         PUSH R0,R1
(2) 000004' 010046                                                                      MOV     R0,-(SP)
(3) 000006' 010146                                                                      MOV     R1,-(SP)


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

000010'                         LET R0 := TABL(R5)
(4) 000010' 016500  000000                                                              MOV     TABL(R5),R0


                                ;+
                                ; IF MEMORY MANAGEMENT WAS ENABLED, RE-ENABLE IT, INCLUDING
                                ; 22-BIT ADDRESSING IF IT EXISTS.
                                ;-

000014'                         IF #KTSTAT SETIN DT.ST0(R0) THEN
(6) 000014' 032760  000020  000010                                                     BIT     #KTSTAT,DT.ST0(R
(9) 000022' 001412                                                                      BEQ     50002$
000024'                                 LET @#SR0 := @#SR0 SET.BY #001001
(6) 000024' 052737  001001  177572                                                      BIS     #001001,@#SR0
000032'                                 IF #ADDR22 SETIN DT.CF0(R0) THEN
(6) 000032' 032760  001000  000014                                                     BIT     #ADDR22,DT.CF0(R
(9) 000040' 001403                                                                      BEQ     50003$
000042'                                         LET @#SR3 := @#SR3 SET.BY #BIT04
(6) 000042' 052737  000020  172516                                                      BIS     #BIT04,@#SR3
000050'                                 ENDIF
(4) 000050'                                                                     50003$:
000050'                         ENDIF
(4) 000050'                                                                     50002$:


                                ;+
                                ; IF THE UNIBUS MAP HAD BEEN ENABLED, RE-ENABLE IT.
                                ;-


000050'                         IF #MAPSTAT SETIN DT.ST0(R0) THEN
(6) 000050' 032760  000200  000010                                                     BIT     #MAPSTAT,DT.ST0(
(9) 000056' 001403                                                                      BEQ     50004$
000060'                                 LET @#SR3 := @#SR3 SET.BY #BIT05
(6) 000060' 052737  000040  172516                                                      BIS     #BIT05,@#SR3
000066'                         ENDIF
```

```
(4)  000066'                                                                          50004$:
623
624
625                                 ;+
626                                 ; IF PARITY OR ECC HAD BEEN ENABLED, RE-LOAD THE PARITY REGISTERS.
627                                 ; BY CALLING KPON ROUTINE AND PASSING THE ADDRESS
628                                 ; OF FAKE BUFFER. THEN CLEAR OUT KEYBOARD RESPONSE LOACATION IN
629                                 ; DTABLE TO ELIMINATE OUTPUTTING MESSAGE.
630                                 ;-
631
632  000066'                        IF #PARSTAT SETIN DT.ST0(R0) OR #ECCSTAT SETIN DT.ST0(R0) THEN
(6)  000066' 032760  000100  000010                                                  BIT     #PARSTAT,DT.ST0(
(8)  000074' 001004                                                                  BNE     50005$
(6)  000076' 032760  000010  000010                                                  BIT     #ECCSTAT,DT.ST0(
(9)  000104' 001411                                                                  BEQ     50006$
(6)  000106'                                                                          50005$:
633  000106'                            CALL KPON IN <R0,#RST.PTR>
(3)  000106' 010546                                                                  MOV     R5,-(SP)
(5)  000110' 012745  000000'                                                         MOV     #RST.PTR,-(R5)
(4)  000114' 010045                                                                  MOV     R0,-(R5)
(3)  000116' 004767  000000G                                                         JSR     PC,KPON
(3)  000122' 012605                                                                  MOV     (SP)+,R5
634  000124'                            LET DT.KBRSP(R0) := #0
(4)  000124' 005060  000022                                                          CLR     DT.KBRSP(R0)
635  000130'                        ENDIF
(4)  000130'                                                                          50006$:
636
637
638                                 ;+
639                                 ; SEE IF CACHE MEMORY IS PRESENT.  IF IT IS, AND IF IT HAD BEEN
640                                 ; DISABLED, THEN RE-DISABLE IT.  ALSO, IF CACHE IS PRESENT AND IF
641                                 ; PARITY HAD BEEN DISABLED, THEN CACHE PARITY MUST BE RE-DISABLED.
642                                 ;-
643
644  000130'                        IF #CAPRES SETIN DT.CF0(R0) THEN
(6)  000130' 032760  000004  000014                                                  BIT     #CAPRES,DT.CF0(R
(9)  000136' 001424                                                                  BEQ     50007$
645  000140'                            IF #CASTAT NOTSETIN DT.ST0(R0) THEN
(6)  000140' 032760  000004  000010                                                  BIT     #CASTAT,DT.ST0(R
(9)  000146' 001006                                                                  BNE     50010$
646  000150'                                    LET KONTRL := KONTRL SET.BY #14
(6)  000150' 052767  000014  000000G                                                 BIS     #14,KONTRL
647  000156'                                    LET @CCNTRL := KONTRL
(4)  000156' 016777  000000G 000000G                                                 MOV     KONTRL,@CCNTRL
648  000164'                            ENDIF
(4)  000164'                                                                          50010$:
649  000164'                            IF #PARSTAT NOTSETIN DT.ST0(R0) THEN
(6)  000164' 032760  000100  000010                                                  BIT     #PARSTAT,DT.ST0(
(9)  000172' 001006                                                                  BNE     50011$
650  000174'                                    LET KONTRL := KONTRL SET.BY #203
(6)  000174' 052767  000203  000000G                                                 BIS     #203,KONTRL
651  000202'                                    LET @CCNTRL := KONTRL
(4)  000202' 016777  000000G 000000G                                                 MOV     KONTRL,@CCNTRL
652  000210'                            ENDIF
(4)  000210'                                                                          50011$:
653  000210'                        ENDIF
```

```
   (4)  000210'                                                                  50007$:
   654
   655
   656                              ;+
   657                              ; RESTORE REGISTERS AND RETURN.
   658                              ;-
   659
   660  000210'                     POP R1,R0
   (2)  000210' 012601                                                                MOV     (SP)+,R1
   (3)  000212' 012600                                                                MOV     (SP)+,R0
   661
   662  000214'                     ENDRTN
   (3)  000214'                                                                  50000$:
   (3)  000214'                                                                  50001$:
   (2)  000214' 000207                                                                RTS     PC
   663          000001              .END
```

```
ACSR   = 000102          CSRC   = 000102          EOPBIT= 000001          MODEXH= 004000          RBUFSZ= 000132
ACTBIT= 004000          CTRLC  = 000003          ERRTYP= 000106          MODHOL= 002000          RBUFVA= 000124
ADDR22= 001000          CTRLO  = 000017          EVNTBE= 000200          MODSEL= 001000          RDSERV= 000101
ADR    = 000006          CTRLU  = 000025          EVNTHD= 000200          MSGCKD= 000010          RDWHMI= 000022
APTFER= 000004          DCEVNT= 000011          EVNTKT= 000203          MSGCKS= 000011          RELERR= 000020
APTPRE= 000200          DEFRTN= 000400          EVNTPE= 000202          MSGDER= 000005          RELMOD= 020000
ASB    = 000106          DIAGMC= 000000          EVNTRE= 000201          MSGDRP= 000017          RELTIM= 010000
ASSEMB= 000010          DROPMO= 100000          FATERR= 100000          MSGECH= 177777          RES1   = 000056
ASTAT  = 000104          DSEVNT= 000014          HRDCNT= 000044          MSGEOP= 000013          RES2   = 000060
AUTO   = 000010          DT.ADD= 000042          HRDPAS= 000050          MSGHDR= 000004          RICHAR= 031060
AUTOST= 020000          DT.AP  = 000100          ICONT  = 000036          MSGHNG= 000022          RPTDAT= 002000
AWAS   = 000110          DT.APK= 000076          ICOUNT= 000040          MSGHRD= 000007          RSTRCY  000004RG
BIT0   = 000001          DT.BLS= 000034          IDNUM  = 000122          MSGMAP= 000021          RSTRT  = 000112
BIT00  = 000001          DT.CF0= 000014          IE     = 000100          MSGNUL= 177775          RST.BU  000002R
BIT01  = 000002          DT.CF1= 000016          INDPAR= 000040          MSGPOP= 000002          RST.PT  000000R
BIT02  = 000004          DT.ERR= 000020          INHDRP= 040000          MSGPRM= 177776          RUBOUT= 000177
BIT03  = 000010          DT.ESI= 000044          INHEPR= 020000          MSGRES= 000001          RUNMOD= 100000
BIT04  = 000020          DT.EVN= 000000          INHREL= 001000          MSGSFT= 000006          R5VALU= 001740
BIT05  = 000040          DT.EXS= 000060          INHRRE= 000400          MSGSKE= 000003          SAM    = 075464
BIT06  = 000100          DT.FCH= 000037          INIT   = 000030          MSGSMB= 000015          SBADR  = 000102
BIT07  = 000200          DT.FCN= 000036          INTR   = 000120          MSGSMH= 000014          SBKMOD= 000000
BIT08  = 000400          DT.HMX= 000104          IOMOD  = 100000          MSGSMS= 000016          SBKSEL= 010000
BIT09  = 001000          DT.KBE= 000024          IOMODP= 102000          MSGSTD= 000000          SC.ADR= 000006
BIT1   = 000002          DT.KBP= 000026          IOMODR= 112000          MSGSYS= 000012          SC.ALC= 000014
BIT10  = 002000          DT.KBR= 000022          IOMODX= 110000          MSGVEC= 000020          SC.APC= 000016
BIT11  = 004000          DT.KBU= 000030          JACK   = 035060          NBKMOD= 001000          SC.CKL= 000002
BIT12  = 010000          DT.MLS= 000032          KIPAR0= 172340          NCPUOP= 000020          SC.CKP= 000004
BIT13  = 020000          DT.MTI= 000110          KIPAR1= 172342          NOAPTY= 000002          SC.CLO= 000000
BIT14  = 040000          DT.OFF= 000070          KIPAR2= 172344          NULL   = 000000          SC.HLD= 000010
BIT15  = 100000          DT.PAS= 000074          KIPAR3= 172346          OWEN   = 024020          SC.SCA= 000012
BIT2   = 000004          DT.PC  = 000002          KIPAR4= 172350          PAERR  = 000010          SENDLS= 177777
BIT3   = 000010          DT.PFL= 000062          KIPAR5= 172352          PARPRE= 002000          SOFCNT= 000042
BIT4   = 000020          DT.PSW= 000004          KIPAR6= 172354          PARSTA= 000100          SOFPAS= 000046
BIT5   = 000040          DT.PTA= 000064          KIPAR7= 172356          PASCNT= 000034          SPACE  = 000040
BIT6   = 000100          DT.RCS= 000102          KIPDR0= 172300          PDPLSI= 020000          SPOINT= 000032
BIT7   = 000200          DT.REL= 000040          KIPDR1= 172302          PDP60  = 004000          SPVALU= 002200
BIT8   = 000400          DT.SCT= 000066          KIPDR2= 172304          PDP70  = 010000          SR0    = 177572
BIT9   = 001000          DT.SMX= 000106          KIPDR3= 172306          PRI0   = 000000          SR1    = 177574
BKDEF  = 000002          DT.SP  = 000006          KIPDR4= 172310          PRI1   = 000040          SR2    = 177576
BKMOD  = 000020          DT.SSI= 000046          KIPDR5= 172312          PRI4   = 000200          SR3    = 172516
BKMODE= 040000          DT.ST0= 000010          KIPDR6= 172314          PRI5   = 000240          STAT   = 000026
BKSLSH= 000134          DT.ST1= 000012          KIPDR7= 172316          PRI6   = 000300          STATBI= 064757
CAPRES= 000004          DT.SWR= 000056          KONTRL= ****** G        PRI7   = 000340          STAT1  = 000027
CASTAT= 000004          DT.SYP= 000072          KPON   = ****** G        PR0    = 000000          SUSPND= 000001
CCNTRL= ****** G        DT.WBU= 000050          KTERRO= 000040          PR4    = 000200          SVR0   = 000062
CDERCT= 000146          DT.WHL= 000054          KTPRES= 000400          PR5    = 000240          SVR1   = 000064
CDWDCT= 000144          DT.WLL= 000052          KTSTAT= 000020          PR6    = 000300          SVR2   = 000066
CKTIM  = 100000          DVID1  = 000014          KTXTND= 040000          PR7    = 000340          SVR3   = 000070
CLKPRE= 000001          ECCMEM= 000100          LF     = 000012          PS     = 177776          SVR4   = 000072
CONFIG= 000056          ECCSTA= 000010          LPSTAT= 000001          PSW    = 177776          SVR5   = 000074
CQOVF  = 000001          ENBEOP= 010000          MAPSTA= 000200          RANNUM= 000054          SVR6   = 000076
CR     = 000015          ENBNUL= 000001          MED    = 076600          RBUFEA= 000130          SYSCNT= 000052
CSRA   = 000100          ENDLST= 000000          MEMPAS= 040000          RBUFPA= 000126          SYSERR= 000100
```

TABLE OF CONTENTS

```
                        .TITLE  ACTIV - ACTIVATE NEXT OPTION MODULE
                        .IDENT  /V0.0/


                        ;++
                        ; MODULE NAME:
                        ;       ACTIV
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE FETCHES FROM THE MODULE QUEUE LIST SUPPLIED BY THE
                        ;       CONFIGURATOR LINKER PROGRAM THE ADDRESS OF THE NEXT OPTION MODULE
                        ;       TO BE ACTIVATED, AND ACTIVATES IT BY  MAKING AN ENTRY FOR IT IN
                        ;       THE CONTROL QUEUE
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.ST0
                        ;       2. DT.ST1
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.ST0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ENQCQ                MAKE ENTRY IN CONTROL QUEUE
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       1. THIS ROUTINE IS RESPONSIBLE FOR SETTING THE FOLLOWING BITS IN THE
                        ;          SYSTEM'S STATUS INDICATOR WORD:
                        ;               MODHOLD         SET TO INDICATE THAT A SBK OR NBK MODULE
                        ;                               IS PRESENTLY RUNNING
                        ;               MODEXH          SET TO INDICATE THAT ALL OF THE EXERCISER'S
                        ;                               OPTION MODULES HAVE BEEN ACTIVATED
                        ;               TMPIO           SET IF BKMOD IS DOING A ONE
                        ;                               ONE ITERATION PASS, BKMOD WILL TAKE
                        ;                               AN APPEARANCE OF AN IOMOD.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ACTIV IN <DTABLE>
                        ;
                        ;               DTABLE = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

564

```
                        .SBTTL  COMMON DEFINITIONS AND REFERENCES

                                .MCALL  STRUCT
                                STRUCT
(1)                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN = 1
        000001                  $LSTTAG = 1


                        ;
                        ;****************
                        ;
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  ACTIV                   ;MODULE'S ENTRY POINT
                        .GLOBL  AC.TPTR
                        .GLOBL  AC.TYPE
                        .GLOBL  AC.MPTR
                        .GLOBL  AC.MODQ
                        .GLOBL  MODQ
                        .GLOBL  STATBITS
                        .GLOBL  SENDLST
                        ;
                        ;****************
                        ;
                        .GLOBL  RESREG                  ;RESTORE REGISTERS ROUTINE
                        .GLOBL  SAVREG                  ;SAVE REGISTERS ROUTINE
                        .GLOBL  ENQCQ                   ;CONTROL QUEUE EN-QUEUE MODULE
                        ;
                        ;****************
                        ;
                        ;       LOCAL EQUATES
                        ;
        001777          NON32K  = 001777                ;NON-32K BOUNDARY
        060757          STATBITS = 060757               ;MODULE'S STATUS WORD
        177777          SENDLST  = 177777               ;SPECIAL END OF LIST INDICATOR
                        ;
                        ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
                        ;
000000' 000000          AC.MPTR: .WORD  0               ;MODULE-QUEUE-LIST POINTER
000002'                 AC.MODQ:
000002'                 MODQ:
        000050                  .REPT   ^D<40>
                                .NLIST
                                .WORD   0
                                .LIST
                                .ENDR
000122' 000000                  .WORD   0               ;END OF LIST INDICATOR

000124' 000000          AC.TPTR: .WORD  SBKMOD          ;MODULE-TYPE-LIST POINTER (ASSEMBLED TO SBKMOD)
000126' 000000          AC.TYPE: .WORD  SBKMOD          ;SPECIAL MODULE TYPE
000130' 001000                  .WORD   NBKMOD          ;NON-BACKGROUND
000132' 000020                  .WORD   BKMOD           ;BACKGROUND
000134' 100000                  .WORD   IOMOD           ;I/O MODULE - INTERRUPT DRIVEN
```

```
621  000136' 110000                  .WORD   IOMODX          ;I/O MODULE - NPR DRIVEN
622  000140' 102000                  .WORD   IOMODP          ;I/O MODULE - RUNS ONLY ON 32K BOUNDARIES
623  000142' 112000                  .WORD   IOMODR          ;I/O MODULE - RUNS ONLY AT LOWEST MEMORY BANK
624  000144' 177777                  .WORD   SENDLST         ;SPECIAL END-OF-LIST INDICATOR
625                                  ;
626                                  ;****************
627                                  ;
628
```

```
                                .SBTTL  ACTIVATE ROUTINE
                                
                                        ROUTINE ACTIV <DTABLE>
000146'                                                                 ACTIV:
                                
                                ;+
                                ; SAVE REGISTERS AND PUT THE DATA TABLE ADDRESS IN R0.
                                ;-
                                
000146'                                 CALL SAVREG
000146' 004767  000000G                                                 JSR     PC,SAVREG
000152'                                 LET R0 := DTABLE(R5)
000152' 016500  000000                                                  MOV     DTABLE(R5),R0
                                ;+
                                ; IF WE'VE BEEN THROUGH THE TYPE LIST ALREADY, RESET THE POINTERS TO
                                ; IOMOD AND TO START OF MODQ LIST, THEN CLEAR THE MODSEL AND MODEXH
                                ; BITS INSTATUS WORD 0.
                                ;-
                                
000156'                                 IF #MODEXH SETIN DT.ST0(R0) THEN
000156' 032760  004000  000010                                          BIT     #MODEXH,DT.ST0(R
000164' 001414                                                          BEQ     50002$
000166'                                         LET AC.TPTR := #AC.TYPE + #6
000166' 012767  000126' 177730                                          MOV     #AC.TYPE,AC.TPTR
000174' 062767  000006  177722                                          ADD     #6,AC.TPTR
000202'                                         LET AC.MPTR := #AC.MODQ
000202' 012767  000002' 177570                                          MOV     #AC.MODQ,AC.MPTR
000210'                                         LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #MODSEL!MODEXH
000210' 042760  005000  000010                                          BIC     #MODSEL!MODEXH,D
000216'                                 ENDIF
000216'                                                         50002$:
                                
                                ;+
                                ; SEARCH UNTIL THE MODULE-TYPE-LIST IS EXHAUSTED
                                ;-
                                
000216'                                 WHILE @AC.TPTR NE #SENDLST DO
000216'                                                         50003$:
000216' 027727  177702  177777                                          CMP     @AC.TPTR,#SENDLS
000224' 001563                                                          BEQ     50004$
                                
                                ;+
                                ; SAVE THE POINTER IN R3.
                                ;-
                                
000226'                                     LET R3 := @AC.TPTR
000226' 017703  177672                                                  MOV     @AC.TPTR,R3
                                
                                ;+
                                ; SEARCH THE MODULE-QUEUE-LIST FOR A MODULE OF THE CURRENT TYPE
                                ;-
                                
000232'                                     WHILE @AC.MPTR NE #0 DO
000232'                                                         50005$:
000232' 005777  177542                                                  TST     @AC.MPTR
000236' 001515                                                          BEQ     50006$
```

```
 369  000240'                                LET R1 := @AC.MPTR
(4)   000240' 017701  177534                                                    MOV     @AC.MPTR,R1
 370  000244'                                LET R4 := R1
(4)   000244' 010104                                                            MOV     R1,R4
 371
 372                                ;+
 373                                ; CLEAR OUT ALL BUT THE MODULE TYPE BITS IN THE MODULE'S STATUS WORD.
 374                                ;-
 375
 376  000246'                                LET R1 := STAT(R1)
(4)   000246' 016101  000026                                                    MOV     STAT(R1),R1
 377  000252'                                LET R2 := R1 CLR.BY #STATBITS
(4)   000252' 010102                                                            MOV     R1,R2
(6)   000254' 042702  060757                                                    BIC     #STATBITS,R2
 378
 379                                ;+
 380                                ; IF ONE IS FOUND MAKE SURE IT IS SELECTED AND THAT IT HAS NOT
 381                                ; BEEN DROPPED AND IS NOT ACTIVE.
 382                                ;-
 383
 384  000260'                                IF R2 EQ R3 THEN
(6)   000260' 020203                                                            CMP     R2,R3
(9)   000262' 001077                                                            BNE     50007$
 385  000264'                                    IF #BIT14 SETIN R1 AND #BIT13!ACTBIT NOTSETIN R1 THEN
(6)   000264' 032701  040000                                                    BIT     #BIT14,R1
(9)   000270' 001474                                                            BEQ     50010$
(6)   000272' 032701  024000                                                    BIT     #BIT13!ACTBIT,R1
(9)   000276' 001071                                                            BNE     50010$
 386
 387                                ;+
 388                                ; SAVE THE MODULE LIST POINTER.
 389                                ;-
 390
 391  000300'                                    LET R1 := @AC.MPTR
(4)   000300' 017701  177474                                                    MOV     @AC.MPTR,R1
 392
 393
 394                                ;+
 395                                ; IF THE OPTION MODULE'S PASS COUNT IS NOT 0, THEN SAVE ITS RESTART ADDRESS.
 396                                ; OTHERWISE, SAVE THE START ADDRESS AS THIS IS THE FIRST PASS.
 397                                ;-
 398
 399  000304'                                    IF PASCNT(R4) NE #0 THEN
(6)   000304' 005764  000034                                                    TST     PASCNT(R4)
(9)   000310' 001403                                                            BEQ     50011$
 400  000312'                                        LET R4 := RSTRT(R1)
(4)   000312' 016104  000112                                                    MOV     RSTRT(R1),R4
 401  000316'                                    ELSE
(4)   000316' 000402                                                            BR      50012$
(3)   000320'                                                           50011$:
 402  000320'                                        LET R4 := INIT(R1)
(4)   000320' 016104  000030                                                    MOV     INIT(R1),R4
 403  000324'                                    ENDIF
(4)   000324'                                                           50012$:
 404
 405                                ;+
```

```
                        ; BEFORE WE EN-QUEUE THE MODULE MAKE SURE ITS STACK WILL BE SETUP PROPERLY
                        ; THEN EN-QUEUE THE MODULE IN THE CONTROL QUEUE
                        ;-

000324'                                 LET SVR6(R1) := SPOINT(R1)
(4) 000324' 016161 000032 000076                                    MOV     SPOINT(R1),SVR6(
000332'                                 CALL ENQCQ IN <R0,R1,R4>
(3) 000332' 010546                                                  MOV     R5,-(SP)
(6) 000334' 010445                                                  MOV     R4,-(R5)
(5) 000336' 010145                                                  MOV     R1,-(R5)
(4) 000340' 010045                                                  MOV     R0,-(R5)
(3) 000342' 004767 000000G                                          JSR     PC,ENQCQ
(3) 000346' 012605                                                  MOV     (SP)+,R5

                        ;+
                        ; MARK THE OPTION MODULE AS BEING ACTIVE
                        ;-

000350'                                 LET STAT(R1) := STAT(R1) SET.BY #ACTBIT
(6) 000350' 052761 004000 000026                                    BIS     #ACTBIT,STAT(R1)

                        ;+
                        ; IF THIS MODULE IS OF THE SBK OR NBK TYPE SET THE MODULE-HOLD-FLAG
                        ; WHICH  WILL ALLOW THE MODULE TO RUN BY ITSELF
                        ;-

000356'                                 IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD OR R2 EQ #BKMOD THEN
(6) 000356' 020227 000000                                           CMP     R2,#SBKMOD
(8) 000362' 001406                                                  BEQ     50013$
(6) 000364' 020227 001000                                           CMP     R2,#NBKMOD
(8) 000370' 001403                                                  BEQ     50013$
(6) 000372' 020227 000020                                           CMP     R2,#BKMOD
(9) 000376' 001003                                                  BNE     50014$
(6) 000400'                                                     50013$:
000400'                                     LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODHOLD
(6) 000400' 052760 002000 000010                                    BIS     #MODHOLD,DT.ST0(
000406'                                 ENDIF
(4) 000406'                                                     50014$:
000406'                                 IF R2 EQ #SBKMOD THEN
(6) 000406' 020227 000000                                           CMP     R2,#SBKMOD
(9) 000412' 001003                                                  BNE     50015$
000414'                                         LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
(6) 000414' 052760 010000 000012                                    BIS     #SBKSEL,DT.ST1(R
000422'                                 ENDIF
(4) 000422'                                                     50015$:

                        ;+
                        ; IF MODULE IS A BKMOD THEN IT IS NECESSARY TO DO A ONE ITERATION
                        ; PASS. THIS IS ACCOMPLISHED BY SETTING THE TMPIO BIT AND CHANGING THE
                        ; BKMOD STAT TO THAT OF AN IOMOD. PRENDIT WILL CHANGE THE MODE
                        ; BACK AT END-OF-PASS TIME.
                        ;-

000422'                                 IF R2 EQ #BKMOD THEN
(6) 000422' 020227 000020                                           CMP     R2,#BKMOD
(9) 000426' 001011                                                  BNE     50016$
```

```
739  000430'                                                    LET DT.ST0(R0) := DT.ST0(R0) SET.BY #TMPIO
(6)  000430' 052760  000002  000010                                                          BIS     #TMPIO,DT.ST0(R0
740  000436'                                                    LET STAT(R1) := STAT(R1) CLR.BY #BKMOD
(6)  000436' 042761  000020  000026                                                          BIC     #BKMOD,STAT(R1)
741  000444'                                                    LET STAT(R1) := STAT(R1) SET.BY #IOMOD
(6)  000444' 052761  100000  000026                                                          BIS     #IOMOD,STAT(R1)
742  000452'                                                ENDIF
(4)  000452'                                                                          50016$:
743
744                                                     ;+
745                                                     ; ADVANCE THE MODULE-QUEUE-LIST POINTER, THEN RESTORE THE
746                                                     ; REGISTERS AND DO A RETURN
747                                                     ;-
748
749  000452'                                                    LET AC.MPTR := AC.MPTR + #2
(6)  000452' 062767  000002  177320                                                          ADD     #2,AC.MPTR
750  000460'                                                    INLINE <BR      1$>
(2)  000460' 000450                                                                          BR      1$
751  000462'                                               ENDIF
(4)  000462'                                                                          50010$:
752
753                                                     ;+
754                                                     ; THE MODULE IS NOT OF THE CURRENT TYPE OR IF IT IS, IT IS
755                                                     ; PRESENTLY DESELECTED OR DROPPED, SO UPDATE THE MODULE-QUEUE-LIST
756                                                     ; POINTER AND KEEP SEARCHING
757                                                     ;-
758
759  000462'                                            ENDIF
(4)  000462'                                                                          50007$:
760  000462'                                            LET AC.MPTR := AC.MPTR + #2
(6)  000462' 062767  000002  177310                                                          ADD     #2,AC.MPTR
761  000470'                                         ENDDO
(4)  000470' 000660                                                                          BR      50005$
(3)  000472'                                                                          50006$:
```

```
                                .SBTTL  SET UP FOR THE NEXT PASS THRU MODULE LIST


                                ;+
                                ; THE MODULE-QUEUE-LIST HAS BEEN EXHAUSTED SO SET THE POINTER BACK TO THE
                                ; BEGINNING OF IT AND UPDATE THE MODULE-TYPE-LIST POINTER
                                ;-

000472'                                 LET AC.MPTR := #AC.MODQ
(4) 000472' 012767 000002' 177300                                               MOV     #AC.MODQ,AC.MPTR
000500'                                 LET AC.TPTR := AC.TPTR + #2
(6) 000500' 062767 000002 177416                                                ADD     #2,AC.TPTR

                                ;+
                                ; IF THE NEW CURRENT TYPE IS THAT OF IOMODP MAKE SURE THE IOMODP
                                ; MODULES CAN RUN AT THIS ADDRESS
                                ;-

000506'                                 IF @AC.TPTR EQ #IOMODP AND DT.ADDR(R0) NE #200 AND #NON32K SETIN DT.ADDR(R0)
(6) 000506' 027727 177412 102000                                                CMP     @AC.TPTR,#IOMODP
(9) 000514' 001013                                                              BNE     50017$
(6) 000516' 026027 000042 000200                                                CMP     DT.ADDR(R0),#200
(9) 000524' 001407                                                              BEQ     50017$
(6) 000526' 032760 001777 000042                                                BIT     #NON32K,DT.ADDR(
(9) 000534' 001403                                                              BEQ     50017$
000536'                                     LET AC.TPTR := AC.TPTR + #2
(6) 000536' 062767 000002 177360                                                ADD     #2,AC.TPTR
000544'                                 ENDIF
(4) 000544'                                                             50017$:

                                ;+
                                ; IF ITS AN IOMODR  TYPE MAKE SURE IT CAN RUN AT THIS ADDRESS
                                ;-

000544'                                 IF @AC.TPTR EQ #IOMODR AND DT.ADDR(R0) NE #200 THEN
(6) 000544' 027727 177354 112000                                                CMP     @AC.TPTR,#IOMODR
(9) 000552' 001007                                                              BNE     50020$
(6) 000554' 026027 000042 000200                                                CMP     DT.ADDR(R0),#200
(9) 000562' 001403                                                              BEQ     50020$
000564'                                     LET AC.TPTR := AC.TPTR + #2
(6) 000564' 062767 000002 177332                                                ADD     #2,AC.TPTR
000572'                                 ENDIF
(4) 000572'                                                             50020$:
000572'                         ENDDO
(4) 000572' 000611                                                              BR      50003$
(3) 000574'                                                             50004$:

                                ;+
                                ; THE MODULE-TYPE-LIST HAS BEEN EXHAUSTED, SO SET THE MODULE-QUEUE-LIST
                                ; EXHAUSTED INDICATOR WHICH SAYS THAT ALL OF THE OPTION MODULES HAVE
                                ; BEEN ACTIVATED
                                ;-

000574'                                 LET DT.STO(R0) := DT.STO(R0) SET.BY #MODEXH
(6) 000574' 052760 004000 000010                                                BIS     #MODEXH,DT.STO(R
```

```
                                ;+
                                ; RETURN TO CALLER
                                ;-

                                INLINE <1$:>
000602'                                                 1$:
(2)  000602'
                                        CALL RESREG
000602'                                                 JSR     PC,RESREG
(3)  000602' 004767  000000G
                                        ENDRTN
000606'                                         50000$:
(3)  000606'                                    50001$:
(3)  000606'                                            RTS     PC
(2)  000606' 000207
                                .END
000001
```

```
ACSR   = 000102        CONFIG= 000056        ECCMEM= 000100        LPSTAT= 000001        PR7    = 000340
ACTBIT= 004000        CQOVF  = 000001        ECCSTA= 000010        MAPSTA= 000200        PS     = 177776
ACTIV   000146RG      CR     = 000015        ENBEOP= 010000        MED    = 076600        PSW    = 177776
AC.MOD  000002RG      CSRA   = 000100        ENBNUL= 000001        MEMPAS= 040000        RANNUM= 000054
AC.MPT  000000RG      CSRC   = 000102        ENDLST= 000000        MODEXH= 004000        RBUFEA= 000130
AC.TPT  000124RG      CTRLC  = 000003        ENQCQ  = ****** G     MODHOL= 002000        RBUFPA= 000126
AC.TYP  000126RG      CTRLO  = 000017        EOPBIT= 000001        MODQ    000002RG      RBUFSZ= 000132
ADDR22= 001000        CTRLU  = 000025        ERRTYP= 000106        MODSEL= 001000        RBUFVA= 000124
ADR    = 000006       DCEVNT= 000011        EVNTBE= 000200        MSGCKD= 000010        RDSERV= 000101
APTFER= 000004        DEFRTN= 000400        EVNTHD= 000200        MSGCKS= 000011        RDWHMI= 000022
APTPRE= 000200        DIAGMC= 000000        EVNTKT= 000203        MSGDER= 000005        RELERR= 000020
ASB    = 000106       DROPMO= 100000        EVNTPE= 000202        MSGDRP= 000017        RELMOD= 020000
ASSEMB= 000010        DSEVNT= 000014        EVNTRE= 000201        MSGECH= 177777        RELTIM= 010000
ASTAT  = 000104       DTABLE= 000000        FATERR= 100000        MSGEOP= 000013        RESREG= ****** G
AUTO   = 000010       DT.ADD= 000042        HRDCNT= 000044        MSGHDR= 000004        RES1   = 000056
AUTOST= 020000        DT.AP  = 000100        HRDPAS= 000050        MSGHNG= 000022        RES2   = 000060
AWAS   = 000110       DT.APK= 000076        ICONT  = 000036        MSGHRD= 000007        RICHAR= 031060
BIT0   = 000001       DT.BLS= 000034        ICOUNT= 000040        MSGMAP= 000021        RPTDAT= 002000
BIT00  = 000001       DT.CF0= 000014        IDNUM  = 000122        MSGNUL= 177775        RSTRT  = 000112
BIT01  = 000002       DT.CF1= 000016        IE     = 000100        MSGPOP= 000002        RUBOUT= 000177
BIT02  = 000004       DT.ERR= 000020        INDPAR= 000040        MSGPRM= 177776        RUNMOD= 100000
BIT03  = 000010       DT.ESI= 000044        INHDRP= 040000        MSGRES= 000001        R5VALU= 001740
BIT04  = 000020       DT.EVN= 000000        INHEPR= 020000        MSGSFT= 000006        SAM    = 075464
BIT05  = 000040       DT.EXS= 000060        INHREL= 001000        MSGSKE= 000003        SAVREG= ****** G
BIT06  = 000100       DT.FCH= 000037        INHRRE= 000400        MSGSMB= 000015        SBADR  = 000102
BIT07  = 000200       DT.FCN= 000036        INIT   = 000030        MSGSMH= 000014        SBKMOD= 000000
BIT08  = 000400       DT.HMX= 000104        INTR   = 000120        MSGSMS= 000016        SBKSEL= 010000
BIT09  = 001000       DT.KBE= 000024        IOMOD  = 100000        MSGSTD= 000000        SC.ADR= 000006
BIT1   = 000002       DT.KBP= 000026        IOMODP= 102000        MSGSYS= 000012        SC.ALC= 000014
BIT10  = 002000       DT.KBR= 000022        IOMODR= 112000        MSGVEC= 000020        SC.APC= 000016
BIT11  = 004000       DT.KBU= 000030        IOMODX= 110000        NBKMOD= 001000        SC.CKL= 000002
BIT12  = 010000       DT.MLS= 000032        JACK   = 035060        NCPUOP= 000020        SC.CKP= 000004
BIT13  = 020000       DT.MTI= 000110        KIPAR0= 172340        NOAPTY= 000002        SC.CLO= 000000
BIT14  = 040000       DT.OFF= 000070        KIPAR1= 172342        NON32K= 001777        SC.HLD= 000010
BIT15  = 100000       DT.PAS= 000074        KIPAR2= 172344        NULL   = 000000        SC.SCA= 000012
BIT2   = 000004       DT.PC  = 000002        KIPAR3= 172346        OWEN   = 024020        SENDLS= 177777 G
BIT3   = 000010       DT.PFL= 000062        KIPAR4= 172350        PAERR  = 000010        SOFCNT= 000042
BIT4   = 000020       DT.PSW= 000004        KIPAR5= 172352        PARPRE= 002000        SOFPAS= 000046
BIT5   = 000040       DT.PTA= 000064        KIPAR6= 172354        PARSTA= 000100        SPACE  = 000040
BIT6   = 000100       DT.RCS= 000102        KIPAR7= 172356        PASCNT= 000034        SPOINT= 000032
BIT7   = 000200       DT.REL= 000040        KIPDR0= 172300        PDPLSI= 020000        SPVALU= 002200
BIT8   = 000400       DT.SCT= 000066        KIPDR1= 172302        PDP60  = 004000        SR0    = 177572
BIT9   = 001000       DT.SMX= 000106        KIPDR2= 172304        PDP70  = 010000        SR1    = 177574
BKDEF  = 000002       DT.SP  = 000006        KIPDR3= 172306        PRI0   = 000000        SR2    = 177576
BKMOD  = 000020       DT.SSI= 000046        KIPDR4= 172310        PRI1   = 000040        SR3    = 172516
BKMODE= 040000        DT.ST0= 000010        KIPDR5= 172312        PRI4   = 000200        STAT   = 000026
BKSLSH= 000134        DT.ST1= 000012        KIPDR6= 172314        PRI5   = 000240        STATBI= 060757 G
CAPRES= 000004        DT.SWR= 000056        KIPDR7= 172316        PRI6   = 000300        STAT1  = 000027
CASTAT= 000004        DT.SYP= 000072        KTERR0= 000040        PRI7   = 000340        SUSPND= 000001
CDERCT= 000146        DT.WBU= 000050        KTPRES= 000400        PR0    = 000000        SVR0   = 000062
CDWDCT= 000144        DT.WHL= 000054        KTSTAT= 000020        PR4    = 000200        SVR1   = 000064
CKTIM  = 100000       DT.WLL= 000052        KTXTND= 040000        PR5    = 000240        SVR2   = 000066
CLKPRE= 000001        DVID1  = 000014        LF     = 000012        PR6    = 000300        SVR3   = 000070
```

```
SVR4   = 000072          UIPDR7= 177616          $F$FAL= 000405          $LSTTA= 000001          $$CASE= 000000
SVR5   = 000074          WASADR= 000104          $F$GOO= 000400          $NESTL= 177777          $$DST = 000000
SVR6   = 000076          WBSTAT= 000040          $F$IF = 000110          $NSK0 = 000300          $$ELOC= 000402
SYSCNT= 000052           WBUFEA= 000136          $F$INC= 000210          $NSK1 = 000120          $$ERFL= 000000
SYSERR= 000100           WBUFPA= 000134          $F$LOO= 000200          $NSK2 = 000110          $$FLAG= 000001
TMPIO  = 000002          WBUFRQ= 000140          $F$NAM= 000160          $NSK3 = 000110          $$FROM= 000000
TQOVF  = 000002          WBUFSZ= 000142          $F$NO  = 000403         $NSK4 = 000110          $$LOC = 000562R
UIPAR0= 177640           WDFR   = 000116         $F$OR  = 000320         $NSK5 = 000110          $$LOCN= 000000
UIPAR1= 177642           WDTO   = 000114         $F$RTI= 000350          $SAVLE= 177777          $$REG = 177777
UIPAR2= 177644           WTINRE= 000352          $F$RTN= 000300          $SSK0 = 050004          $$RETU= 000000
UIPAR3= 177646           WTWHMI= 000222          $F$SEL= 000140          $TAGLE= 177777          $$RTN1= 050000
UIPAR4= 177650           XFLAG  = 000005         $F$THE= 000330          $TAGNU= 050021          $$RTN2= 050001
UIPAR5= 177652           XOFF   = 000023         $F$TRU= 000404          $TEMP  = 000300         $$SRC = 000000
UIPAR6= 177654           XON    = 000021         $F$UNT= 000130          $TSK0 = 050003          $$TGSV= 000000
UIPAR7= 177656           $BGNLE= 177777          $F$WHI= 000120          $TSK1 = 050004          $$TGS1= 000000
UIPDR0= 177600           $ERFLG= 000400          $F$YES= 000402          $TSK2 = 050020          $$TGS2= 000000
UIPDR1= 177602           $F$AND= 000310          $IFLEV= 177777          $TSK3 = 050006          $$TO   = 000000
UIPDR2= 177604           $F$BAD= 000401          $ISK0 = 000001          $TSK4 = 050007          $$$TAG= 050000
UIPDR3= 177606           $F$BLA= 000170          $ISK1 = 000001          $TSK5 = 050010          .      = 000610R
UIPDR4= 177610           $F$CAS= 000150          $ISK2 = 000001          $TSK6 = 050016
UIPDR5= 177612           $F$DEC= 000220          $LOCTA= 177777          $$ARGC= 000002
UIPDR6= 177614           $F$DO  = 000340         $LSTIN= 000001          $$BYTE= 000403

. ABS.  000000     000
        000510     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:ACTIV,DSKZ:ACTIV=SPMAC/ML,EQUATE,ACTIV
 RUN-TIME: 19 9 .4 SECONDS
 RUN-TIME RATIO: 49/28=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  ARGCHK CHECK A KEYBOARD COMMAND ARGUMENT
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       ARGCHK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE IS USED TO VERIFY THE EXISTENCE OF A KEYBOARD
                        ;       COMMAND ARGUMENT. IT IS GIVEN THE DECODE BUFFER POINTER.
                        ;       IT WILL SEARCH FOR THE FIRST NON SPACE
                        ;       CHARACTER. IF IT IS A <CR> A RETURN WITH ERROR OCCURS. IN EITHER
                        ;       CASE, THE UPDATED COMMAND DECODE BUFFER PTR IS RETURNED
                        ;       POINTING TO THE FIRST NON SPACE CHARACTER.
                        ;
                        ; INPUTS:
                        ;       1. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       UPDATED COMMAND DECODE BUFFER POINTER
                        ;       ERROR INDICATION
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ARGCHK IN <PTR> OUT <NEWPTR>
                        ;       WHERE PTR = CURRENT COMMAND DECODE BUFFER POINTER
                        ;             NEWPTR = UPDATED COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR ARGCHK

                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTIN=1
                        000001  $LSTTAG=1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  ARGCHK          ;MODULE ENTRY POINT
                                ;
                                ;***************

```

```
                                .SBTTL  ARGCHK ROUTINE

                                ROUTINE ARGCHK <PTR,NEWPTR>
                                                                        ARGCHK:

                                ;+
                                ; INITIALIZE AND SAVE CURRENT DECODE BUFFER POINTER
                                ;-

                                PUSH R0
                                                                                MOV     R0,-(SP)
                                LET R0 := PTR(R5)
                                                                                MOV     PTR(R5),R0

                                ;+
                                ; WHILE SPACE, ADVANCE TO NEXT CHARACTER
                                ;-

                                WHILEB (R0) EQ #SPACE DO
                                                                        50002$:
                                                                                CMPB    (R0),#SPACE
                                                                                BNE     50003$
                                        LET R0 := R0 + #1
                                                                                INC     R0
                                ENDDO
                                                                                BR      50002$
                                                                        50003$:

                                ;+
                                ; UPDATE THE CMD DECODE BUFFER POINTER
                                ; AND IF A <CR> RETURN WITH ERROR. IF NOT A <CR> RETURN
                                ; NO ERROR
                                ;-

                                LET NEWPTR(R5) := R0
                                                                                MOV     R0,NEWPTR(R5)

                                IFB (R0) EQ #CR THEN
                                                                                CMPB    (R0),#CR
                                                                                BNE     50004$
                                        POP R0
                                                                                MOV     (SP)+,R0
                                        RETURN ERROR
                                                                                SEC
                                                                                BR      50001$

                                ELSE
                                                                        50004$:

                                        POP R0
                                                                                MOV     (SP)+,R0
                                        RETURN NO.ERROR
                                                                                BR      50000$

                                ENDIF
```

```
(4)  000044'                                                    50005$:
614
615  000044'                        ENDRTN
(3)  000044'                                                    50000$:
(2)  000044' 000241                                                 CLC
(3)  000046'                                                    50001$:
(2)  000046' 000207                                                 RTS     PC
616
617          000001                .END
```

```
ACSR   = 000102  CSRC   = 000102  EOPBIT= 000001  MODSEL= 001000  RBUFSZ= 000132
ACTBIT= 004000  CTRLC  = 000003  ERRTYP= 000106  MSGCKD= 000010  RBUFVA= 000124
ADDR22= 001000  CTRLO  = 000017  EVNTBE= 000200  MSGCKS= 000011  RDSERV= 000101
ADR    = 000006  CTRLU  = 000025  EVNTHD= 000200  MSGDER= 000005  RDWHMI= 000022
APTFER= 000004  DCEVNT= 000011  EVNTKT= 000203  MSGDRP= 000017  RELERR= 000020
APTPRE= 000200  DEFRTN= 000400  EVNTPE= 000202  MSGECH= 177777  RELMOD= 020000
ARGCHK  000000RG  DIAGMC= 000000  EVNTRE= 000201  MSGEOP= 000013  RELTIM= 010000
ASB    = 000106  DROPMO= 100000  FATERR= 100000  MSGHDR= 000004  RES1   = 000056
ASSEMB= 000010  DSEVNT= 000014  HRDCNT= 000044  MSGHNG= 000022  RES2   = 000060
ASTAT  = 000104  DT.ADD= 000042  HRDPAS= 000050  MSGHRD= 000007  RICHAR= 031060
AUTO   = 000010  DT.AP  = 000100  ICONT  = 000036  MSGMAP= 000021  RPTDAT= 002000
AUTOST= 020000  DT.APK= 000076  ICOUNT= 000040  MSGNUL= 177775  RSTRT  = 000112
AWAS   = 000110  DT.BLS= 000034  IDNUM  = 000122  MSGPOP= 000002  RUBOUT= 000177
BIT0   = 000001  DT.CF0= 000014  IE     = 000100  MSGPRM= 177776  RUNMOD= 100000
BIT00  = 000001  DT.CF1= 000016  INDPAR= 000040  MSGRES= 000001  R5VALU= 001740
BIT01  = 000002  DT.ERR= 000020  INHDRP= 040000  MSGSFT= 000006  SAM    = 075464
BIT02  = 000004  DT.ESI= 000044  INHEPR= 020000  MSGSKE= 000003  SBADR  = 000102
BIT03  = 000010  DT.EVN= 000000  INHREL= 001000  MSGSMB= 000015  SBKMOD= 000000
BIT04  = 000020  DT.EXS= 000060  INHRRE= 000400  MSGSMH= 000014  SBKSEL= 010000
BIT05  = 000040  DT.FCH= 000037  INIT   = 000030  MSGSMS= 000016  SC.ADR= 000006
BIT06  = 000100  DT.FCN= 000036  INTR   = 000120  MSGSTD= 000000  SC.ALC= 000014
BIT07  = 000200  DT.HMX= 000104  IOMOD  = 100000  MSGSYS= 000012  SC.APC= 000016
BIT08  = 000400  DT.KBE= 000024  IOMODP= 102000  MSGVEC= 000020  SC.CKL= 000002
BIT09  = 001000  DT.KBP= 000026  IOMODR= 112000  NBKMOD= 001000  SC.CKP= 000004
BIT1   = 000002  DT.KBR= 000022  IOMODX= 110000  NCPUOP= 000020  SC.CLO= 000000
BIT10  = 002000  DT.KBU= 000030  JACK   = 035060  NEWPTR= 000002  SC.HLD= 000010
BIT11  = 004000  DT.MLS= 000032  KIPAR0= 172340  NOAPTY= 000002  SC.SCA= 000012
BIT12  = 010000  DT.MTI= 000110  KIPAR1= 172342  NULL   = 000000  SENDLS= 177777
BIT13  = 020000  DT.OFF= 000070  KIPAR2= 172344  OWEN   = 024020  SOFCNT= 000042
BIT14  = 040000  DT.PAS= 000074  KIPAR3= 172346  PAERR  = 000010  SOFPAS= 000046
BIT15  = 100000  DT.PC  = 000002  KIPAR4= 172350  PARPRE= 002000  SPACE  = 000040
BIT2   = 000004  DT.PFL= 000062  KIPAR5= 172352  PARSTA= 000100  SPOINT= 000032
BIT3   = 000010  DT.PSW= 000004  KIPAR6= 172354  PASCNT= 000034  SPVALU= 002200
BIT4   = 000020  DT.PTA= 000064  KIPAR7= 172356  PDPLSI= 020000  SR0    = 177572
BIT5   = 000040  DT.RCS= 000102  KIPDR0= 172300  PDP60  = 004000  SR1    = 177574
BIT6   = 000100  DT.REL= 000040  KIPDR1= 172302  PDP70  = 010000  SR2    = 177576
BIT7   = 000200  DT.SCT= 000066  KIPDR2= 172304  PRI0   = 000000  SR3    = 172516
BIT8   = 000400  DT.SMX= 000106  KIPDR3= 172306  PRI1   = 000040  STAT   = 000026
BIT9   = 001000  DT.SP  = 000006  KIPDR4= 172310  PRI4   = 000200  STATBI= 064757
BKDEF  = 000002  DT.SSI= 000046  KIPDR5= 172312  PRI5   = 000240  STAT1  = 000027
BKMOD  = 000020  DT.ST0= 000010  KIPDR6= 172314  PRI6   = 000300  SUSPND= 000001
BKMODE= 040000  DT.ST1= 000012  KIPDR7= 172316  PRI7   = 000340  SVR0   = 000062
BKSLSH= 000134  DT.SWR= 000056  KTERRO= 000040  PR0    = 000000  SVR1   = 000064
CAPRES= 000004  DT.SYP= 000072  KTPRES= 000400  PR4    = 000200  SVR2   = 000066
CASTAT= 000004  DT.WBU= 000050  KTSTAT= 000020  PR5    = 000240  SVR3   = 000070
CDERCT= 000146  DT.WHL= 000054  KTXTND= 040000  PR6    = 000300  SVR4   = 000072
CDWDCT= 000144  DT.WLL= 000052  LF     = 000012  PR7    = 000340  SVR5   = 000074
CKTIM  = 100000  DVID1  = 000014  LPSTAT= 000001  PS     = 177776  SVR6   = 000076
CLKPRE= 000001  ECCMEM= 000100  MAPSTA= 000200  PSW    = 177776  SYSCNT= 000052
CONFIG= 000056  ECCSTA= 000010  MED    = 076600  PTR    = 000000  SYSERR= 000100
CQOVF  = 000001  ENBEOP= 010000  MEMPAS= 040000  RANNUM= 000054  TMPIO  = 000002
CR     = 000015  ENBNUL= 000001  MODEXH= 004000  RBUFEA= 000130  TQOVF  = 000002
CSRA   = 000100  ENDLST= 000000  MODHOL= 002000  RBUFPA= 000126  UIPAR0= 177640
```

```
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LOCTA= 177777          $$ERFL= 000000
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000001
UIPAR3= 177646          WBUFSZ= 000142          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR4= 177650          WDFR  = 000116          $F$INC= 000210          $NESTL= 177777          $$LOC = 000030R
UIPAR5= 177652          WDTO  = 000114          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $NSK1 = 000110          $$REG = 177777
UIPAR7= 177656          WTWHMI= 000222          $F$NO = 000403          $SAVLE= 177777          $$RETU= 000001
UIPDR0= 177600          XFLAG = 000005          $F$OR = 000320          $SSK0 = 050003          $$RTN1= 050000
UIPDR1= 177602          XOFF  = 000023          $F$RTI= 000350          $TAGLE= 177777          $$RTN2= 050001
UIPDR2= 177604          XON   = 000021          $F$RTN= 000300          $TAGNU= 050006          $$SRC = 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TEMP = 000300          $$TGSV= 000000
UIPDR4= 177610          $ERFLG= 000000          $F$THE= 000330          $TSK0 = 050005          $$TGS1= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $TSK1 = 050003          $$TGS2= 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$ARGC= 000004          $$TO  = 000000
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$BYTE= 000402          $$$TAG= 050000
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$CASE= 000000          .     = 000050R
WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$DST = 000000
WBUFEA= 000136          $F$DO = 000340          $ISK0 = 000001          $$ELOC= 000402

. ABS.  000000      000
        000050      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:ARGCHK,DSKZ:ARGCHK=SPMAC/ML,EQUATE,ARGCHK
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 29/14=2.1
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  BACTIV - ACTIVATE NEXT BACKGROUND MODULE
                        .IDENT  /V0.0/


                        ;++
                        ; MODULE NAME:
                        ;       BACTIV
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE FETCHES THE ADDRESS OF THE NEXT BACKGROUND OPTION MODULE
                        ;       TO BE ACTIVATED FROM THE BACKGROUND JOB LIST CONTAINED WITHIN THE MODULE
                        ;       LIST, AND ACTIVATES THE BACKGROUND MODULE BY PLACING IT IN THE MONITOR'S
                        ;       NEXT-TO-EXECUTE SLOT
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.BLST
                        ;       2. MODULE'S STATUS WORD (STAT)
                        ;       3. MODULE'S START ADDRESS (INIT)
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. ERROR/NO ERROR INDICATOR
                        ;               WHERE   ERROR   -       NO BKGRD MODULE TO ACTIVATE
                        ;                       NO ERROR -      BKGRD MODULE HAS BEEN ACTIVATED
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. DX.HDR
                        ;       2. DX.RET
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL BACTIV IN <DTABLE>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
000000'                                 STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN = 1
        000001                          $LSTTAG = 1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  BACTIV                  ;MODULE'S ENTRY POINT
                                .GLOBL  BA.MPTP                 ;MODULE'S BKGRD LIST POINTER
                                .GLOBL  BA.HDR                  ;SUSPENDED BKGRD MODULE'S HEADER ADDRESS
                                .GLOBL  BA.RET                  ;SUSPENDED BKGRD MODULE'S RETURN ADDRESS
                                .GLOBL  BA.STAT                 ;MODULE'S STATUS WORD
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  DX.HDR                  ;MONITOR'S NEXT-TO-EXECUTE
                                .GLOBL  DX.RET                  ;  SLOT
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                ;
                                ;****************
                                ;
                                ; GLOBAL STORAGE
                                ;
000000' 000000                  BA.MPTR: .WORD  0               ;BACKGROUND-JOB-LIST POINTER
000002' 000000                  BA.HDR: .WORD   0               ;STORAGE FOR SUSPENDED MODULE'S HEADER ADDRESS
000004' 000000                  BA.RET: .WORD   0               ;STORAGE FOR SUSPENDED MODULE'S RESUME ADDRESS
000006' 000000                  BA.STAT: .WORD  0               ;BACKGROUND LIST STATUS WORD
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL  ACTIVATE BACKGROUND MODULE  ROUTINE

                                ROUTINE BACTIV <DTABLE>
(2) 000010'                                                                     BACTIV:

                                ;+
                                ; IS THERE A BACKGROUND MODULE CURRENTLY WORKING IT'S WAY THRU
                                ; THE QUEUING MECHANISM?
                                ;-

000010'                         IF #BKDEF SETIN BA.STAT THEN
(6) 000010' 032767  000002  177770                                              BIT     #BKDEF,BA.STAT
(9) 000016' 001402                                                              BEQ     50002$
000020'                                 RETURN ERROR
(2) 000020' 000261                                                              SEC
(4) 000022' 000513                                                              BR      50001$
000024'                         ENDIF
(4) 000024'                                                                     50002$:

                                ;+
                                ; DO THE NECESSARY SET UP STUFF
                                ;-

000024'                         CALL SAVREG
(3) 000024' 004767  000000G                                                     JSR     PC,SAVREG
000030'                         LET R0 := DTABLE(R5)
(4) 000030' 016500  000000                                                      MOV     DTABLE(R5),R0

                                ;+
                                ; IS THERE A BACKGROUND MODULE WAITING IN A SUSPEND STATE? IF SO, ACTIVATE IT BY
                                ; ENTERING IT INTO THE MONITOR'S NEXT-TO-EXECUTE SLOT AND RETURN TO CALLER
                                ;-
000034'                         IF #SUSPND SETIN BA.STATUS THEN
(6) 000034' 032767  000001  177744                                              BIT     #SUSPND,BA.STATU
(9) 000042' 001413                                                              BEQ     50003$
000044'                                 LET R1 := BA.HDR
(4) 000044' 016701  177732                                                      MOV     BA.HDR,R1
000050'                                 LET DX.HDR := R1
(4) 000050' 010167  000000G                                                     MOV     R1,DX.HDR
000054'                                 LET DX.RET := BA.RET
(4) 000054' 016767  177724  000000G                                             MOV     BA.RET,DX.RET
000062'                                 LET BA.STATUS := BA.STATUS CLR.BY #SUSPND
(6) 000062' 042767  000001  177716                                              BIC     #SUSPND,BA.STATU
000070'                                 INLINE  <BR     1$>
(2) 000070' 000461                                                              BR      1$
000072'                         ENDIF
(4) 000072'                                                                     50003$:

                                ;+
                                ; OTHERWISE SEARCH THE BACKGROUND LIST LOOKING FOR A BACKGROUND MODULE TO
                                ; ACTIVATE
                                ;-

000072'                         IF @BA.MPTR EQ #ENDLST THEN
(6) 000072' 027727  177702  000000                                              CMP     @BA.MPTR,#ENDLST
```

```
(9)  000100' 001003                                                          BNE     50004$
640
641                                 ;+
642                                 ; IF THE LIST ENTRY IS A ZERO, RESET IT TO THE TOP OF LIST
643                                 ;-
644
645  000102'                                LET BA.MPTR := DT.BLST(R0)
(4)  000102' 016067  000034  177670                                          MOV     DT.BLST(R0),BA.M
646  000110'                        ENDIF
(4)  000110'                                                          50004$:
647  000110'                        LET R3 := BA.MPTR
(4)  000110' 016703  177664                                          MOV     BA.MPTR,R3
648
649                                 ;+
650                                 ; SEARCH LIST UNTIL WE WIND UP BACK WHERE WE STARTED, OR UNTIL
651                                 ; WE FIND A MODULE THAT CAN BE ACTIVATED
652                                 ;-
653
654  000114'                        REPEAT
(3)  000114'                                                          50005$:
655  000114'                                LET R1 := @BA.MPTR
(4)  000114' 017701  177660                                          MOV     @BA.MPTR,R1
656
657                                 ;+
658                                 ; ONE HAS BEEN FOUND, NOW MAKE SURE THAT IT IS ELIGIBLE TO RUN
659                                 ;-
660
661  000120'                                LET R2 := STAT(R1)
(4)  000120' 016102  000026                                          MOV     STAT(R1),R2
662  000124'                                IF #BIT14 SETIN R2 AND #BIT13!ACTBIT NOTSETIN R2 THEN
(6)  000124' 032702  040000                                          BIT     #BIT14,R2
(9)  000130' 001421                                                  BEQ     50006$
(6)  000132' 032702  024000                                          BIT     #BIT13!ACTBIT,R2
(9)  000136' 001016                                                  BNE     50006$
663
664                                 ;+
665                                 ; IF IT IS ELIGIBLE, ENTER IT INTO THE MONITOR'S NEXT-TO-EXECUTE SLOT, SET ITS
666                                 ; ACTIVE BIT, AND UPDATE THE
667                                 ; BACKGROUND LIST POINTER, AND RETURN TO THE CALLER
668                                 ;-
669
670  000140'                                    LET DX.HDR := R1
(4)  000140' 010167  000000G                                         MOV     R1,DX.HDR
671  000144'                                    LET BA.HDR := R1
(4)  000144' 010167  177632                                          MOV     R1,BA.HDR
672  000150'                                    LET DX.RET := INIT(R1)
(4)  000150' 016167  000030  000000G                                 MOV     INIT(R1),DX.RET
673  000156'                                    LET SVR6(R1) := SPOINT(R1)
(4)  000156' 016161  000032  000076                                  MOV     SPOINT(R1),SVR6(
674  000164'                                    LET BA.MPTR := BA.MPTR + #2
(6)  000164' 062767  000002  177606                                  ADD     #2,BA.MPTR
675  000172'                                    INLINE  <BR     1$>
(2)  000172' 000420                                                  ER      1$
676  000174'                                ENDIF
(4)  000174'                                                          50006$:
677
```

```
                                ;+
                                ; UPDATE THE LIST POINTER AND CONTINUE SEARCHING
                                ;-
000174'                                 LET BA.MPTR := BA.MPTR + #2
000174' 062767  000002  177576                                          ADD     #2,BA.MPTR

                                ;+
                                ; IF ENTRY IS ZERO, RESET TO TOP OF LIST
                                ;-

000202'                                 IF @BA.MPTR EQ #0 THEN
000202' 005777  177572                                                  TST     @BA.MPTR
000206' 001003                                                          BNE     50007$
000210'                                         LET BA.MPTR := DT.BLST(R0)
000210' 016067  000034  177562                                          MOV     DT.BLST(R0),BA.M
000216'                                 ENDIF
000216'                                                         50007$:
000216'                         UNTIL BA.MPTR EQ R3
000216' 026703  177556                                                  CMP     BA.MPTR,R3
000222' 001334                                                          BNE     50005$


                                ;+
                                ; DO THE NECESSARY CLEAN-UP AND RETURN
                                ;-

000224'                         CALL RESREG
000224' 004767  000000G                                                 JSR     PC,RESREG
000230'                         RETURN ERROR
000230' 000261                                                          SEC
000232' 000407                                                          BR      50001$


                                ;+
                                ; SET THE ACTBIT AND RETURN
                                ;-

000234'                         INLINE <1$:>
000234'                                                                 1$:
000234'                         LET STAT(R1) := STAT(R1) SET.BY #ACTBIT
000234' 052761  004000  000026                                          BIS     #ACTBIT,STAT(R1)

000242'                         CALL RESREG
000242' 004767  000000G                                                 JSR     PC,RESREG
000246'                         RETURN NO.ERROR
000246' 000400                                                          BR      50000$
000250'                         ENDRTN
000250'                                                         50000$:
000250' 000241                                                          CLC
000252'                                                         50001$:
000252' 000207                                                          RTS     PC


000001                                  .END
```

```
ACSR   = 000102          CONFIG= 000056          DX.HDR= ****** G        LF     = 000012          PS     = 177776
ACTBIT= 004000          CQOVF  = 000001          DX.RET= ****** G        LPSTAT= 000001          PSW    = 177776
ADDR22= 001000          CR     = 000015          ECCMEM= 000100          MAPSTA= 000200          RANNUM= 000054
ADR    = 000006          CSRA   = 000100          ECCSTA= 000010          MED    = 076600          RBUFEA= 000130
APTFER= 000004          CSRC   = 000102          ENBEOP= 010000          MEMPAS= 040000          RBUFPA= 000126
APTPRE= 000200          CTRLC  = 000003          ENBNUL= 000001          MODEXH= 004000          RBUFSZ= 000132
ASB    = 000106          CTRLO  = 000017          ENDLST= 000000          MODHOL= 002000          RBUFVA= 000124
ASSEMB= 000010          CTRLU  = 000025          EOPBIT= 000001          MODSEL= 001000          RDSERV= 000101
ASTAT  = 000104          DCEVNT= 000011          ERRTYP= 000106          MSGCKD= 000010          RDWHMI= 000022
AUTO   = 000010          DEFRTN= 000400          EVNTBE= 000200          MSGCKS= 000011          RELERR= 000020
AUTOST= 020000          DIAGMC= 000000          EVNTHD= 000200          MSGDER= 000005          RELMOD= 020000
AWAS   = 000110          DROPMO= 100000          EVNTKT= 000203          MSGDRP= 000017          RELTIM= 010000
BACTIV  000010RG        DSEVNT= 000014          EVNTPE= 000202          MSGECH= 177777          RESREG= ****** G
BA.HDR  000002RG        DTABLE= 000000          EVNTRE= 000201          MSGEOP= 000013          RES1   = 000056
BA.MPT  000000RG        DT.ADD= 000042          FATERR= 100000          MSGHDR= 000004          RES2   = 000060
BA.RET  000004RG        DT.AP  = 000100          HRDCNT= 000044          MSGHNG= 000022          RICHAR= 031060
BA.STA  000006RG        DT.APK= 000076          HRDPAS= 000050          MSGHRD= 000007          RPTDAT= 002000
BIT0   = 000001          DT.BLS= 000034          ICONT  = 000036          MSGMAP= 000021          RSTRT  = 000112
BIT00  = 000001          DT.CF0= 000014          ICOUNT= 000040          MSGNUL= 177775          RUBOUT= 000177
BIT01  = 000002          DT.CF1= 000016          IDNUM  = 000122          MSGPOP= 000002          RUNMOD= 100000
BIT02  = 000004          DT.ERR= 000020          IE     = 000100          MSGPRM= 177776          R5VALU= 001740
BIT03  = 000010          DT.ESI= 000044          INDPAR= 000040          MSGRES= 000001          SAM    = 075464
BIT04  = 000020          DT.EVN= 000000          INHDRP= 040000          MSGSFT= 000006          SAVREG= ****** G
BIT05  = 000040          DT.EXS= 000060          INHEPR= 020000          MSGSKE= 000003          SBADR  = 000102
BIT06  = 000100          DT.FCH= 000037          INHREL= 001000          MSGSMB= 000015          SBKMOD= 000000
BIT07  = 000200          DT.FCN= 000036          INHRRE= 000400          MSGSMH= 000014          SBKSEL= 010000
BIT08  = 000400          DT.HMX= 000104          INIT   = 000030          MSGSMS= 000016          SC.ADR= 000006
BIT09  = 001000          DT.KBE= 000024          INTR   = 000120          MSGSTD= 000000          SC.ALC= 000014
BIT1   = 000002          DT.KBP= 000026          IOMOD  = 100000          MSGSYS= 000012          SC.APC= 000016
BIT10  = 002000          DT.KBR= 000022          IOMODP= 102000          MSGVEC= 000020          SC.CKL= 000002
BIT11  = 004000          DT.KBU= 000030          IOMODR= 112000          NBKMOD= 001000          SC.CKP= 000004
BIT12  = 010000          DT.MLS= 000032          IOMODX= 110000          NCPUOP= 000020          SC.CLO= 000000
BIT13  = 020000          DT.MTI= 000110          JACK   = 035060          NOAPTY= 000002          SC.HLD= 000010
BIT14  = 040000          DT.OFF= 000070          KIPAR0= 172340          NULL   = 000000          SC.SCA= 000012
BIT15  = 100000          DT.PAS= 000074          KIPAR1= 172342          OWEN   = 024020          SENDLS= 177777
BIT2   = 000004          DT.PC  = 000002          KIPAR2= 172344          PAERR  = 000010          SOFCNT= 000042
BIT3   = 000010          DT.PFL= 000062          KIPAR3= 172346          PARPRE= 002000          SOFPAS= 000046
BIT4   = 000020          DT.PSW= 000004          KIPAR4= 172350          PARSTA= 000100          SPACE  = 000040
BIT5   = 000040          DT.PTA= 000064          KIPAR5= 172352          PASCNT= 000034          SPOINT= 000032
BIT6   = 000100          DT.RCS= 000102          KIPAR6= 172354          PDPLSI= 020000          SPVALU= 002200
BIT7   = 000200          DT.REL= 000040          KIPAR7= 172356          PDP60  = 004000          SR0    = 177572
BIT8   = 000400          DT.SCT= 000066          KIPDR0= 172300          PDP70  = 010000          SR1    = 177574
BIT9   = 001000          DT.SMX= 000106          KIPDR1= 172302          PRI0   = 000000          SR2    = 177576
BKDEF  = 000002          DT.SP  = 000006          KIPDR2= 172304          PRI1   = 000040          SR3    = 172516
BKMOD  = 000020          DT.SSI= 000046          KIPDR3= 172306          PRI4   = 000200          STAT   = 000026
BKMODE= 040000          DT.ST0= 000010          KIPDR4= 172310          PRI5   = 000240          STATBI= 064757
BKSLSH= 000134          DT.ST1= 000012          KIPDR5= 172312          PRI6   = 000300          STAT1  = 000027
CAPRES= 000004          DT.SWR= 000056          KIPDR6= 172314          PRI7   = 000340          SUSPND= 000001
CASTAT= 000004          DT.SYP= 000072          KIPDR7= 172316          PR0    = 000000          SVR0   = 000062
CDERCT= 000146          DT.WBU= 000050          KTERRO= 000040          PR4    = 000200          SVR1   = 000064
CDWDCT= 000144          DT.WHL= 000054          KTPRES= 000400          PR5    = 000240          SVR2   = 000066
CKTIM  = 100000          DT.WLL= 000052          KTSTAT= 000020          PR6    = 000300          SVR3   = 000070
CLKPRE= 000001          DVID1  = 000014          KTXTND= 040000          PR7    = 000340          SVR4   = 000072
```

```
SVR5  = 000074        UIPDR5= 177612        $F$BAD= 000401        $F$WHI= 000120        $$CASE= 000000
SVR6  = 000076        UIPDR6= 177614        $F$BLA= 000170        $F$YES= 000402        $$DST = 000000
SYSCNT= 000052        UIPDR7= 177616        $F$CAS= 000150        $IFLEV= 177777        $$ELOC= 000403
SYSERR= 000100        WASADR= 000104        $F$DEC= 000220        $ISK0 = 000001        $$ERFL= 000000
TMPIO = 000002        WBSTAT= 000040        $F$DO = 000340        $LOCTA= 177777        $$FLAG= 000001
TQOVF = 000002        WBUFEA= 000136        $F$FAL= 000405        $LSTIN= 000001        $$FROM= 000000
UIPAR0= 177640        WBUFPA= 000134        $F$GOO= 000400        $LSTTA= 000001        $$LOC = 000222R
UIPAR1= 177642        WBUFRQ= 000140        $F$IF = 000110        $NESTL= 177777        $$LOCN= 000000
UIPAR2= 177644        WBUFSZ= 000142        $F$INC= 000210        $NSK0 = 000300        $$REG = 177777
UIPAR3= 177646        WDFR  = 000116        $F$LOO= 000200        $NSK1 = 000130        $$RETU= 000001
UIPAR4= 177650        WDTO  = 000114        $F$NAM= 000160        $NSK2 = 000110        $$RTN1= 050000
UIPAR5= 177652        WTINRE= 000352        $F$NO = 000403        $SAVLE= 177777        $$RTN2= 050001
UIPAR6= 177654        WTWHMI= 000222        $F$OR = 000320        $TAGLE= 177777        $$SRC = 000000
UIPAR7= 177656        XFLAG = 000005        $F$RTI= 000350        $TAGNU= 050010        $$TGSV= 000000
UIPDR0= 177600        XOFF  = 000023        $F$RTN= 000300        $TEMP = 000300        $$TGS1= 000000
UIPDR1= 177602        XON   = 000021        $F$SEL= 000140        $TSK0 = 050005        $$TGS2= 000000
UIPDR2= 177604        $BGNLE= 177777        $F$THE= 000330        $TSK1 = 050007        $$TO  = 000000
UIPDR3= 177606        $ERFLG= 000000        $F$TRU= 000404        $$ARGC= 000002        $$$TAG= 050000
UIPDR4= 177610        $F$AND= 000310        $F$UNT= 000130        $$BYTE= 000403        .     = 000254R

. ABS.  000000     000
        000254     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BACTIV,DSKZ:BACTIV=SPMAC/ML,EQUATE,BACTIV
 RUN-TIME: 14 4 .3 SECONDS
 RUN-TIME RATIO: 35/19=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE BADMEM - REPORT BAD MEMORY TRANSFER
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       BADMEM
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS USED TO REPORT A BAD MEMORY TRANSFER.
                        ;       IT IS PASSED THE ADDRESS A TABLE CONTAINING THE FAILING ADDRESS, ALONG
                        ;       WITH THE ACTUAL AND SHOULD-BE CONTENTS FOR THAT ADDRESS.  THE ROUTINE
                        ;       CONVERTS THESE VALUES TO ASCII AND OUTPUTS THEM.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;       ADDRESS OF TABLE CONTAINING LOW-ORDER AND HIGH-ORDER BITS OF PA
                        ;       ACTUAL CONTENTS
                        ;       SHOULD-BE CONTENTS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       BOAC                            ;BINARY TO OCTAL-ASCII CONVERSION
                        ;       BOA16                           ;16-BIT BINARY TO OCTAL-ASCII CONVERSION
                        ;       MSGDHOOK                        ;HOOK MESSAGE TO PROPER DRIVER
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL BADMEM IN <A,B,C,D>
                        ;               A=ADDRESS OF DATA TABLE
                        ;               B=ADDRESS OF LOW ORDER 16 BITS OF PA
                        ;               C=ACTUAL CONTENTS
                        ;               D=SHOULD-BE CONTENTS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  BADMEM
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  BOAC                    ;BINARY TO OCTAL-ASCII CONVERSION
                                .GLOBL  BOA16                   ;16-BIT BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  MSGDHOOK                ;HOOK MESSAGE TO PROPER DRIVER
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 041045  042101  046440  BM.MSG: .ASCII /%BAD MEMORY TRANSFER%ADDR: /
000006' 046505  051117  020131
000014' 051124  047101  043123
000022' 051105  040445  042104
000030' 035122     040
000033' 000010                  BM.ADR: .BLKB 10
000043'    040  020040  047503          .ASCII /   CONTENTS: /
000050' 052116  047105  051524
000056' 020072
000060' 000006                  BM.BD:  .BLKB 6
000066' 020040  051440  047510          .ASCII /   SHOULD BE: /
000074' 046125  020104  042502
000102' 020072
000104' 000006                  BM.GD:  .BLKB 6
000112' 000045                          .ASCIZ /%/

                                .EVEN
```

```
                                .SBTTL BADMEM ROUTINE

                                ROUTINE BADMEM <TABL,PA,BD,GD>
(2)  000114'                                                            BADMEM:

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
(3)  000114' 004767  000000G                                           JSR     PC,SAVREG


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE, R1 TO THE ADDRESS OF THE PA TABLE,
                                ; R2 TO THE ACTUAL CONTENTS, AND R3 TO THE SHOULD-BE CONTENTS.
                                ;-

                                LET R0 := TABL(R5)
(4)  000120' 016500  000000                                             MOV     TABL(R5),R0
                                LET R1 := PA(R5)
(4)  000124' 016501  000002                                             MOV     PA(R5),R1
                                LET R2 := BD(R5)
(4)  000130' 016502  000004                                             MOV     BD(R5),R2
                                LET R3 := GD(R5)
(4)  000134' 016503  000006                                             MOV     GD(R5),R3


                                ;+
                                ; CONVERT THE PA TO ASCII.
                                ;-

                                CALL BOAC IN <R0,(R1),2(R1),#BM.ADR>
(3)  000140' 010546                                                     MOV     R5,-(SP)
(7)  000142' 012745  000033'                                            MOV     #BM.ADR,-(R5)
(6)  000146' 016145  000002                                             MOV     2(R1),-(R5)
(5)  000152' 011145                                                     MOV     (R1),-(R5)
(4)  000154' 010045                                                     MOV     R0,-(R5)
(3)  000156' 004767  000000G                                           JSR     PC,BOAC
(3)  000162' 012605                                                     MOV     (SP)+,R5


                                ;+
                                ; CONVERT THE ACTUAL AND SHOULD-BE DATA TO ASCII.
                                ;-

                                CALL BOA16 IN <R2,#BM.BD>
(3)  000164' 010546                                                     MOV     R5,-(SP)
(5)  000166' 012745  000060'                                            MOV     #BM.BD,-(R5)
(4)  000172' 010245                                                     MOV     R2,-(R5)
(3)  000174' 004767  000000G                                           JSR     PC,BOA16
(3)  000200' 012605                                                     MOV     (SP)+,R5
                                CALL BOA16 IN <R3,#BM.GD>
(3)  000202' 010546                                                     MOV     R5,-(SP)
(5)  000204' 012745  000104'                                            MOV     #BM.GD,-(R5)
(4)  000210' 010345                                                     MOV     R3,-(R5)
```

```
(3)  000212' 004767  000000G                                                    JSR     PC,BOA16
(3)  000216' 012605                                                             MOV     (SP)+,R5
636
637
638                          ;+
639                          ; OUTPUT THE MESSAGE.
640                          ;-
641
642  000220'                 CALL MSGDHOOK IN <R0,#MSGPOP,#BM.MSG,#2$>
(3)  000220' 010546                                                             MOV     R5,-(SP)
(7)  000222' 012745  000252'                                                    MOV     #2$,-(R5)
(6)  000226' 012745  000000'                                                    MOV     #BM.MSG,-(R5)
(5)  000232' 012745  000002                                                     MOV     #MSGPOP,-(R5)
(4)  000236' 010045                                                             MOV     R0,-(R5)
(3)  000240' 004767  000000G                                                    JSR     PC,MSGDHOOK
(3)  000244' 012605                                                             MOV     (SP)+,R5
643  000246'                 INLINE <1$:     NOP>
(2)  000246' 000240                                                             1$:     NOP
644  000250'                 INLINE <BR      1$>
(2)  000250' 000776                                                             BR      1$
645  000252'                 INLINE <2$:>
(2)  000252'                                                                    2$:
646
647
648                          ;+
649                          ; RESTORE REGISTERS AND RETURN.
650                          ;-
651
652  000252'                 CALL RESREG
(3)  000252' 004767  000000G                                                    JSR     PC,RESREG
653
654  000256'                 ENDRTN
(3)  000256'                                                            50000$:
(3)  000256'                                                            50001$:
(2)  000256' 000207                                                             RTS     PC
655          000001         .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BADMEM  000114RG
BD     = 000004
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BM.ADR  000033R
BM.BD   000060R
BM.GD   000104R
BM.MSG  000000R
BOAC   = ****** G
BOA16  = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GD     = 000006
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PA     = 000002
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
```

```
SVR2   = 000066          UIPDR1= 177602          XON    = 000021          $F$SEL= 000140          $$DST  = 000000
SVR3   = 000070          UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $$ELOC= 000000
SVR4   = 000072          UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $$ERFL= 000000
SVR5   = 000074          UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $$FLAG= 000000
SVR6   = 000076          UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$FROM= 000000
SYSCNT= 000052           UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$LOC  = 000000
SYSERR= 000100           UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$LOCN= 000000
TABL   = 000000          WASADR= 000104          $F$DEC= 000220          $LOCTA= 177777          $$REG  = 177777
TMPIO  = 000002          WBSTAT= 000040          $F$DO  = 000340          $LSTIN= 000001          $$RETU= 000000
TQOVF  = 000002          WBUFEA= 000136          $F$FAL= 000405          $LSTTA= 000001          $$RTN1= 050000
UIPAR0= 177640           WBUFPA= 000134          $F$GOO= 000400          $NESTL= 177777          $$RTN2= 050001
UIPAR1= 177642           WBUFRQ= 000140          $F$IF  = 000110          $NSKO  = 000300          $$SRC  = 000000
UIPAR2= 177644           WBUFSZ= 000142          $F$INC= 000210          $SAVLE= 177777          $$TGSV= 000000
UIPAR3= 177646           WDFR   = 000116          $F$LOO= 000200          $TAGLE= 177777          $$TGS1= 000000
UIPAR4= 177650           WDTO   = 000114          $F$NAM= 000160          $TAGNU= 050002          $$TGS2= 000000
UIPAR5= 177652           WTINRE= 000352          $F$NO  = 000403          $TEMP  = 000300          $$TO   = 000000
UIPAR6= 177654           WTWHMI= 000222          $F$OR  = 000320          $$ARGC= 000010          $$$TAG= 050000
UIPAR7= 177656           XFLAG  = 000005          $F$RTI= 000350          $$BYTE= 000000          .      = 000260R
UIPDR0= 177600           XOFF   = 000023          $F$RTN= 000300          $$CASE= 000000

. ABS.  000000    000
        000260    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BADMEM,DSKZ:BADMEM=SPMAC/ML,EQUATE,BADMEM
 RUN-TIME:  11  1  .3 SECONDS
 RUN-TIME RATIO: 28/14=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  BDACNV  -  BINARY TO DECIMAL-ASCII CONVERSION ROUTINE
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE NAME:
                        ;       BDACNV
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE CONVERTS A 16-BIT UNSIGNED BINARY NUMBER
                        ;       TO A 5-DIGIT DECIMAL-ASCII NUMBER. THE TECHNIQUE USED
                        ;       IS THAT OF REPEATED DIVISION BY POWERS OF 10 (10), WITH THE RE-
                        ;       MAINDERS, WHICH COMPOSE THE DECIMAL NUMBER, BEING
                        ;       CONVERTED TO ASCII.
                        ;
                        ; INPUTS:
                        ;       1. 16-BIT BINARY NUMBER THAT IS TO BE CONVERTED
                        ;       2. ADDRESS AT WHICH THE RESULT IS TO BE STORED
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. 5-CHARACTER DECIMAL-ASCII RESULT
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       1. SAVREG               SAVE REGISTERS
                        ;       2. RESREG               RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL BDACNV IN <NUMBER,ADDRESS>
                        ;               NUMBER  = 16-BIT BINARY NUMBER TO BE CONVERTED
                        ;               ADDRESS = ADDRESS TO STORE 5-CHAR RESULT
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
000000'                                 STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN = 1
        000001                          $LSTTAG = 1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  BDACNV                  ;MODULE'S ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 023420  001750  000144  BD.TEN: ^D10000,^D1000,^D100,^D10,^D1
000006' 000012  000001
                                ;
```

```
(4)  000040' 005003                                                          CLR     R3
641
642  000042'                        WHILE NUM(R5) HIS (R1) DO
(4)  000042'                                                          50004$:
(6)  000042' 026511  000000                                                  CMP     NUM(R5),(R1)
(9)  000046' 103404                                                          BLO     50005$
643  000050'                                LET NUM(R5) := NUM(R5) - (R1)
(6)  000050' 161165  000000                                                  SUB     (R1),NUM(R5)
644  000054'                                LET R3 := R3 + #1
(6)  000054' 005203                                                          INC     R3
645  000056'                        ENDDO
(4)  000056' 000771                                                          BR      50004$
(3)  000060'                                                          50005$:
646
647  000060'                        LET R3 := R3 + #60
(6)  000060' 062703  000060                                                  ADD     #60,R3
648  000064'                        LET (R0)+ :B= R3
(4)  000064' 110320                                                          MOVB    R3,(R0)+
649
650  000066'                        LET R1 := R1 + #2
(6)  000066' 062701  000002                                                  ADD     #2,R1
651  000072'                ENDDO
(4)  000072' 000760                                                          BR      50002$
(3)  000074'                                                          50003$:
652
653
654
655                         ;+
656                         ; RESTORE REGISTERS AND RETURN.
657                         ;-
658
659  000074'                CALL RESREG
(3)  000074' 004767  000000G                                                 JSR     PC,RESREG
660
661
662  000100'                ENDRTN
(3)  000100'                                                          50000$:
(3)  000100'                                                          50001$:
(2)  000100' 000207                                                          RTS     PC
663
664          000001         .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000002
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BDACNV  000012RG
BD.TEN  000000R
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQDVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
NUM    = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
```

```
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LSTTA= 000001          $$ERFL= 000000
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $NESTL= 177777          $$FLAG= 000340
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $NSK0 = 000300          $$FROM= 000000
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NSK1 = 000120          $$LOC = 000046R
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK2 = 000120          $$LOCN= 000000
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $SAVLE= 177777          $$REG = 177777
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $SSK0 = 050003          $$RETU= 000000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $TAGLE= 177777          $$RTN1= 050000
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $TAGNU= 050006          $$RTN2= 050001
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $TEMP = 000300          $$SRC = 000000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TSK0 = 050002          $$TGSV= 000000
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TSK1 = 050003          $$TGS1= 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TSK2 = 050004          $$TGS2= 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $TSK3 = 050005          $$TO  = 000000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$ARGC= 000004          $$$TAG= 050000
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$BYTE= 000403          .     = 000102R
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$CASE= 000000
WASADR= 000104          $F$DEC= 000220          $LOCTA= 177777          $$DST = 000000
WBSTAT= 000040          $F$DO = 000340          $LSTIN= 000001          $$ELOC= 000000

. ABS.  000000      000
        000102      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BDACNV,DSKZ:BDACNV=SPMAC/ML,EQUATE,BDACNV
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 30/15=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE  BDVSRV  SERVICE INTERRUPTS THROUGH INCORRECT VECTORS
                                .IDENT  /V0.0/

                                ;++
                                ; MODULE PACKAGE NAME:
                                ;       BDVSRV
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS PACKAGE CONSISTS OF THE FOLLOWING ROUTINES:
                                ;       1. BADVEC - ISSUES AN EMT INSTRUCTION
                                ;       2. PBDVEC - FORMS BAD VECTOR MESSAGE
                                ;       3. SETVEC - SETS UP VECTORS
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND STORAGE
                                ;
                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTIN=1
        000001                  $LSTTAG=1


                                ;REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  BADVEC          ;MODULE ENTRY POINT
                                .GLOBL  BADVC1          ;MODULE ENTRY POINT
                                .GLOBL  PBDVEC          ;MODULE ENTRY POINT
                                .GLOBL  SETVEC          ;MODULE ENTRY POINT

                                ;****************

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQTQ           ;MESSAGE ENQUEUER
                                .GLOBL  DTABLE          ;ADDRESS OF DTABLE
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS


                                ;
                                ;****************
                                ; LOCAL STORAGE
                                ;

000000' 000000                  BD.VCT: .WORD 0         ;HIGH/LOW VECTOR FLAG
000002' 000000                          .WORD   0       ;SAVE R5 STUFF HERE , NOT IN INTERRUPTED MODULE
000004' 000000                          .WORD   0
000006' 000000                          .WORD   0
000010' 000000                          .WORD   0
000012' 000000                          .WORD   0
000014'                         SVR5S:
```

```
                        .SBTTL  BADVEC FIELD AN INTERRUPT THROUGH AN INCORRECT VECTOR
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       BADVEC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE IS ENTERED AS A RESULT OF AN INTERRUPT THROUGH
                        ;       AN INCORRECT VECTOR. BY ISSUING AN EMT INSTRUCTION,THE PSW
                        ;       IS PUSHED ONTO THE STACK. THE PSW WILL
                        ;       CONTAIN THE LOCATION OF THE BAD VECTOR.
                        ;       THE ROUTINE INVOKED BY THE EMT DETERMINES THE BAD VECTOR
                        ;       AND QUEUES A BAD VECTOR MESSAGE. CONTROL IS RETURNED TO THIS
                        ;       ROUTINE AND A RETURN TO THE INTERRUPTED ROUTINE THEN OCCURS.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       SINCE THE HARDWARE HAS INTERRUPTED THROUGH THE WRONG VECTOR
                        ;       IT'S INTERRUPT ROUTINE WILL NOT BE EXECUTED.
                        ;       THE DECX OPTION MODULE CONTROLLING THE
                        ;       HARDWARE WHICH INTERRUPTED THROUGH THE WRONG VECTOR
                        ;       ADDRESS WILL BECOME SUSPENDED.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALLED BY A HARDWARE INTERRUPT THROUGH AN INCORRECT VECTOR
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                    ;
```

```
                                .SBTTL  PBDVEC PROCESS EMTS(& INTERRUPTS THRU UNEXPECTED VECTORS)
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       PBDVEC
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       IF THE SYSTEM HARDWARE INTERRUPTS THRU AN UNEXPECTED VECTOR, AN
                                ;       EMT INSTRUCTION IS EXECUTED WHICH PUSHES THE PSW AND PC
                                ;       ONTO THE STACK AND THIS ROUTINE IS ENTERED. THIS PSW ON THE
                                ;       STACK CONTAINS THE UNEXPECTED VECTOR ADDRESS. A MESSAGE
                                ;       CONTAINING THE UNEXPECTED VECTOR IS QUEUED AND A RETURN TO THE
                                ;       ROUTINE WHICH ISSUED THE EMT TAKES PLACE.
                                ;
                                ; INPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       ADDRESS OF DATA TABLE
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       ENQTQ     MESSAGE ENQUEUER
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       ISSUE AN EMT INSTRUCTION
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                .SBTTL PBDVEC ROUTINE

                                IROUTINE PBDVEC
(2)  000024'                                                                    PBDVEC:

                                ;+
                                ; INITIALIZATION - SAVE R0,R1 AND SAVE R5 HERE
                                ;-

000024'                         PUSH R0,R1,R5
(2)  000024'  010046                                                            MOV     R0,-(SP)
(3)  000026'  010146                                                            MOV     R1,-(SP)
(4)  000030'  010546                                                            MOV     R5,-(SP)

                                ;+
                                ; INIT THE NEW R5 STACK
                                ;-

000032'                         LET R5 := #SVR5S
(4)  000032'  012705  000014'                                                   MOV     #SVR5S,R5

                                ;+
                                ; GET THE UNEXPECTED VECTOR CODE FROM THE STACK
                                ; SAVE IT IN R0,R1. CLEAR OUT ALL BUT BITS 0-3 IN R1
                                ; AND ALL BUT BITS 4-7 IN R0. SHIFT R1 TO LEFT.
                                ; AND COMBINE THE TWO WORDS INTO R0.
                                ; THIS IS DONE TO CONVERT THE SPECIAL VECTOR CODE BACK
                                ; INTO THE ACTUAL VECTOR IT REPRESENTS.
                                ;-


000036'                         LET R0 := 10(SP)
(4)  000036'  016600  000010                                                    MOV     10(SP),R0
000042'                         LET R1 := 10(SP)
(4)  000042'  016601  000010                                                    MOV     10(SP),R1
000046'                         LET R1 := R1 CLR.BY #177760
(6)  000046'  042701  177760                                                    BIC     #177760,R1
000052'                         LET R0 := R0 CLR.BY #177417
(6)  000052'  042700  177417                                                    BIC     #177417,R0
000056'                         LET R1 := R1 SHIFT +1
(7)  000056'  006301                                                            ASL     R1
000060'                         LET R0 := R0 SET.BY R1
(6)  000060'  050100                                                            BIS     R1,R0

                                ;+
                                ; IF INTERRUPT VECTOR GREATER THEN 374 THEN ADD 400 TO BAD VECTOR
                                ;-

000062'                         IF 6(SP) HI #BADVC1 AND 6(SP) LO #BADVEC THEN
(6)  000062'  026627  000006  000014'                                           CMP     6(SP),#BADVC1
(9)  000070'  101406                                                            BLOS    50002$
(6)  000072'  026627  000006  000020'                                           CMP     6(SP),#BADVEC
(9)  000100'  103002                                                            BHIS    50002$
000102'                                 LET R0 := R0 + #400
(6)  000102'  062700  000400                                                    ADD     #400,R0
000106'                         ENDIF
```

```
(4)  000106'                                                                          50002$:
741
742
743                                 ;+
744                                 ; ENQUEUE THE MESSAGE
745                                 ;-
746  000106'                        CALL ENQTQ IN <#DTABLE,#MSGVEC,R0,#0,#0>
(3)  000106' 010546                                                                            MOV     R5,-(SP)
(8)  000110' 012745  000000                                                                    MOV     #0,-(R5)
(7)  000114' 012745  000000                                                                    MOV     #0,-(R5)
(6)  000120' 010045                                                                            MOV     R0,-(R5)
(5)  000122' 012745  000020                                                                    MOV     #MSGVEC,-(R5)
(4)  000126' 012745  000000G                                                                   MOV     #DTABLE,-(R5)
(3)  000132' 004767  000000G                                                                   JSR     PC,ENQTQ
(3)  000136' 012605                                                                            MOV     (SP)+,R5
747
748                                 ;+
749                                 ; CLEAN UP
750                                 ;-
751
752  000140'                        LET BD.VCT := #0
(4)  000140' 005067  177634                                                                    CLR     BD.VCT
753
754
755  000144'                        POP R5,R1,R0
(2)  000144' 012605                                                                            MOV     (SP)+,R5
(3)  000146' 012601                                                                            MOV     (SP)+,R1
(4)  000150' 012600                                                                            MOV     (SP)+,R0
756                                 ;+
757                                 ; RETURN
758                                 ;-
759
760  000152'                        ENDRTI
(3)  000152'                                                                          50000$:
(3)  000152'                                                                          50001$:
(2)  000152' 000002                                                                            RTI
761
762
```

```
                        .SBTTL  SETVEC INITIALIZE ALL UNUSED VECTOR LOCATIONS (0-774)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SETVEC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL LOAD THE ADDRESS OF THE INCORRECT VECTOR
                        ;       ROUTINE IN ALL UNUSED VECTOR LOCATIONS FROM 0 TO 774. IT WILL
                        ;       PLACE A UNIQUE CODE IN ALL VECTOR + 2 LOCATIONS. THE CODE
                        ;       REPRESENTS THE VECTOR LOCATION .
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL <SETVEC>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
  309
  310                                   .SBTTL SETVEC ROUTINE
  311  000154'                          ROUTINE SETVEC
  (2)  000154'                                                                  SETVEC:
  312
  313                                   ;+
  314                                   ; SAVE REGISTERS AND FLAG TO LOW VECTORS
  315                                   ;-
  316
  317  000154'                          CALL SAVREG
  (3)  000154' 004767  000000G                                                 JSR     PC,SAVREG
  318  000160'                          LET BD.VCT := #0
  (4)  000160' 005067  177614                                                  CLR     BD.VCT
  319
  320                                   ;+
  321                                   ; LOAD LOC. 0 WITH BADVEC AND LOC. 2 WITH CODE
  322                                   ;-
  323
  324  000164'                          LET @#0 := #BADVEC
  (4)  000164' 012737  000020' 000000                                          MOV     #BADVEC,@#0
  325  000172'                          LET @#2 := #0
  (4)  000172' 005037  000002                                                  CLR     @#2
  326
  327
  328                                   ;+
  329                                   ; LOAD TRACE TRAP VECTOR WITH BADVEC, AND TRACE TRAP'S PSW WITH CODE
  330                                   ;-
  331
  332  000176'                          LET @#14 := #BADVEC
  (4)  000176' 012737  000020' 000014                                          MOV     #BADVEC,@#14
  333  000204'                          LET @#16 := #6
  (4)  000204' 012737  000006  000016                                          MOV     #6,@#16
  334
  335                                   ;+
  336                                   ; START AT LOC. 70 AND START LOADING CODES
  337                                   ;-
  338
  339  000212'                          LET R0 := #70
  (4)  000212' 012700  000070                                                  MOV     #70,R0
  340
  341                                   ;+
  342                                   ; LOAD CODE UP TO LOCATION 1000 (WILL NOT LOAD 114,200  OR 250)
  343                                   ;-
  344
  345  000216'                          WHILE R0 NE #1000 DO
  (4)  000216'                                                                  50002$:
  (6)  000216' 020027  001000                                                  CMP     R0,#1000
  (9)  000222' 001446                                                          BEQ     50003$
  346  000224'                                IF R0 NE #114 AND R0 NE #200 AND R0 NE #250 THEN
  (6)  000224' 020027  000114                                                  CMP     R0,#114
  (9)  000230' 001441                                                          BEQ     50004$
  (6)  000232' 020027  000200                                                  CMP     R0,#200
  (9)  000236' 001436                                                          BEQ     50004$
  (6)  000240' 020027  000250                                                  CMP     R0,#250
  (9)  000244' 001433                                                          BEQ     50004$
  347
```

```
  848                                   ;+
  849                                   ; DETERMINE WHETHER WORKING VECTORS 0-376 OR 400-776
  850                                   ;-
  851
  852  000246'                                  IF BD.VCT NE #0 THEN
(6)    000246' 005767  177526                                                    TST     BD.VCT
(9)    000252' 001403                                                            BEQ     50005$
  853  000254'                                        LET (R0) := #BADVC1
(4)    000254' 012710  000014'                                                   MOV     #BADVC1,(R0)
  854  000260'                                  ELSE
(4)    000260' 000410                                                            BR      50006$
(3)    000262'                                                          50005$:
  855  000262'                                        LET (R0) := #BADVEC
(4)    000262' 012710  000020'                                                   MOV     #BADVEC,(R0)
  856
  857                                   ;+
  858                                   ; IF AT VECTOR 374 THEN SET FLAG TO HIGH VECTORS
  859                                   ;-
  860
  861  000266'                                  IF R0 EQ #374 THEN
(6)    000266' 020027  000374                                                    CMP     R0,#374
(9)    000272' 001003                                                            BNE     50007$
  862  000274'                                        LET BD.VCT := #1
(4)    000274' 012767  000001  177476                                            MOV     #1,BD.VCT
  863  000302'                                  ENDIF
(4)    000302'                                                          50007$:
  864  000302'                                  ENDIF
(4)    000302'                                                          50006$:
  865
  866                                   ;+
  867                                   ; STRIP OFF BIT 08 FOR CONVERSION
  868                                   ;-
  869
  870  000302'                                  LET R1 := R0 CLR.BY #BIT08
(4)    000302' 010001                                                            MOV     R0,R1
(6)    000304' 042701  000400                                                    BIC     #BIT08,R1
  871  000310'                                  INLINE <TST (R0)+>
(2)    000310' 005720                                                            TST (R0)+
  872
  873                                   ;+
  874                                   ; SHIFT RIGHT ONCE (DIVIDE BY 2) AND ONLY SAVE LOWER 4 BITS
  875                                   ;-
  876
  877  000312'                                  LET R2 := R1 SHIFT -1
(4)    000312' 010102                                                            MOV     R1,R2
(7)    000314' 006202                                                            ASR     R2
  878  000316'                                  LET R2 := R2 CLR.BY #177760
(6)    000316' 042702  177760                                                    BIC     #177760,R2
  879
  880                                   ;+
  881                                   ; NOW STRIP UNWANTED BITS FROM R1 AND "OR" TOGETHER R1 AND R2
  882                                   ;-
  883
  884  000322'                                  LET R1 := R1 CLR.BY #177037
(6)    000322' 042701  177037                                                    BIC     #177037,R1
  885  000326'                                  LET R1 := R1 SET.BY R2
```

```
(6)  000326' 050201                                                                  BIS     R2,R1
886
887                               ;+
888                               ; LOAD CODE INTO PSW WORD OF VECTOR
889                               ;-
890
891  000330'                                  LET (R0)+ := R1
(4)  000330' 010120                                                                  MOV     R1,(R0)+
892  000332'                           ELSE
(4)  000332' 000401                                                                  BR      50010$
(3)  000334'                                                                 50004$:
893
894                               ;+
895                               ; UPDATE R0 BY 4
896                               ;-
897
898  000334'                                  INLINE <CMP (R0)+,(R0)+>
(2)  000334' 022020                                                                  CMP (R0)+,(R0)+
899  000336'                           ENDIF
(4)  000336'                                                                 50010$:
900  000336'                          ENDDO
(4)  000336' 000727                                                                  BR      50002$
(3)  000340'                                                                 50003$:
901
902                               ;+
903                               ; CLEAN UP
904                               ;-
905
906  000340'                          LET BD.VCT := #0
(4)  000340' 005067  177434                                                          CLR     BD.VCT
907  000344'                          CALL RESREG
(3)  000344' 004767  000000G                                                         JSR     PC,RESREG
908  000350'                          ENDRTN
(3)  000350'                                                                 50000$:
(3)  000350'                                                                 50001$:
(2)  000350' 000207                                                                  RTS     PC
909          000001              .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BADVC1  000014RG
BADVEC  000020RG
BD.VCT  000000R
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= ****** G
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PBDVEC  000024RG
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SETVEC  000154RG
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
```

```
SVR5S   000014R          UIPDR7= 177616          $F$DO  = 000340          $LOCTA= 177777          $$CASE= 000000
SVR6  = 000076           WASADR= 000104          $F$FAL= 000405          $LSTIN= 000001          $$DST = 000000
SYSCNT= 000052           WBSTAT= 000040          $F$GOO= 000400          $LSTTA= 000001          $$ELOC= 000402
SYSERR= 000100           WBUFEA= 000136          $F$IF  = 000110          $NESTL= 177777          $$ERFL= 000000
TMPIO  = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK0  = 000300         $$FLAG= 000001
TQOVF  = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK1  = 000120         $$FROM= 000000
UIPAR0= 177640           WBUFSZ= 000142          $F$NAM= 000160          $NSK2  = 000110         $$LOC  = 000272R
UIPAR1= 177642           WDFR   = 000116         $F$NO  = 000403         $NSK3  = 000110         $$LOCN= 000000
UIPAR2= 177644           WDTO   = 000114         $F$OR  = 000320         $NSK4  = 000110         $$REG  = 177777
UIPAR3= 177646           WTINRE= 000352          $F$RTI= 000350          $SAVLE= 177777          $$RETU= 000000
UIPAR4= 177650           WTWHMI= 000222          $F$RTN= 000300          $SSK0  = 050003         $$RTN1= 050000
UIPAR5= 177652           XFLAG  = 000005         $F$SEL= 000140          $TAGLE= 177777          $$RTN2= 050001
UIPAR6= 177654           XOFF   = 000023         $F$THE= 000330          $TAGNU= 050011          $$SRC  = 000000
UIPAR7= 177656           XON    = 000021         $F$TRU= 000404          $TEMP  = 000300         $$TGSV= 000000
UIPDR0= 177600           $BGNLE= 177777          $F$UNT= 000130          $TSK0  = 050002         $$TGS1= 000000
UIPDR1= 177602           $ERFLG= 000400          $F$WHI= 000120          $TSK1  = 050003         $$TGS2= 000000
UIPDR2= 177604           $F$AND= 000310          $F$YES= 000402          $TSK2  = 050010         $$TO   = 000000
UIPDR3= 177606           $F$BAD= 000401          $IFLEV= 177777          $TSK3  = 050006         $$$TAG= 050000
UIPDR4= 177610           $F$BLA= 000170          $ISK0  = 000001         $TSK4  = 050007         .      = 000352R
UIPDR5= 177612           $F$CAS= 000150          $ISK1  = 000001         $$ARGC= 000000
UIPDR6= 177614           $F$DEC= 000220          $ISK2  = 000001         $$BYTE= 000403

. ABS.  000000     000
        000352     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BDVSRV,DSKZ:BDVSRV=SPMAC/ML,EQUATE,BDVSRV
 RUN-TIME: 16 6 .4 SECONDS
 RUN-TIME RATIO: 40/23=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE BOAC BINARY TO OCTAL ASCII CONVERSION FOR 18 & 22 BIT NUMBERS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       BOAC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL CONVERT AN 18 OR 22 BIT BINARY NUMBER TO A
                        ;       8 CHARACTER OCTAL ASCII STRING STORED AT THE CALLER SPECIFIED
                        ;       ADDRESS.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE.
                        ;       2. BITS 0 THRU 15 OF THE NUMBER TO BE CONVERTED
                        ;       3. BITS 16 & 17 OR BITS 16-21 OF THE NUMBER TO BE CONVERTED
                        ;             (SEE CALLING SEQUENCE BELOW)
                        ;       4. STARTING ADDRESS OF THE 8 BYTE STORAGE AREA WHERE
                        ;          THE OCTAL ASCII CHARACTERS ARE TO BE STORED.
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       OCTAL ASCII CHARACTERS
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       BOA16    16 BIT BINARY TO OCTAL ASCII CONVERT ROUTINE
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL BOAC IN <DTADR,NUM1,NUM2,ADDR>
                        ;       WHERE DTADR = ADDRESS OF DTABLE
                        ;             NUM1 = BITS 0-15 OF THE OCTAL NUMBER TO BE CONVERTED
                        ;
                        ;                    F O R   1 8   B I T S
                        ;
                        ;             NUM2 = BITS 16 & 17 OF THE OCTAL NUMBER TO BE CONVERTED
                        ;                  ( BITS 16 & 17 MUST BE LINED UP IN BIT POSITIONS
                        ;                    4 AND 5  I.E.  0 000 000 000 XY0 000 WHERE
                        ;                    X = BIT 17 AND Y = BIT 16 )
                        ;
                        ;                    F O R   2 2   B I T S
                        ;
                        ;              NUM2 = BITS 16 - 21 OF THE # TO BE CONVERTED.
```

```
                        ;                       (NUM2 MUST BE IN THE FOLLOWING FORMAT:
                        ;                        0 000 000 000 FED CBA
                        ;                        WHERE F = BIT 21
                        ;                              E = BIT 20
                        ;                              D = BIT 19
                        ;                              C = BIT 18
                        ;                              B = BIT 17
                        ;                              A = BIT 16
                        ;
                        ;               ADDR = THE START ADDRESS OF THE 8 BYTE STORAGE AREA
                        ;                      WHERE THE OCTAL ASCII CHARACTERS ARE TO BE STORED.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR BOAC

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;***************
                                ;
                                ;REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  BOAC            ;MODULE ENTRY POINT
                                ;
                                ;*****************
                                ;
                                ;GLOBAL REFERENCES
                                ;
                                .GLOBL  BOA16           ;CALLED TO CONVERT BITS 0-15 TO ASCII
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;
```

```
                                .SBTTL  BOAC ROUTINE

                                ROUTINE BOAC <DTADR,NUMB1,NUMB2,ADDR>
                                                                                BOAC:

                                ;+
                                ;SAVE REGISTERS
                                ;-

                                CALL SAVREG
                                                                                JSR     PC,SAVREG

                                ;+
                                ;GET DTABLE ADDRESS TO R0
                                ;-

                                LET R4 := DTADR(R5)
                                                                                MOV     DTADR(R5),R4

                                ;+
                                ; PUT THE LOW ORDER BITS IN R0, THE HIGH ORDER BITS IN R3, AND
                                ; THE ADDRESS OF THESTORAGE AREA IN R1.
                                ;-

                                LET R0 := NUMB1(R5)
                                                                                MOV     NUMB1(R5),R0
                                LET R3 := NUMB2(R5)
                                                                                MOV     NUMB2(R5),R3
                                LET R1 := ADDR(R5)
                                                                                MOV     ADDR(R5),R1

                                ;+
                                ; IF NOT 18-BIT ADDRESSING GO TO 22-BIT SECTION.
                                ;-

                                IF #MAPSTAT NOTSETIN DT.ST0(R4) THEN
                                                                                BIT     #MAPSTAT,DT.ST0(
                                                                                BNE     50002$

                                ;+
                                ;SINCE ONLY 18 BITS REQUIRED, ZERO THE UPPER 2 MOST SIGNIFICANT
                                ;ASCII LOCATIONS AND UPDATE TABLE ADDRESS
                                ;-

                                        LET (R1)+ :B= #60
                                                                                MOVB    #60,(R1)+
                                        LET (R1)+ :B= #60
                                                                                MOVB    #60,(R1)+


                                ;+
                                ;CONVERT BITS 0-15 TO ASCII
```

```
                                ;-
                                ;
000044'                                 CALL BOA16  IN <R0,R1>
(3)  000044' 010546                                                     MOV     R5,-(SP)
(5)  000046' 010145                                                     MOV     R1,-(R5)
(4)  000050' 010045                                                     MOV     R0,-(R5)
(3)  000052' 004767  000000G                                           JSR     PC,BOA16
(3)  000056' 012605                                                     MOV     (SP)+,R5
                                ;
                                ;+
                                ;CLEAR OUT ANY UNNEEDED BITS IF ANY ARE ACCIDENTALLY SET
                                ;IN EXTENDED ADDRESS WORD
                                ;-
                                ;
000060'                                 LET R3 := R3 CLR.BY #177717
(6)  000060' 042703  177717                                             BIC     #177717,R3
                                ;
                                ;+
                                ;SHIFT EXTENDED ADDRESS BITS FROM POSITION 4 & 5  TO 1 & 2
                                ;-
                                ;
000064'                                 LET R3 := R3 SHIFT -3
(7)  000064' 006203                                                     ASR     R3
(7)  000066' 006203                                                     ASR     R3
(7)  000070' 006203                                                     ASR     R3
                                ;
                                ;+
                                ;ADD EA BITS TO FIRST ASCII CHARACTER(MOST SIGNIFICANT CHARACTER)
                                ;THE SEQUENCE SHOWN IS TO HANDLE TABLES STARTING ON ODD BOUNDARIES
                                ;-
                                ;
000072'                                 LET R0 :B= (R1)
(4)  000072' 111100                                                     MOVB    (R1),R0
000074'                                 LET R0 := R0 + R3
(6)  000074' 060300                                                     ADD     R3,R0
000076'                                 LET (R1) :B= R0
(4)  000076' 110011                                                     MOVB    R0,(R1)
                                ;
000100'                         ELSE
(4)  000100' 000443                                                     BR      50003$
(3)  000102'                                                    50002$:
                                ;
```

```
                                .SBTTL  22 BIT ADDRESSING CONVERSION
                                ;+
                                ;               2 2   B I T   A D D R E S S I N G
                                ;-


                                ;+
                                ;INITIALIZE AND SAVE ADDRESS + 2 OF ASCII STORAGE AREA
                                ;TO CONVERT LOWER BITS(0-15) TO OCTAL ASCII
                                ;ALSO CLEAN UP ANY UNWANTED BITS FROM BITS 16-21
                                ;-

000102'                                 LET R1 := R1 + #2
(6) 000102' 062701  000002                                              ADD     #2,R1
000106'                                 LET R3 := R3 CLR.BY #177700
(6) 000106' 042703  177700                                              BIC     #177700,R3

                                ;+
                                ;SAVE BITS 0-15 AND CONVERT TO OCTAL ASCII
                                ;-

000112'                                 CALL BOA16 IN <R0,R1>
(3) 000112' 010546                                                      MOV     R5,-(SP)
(5) 000114' 010145                                                      MOV     R1,-(R5)
(4) 000116' 010045                                                      MOV     R0,-(R5)
(3) 000120' 004767  000000G                                             JSR     PC,BOA16
(3) 000124' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ;SHIFT BITS 16-21 LEFT ONE TIME
                                ;AND SAVE TO ALIGN BITS 16,17 WITH BITS 1,2
                                ;-

000126'                                 LET R3 := R3 SHIFT +1
(7) 000126' 006303                                                      ASL     R3
000130'                                 LET R0 := R3
(4) 000130' 010300                                                      MOV     R3,R0


                                ;+
                                ;MASK OFF ALL BUT BITS 0-2 IN REG AND ADD TO MOST
                                ;SIGNIFICANT OCTAL ASCII CHARACTER
                                ;THE SEQUENCE SHOWN IS FOR HANDLING ODD ADDRESSES AS WELL AS EVEN
                                ;-

000132'                                 LET R0 := R0 CLR.BY #177770
(6) 000132' 042700  177770                                              BIC     #177770,R0
000136'                                 LET R2 :B= (R1)
(4) 000136' 111102                                                      MOVB    (R1),R2
```

```
                                        LET R2 := R2 + R0
(6)  000140' 060002                                                     ADD     R0,R2
                                        LET (R1) :B= R2
(4)  000142' 110211                                                     MOVB    R2,(R1)

                                ;+
                                ;SHIFT BITS 16-21 3 TIMES TO RIGHT
                                ;-

                                        LET R3 := R3 SHIFT -3
(7)  000144' 006203                                                     ASR     R3
(7)  000146' 006203                                                     ASR     R3
(7)  000150' 006203                                                     ASR     R3

                                ;+
                                ;SET UP COUNTER
                                ;-

                                        LET R2 := #2
(4)  000152' 012702  000002                                             MOV     #2,R2

                                ;+
                                ;WHILE COUNTER NOT = 0 ,SAVE BITS 18-21 IN R0,MASK OFF ALL
                                ;BUT BITS 0-2 IN R0,ADD #60 TO R0, STORE R0 IN ASCII AREA
                                ;SHIFT BITS 18-21 3 TIMES TO RIGHT AND DECREMENT COUNTER
                                ;-

                                        WHILE R2 NE #0 DO
(4)  000156'                                                    50004$:
(6)  000156' 005702                                                     TST     R2
(9)  000160' 001413                                                     BEQ     50005$

                                ;+
                                ;SAVE BITS 18-21
                                ;-

                                                LET R0 := R3
(4)  000162' 010300                                                     MOV     R3,R0

                                ;+
                                ;MASK OFF UNNEEDED BITS AND ADD # 60
                                ;-

                                                LET R0 := R0 CLR.BY #177770
(6)  000164' 042700  177770                                             BIC     #177770,R0
                                                LET R0 := R0 + #60
(6)  000170' 062700  000060                                             ADD     #60,R0

                                ;+
                                ;STORE IN ASCII AREA, SHIFT BITS AND DECREMENT COUNTER
                                ;-

                                                LET -(R1) :B= R0
(4)  000174' 110041                                                     MOVB    R0,-(R1)
                                                LET R3 := R3 SHIFT -3
```

```
(7)  000176' 006203                                                          ASR     R3
(7)  000200' 006203                                                          ASR     R3
(7)  000202' 006203                                                          ASR     R3
770  000204'                                LET R2 := R2 - #1
(6)  000204' 005302                                                          DEC     R2
771
772  000206'                            ENDDO
(4)  000206' 000763                                                          BR      50004$
(3)  000210'                                                          50005$:
773
774
775
776
777
778  000210'                        ENDIF
(4)  000210'                                                          50003$:
779
780                                 ;+
781                                 ; RESTORE REGISTERS
782                                 ;-
783
784  000210'                        CALL RESREG
(3)  000210' 004767  000000G                                                 JSR     PC,RESREG
785
786  000214'                        ENDRTN
(3)  000214'                                                          50000$:
(3)  000214'                                                          50001$:
(2)  000214' 000207                                                          RTS     PC
787          000001                .END
```

```
ACSR  = 000102          CR    = 000015          ENBEOP= 010000          MEMPAS= 040000          RANNUM= 000054
ACTBIT= 004000          CSRA  = 000100          ENBNUL= 000001          MODEXH= 004000          RBUFEA= 000130
ADDR  = 000006          CSRC  = 000102          ENDLST= 000000          MODHOL= 002000          RBUFPA= 000126
ADDR22= 001000          CTRLC = 000003          EOPBIT= 000001          MODSEL= 001000          RBUFSZ= 000132
ADR   = 000006          CTRLO = 000017          ERRTYP= 000106          MSGCKD= 000010          RBUFVA= 000124
APTFER= 000004          CTRLU = 000025          EVNTBE= 000200          MSGCKS= 000011          RDSERV= 000101
APTPRE= 000200          DCEVNT= 000011          EVNTHD= 000200          MSGDER= 000005          RDWHMI= 000022
ASB   = 000106          DEFRTN= 000400          EVNTKT= 000203          MSGDRP= 000017          RELERR= 000020
ASSEMB= 000010          DIAGMC= 000000          EVNTPE= 000202          MSGECH= 177777          RELMOD= 020000
ASTAT = 000104          DROPMO= 100000          EVNTRE= 000201          MSGEOP= 000013          RELTIM= 010000
AUTO  = 000010          DSEVNT= 000014          FATERR= 100000          MSGHDR= 000004          RESREG= ****** G
AUTOST= 020000          DTADR = 000000          HRDCNT= 000044          MSGHNG= 000022          RES1  = 000056
AWAS  = 000110          DT.ADD= 000042          HRDPAS= 000050          MSGHRD= 000007          RES2  = 000060
BIT0  = 000001          DT.AP = 000100          ICONT = 000036          MSGMAP= 000021          RICHAR= 031060
BIT00 = 000001          DT.APK= 000076          ICOUNT= 000040          MSGNUL= 177775          RPTDAT= 002000
BIT01 = 000002          DT.BLS= 000034          IDNUM = 000122          MSGPOP= 000002          RSTRT = 000112
BIT02 = 000004          DT.CFO= 000014          IE    = 000100          MSGPRM= 177776          PUBOUT= 000177
BIT03 = 000010          DT.CF1= 000016          INDPAR= 000040          MSGRES= 000001          RUNMOD= 100000
BIT04 = 000020          DT.ERR= 000020          INHDRP= 040000          MSGSFT= 000006          R5VALU= 001740
BIT05 = 000040          DT.ESI= 000044          INHEPR= 020000          MSGSKE= 000003          SAM   = 075464
BIT06 = 000100          DT.EVN= 000000          INHREL= 001000          MSGSMB= 000015          SAVREG= ****** G
BIT07 = 000200          DT.EXS= 000060          INHRRE= 000400          MSGSMH= 000014          SBADR = 000102
BIT08 = 000400          DT.FCH= 000037          INIT  = 000030          MSGSMS= 000016          SBKMOD= 000000
BIT09 = 001000          DT.FCN= 000036          INTR  = 000120          MSGSTD= 000000          SBKSEL= 010000
BIT1  = 000002          DT.HMX= 000104          IOMOD = 100000          MSGSYS= 000012          SC.ADR= 000006
BIT10 = 002000          DT.KBE= 000024          IOMODP= 102000          MSGVEC= 000020          SC.ALC= 000014
BIT11 = 004000          DT.KBP= 000026          IOMODR= 112000          NBKMOD= 001000          SC.APC= 000016
BIT12 = 010000          DT.KBR= 000022          IOMODX= 110000          NCPUOP= 000020          SC.CKL= 000002
BIT13 = 020000          DT.KBU= 000030          JACK  = 035060          NOAPTY= 000002          SC.CKP= 000004
BIT14 = 040000          DT.MLS= 000032          KIPAR0= 172340          NULL  = 000000          SC.CLO= 000000
BIT15 = 100000          DT.MTI= 000110          KIPAR1= 172342          NUMB1 = 000002          SC.HLD= 000010
BIT2  = 000004          DT.OFF= 000070          KIPAR2= 172344          NUMB2 = 000004          SC.SCA= 000012
BIT3  = 000010          DT.PAS= 000074          KIPAR3= 172346          OWEN  = 024020          SENDLS= 177777
BIT4  = 000020          DT.PC = 000002          KIPAR4= 172350          PAERR = 000010          SOFCNT= 000042
BIT5  = 000040          DT.PFL= 000062          KIPAR5= 172352          PARPRE= 002000          SOFPAS= 000046
BIT6  = 000100          DT.PSW= 000004          KIPAR6= 172354          PARSTA= 000100          SPACE = 000040
BIT7  = 000200          DT.PTA= 000064          KIPAR7= 172356          PASCNT= 000034          SPOINT= 000032
BIT8  = 000400          DT.RCS= 000102          KIPDR0= 172300          PDPLSI= 020000          SPVALU= 002200
BIT9  = 001000          DT.REL= 000040          KIPDR1= 172302          PDP60 = 004000          SR0   = 177572
BKDEF = 000002          DT.SCT= 000066          KIPDR2= 172304          PDP70 = 010000          SR1   = 177574
BKMOD = 000020          DT.SMX= 000106          KIPDR3= 172306          PRI0  = 000000          SR2   = 177576
BKMODE= 040000          DT.SP = 000006          KIPDR4= 172310          PRI1  = 000040          SR3   = 172516
BKSLSH= 000134          DT.SSI= 000046          KIPDR5= 172312          PRI4  = 000200          STAT  = 000026
BOAC    000000RG        DT.ST0= 000010          KIPDR6= 172314          PRI5  = 000240          STATBI= 064757
BOA16 = ****** G        DT.ST1= 000012          KIPDR7= 172316          PRI6  = 000300          STAT1 = 000027
CAPRES= 000004          DT.SWR= 000056          KTERRO= 000040          PRI7  = 000340          SUSPND= 000001
CASTAT= 000004          DT.SYP= 000072          KTPRES= 000400          PR0   = 000000          SVR0  = 000062
CDERCT= 000146          DT.WBU= 000050          KTSTAT= 000020          PR4   = 000200          SVR1  = 000064
CDWDCT= 000144          DT.WHL= 000054          KTXTND= 040000          PR5   = 000240          SVR2  = 000066
CKTIM = 100000          DT.WLL= 000052          LF    = 000012          PR6   = 000300          SVR3  = 000070
CLKPRE= 000001          DVID1 = 000014          LPSTAT= 000001          PR7   = 000340          SVR4  = 000072
CONFIG= 000056          ECCMEM= 000100          MAPSTA= 000200          PS    = 177776          SVR5  = 000074
CQOVF = 000001          ECCSTA= 000010          MED   = 076600          PSW   = 177776          SVR6  = 000076
```

```
SYSCNT= 000052   UIPDR7= 177616   $F$CAS= 000150   $IFLEV= 177777   $$CASE= 000000
SYSERR= 000100   WASADR= 000104   $F$DEC= 000220   $ISK0 = 000001   $$DST = 000000
TMPIO = 000002   WBSTAT= 000040   $F$DO = 000340   $LOCTA= 177777   $$ELOC= 000402
TQOVF = 000002   WBUFEA= 000136   $F$FAL= 000405   $LSTIN= 000001   $$ERFL= 000000
UIPAR0= 177640   WBUFPA= 000134   $F$GOO= 000400   $LSTTA= 000001   $$FLAG= 000001
UIPAR1= 177642   WBUFRQ= 000140   $F$IF = 000110   $NESTL= 177777   $$FROM= 000000
UIPAR2= 177644   WBUFSZ= 000142   $F$INC= 000210   $NSK0 = 000300   $$LOC = 000160R
UIPAR3= 177646   WDFR  = 000116   $F$LOO= 000200   $NSK1 = 000110   $$LOCN= 000000
UIPAR4= 177650   WDTO  = 000114   $F$NAM= 000160   $NSK2 = 000120   $$REG = 177777
UIPAR5= 177652   WTINRE= 000352   $F$NO = 000403   $SAVLE= 177777   $$RETU= 000000
UIPAR6= 177654   WTWHMI= 000222   $F$OR = 000320   $SSK0 = 050005   $$RTN1= 050000
UIPAR7= 177656   XFLAG = 000005   $F$RTI= 000350   $TAGLE= 177777   $$RTN2= 050001
UIPDR0= 177600   XOFF  = 000023   $F$RTN= 000300   $TAGNU= 050006   $$SRC = 000000
UIPDR1= 177602   XON   = 000021   $F$SEL= 000140   $TEMP = 000300   $$TGSV= 000000
UIPDR2= 177604   $BGNLE= 177777   $F$THE= 000330   $TSK0 = 050003   $$TGS1= 000000
UIPDR3= 177606   $ERFLG= 000400   $F$TRU= 000404   $TSK1 = 050004   $$TGS2= 000000
UIPDR4= 177610   $F$AND= 000310   $F$UNT= 000130   $TSK2 = 050005   $$TJ  = 000000
UIPDR5= 177612   $F$BAD= 000401   $F$WHI= 000120   $$ARGC= 000010   $$$TAG= 050000
UIPDR6= 177614   $F$BLA= 000170   $F$YES= 000402   $$BYTE= 000403   .     = 000216R

. ABS.  000000   000
        000216   001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BOAC,DSKZ:BOAC=SPMAC/ML,EQUATE,BOAC
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 36/20=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE BOA16 OCTAL TO ASCII CONVERSION ROUTINE FOR 16 BIT NUMBERS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       BOA16
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL CONVERT A SIX CHARACTER (SIXTEEN BIT) OCTAL
                        ;       NUMBER TO A SIX CHARACTER ASCII STRING STORED AT THE CALLER
                        ;       SPECIFIED ADDRESS.
                        ;
                        ; INPUTS:
                        ;       1. NUMBER TO BE CONVERTED.
                        ;       2. START ADDRESS OF 6 BYTE STORAGE AREA WHERE
                        ;          ASCII CHARACTERS ARE TO BE STORED.
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG  - SAVE GPRS
                        ;       RESREG  - RESTORE GPRS

                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL BOA16  IN <A,B>
                        ;       WHERE A = NUMBER TO BE CONVERTED
                        ;             B = START ADDRESS OF 6 BYTE STORAGE AREA WHERE ASCII
                        ;                 CHARACTERS ARE TO BE STORED
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;       ----            ----            --              ------
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES FOR BOA16

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
          000001                $LSTTAG=1
          000001                $LSTIN=1
                                ; ****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL BOA16            ;ENTRY POINT FOR THIS MODULE
                                ;
                                ; ****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG          ;SAVE GPRS ROUTINE
                                .GLOBL  RESREG          ;RESTORE GPRS ROUTINE
```

```
                                .SBTTL BOA16 ROUTINE
000000'                         ROUTINE BOA16 <NUM,ADRESS>
(2) 000000'                                                             BOA16:

                                ;+
                                ; SAVE REGISTERS
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                                     JSR     PC,SAVREG

                                ;+
                                ; GET 6 BYTE STORAGE ADDRESS + 6 AND SET CHARACTER COUNT
                                ;-

000004'                         LET R0 := ADRESS(R5) + #6
(4) 000004' 016500  000002                                                      MOV     ADRESS(R5),R0
(6) 000010' 062700  000006                                                      ADD     #6,R0
000014'                         LET R1 := #6
(4) 000014' 012701  000006                                                      MOV     #6,R1
000020'                         LET R3 := NUM(R5)
(4) 000020' 016503  000000                                                      MOV     NUM(R5),R3

                                ;+
                                ; WHILE R1 IS NOT = 0,GET THE OCTAL #,CLEAR OUT ALL BUT LSB 0-2,
                                ; ADD 60 (CONVERT TO ASCII), STORE THE ASCII ,UPDATE POINTER,CLEAR
                                ; OUT 3 BITS JUST CONVERTED, SHIFT NEXT 3 BITS, DECREMENT COUNTER
                                ;-

000024'                         WHILE R1 NE #0 DO
(4) 000024'                                                             50002$:
(6) 000024' 005701                                                              TST     R1
(9) 000026' 001415                                                              BEQ     50003$
000030'                                 LET R2 := R3
(4) 000030' 010302                                                              MOV     R3,R2
000032'                                 LET R2 := R2 CLR.BY #177770
(6) 000032' 042702  177770                                                      BIC     #177770,R2
000036'                                 LET R2 := R2 + #60
(6) 000036' 062702  000060                                                      ADD     #60,R2
000042'                                 LET -(R0) :B= R2
(4) 000042' 110240                                                              MOVB    R2,-(R0)
000044'                                 LET R3 := R3 CLR.BY #7
(6) 000044' 042703  000007                                                      BIC     #7,R3
000050'                                 LET R3 := R3 ROTATE -3
(7) 000050' 006003                                                              ROR     R3
(7) 000052' 006003                                                              ROR     R3
(7) 000054' 006003                                                              ROR     R3
000056'                                 LET R1 := R1 - #1
(6) 000056' 005301                                                              DEC     R1
000060'                         ENDDO
(4) 000060' 000761                                                              BR      50002$
(3) 000062'                                                             50003$:


                                ;+
```

```
                                ; CLEAN UP
                                ;-

000062'                         CALL RESREG
(3) 000062' 004767  000000G                                         JSR     PC,RESREG

000066'                         ENDRTN
(3) 000066'                                                     50000$:
(3) 000066'                                                     50001$:
(2) 000066' 000207                                                  RTS     PC
          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
ADRESS= 000002
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16   000000RG
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CFO= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.STO= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
NUM   = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
```

```
TQOVF = 000002          WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$ELOC= 000000
UIPAR0= 177640          WBUFEA= 000136          $F$DO = 000340          $LOCTA= 177777          $$ERFL= 000000
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000340
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
UIPAR3= 177646          WBUFSZ= 000142          $F$IF = 000110          $NESTL= 177777          $$LOC = 000026R
UIPAR4= 177650          WDFR  = 000116          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
UIPAR5= 177652          WDTO  = 000114          $F$LOO= 000200          $NSK1 = 000120          $$REG = 177777
UIPAR6= 177654          WTINRS= 000352          $F$NAM= 000160          $SAVLE= 177777          $$RETU= 000000
UIPAR7= 177656          WTWHMI= 000222          $F$NO = 000403          $SSK0 = 050003          $$RTN1= 050000
UIPDR0= 177600          XFLAG = 000005          $F$OR = 000320          $TAGLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XOFF  = 000023          $F$RTI= 000350          $TAGNU= 050004          $$SRC = 000000
UIPDR2= 177604          XON   = 000021          $F$RTN= 000300          $TEMP = 000300          $$TGSV= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TSK0 = 050002          $$TGS1= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $TSK1 = 050003          $$TGS2= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000004          $$ARGC= 000004          $$TO  = 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$BYTE= 000403          $$$TAG= 050000
UIPDR7= 177516          $F$BLA= 000170          $F$WHI= 000120          $$CASE= 000000          .     = 000070R
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$DST = 000000

. ABS.  000000     000
        000070     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:BOA16,DSKZ:BOA16=SPMAC/ML,EQUATE,BOA16
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 30/15=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE CLREOP - CLEAR END-OF-PASS INDICATORS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CLREOP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CLEARS THE 1ST-END-OF-PASS INDICATOR IN EACH OPTION
                        ;       MODULE OF THE RUNTIME EXERCISER
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       MODULES' 1ST-EOP INDICATOR, LOCATED IN MODULE'S HEADER WORD, XFLAG
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CLREOP IN <A>
                        ;               A=DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES
                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  CLREOP                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                          ;SAVE GPR'S
                                .GLOBL  RESREG                          ;RESTORE GPR'S
```

```
                                .SBTTL CLREOP ROUTINE

                                ROUTINE CLREOP <TABL>
(2)  000000'                                                    CLREOP:


                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
(3)  000000' 004767  000000G                                            JSR     PC,SAVREG


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := TABL(R5)
(4)  000004' 016500  000000                                             MOV     TABL(R5),R0


                                ;+
                                ; GET THE MODULE LIST
                                ;-

                                LET R1 := DT.MLST(R0)
(4)  000010' 016001  000032                                             MOV     DT.MLST(R0),R1


                                ;+
                                ; CLEAR EOPBIT IN XFLAG IN EACH MODULE HEADER
                                ;-

                                WHILE (R1) NE #ENDLST DO
(4)  000014'                                                    50002$:
(6)  000014' 021127  000000                                             CMP     (R1),#ENDLST
(9)  000020' 001405                                                     BEQ     50003$
                                        LET R2 := (R1)+
(4)  000022' 012102                                                     MOV     (R1)+,R2
                                        LET XFLAG(R2) :B= XFLAG(R2) CLR.BY #EOPBIT
(6)  000024' 142762  000001  000005                                    BICB    #EOPBIT,XFLAG(R2
                                ENDDO
(4)  000032' 000770                                                     BR      50002$
(3)  000034'                                                    50003$:


                                ;+
                                ; RESTORE REGISTERS AND RETURN
                                ;-

                                CALL RESREG
(3)  000034' 004767  000000G                                           JSR     PC,RESREG

                                ENDRTN
(3)  000040'                                                    50000$:
```

```
   (3)  000040'                                                    50001$:
   (2)  000040' 000207                                                     RTS     PC
   615
   616          000001                  .END
```

```
ACSR   = 000102          CSRC   = 000102          EOPBIT= 000001          MODSEL= 001000          RDSERV= 000101
ACTBIT= 004000          CTRLC  = 000003          ERRTYP= 000106          MSGCKD= 000010          RDWHMI= 000022
ADDR22= 001000          CTRLO  = 000017          EVNTBE= 000200          MSGCKS= 000011          RELERR= 000020
ADR    = 000006          CTRLU  = 000025          EVNTHD= 000200          MSGDER= 000005          RELMOD= 020000
APTFER= 000004          DCEVNT= 000011          EVNTKT= 000203          MSGDRP= 000017          RELTIM= 010000
APTPRE= 000200          DEFRTN= 000400          EVNTPE= 000202          MSGECH= 177777          RESREG= ****** G
ASB    = 000106          DIAGMC= 000000          EVNTRE= 000201          MSGEOP= 000013          RES1   = 000056
ASSEMB= 000010          DROPMO= 100000          FATERR= 100000          MSGHDR= 000004          RES2   = 000060
ASTAT  = 000104          DSEVNT= 000014          HRDCNT= 000044          MSGHNG= 000022          RICHAR= 031060
AUTO   = 000010          DT.ADD= 000042          HRDPAS= 000050          MSGHRD= 000007          RPTDAT= 002000
AUTOST= 020000          DT.AP  = 000100          ICONT  = 000036          MSGMAP= 000021          RSTRT  = 000112
AWAS   = 000110          DT.APK= 000076          ICOUNT= 000040          MSGNUL= 177775          RUBOUT= 000177
BIT0   = 000001          DT.BLS= 000034          IDNUM  = 000122          MSGPOP= 000002          RUNMOD= 100000
BIT00  = 000001          DT.CF0= 000014          IE     = 000100          MSGPRM= 177776          RSVALU= 001740
BIT01  = 000002          DT.CF1= 000016          INDPAR= 000040          MSGRES= 000001          SAM    = 075464
BIT02  = 000004          DT.ERR= 000020          INHDRP= 040000          MSGSFT= 000006          SAVREG= ****** G
BIT03  = 000010          DT.ESI= 000044          INHEPR= 020000          MSGSKE= 000003          SBADR  = 000102
BIT04  = 000020          DT.EVN= 000000          INHREL= 001000          MSGSMB= 000015          SBKMOD= 000000
BIT05  = 000040          DT.EXS= 000060          INHRRE= 000400          MSGSMH= 000014          SBKSEL= 010000
BIT06  = 000100          DT.FCH= 000037          INIT   = 000030          MSGSMS= 000016          SC.ADR= 000006
BIT07  = 000200          DT.FCN= 000036          INTR   = 000120          MSGSTD= 000000          SC.ALC= 000014
BIT08  = 000400          DT.HMX= 000104          IOMOD  = 100000          MSGSYS= 000012          SC.APC= 000016
BIT09  = 001000          DT.KBE= 000024          IOMODP= 102000          MSGVEC= 000020          SC.CKL= 000002
BIT1   = 000002          DT.KBP= 000026          IOMODR= 112000          NBKMOD= 001000          SC.CKP= 000004
BIT10  = 002000          DT.KBR= 000022          IOMODX= 110000          NCPUOP= 000020          SC.CLO= 000000
BIT11  = 004000          DT.KBU= 000030          JACK   = 035060          NDAPTY= 000002          SC.HLD= 000010
BIT12  = 010000          DT.MLS= 000032          KIPAR0= 172340          NULL   = 000000          SC.SCA= 000012
BIT13  = 020000          DT.MTI= 000110          KIPAR1= 172342          OWEN   = 024020          SENDLS= 177777
BIT14  = 040000          DT.OFF= 000070          KIPAR2= 172344          PAERR  = 000010          SOFCNT= 000042
BIT15  = 100000          DT.PAS= 000074          KIPAR3= 172346          PARPRE= 002000          SOFPAS= 000046
BIT2   = 000004          DT.PC  = 000002          KIPAR4= 172350          PARSTA= 000100          SPACE  = 000040
BIT3   = 000010          DT.PFL= 000062          KIPAR5= 172352          PASCNT= 000034          SPOINT= 000032
BIT4   = 000020          DT.PSW= 000004          KIPAR6= 172354          PDPLSI= 020000          SPVALU= 002200
BIT5   = 000040          DT.PTA= 000064          KIPAR7= 172356          PDP60  = 004000          SR0    = 177572
BIT6   = 000100          DT.RCS= 000102          KIPDR0= 172300          PDP70  = 010000          SR1    = 177574
BIT7   = 000200          DT.REL= 000040          KIPDR1= 172302          PRI0   = 000000          SR2    = 177576
BIT8   = 000400          DT.SCT= 000066          KIPDR2= 172304          PRI1   = 000040          SR3    = 172516
BIT9   = 001000          DT.SMX= 000106          KIPDR3= 172306          PRI4   = 000200          STAT   = 000026
BKDEF  = 000002          DT.SP  = 000006          KIPDR4= 172310          PRI5   = 000240          STATBI= 064757
BKMOD  = 000020          DT.SSI= 000046          KIPDR5= 172312          PRI6   = 000300          STAT1  = 000027
BKMODE= 040000          DT.ST0= 000010          KIPDR6= 172314          PRI7   = 000340          SUSPND= 000001
BKSLSH= 000134          DT.ST1= 000012          KIPDR7= 172316          PR0    = 000000          SVR0   = 000062
CAPRES= 000004          DT.SWR= 000056          KTERRO= 000040          PR4    = 000200          SVR1   = 000064
CASTAT= 000004          DT.SYP= 000072          KTPRES= 000400          PR5    = 000240          SVR2   = 000066
CDERCT= 000146          DT.WBU= 000050          KTSTAT= 000020          PR6    = 000300          SVR3   = 000070
CDWDCT= 000144          DT.WHL= 000054          KTXTND= 040000          PR7    = 000340          SVR4   = 000072
CKTIM  = 100000          DT.WLL= 000052          LF     = 000012          PS     = 177776          SVR5   = 000074
CLKPRE= 000001          DVID1  = 000014          LPSTAT= 000001          PSW    = 177776          SVR6   = 000076
CLREOP  000000RG        ECCMEM= 000100          MAPSTA= 000200          RANNUM= 000054          SYSCNT= 000052
CONFIG= 000056          ECCSTA= 000010          MED    = 076600          RBUFEA= 000130          SYSERR= 000100
CQOVF  = 000001          ENBEOP= 010000          MEMPAS= 040000          RBUFPA= 000126          TABL   = 000000
CR     = 000015          ENBNUL= 000001          MODEXH= 004000          RBUFSZ= 000132          TMPIO  = 000002
CSRA   = 000100          ENDLST= 000000          MODHOL= 002000          RBUFVA= 000124          TQOVF  = 000002
```

```
UIPAR0= 177640          WBUFEA= 000136          $F$DO  = 000340          $LOCTA= 177777          $$ERFL= 000000
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000340
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
UIPAR3= 177646          WBUFSZ= 000142          $F$IF  = 000110          $NESTL= 177777          $$LOC = 000020R
UIPAR4= 177650          WDFR   = 000116          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
UIPAR5= 177652          WDTO   = 000114          $F$LOO= 000200          $NSK1 = 000120          $$REG = 177777
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $SAVLE= 177777          $$RETU= 000000
UIPAR7= 177656          WTWHMI= 000222          $F$NO  = 000403          $SSK0 = 050003          $$RTN1= 050000
UIPDR0= 177600          XFLAG  = 000005          $F$OR  = 000320          $TAGLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XOFF   = 000023          $F$RTI= 000350          $TAGNU= 050004          $$SRC = 000000
UIPDR2= 177604          XON    = 000021          $F$RTN= 000300          $TEMP  = 000300          $$TGSV= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TSK0  = 050002          $$TGS1= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $TSK1  = 050003          $$TGS2= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $$ARGC= 000002          $$TO   = 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$BYTE= 000403          $$$TAG= 050000
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$CASE= 000000          .      = 000042R
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$DST  = 000000
WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$ELOC= 000000

. ABS.  000000    000
        000042    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CLREOP,DSKZ:CLREOP=SPMAC/ML,EQUATE,CLREOP
 RUN-TIME: 11  1  .4 SECONDS
 RUN-TIME RATIO: 31/13=2.2
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE CMDCPY - COMMAND COPY
                        .IDENT /V0.0/
                        ;++
                        ; MODULE NAME:
                        ;       CMDCPY
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       COPIES A COMMAND FROM THE INPUT BUFFER
                        ;       TO THE DECODE BUFFER, EXCLUDING RUBBED OUT CHARACTERS
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS, DECODE BUFFER ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.KBUF
                        ;
                        ; OUTPUTS:
                        ;       ABORT FLAG
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       OV.HIKB
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CMDCPY IN <A,B> OUT <C>
                        ;               A=DATA TABLE ADDRESS
                        ;               B=DECODE BUFFER ADDRESS
                        ;               C=ABORT FLAG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES


                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CMDCPY                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                .GLOBL  OV.HIKB         ;KB BUF HIGH ADDR
                                ;
                                ;****************
                                ; LOCAL STORAGE:
                                ;
000000' 000000G                 CM.KSIZ:        OV.HIKB                 ;KB HIGH LIMIT
                                ;
```

```
                                .SBTTL CMDCPY ROUTINE
000002'                         ROUTINE CMDCPY <TABL,DECBUF,ABORT>
000002'                                                         CMDCPY:

                                ;+
                                ;SAVE REGISTERS
                                ;-

000002'                         CALL SAVREG
000002' 004767  000000G                                                 JSR     PC,SAVREG

                                ;+
                                ;SET R0 TO START OF DTABLE
                                ;-

000006'                         LET R0 := TABL(R5)
000006' 016500  000000                                                  MOV     TABL(R5),R0

                                ;+
                                ;CLEAR THE ABORT FLAG
                                ;-

000012'                         LET ABORT(R5) := #0
000012' 005065  000004                                                  CLR     ABORT(R5)

                                ;+
                                ;INITIALIZE INPUT BUFFER POINTER
                                ;-

000016'                         LET R1 := DT.KBUF(R0)
000016' 016001  000030                                                  MOV     DT.KBUF(R0),R1

                                ;+
                                ;INITALIZE DECODE BUFFER POINTER
                                ;-

000022'                         LET R2 := DECBUF(R5)
000022' 016502  000002                                                  MOV     DECBUF(R5),R2

                                ;+
                                ;IF FIRST CHARACTER IS <CR> OR <RUBOUT>, SET
                                ;ABORT FLAG
                                ;-

000026'                         IFB (R1) EQ #CR ORB (R1) EQ #RUBOUT THEN
000026' 121127  000015                                                  CMPB    (R1),#CR
000032' 001403                                                          BEQ     50002$
000034' 121127  000177                                                  CMPB    (R1),#RUBOUT
000040' 001004                                                          BNE     50003$
000042'                                                         50002$:

000042'                                 LET ABORT(R5) := #1
000042' 012765  000001  000004                                          MOV     #1,ABORT(R5)

000050'                         ELSE
000050' 000441                                                          BR      50004$
```

```
(3)  000052'                                                                    50003$:
627
628                                 ;+
629                                 ;ELSE GO THROUGH INPUT STRING UNTIL END
630                                 ;-
631
632  000052'                                WHILE R1 LO CM.KSIZ DO
(4)  000052'                                                                    50005$:
(6)  000052' 020167  177722                                                             CMP     R1,CM.KSIZ
(9)  000056' 103026                                                                     BHIS    50006$
633
634                                 ;+
635                                 ;IF CHARACTER IS ^U, SET ABORT FLAG, RESTORE REGISTERS, AND RETURN
636                                 ;-
637
638  000060'                                    IFB (R1) EQ #CTRLU THEN
(6)  000060' 121127  000025                                                             CMPB    (R1),#CTRLU
(9)  000064' 001005                                                                     BNE     50007$
639  000066'                                           LET ABORT(R5) := #1
(4)  000066' 012765  000001  000004                                                     MOV     #1,ABORT(R5)
640  000074'                                           INLINE <BR      1$>
(2)  000074' 000427                                                                     BR      1$
641  000076'                                    ELSE
(4)  000076' 000415                                                                     BR      50010$
(3)  000100'                                                                    50007$:
642
643                                 ;+
644                                 ;ELSE IF CHARACTER IS <RUBOUT>, LOOK FOR NEXT RUBOUT AND
645                                 ;BACK UP DECODE BUFFER POINTER AS FAR AS NECESSARY
646                                 ;-
647
648  000100'                                    IFB (R1) EQ #RUBOUT THEN
(6)  000100' 121127  000177                                                             CMPB    (R1),#RUBOUT
(9)  000104' 001010                                                                     BNE     50011$
649  000106'                                           LET R1 := R1 + #1
(6)  000106' 005201                                                                     INC     R1
650  000110'                                           WHILEB (R1) NE #RUBOUT DO
(4)  000110'                                                                    50012$:
(6)  000110' 121127  000177                                                             CMPB    (R1),#RUBOUT
(9)  000114' 001403                                                                     BEQ     50013$
651  000116'                                                 LET R2 := R2 - #1
(6)  000116' 005302                                                                     DEC     R2
652  000120'                                                 LET R1 := R1 + #1
(6)  000120' 005201                                                                     INC     R1
653  000122'                                           ENDDO
(4)  000122' 000772                                                                     BR      50012$
(3)  000124'                                                                    50013$:
654
655  000124'                                    ELSE
(4)  000124' 000401                                                                     BR      50014$
(3)  000126'                                                                    50011$:
656
657                                 ;+
658                                 ;ELSE COPY CHARACTER FROM INPUT BUFFER TO DECODE BUFFER AND
659                                 ;INCREMENT DECODE BUFFER POINTER
660                                 ;-
```

```
                                                        LET (R2)+ :B= (R1)
000126'                                                                         MOVB    (R1),(R2)+
(4) 000126' 111122
                                                    ENDIF
000130'                                                                 50014$:
(4) 000130'

                        ;+
                        ;GET NEXT INPUT BUFFER CHARACTER
                        ;-

                                    LET R1 := R1 + #1
000130'                                                                         INC     R1
(6) 000130' 005201

                                ENDIF
000132'                                                                 50010$:
(4) 000132'
                            ENDDO
000132'                                                                         BR      50005$
(4) 000132' 000747
000134'                                                                 50006$:
(3) 000134'


                        ;+
                        ;IF THE ENTIRE INPUT STRING WAS RUBBED OUT, SET THE ABORT FLAG
                        ;(FIRST, BACK UP THE DECODE BUFFER POINTER TO ALLOW
                        ;FOR THE CR AND LF, THEN SEE IF THE POINTER IS AT THE
                        ;BEGINNING OF THE BUFFER.)
                        ;-

                            LET R2 := R2 - #2
C00134'                                                                         SUB     #2,R2
(6) 000134' 162702  000002
                            IF R2 LOS DECBUF(R5) THEN
000140'                                                                         CMP     R2,DECBUF(R5)
(6) 000140' 020265  000002                                                      BHI     50015$
(9) 000144' 101003
                                    LET ABORT(R5) := #1
000146'                                                                         MOV     #1,ABORT(R5)
(4) 000146' 012765  000001  000004
                            ENDIF
000154'                                                                 50015$:
(4) 000154'


                        ENDIF
000154'                                                                 50004$:
(4) 000154'

                        ;+
                        ;RESTORE REGISTERS AND RETURN
                        ;-

                        INLINE <1$:>
000154'                                                                         1$:
(2) 000154'
                        CALL RESREG
000154'                                                                         JSR     PC,RESREG
(3) 000154' 004767  000000G

                        ENDRTN
000160'                                                                 50000$:
(3) 000160'                                                             50001$:
(3) 000160'                                                                     RTS     PC
(2) 000160' 000207
            000001      .END
```

```
ABORT = 000004
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CMDCPY  000002RG
CM.KSI  000000R
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DECBUF= 000002
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OV.HIK= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          WBSTAT= 000040          $F$GOO= 000400          $LSTTA= 000001          $$BYTE= 000403
TABL  = 000000          WBUFEA= 000136          $F$IF = 000110          $NESTL= 177777          $$CASE= 000000
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK0 = 000300          $$DST = 000000
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK1 = 000110          $$ELOC= 000402
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $NSK2 = 000110          $$ERFL= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $NSK3 = 000110          $$FLAG= 000001
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $NSK4 = 000110          $$FROM= 000000
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $NSK5 = 000120          $$LOC = 000144R
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $SAVLE= 177777          $$LOCN= 000000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $SSK0 = 050006          $$REG = 177777
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $TAGLE= 177777          $$RETU= 000000
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $TAGNU= 050016          $$RTN1= 050000
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $TEMP = 000300          $$RTN2= 050001
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $TSK0 = 050004          $$SRC = 000000
UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $TSK1 = 050015          $$TGSV= 000000
UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $TSK2 = 050006          $$TGS1= 000000
UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $TSK3 = 050010          $$TGS2= 000000
UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $TSK4 = 050014          $$TO  = 000000
UIPDR6= 177614          $F$DEC= 000220          $ISK2 = 000001          $TSK5 = 050012          $$$TAG= 050000
UIPDR7= 177616          $F$DO = 000340          $LOCTA= 177777          $TSK6 = 050013          .     = 000162R
WASADR= 000104          $F$FAL= 000405          $LSTIN= 000001          $$ARGC= 000006

. ABS.  000000    000
        000162    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDCPY,DSKZ:CMDCPY=SPMAC/ML,EQUATE,CMDCPY
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 39/19=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE CMDDCD - DECODE COMMAND
                                .IDENT /V0.0/

                                ;++
                                ;
                                ; MODULE NAME:
                                ;       CMDDCD
                                ;
                                ; FUNTIONAL DESCRIPTION:
                                ;       DECODES THE CONTENTS OF THE DECODE BUFFER AND
                                ;       RETURNS A COMMAND CODE
                                ;
                                ; INPUTS:
                                ;       DECODE BUFFER ADDRESS
                                ;
                                ; IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; OUTPUTS:
                                ;       COMMAND CODE, UPDATED DECODE BUFFER POINTER
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE MODULES CALLED:
                                ;       NONE
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL CMDDCD IN <A> OUT <B,C>
                                ;               A=DECODE BUFFER ADDRESS
                                ;               B=UPDATED DECODE BUFFER POINTER
                                ;               C=COMMAND CODE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;               EDIT            BY              DATE            REASON
                                ;
                                ;--
                                ;
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES


                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CMDDCD                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  CM.TBL                          ;VALID COMMAND TABLE
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL CMDDCD ROUTINE
                                .EVEN

000000'                         ROUTINE CMDDCD <DECBUF,DBFPTR,CDCODE>
(2) 000000'                                                                     CMDDCD:


                                ;+
                                ;SAVE REGISTERS
                                ;-

000000'                         PUSH R1,R2
(2) 000000' 010146                                                                      MOV     R1,-(SP)
(3) 000002' 010246                                                                      MOV     R2,-(SP)

                                ;+
                                ;LET R1 BE THE POINTER FOR THE VALID COMMAND TABLE
                                ;-

000004'                         LET R1 := #CM.TBL
(4) 000004' 012701  000000G                                                             MOV     #CM.TBL,R1

                                ;+
                                ;SET THE COMMAND CODE TO 0
                                ;-

000010'                         LET CDCODE(R5) := #0
(4) 000010' 005065  000004                                                              CLR     CDCODE(R5)


                                ;+
                                ;GO THROUGH THE VALID COMMAND TABLE UNTIL THE END
                                ;-

000014'                         REPEAT
(3) 000014'                                                                     50002$:

                                ;+
                                ;LET R2 BE THE DECODE BUFFER POINTER
                                ;-

000014'                                 LET R2 := DECBUF(R5)
(4) 000014' 016502  000000                                                              MOV     DECBUF(R5),R2

                                ;+
                                ;FIND THE FIRST CHARACTER IN THE DECODE BUFFER THAT
                                ;IS NOT A SPACE
                                ;-

000020'                                 WHILEB (R2) EQ #SPACE DO
(4) 000020'                                                                     50003$:
(6) 000020' 121227  000040                                                              CMPB    (R2),#SPACE
(9) 000024' 001002                                                                      BNE     50004$
000026'                                         LET R2 := R2 + #1
(6) 000026' 005202                                                                      INC     R2
000030'                                 ENDDO
```

```
(4)  000030' 000773                                                                    BR      50003$
(3)  000032'                                                                50004$:
626
627                                 ;+
628                                 ;COMPARE CHARACTERS OF VALID COMMAND TO CHARACTERS IN
629                                 ;DECODE BUFFER UNTIL A MISMATCH IS FOUND
630                                 ;-
631
632  000032'                                WHILEB (R1) EQ (R2) DO
(4)  000032'                                                                50005$:
(6)  000032' 121112                                                             CMPB    (R1),(R2)
(9)  000034' 001003                                                             BNE     50006$
633  000036'                                        LET R1 := R1 + #1
(6)  000036' 005201                                                             INC     R1
634  000040'                                        LET R2 := R2 + #1
(6)  000040' 005202                                                             INC     R2
635  000042'                                ENDDO
(4)  000042' 000773                                                             BR      50005$
(3)  000044'                                                                50006$:
636
637                                 ;+
638                                 ;IF AT THE END OF A VALID COMMAND, THEN CHECK IF AT
639                                 ;END OF DECODE COMMAND.  IF YES,
640                                 ;SAVE DECODE BUFFER POINTER, SAVE REGISTERS, AND RETURN
641                                 ;-
642
643  000044'                                IFB (R1) EQ #0 THEN
(6)  000044' 105711                                                             TSTB    (R1)
(9)  000046' 001007                                                             BNE     50007$
644
645  000050'                                        IFB (R2) EQ #SPACE ORB (R2) EQ #CR THEN
(6)  000050' 121227  000040                                                     CMPB    (R2),#SPACE
(8)  000054' 001403                                                             BEQ     50010$
(6)  000056' 121227  000015                                                     CMPB    (R2),#CR
(9)  000062' 001001                                                             BNE     50011$
(6)  000064'                                                                50010$:
646  000064'                                                INLINE <BR      1$>
(2)  000064' 000415                                                             BR      1$
647  000066'                                        ENDIF
(4)  000066'                                                                50011$:
648
649  000066'                                ENDIF
(4)  000066'                                                                50007$:
650
651                                 ;+
652                                 ;ELSE THE COMMANDS DID NOT MATCH, SO INCREMENT THE COMMAND CODE
653                                 ;AND GET THE NEXT VALID COMMAND FROM THE TABLE
654                                 ;-
655
656  000066'                                WHILEB (R1) NE #0 DO
(4)  000066'                                                                50012$:
(6)  000066' 105711                                                             TSTB    (R1)
(9)  000070' 001402                                                             BEQ     50013$
657  000072'                                        LET R1 := R1 + #1
(6)  000072' 005201                                                             INC     R1
658  000074'                                ENDDO
```

```
(4)  000074' 000774                                                                  BR      50012$
(3)  000076'                                                                 50013$:
659
660
661  000076'                        LET CDCODE(R5) := CDCODE(R5) + #1
(6)  000076' 005265  000004                                                          INC     CDCODE(R5)
662
663  000102'                        LET R1 := R1 + #1
(6)  000102' 005201                                                                  INC     R1
664
665                                 ;+
666                                 ;END OF TABLE YET?
667                                 ;-
668
669  000104'                        UNTILB (R1) EQ #-1
(3)  000104' 121127  177777                                                          CMPB    (R1),#-1
(6)  000110' 001341                                                                  BNE     50002$
670
671                                 ;+
672                                 ;YES - ILLEGAL COMMAND
673                                 ;SET COMMAND CODE TO -1
674                                 ;-
675
676  000112'                        LET CDCODE(R5) := #-1
(4)  000112' 012765  177777  000004                                                  MOV     #-1,CDCODE(R5)
677
678                                 ;+
679                                 ;SAVE THE DECODE BUFFER POINTER
680                                 ;-
681
682  000120'                        INLINE <1$:>
(2)  000120'                                                                         1$:
683  000120'                        LET DBFPTR(R5) := R2
(4)  000120' 010265  000002                                                          MOV     R2,DBFPTR(R5)
684
685
686                                 ;+
687                                 ;SAVE REGISTERS AND RETURN
688                                 ;-
689
690  000124'                        POP R2,R1
(2)  000124' 012602                                                                  MOV     (SP)+,R2
(3)  000126' 012601                                                                  MOV     (SP)+,R1
691
692  000130'                        ENDRTN
(3)  000130'                                                                 50000$:
(3)  000130'                                                                 50001$:
(2)  000130' 000207                                                                  RTS     PC
693
694                                 ;+
695                                 ;VALID COMMAND TABLE
696                                 ;-
697
698
699          000001                 .END
```

SEQ 0340

```
ACSR  = 000102        CR    = 000015        ECCSTA= 000010        MED   = 076600        RBUFEA= 000130
ACTBIT= 004000        CSRA  = 000100        ENBEOP= 010000        MEMPAS= 040000        RBUFPA= 000126
ADDR22= 001000        CSRC  = 000102        ENBNUL= 000001        MODEXH= 004000        RBUFSZ= 000132
ADR   = 000006        CTRLC = 000003        ENDLST= 000000        MODHOL= 002000        RBUFVA= 000124
APTFER= 000004        CTRLO = 000017        EOPBIT= 000001        MODSEL= 001000        RDSERV= 000101
APTPRE= 000200        CTRLU = 000025        ERRTYP= 000106        MSGCKD= 000010        RDWHMI= 000022
ASB   = 000106        DBFPTR= 000002        EVNTBE= 000200        MSGCKS= 000011        RELERR= 000020
ASSEMB= 000010        DCEVNT= 000011        EVNTHD= 000200        MSGDER= 000005        RELMOD= 020000
ASTAT = 000104        DECBUF= 000000        EVNTKT= 000203        MSGDRP= 000017        RELTIM= 010000
AUTO  = 000010        DEFRTN= 000400        EVNTPE= 000202        MSGECH= 177777        RES1  = 000056
AUTOST= 020000        DIAGMC= 000000        EVNTRE= 000201        MSGEOP= 000013        RES2  = 000060
AWAS  = 000110        DROPMO= 100000        FATERR= 100000        MSGHDR= 000004        RICHAR= 031060
BIT0  = 000001        DSEVNT= 000014        HRDCNT= 000044        MSGHNG= 000022        RPTDAT= 002000
BIT00 = 000001        DT.ADD= 000042        HRDPAS= 000050        MSGHRD= 000007        RSTRT = 000112
BIT01 = 000002        DT.AP = 000100        ICONT = 000036        MSGMAP= 000021        RUBOUT= 000177
BIT02 = 000004        DT.APK= 000076        ICOUNT= 000040        MSGNUL= 177775        RUNMOD= 100000
BIT03 = 000010        DT.BLS= 000034        IDNUM = 000122        MSGPOP= 000002        R5VALU= 001740
BIT04 = 000020        DT.CF0= 000014        IE    = 000100        MSGPRM= 177776        SAM   = 075464
BIT05 = 000040        DT.CF1= 000016        INDPAR= 000040        MSGRES= 000001        SBADR = 000102
BIT06 = 000100        DT.ERR= 000020        INHDRP= 040000        MSGSFT= 000006        SBKMOD= 000000
BIT07 = 000200        DT.ESI= 000044        INHEPR= 020000        MSGSKE= 000003        SBKSEL= 010000
BIT08 = 000400        DT.EVN= 000000        INHREL= 001000        MSGSMB= 000015        SC.ADR= 000006
BIT09 = 001000        DT.EXS= 000060        INHRRE= 000400        MSGSMH= 000014        SC.ALC= 000014
BIT1  = 000002        DT.FCH= 000037        INIT  = 000030        MSGSMS= 000016        SC.APC= 000016
BIT10 = 002000        DT.FCN= 000036        INTR  = 000120        MSGSTD= 000000        SC.CKL= 000002
BIT11 = 004000        DT.HMX= 000104        IOMOD = 100000        MSGSYS= 000012        SC.CKP= 000004
BIT12 = 010000        DT.KBE= 000024        IOMODP= 102000        MSGVEC= 000020        SC.CLO= 000000
BIT13 = 020000        DT.KBP= 000026        IOMODR= 112000        NBKMOD= 001000        SC.HLD= 000010
BIT14 = 040000        DT.KBR= 000022        IOMODX= 110000        NCPUOP= 000020        SC.SCA= 000012
BIT15 = 100000        DT.KBU= 000030        JACK  = 035060        NOAPTY= 000002        SENDLS= 177777
BIT2  = 000004        DT.MLS= 000032        KIPAR0= 172340        NULL  = 000000        SOFCNT= 000042
BIT3  = 000010        DT.MTI= 000110        KIPAR1= 172342        OWEN  = 024020        SOFPAS= 000046
BIT4  = 000020        DT.OFF= 000070        KIPAR2= 172344        PAERR = 000010        SPACE = 000040
BIT5  = 000040        DT.PAS= 000074        KIPAR3= 172346        PARPRE= 002000        SPOINT= 000032
BIT6  = 000100        DT.PC = 000002        KIPAR4= 172350        PARSTA= 000100        SPVALU= 002200
BIT7  = 000200        DT.PFL= 000062        KIPAR5= 172352        PASCNT= 000034        SR0   = 177572
BIT8  = 000400        DT.PSW= 000004        KIPAR6= 172354        PDPLSI= 020000        SR1   = 177574
BIT9  = 001000        DT.PTA= 000064        KIPAR7= 172356        PDP60 = 004000        SR2   = 177576
BKDEF = 000002        DT.RCS= 000102        KIPDR0= 172300        PDP70 = 010000        SR3   = 172516
BKMOD = 000020        DT.REL= 000040        KIPDR1= 172302        PRI0  = 000000        STAT  = 000026
BKMODE= 040000        DT.SCT= 000066        KIPDR2= 172304        PRI1  = 000040        STATBI= 064757
BKSLSH= 000134        DT.SMX= 000106        KIPDR3= 172306        PRI4  = 000200        STAT1 = 000027
CAPRES= 000004        DT.SP = 000006        KIPDR4= 172310        PRI5  = 000240        SUSPND= 000001
CASTAT= 000004        DT.SSI= 000046        KIPDR5= 172312        PRI6  = 000300        SVR0  = 000062
CDCODE= 000004        DT.STO= 000010        KIPDR6= 172314        PRI7  = 000340        SVR1  = 000064
CDERCT= 000146        DT.ST1= 000012        KIPDR7= 172316        PR0   = 000000        SVR2  = 000066
CDWDCT= 000144        DT.SWR= 000056        KTERRO= 000040        PR4   = 000200        SVR3  = 000070
CKTIM = 100000        DT.SYP= 000072        KTPRES= 000400        PR5   = 000240        SVR4  = 000072
CLKPRE= 000001        DT.WBU= 000050        KTSTAT= 000020        PR6   = 000300        SVR5  = 000074
CMDDCD  000000RG      DT.WHL= 000054        KTXTND= 040000        PR7   = 000340        SVR6  = 000076
CM.TBL= ****** G      DT.WLL= 000052        LF    = 000012        PS    = 177776        SYSCNT= 000052
CONFIG= 000056        DVID1 = 000014        LPSTAT= 000001        PSW   = 177776        SYSERR= 000100
CQOVF = 000001        ECCMEM= 000100        MAPSTA= 000200        RANNUM= 000054        TMPIO = 000002
```

```
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LOCTA= 177777          $$DST = 000000
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LSTIN= 000001          $$ELOC= 000402
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $LSTTA= 000001          $$ERFL= 000000
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NESTL= 177777          $$FLAG= 000340
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK0 = 000300          $$FROM= 000000
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK1 = 000130          $$LOC = 000110R
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $NSK2 = 000120          $$LOCN= 000000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $NSK3 = 000110          $$REG = 177777
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $SAVLE= 177777          $$RETU= 000000
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $SSK0 = 050013          $$RTN1= 050000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TAGLE= 177777          $$RTN2= 050001
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TAGNU= 050014          $$SRC = 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TEMP = 000300          $$TGSV= 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $TSK0 = 050002          $$TGS1= 000000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $TSK1 = 050012          $$TGS2= 000000
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $TSK2 = 050013          $$TO  = 000000
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$ARGC= 000006          $$$TAG= 050000
WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$BYTE= 000402          .     = 000132R
WBSTAT= 000040          $F$DO = 000340          $ISK1 = 000001          $$CASE= 000000

. ABS.  000000     000
        000132     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDDCD,DSKZ:CMDDCD=SPMAC/ML,EQUATE,CMDDCD
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 37/18=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE CMDPRC - PROCESS KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CMDPRC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTROLS THE DECODING AND PROCESSING OF KEYBOARD COMMANDS
                        ;       BY IDENTIFYING THE COMMAND AND THEN CALLING THE PROPER
                        ;       SERVICE ROUTINE
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       CMDCPY, CMDDCD, CMDRST, ALL COMMAND SERVICE ROUTINES
                        ;       SAVRES
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CMDPRC IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES


                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                    000001      $LSTIN=1
                    000001      $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CMDPRC                  ;MODULE ENTRY POINT
                                .GLOBL  CM.BADNAME              ; - MESSAGES
                                .GLOBL  CM.ARG                  ;
                                .GLOBL  CM.NUM                  ;
                                .GLOBL  CM.ODD                  ;
                                .GLOBL  CM.RUN                  ;
                                .GLOBL  CM.ADR                  ;
                                .GLOBL  CM.NBAD                 ;
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  CMDCPY                  ;COMMAND COPY MODULE ENTRY POINT
                                .GLOBL  CMDDCD                  ;COMMAND DECODE MODULE ENTRY POINT
                                .GLOBL  CMDRST                  ;COMMAND RESET MODULE ENTRY POINT
                                .GLOBL  CM.DIS                  ;COMMAND DISPATCH TABLE
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ;
                                ;****************
                                ;
                                ; LOCAL EQUATES:
                                ;
                    000122      CM.BUFSIZ=^D<82>                ;BUFFER SIZE 82 (10) BYTES
                                                                ;MUST ALWAYS=KEYBD. BUF SIZE


                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  CM.ABORT:       .WORD 0         ;ABORT FLAG
000002' 000000                  CM.CDCODE:      .WORD 0         ;COMMAND CODE
000004' 000000                  CM.DBFPTR:      .WORD 0         ;DECODE BUFFER POINTER
000006' 047111  040526  044514  CM.BADNAME:     .ASCIZ /INVALID MODULE NAME%/           ; - MESSAGES
000014' 020104  047515  052504
```

```
                000022' 042514  047040  046501
                000030' 022505     000
     607        000033'    111  053116  046101  CM.ARG:         .ASCIZ /INVALID OR MISSING ARGUMENT%/   ;
                000040' 042111  047440  020122
                000046' 044515  051523  047111
                000054' 020107  051101  052507
                000062' 042515  052116  000045
     608        000070' 052516  041115  051105  CM.NUM:         .ASCIZ /NUMBER TOO LARGE%/              ;
                000076' 052040  047517  046040
                000104' 051101  042507  000045
     609        000112' 052515  052123  041040  CM.ODD:         .ASCIZ /MUST BE EVEN ADDRESS%/          ;
                000120' 020105  053105  047105
                000126' 040440  042104  042522
                000134' 051523  000045
     610        000140' 047111  040526  044514  CM.RUN:         .ASCIZ /INVALID COMMAND IN RUN MODE%/   ;
                000146' 020104  047503  046515
                000154' 047101  020104  047111
                000162' 051040  047125  046440
                000170' 042117  022505     000
     611        000175'    111  053116  046101  CM.ADR:         .ASCIZ /INVALID ADDRESS%/               ;
                000202' 042111  040440  042104
                000210' 042522  051523  000045
     612        000216' 047516  020124  047101  CM.NBAD:        .ASCIZ /NOT AN OCTAL NUMBER%/           ;
                000224' 047440  052103  046101
                000232' 047040  046525  042502
                000240' 022522     000
     613        000243'    111  053116  046101  CM.INVCMD:      .ASCII /INVALID COMMAND - /     ; ***NOTE - ASCII INSTEAD OF ASCIZ
                000250' 042111  041440  046517
                000256' 040515  042116  026440
                000264'    040
     614        000265' 000122                  CM.DECBUF:      .BLKB CM.BUFSIZ         ;KEYBOARD COMMAND DECODE BUFFER
     615        000407'    000                  .BYTE   0               ;GUARANTEE 0 TERMINATION
     616        000410'    000                  .BYTE   0
     617                000412'                 .EVEN
     618                                        ;
     619                                        ;****************
     620                                        ;
```

```
                        .SBTTL CMDPRC ROUTINE
000412'                 ROUTINE CMDPRC <TABL>
(2) 000412'                                                     CMDPRC:

                        ;+
                        ;SAVE REGISTERS
                        ;-
000412'                 CALL SAVREG
(3) 000412' 004767 000000G                                              JSR     PC,SAVREG

                        ;+
                        ;SET R0 TO START OF DTABLE
                        ;-
000416'                 LET R0 := TABL(R5)
(4) 000416' 016500 000000                                               MOV     TABL(R5),R0

                        ;+
                        ;CLEAR THE DECODE BUFFER AND LOCAL STORAGE WORDS
                        ;-

000422'                 LET R1 := #CM.DECBUF
(4) 000422' 012701 000265'                                              MOV     #CM.DECBUF,R1
000426'                 LET R2 := R1 + #CM.BUFSIZ
(4) 000426' 010102                                                      MOV     R1,R2
(6) 000430' 062702 000122                                               ADD     #CM.BUFSIZ,R2
000434'                 WHILEB R1 NE R2 DO
(4) 000434'                                                     50002$:
(6) 000434' 120102                                                      CMPB    R1,R2
(9) 000436' 001402                                                      BEQ     50003$
000440'                         LET (R1)+ :B= #0
(4) 000440' 105021                                                      CLRB    (R1)+
000442'                 ENDDO
(4) 000442' 000774                                                      BR      50002$
(3) 000444'                                                     50003$:
000444'                 LET CM.ABORT := #0
(4) 000444' 005067 177330                                               CLR     CM.ABORT
000450'                 LET CM.CDCODE := #0
(4) 000450' 005067 177326                                               CLR     CM.CDCODE
000454'                 LET CM.DBFPTR := #0
(4) 000454' 005067 177324                                               CLR     CM.DBFPTR

                        ;+
                        ; COPY INPUT BUFFER INTO DECODE BUFFER
                        ;-

000460'                 CALL CMDCPY IN <R0,#CM.DECBUF> OUT <CM.ABORT>
(4) 000460' 162705 000002                                               SUB     #1*2,R5
(3) 000464' 010546                                                      MOV     R5,-(SP)
(5) 000466' 012745 000265'                                              MOV     #CM.DECBUF,-(R5)
(4) 000472' 010045                                                      MOV     R0,-(R5)
(3) 000474' 004767 000000G                                              JSR     PC,CMDCPY
(3) 000500' 012605                                                      MOV     (SP)+,R5
(4) 000502' 012567 177272                                               MOV     (R5)+,CM.ABORT

                        ;+
```

```
                                ; IF ABORT FLAG NOT SET, DECODE COMMAND
                                ;-

000506'                         IF CM.ABORT NE #1 THEN
(6) 000506' 026727 177266 000001                                        CMP     CM.ABORT,#1
(9) 000514' 001436                                                      BEQ     50004$


                                        CALL CMDDCD IN <#CM.DECBUF> OUT <CM.DBFPTR,CM.CDCODE>
(4) 000516' 162705 000004                                               SUB     #2*2,R5
(3) 000522' 010546                                                      MOV     R5,-(SP)
(4) 000524' 012745 000265'                                              MOV     #CM.DECBUF,-(R5)
(3) 000530' 004767 000000G                                              JSR     PC,CMDDCD
(3) 000534' 012605                                                      MOV     (SP)+,R5
(4) 000536' 012567 177242                                               MOV     (R5)+,CM.DBFPTR
(4) 000542' 012567 177234                                               MOV     (R5)+,CM.CDCODE

                                ;+
                                ; IF THERE WAS AN ERROR (CM.CDCODE=-1), THEN OUTPUT THE DECODE BUFFER
                                ;ALONG WITH AN ERROR MESSAGE.  IF THE COMMAND
                                ;WAS VALID, DISPATCH CONTROL TO THE PROPER ROUTINE.
                                ;-

000546'                                 IF CM.CDCODE EQ #-1 THEN
(6) 000546' 026727 177230 177777                                        CMP     CM.CDCODE,#-1
(9) 000554' 001004                                                      BNE     50005$

000556'                                         LET DT.KBRSP(R0) := #CM.INVCMD
(4) 000556' 012760 000243' 000022                                       MOV     #CM.INVCMD,DT.KB

000564'                                 ELSE
(4) 000564' 000412                                                      BR      50006$
(3) 000566'                                                     50005$:
000566'                                         LET R1 := CM.CDCODE SHIFT #+1
(4) 000566' 016701 177210                                               MOV     CM.CDCODE,R1
(7) 000572' 006301                                                      ASL     R1
000574'                                         CALL @CM.DIS(R1) IN <R0,CM.DBFPTR>
(3) 000574' 010546                                                      MOV     R5,-(SP)
(5) 000576' 016745 177202                                               MOV     CM.DBFPTR,-(R5)
(4) 000602' 010045                                                      MOV     R0,-(R5)
(3) 000604' 004771 000000G                                              JSR     PC,@CM.DIS(R1)
(3) 000610' 012605                                                      MOV     (SP)+,R5
000612'                                 ENDIF
(4) 000612'                                                     50006$:
000612'                         ENDIF
(4) 000612'                                                     50004$:


                                ;+
                                ; CALL COMMAND RESET MODULE WHICH WILL OUTPUT PROMPT
                                ;-

000612'                         CALL CMDRST IN <R0>
(3) 000612' 010546                                                      MOV     R5,-(SP)
```

```
   (4)  000614' 010045                                                  MOV     R0,-(R5)
   (3)  000616' 004767  000000G                                         JSR     PC,CMDRST
   (3)  000622' 012605                                                  MOV     (SP)+,R5
   688
   689                              ;+
   690                              ;RESTORE REGISTERS
   691                              ;-
   692
   693  000624'                     CALL RESREG
   (3)  000624' 004767  000000G                                         JSR     PC,RESREG
   694
   695                              ;+
   696                              ;RETURN
   697                              ;-
   698
   699  000630'                     ENDRTN
   (3)  000630'                                                 50000$:
   (3)  000630'                                                 50001$:
   (2)  000630' 000207                                                  RTS     PC
   700          000001              .END
```

```
ACSR   = 000102          CM.ADR  000175RG        DT.SCT= 000066          KIPDR2= 172304          PRI1   = 000040
ACTBIT= 004000          CM.ARG  000033RG        DT.SMX= 000106          KIPDR3= 172306          PRI4   = 000200
ADDR22= 001000          CM.BAD  000006RG        DT.SP = 000006          KIPDR4= 172310          PRI5   = 000240
ADR    = 000006          CM.BUF= 000122          DT.SSI= 000046          KIPDR5= 172312          PRI6   = 000300
APTFER= 000004          CM.CDC  000002R         DT.ST0= 000010          KIPDR6= 172314          PRI7   = 000340
APTPRE= 000200          CM.DBF  000004R         DT.ST1= 000012          KIPDR7= 172316          PR0    = 000000
ASB    = 000106          CM.DEC  000265R         DT.SWR= 000056          KTERRO= 000040          PR4    = 000200
ASSEMB= 000010          CM.DIS= ****** G        DT.SYP= 000072          KTPRES= 000400          PR5    = 000240
ASTAT  = 000104          CM.INV  000243R         DT.WBU= 000050          KTSTAT= 000020          PR6    = 000300
AUTO   = 000010          CM.NBA  000216RG        DT.WHL= 000054          KTXTND= 040000          PR7    = 000340
AUTOST= 020000          CM.NUM  000070RG        DT.WLL= 000052          LF     = 000012          PS     = 177776
AWAS   = 000110          CM.ODD  000112RG        DVID1 = 000014          LPSTAT= 000001          PSW    = 177776
BIT0   = 000001          CM.RUN  000140RG        ECCMEM= 000100          MAPSTA= 000200          RANNUM= 000054
BIT00  = 000001          CONFIG= 000056          ECCSTA= 000010          MED    = 076600          RBUFEA= 000130
BIT01  = 000002          CQOVF  = 000001         ENBEOP= 010000          MEMPAS= 040000          RBUFPA= 000126
BIT02  = 000004          CR     = 000015         ENBNUL= 000001          MODEXH= 004000          RBUFSZ= 000132
BIT03  = 000010          CSRA   = 000100         ENDLST= 000000          MODHOL= 002000          RBUFVA= 000124
BIT04  = 000020          CSRC   = 000102         EOPBIT= 000001          MODSEL= 001000          RDSERV= 000101
BIT05  = 000040          CTRLC  = 000003         ERRTYP= 000106          MSGCKD= 000010          RDWHMI= 000022
BIT06  = 000100          CTRLO  = 000017         EVNTBE= 000200          MSGCKS= 000011          RELERR= 000020
BIT07  = 000200          CTRLU  = 000025         EVNTHD= 000200          MSGDER= 000005          RELMOD= 020000
BIT08  = 000400          DCEVNT= 000011          EVNTKT= 000203          MSGDRP= 000017          RELTIM= 010000
BIT09  = 001000          DEFRTN= 000400          EVNTPE= 000202          MSGECH= 177777          RESREG= ****** G
BIT1   = 000002          DIAGMC= 000000          EVNTRE= 000201          MSGEOP= 000013          RES1   = 000056
BIT10  = 002000          DROPMO= 100000          FATERR= 100000          MSGHDR= 000004          RES2   = 000060
BIT11  = 004000          DSEVNT= 000014          HRDCNT= 000044          MSGHNG= 000022          RICHAR= 031060
BIT12  = 010000          DT.ADD= 000042          HRDPAS= 000050          MSGHRD= 000007          RPTDAT= 002000
BIT13  = 020000          DT.AP  = 000100         ICONT  = 000036         MSGMAP= 000021          RSTRT  = 000112
BIT14  = 040000          DT.APK= 000076          ICOUNT= 000040          MSGNUL= 177775          RUBOUT= 000177
BIT15  = 100000          DT.BLS= 000034          IDNUM  = 000122         MSGPOP= 000002          RUNMOD= 100000
BIT2   = 000004          DT.CF0= 000014          IE     = 000100         MSGPRM= 177776          R5VALU= 001740
BIT3   = 000010          DT.CF1= 000016          INDPAR= 000040          MSGRES= 000001          SAM    = 075464
BIT4   = 000020          DT.ERR= 000020          INHDRP= 040000          MSGSFT= 000006          SAVREG= ****** G
BIT5   = 000040          DT.ESI= 000044          INHEPR= 020000          MSGSKE= 000003          SBADR  = 000102
BIT6   = 000100          DT.EVN= 000000          INHREL= 001000          MSGSMB= 000015          SBKMOD= 000000
BIT7   = 000200          DT.EXS= 000060          INHRRE= 000400          MSGSMH= 000014          SBKSEL= 010000
BIT8   = 000400          DT.FCH= 000037          INIT   = 000030         MSGSMS= 000016          SC.ADR= 000006
BIT9   = 001000          DT.FCN= 000036          INTR   = 000120         MSGSTD= 000000          SC.ALC= 000014
BKDEF  = 000002          DT.HMX= 000104          IOMOD  = 100000         MSGSYS= 000012          SC.APC= 000016
BKMOD  = 000020          DT.KBE= 000024          IOMODP= 102000          MSGVEC= 000020          SC.CKL= 000002
BKMODE= 040000          DT.KBP= 000026          IOMODR= 112000          NBKMOD= 001000          SC.CKP= 000004
BKSLSH= 000134          DT.KBR= 000022          IOMODX= 110000          NCPUOP= 000020          SC.CLO= 000000
CAPRES= 000004          DT.KBU= 000030          JACK   = 035060         NOAPTY= 000002          SC.HLD= 000010
CASTAT= 000004          DT.MLS= 000032          KIPAR0= 172340          NULL   = 000000          SC.SCA= 000012
CDERCT= 000146          DT.MTI= 000110          KIPAR1= 172342          OWEN   = 024020          SENDLS= 177777
CDWDCT= 000144          DT.OFF= 000070          KIPAR2= 172344          PAERR  = 000010          SOFCNT= 000042
CKTIM  = 100000          DT.PAS= 000074          KIPAR3= 172346          PARPRE= 002000          SOFPAS= 000046
CLKPRE= 000001          DT.PC  = 000002         KIPAR4= 172350          PARSTA= 000100          SPACE  = 000040
CMDCPY= ****** G        DT.PFL= 000062          KIPAR5= 172352          PASCNT= 000034          SPOINT= 000032
CMDDCD= ****** G        DT.PSW= 000004          KIPAR6= 172354          PDPLSI= 020000          SPVALU= 002200
CMDPRC  000412RG        DT.PTA= 000064          KIPAR7= 172356          PDP60  = 004000          SR0    = 177572
CMDRST= ****** G        DT.RCS= 000102          KIPDR0= 172300          PDP70  = 010000          SR1    = 177574
CM.ABO  000000R         DT.REL= 000040          KIPDR1= 172302          PRI0   = 000000          SR2    = 177576
```

```
SR3    = 172516          UIPAR5= 177652          XOFF   = 000023          $F$TRU= 000404          $$BYTE= 000403
STAT   = 000026          UIPAR6= 177654          XON    = 000021          $F$UNT= 000130          $$CASE= 000000
STATBI= 064757           UIPAR7= 177656          $BGNLE= 177777           $F$WHI= 000120          $$DST = 000000
STAT1  = 000027          UIPDR0= 177600          $ERFLG= 000400           $F$YES= 000402          $$ELOC= 000402
SUSPND= 000001           UIPDR1= 177602          $F$AND= 000310           $IFLEV= 177777          $$ERFL= 000000
SVR0   = 000062          UIPDR2= 177604          $F$BAD= 000401           $ISK0 = 000001          $$FLAG= 000001
SVR1   = 000064          UIPDR3= 177606          $F$BLA= 000170           $ISK1 = 000001          $$FROM= 000000
SVR2   = 000066          UIPDR4= 177610          $F$CAS= 000150           $LOCTA= 177777          $$LOC = 000554R
SVR3   = 000070          UIPDR5= 177612          $F$DEC= 000220           $LSTIN= 000001          $$LOCN= 000000
SVR4   = 000072          UIPDR6= 177614          $F$DO  = 000340          $LSTTA= 000001          $$REG = 177777
SVR5   = 000074          UIPDR7= 177616          $F$FAL= 000405           $NESTL= 177777          $$RETU= 000000
SVR6   = 000076          WASADR= 000104          $F$GOO= 000400           $NSK0 = 000300          $$RTN1= 050000
SYSCNT= 000052           WBSTAT= 000040          $F$IF  = 000110          $NSK1 = 000110          $$RTN2= 050001
SYSERR= 000100           WBUFEA= 000136          $F$INC= 000210           $NSK2 = 000110          $$SRC = 000000
TABL   = 000000          WBUFPA= 000134          $F$LOO= 000200           $SAVLE= 177777          $$TGSV= 000000
TMPIO  = 000002          WBUFRQ= 000140          $F$NAM= 000160           $SSK0 = 050003          $$TGS1= 000000
TQOVF  = 000002          WBUFSZ= 000142          $F$NO  = 000403          $TAGLE= 177777          $$TGS2= 000000
UIPAR0= 177640           WDFR   = 000116         $F$OR  = 000320          $TAGNU= 050007          $$TO  = 000000
UIPAR1= 177642           WDTO   = 000114         $F$RTI= 000350           $TEMP  = 000300         $$$TAG= 050000
UIPAR2= 177644           WTINRE= 000352          $F$RTN= 000300           $TSK0  = 050004         .     = 000632R
UIPAR3= 177646           WTWHMI= 000222          $F$SEL= 000140           $TSK1  = 050006
UIPAR4= 177650           XFLAG  = 000005         $F$THE= 000330           $$ARGC= 000002

. ABS.  000000    000
        000632    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDPRC,DSKZ:CMDPRC=SPMAC/ML,EQUATE,CMDPRC
 RUN-TIME: 13 3 .4 SECONDS
 RUN-TIME RATIO: 36/17=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE CMDRST - KEYBOARD RESET
                                .IDENT  /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       CMDRST
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       OUTPUTS A PROMPT TO THE KEYBOARD
                                ;
                                ; INPUTS:
                                ;       ADDRESS OF DATA TABLE
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.STO
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       DT.KBPRM
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE MODULES CALLED:
                                ;       KBINI
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL CMDRST IN <A>
                                ;               A=ADDRESS OF DATA TABLE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;               EDIT            BY              DATE            REASON
                                ;--
```

```
                                        .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL STRUCT
000000'                                 STRUCT
(1) 000000'                             .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN=1
        000001                          $LSTTAG=1


                                        ;
                                        ;****************
                                        ;
                                        ; REFERENCED BY OTHER MODULES
                                        ;
                                        .GLOBL  CMDRST                          ;MODULE ENTRY POINT
                                        ;
                                        ;****************
                                        ;
                                        ; LOCAL STORAGE
                                        ;
000000' 051502  037131     000  CM.RUNMSG:      .ASCIZ /BSY>/                   ;RUN MODE PROMPT
000005'    103  042115  000076  CM.CMDMSG:      .ASCIZ /CMD>/                   ;COMMAND MODE PROMPT
                                        ;
                                        ;****************
                                        ;
```

```
                                .SBTTL CMDRST ROUTINE

000012'                         ROUTINE CMDRST <TABL>
000012'                                                                 CMDRST:

                                ;+
                                ;SET R0 TO START OF DATA TABLE
                                ;-

000012'                         PUSH R0
000012' 010046                                                                  MOV     R0,-(SP)

000014'                         LET R0 := TABL(R5)
000014' 016500  000000                                                          MOV     TABL(R5),R0

                                ;+
                                ;IF IN RUN MODE, OUTPUT RUN PROMPT
                                ;-

000020'                         IF #RUNMOD SETIN DT.ST0(R0) THEN
000020' 032760  100000  000010                                                  BIT     #RUNMOD,DT.ST0(R
000026' 001404                                                                  BEQ     50002$
000030'                                 LET DT.KBPRM(R0) := #CM.RUNMSG
000030' 012760  000000' 000026                                                  MOV     #CM.RUNMSG,DT.KB

                                ;+
                                ;ELSE OUTPUT COMMAND MODE PROMPT
                                ;-

000036'                         ELSE
000036' 000403                                                                  BR      50003$
000040'                                                                 50002$:
000040'                                 LET DT.KBPRM(R0) := #CM.CMDMSG
000040' 012760  000005' 000026                                                  MOV     #CM.CMDMSG,DT.KB
000046'                         ENDIF
000046'                                                                 50003$:

000046'                         POP R0
000046' 012600                                                                  MOV     (SP)+,R0

000050'                         ENDRTN
000050'                                                                 50000$:
000050'                                                                 50001$:
000050' 000207                                                                  RTS     PC

                000001          .END
```

SEQ 0355

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CMDRST  000012RG
CM.CMD  000005R
CM.RUN  000000R
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NDAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TABL   = 000000
TMPIO  = 000002
TQOVF  = 000002
```

```
UIPAR0= 177640          WBUFEA= 000136          $F$DO  = 000340          $ISK0  = 000001          $$FLAG= 000001
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LOCTA= 177777          $$FROM= 000000
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTIN= 000001          $$LOC = 000026R
UIPAR3= 177646          WBUFSZ= 000142          $F$IF  = 000110          $LSTTA= 000001          $$LOCN= 000000
UIPAR4= 177650          WDFR   = 000116          $F$INC= 000210          $NESTL= 177777          $$REG = 177777
UIPAR5= 177652          WDTO   = 000114          $F$LOO= 000200          $NSK0  = 000300          $$RETU= 000000
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $NSK1  = 000110          $$RTN1= 050000
UIPAR7= 177656          WTWHMI= 000222          $F$NO  = 000403          $SAVLE= 177777          $$RTN2= 050001
UIPDR0= 177600          XFLAG  = 000005          $F$OR  = 000320          $TAGLE= 177777          $$SRC = 000000
UIPDR1= 177602          XOFF   = 000023          $F$RTI= 000350          $TAGNU= 050004          $$TGSV= 000000
UIPDR2= 177604          XON    = 000021          $F$RTN= 000300          $TEMP  = 000300          $$TGS1= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TSK0  = 050003          $$TGS2= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $$ARGC= 000002          $$TO  = 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $$BYTE= 000403          $$$TAG= 050000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$CASE= 000000          .      = 000052R
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$DST  = 000000
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$ELOC= 000402
WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$ERFL= 000000

. ABS.  000000      000
        000052      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDRST,DSKZ:CMDRST=SPMAC/ML,EQUATE,CMDRST
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 30/13=2.2
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE CMDTB1 - COMMAND TABLE 1
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CMDTB1
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTAINS THE COMPLETE VALID COMMAND AND COMMAND
                        ;       DISPATCH TABLES
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CM.DIS
                                .GLOBL  CM.TBL
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  KCOFF                   ;CACHE OFF ROUTINE
                                .GLOBL  KCON                    ;CACHE ON ROUTINE
                                .GLOBL  KDES                    ;DESELECT ROUTINE
                                .GLOBL  KEXAM                   ;EXAM ROUTINE
                                .GLOBL  KFILL                   ;FILL ROUTINE
                                .GLOBL  KKTOFF                  ;KT OFF ROUTINE
                                .GLOBL  KKTON                   ;KT ON ROUTINE
                                .GLOBL  KLPOFF                  ;LP OFF ROUTINE
                                .GLOBL  KLPON                   ;LP ON ROUTINE
                                .GLOBL  KMAP                    ;MAP ROUTINE
                                .GLOBL  KMOD                    ;MODIFY ROUTINE
                                .GLOBL  KMOFF                   ;UNIBUS MAP OFF ROUTINE
                                .GLOBL  KMON                    ;UNIBUS MAP ON ROUTINE
                                .GLOBL  KPOFF                   ;PARITY OFF ROUTINE
                                .GLOBL  KPON                    ;PARITY ON ROUTINE
                                .GLOBL  KROTOF                  ;WRITE BUFFER ROTATION OFF ROUTINE
                                .GLOBL  KROTON                  ;WRITE BUFFER ROTATION ON ROUTINE
                                .GLOBL  KRUN                    ;RUN ROUTINE
                                .GLOBL  KRUNL                   ;RUN LOCKED ROUTINE
                                .GLOBL  KSEL                    ;SELECT ROUTINE
                                .GLOBL  KSUM                    ;SUMMARY ROUTINE
                                .GLOBL  KSWR                    ;MODIFY SWITCH REGISTER ROUTINE
                                ;
                                ;***************
                                ;

000000'                         CM.DIS:                         ;COMMAND DISPATCH TABLE
000000' 000000G                         .WORD   KCOFF
000002' 000000G                         .WORD   KCON
000004' 000000G                         .WORD   KDES
000006' 000000G                         .WORD   KEXAM
000010' 000000G                         .WORD   KFILL
000012' 000000G                         .WORD   KKTOFF
000014' 000000G                         .WORD   KKTON
000016' 000000G                         .WORD   KLPOFF
000020' 000000G                         .WORD   KLPON
000022' 000000G                         .WORD   KMAP
000024' 000000G                         .WORD   KMOD
000026' 000000G                         .WORD   KMOFF
000030' 000000G                         .WORD   KMON
000032' 000000G                         .WORD   KPOFF
000034' 000000G                         .WORD   KPON
000036' 000000G                         .WORD   KROTOF
```

```
582  000040' 000000G                        .WORD   KROTON
583  000042' 000000G                        .WORD   KRUN
584  000044' 000000G                        .WORD   KRUNL
585  000046' 000000G                        .WORD   KSEL
586  000050' 000000G                        .WORD   KSUM
587  000052' 000000G                        .WORD   KSWR
588
589
590  000054'                        CM.TBL:                  ;VALID COMMAND TABLE
591  000054' 047503  043106     000         .ASCIZ /COFF/
592  000061'    103  047117     000         .ASCIZ /CON/
593  000065'    104  051505     000         .ASCIZ /DES/
594  000071'    105  040530  000115         .ASCIZ /EXAM/
595  000076' 044506  046114     000         .ASCIZ /FILL/
596  000103'    113  047524  043106         .ASCIZ /KTOFF/
     000110'    000
597  000111'    113  047524  000116         .ASCIZ /KTON/
598  000116' 050114  043117  000106         .ASCIZ /LPOFF/
599  000124' 050114  047117     000         .ASCIZ /LPON/
600  000131'    115  050101     000         .ASCIZ /MAP/
601  000135'    115  042117     000         .ASCIZ /MOD/
602  000141'    115  043117  000106         .ASCIZ /MOFF/
603  000146' 047515  000116                 .ASCIZ /MON/
604  000152' 047520  043106     000         .ASCIZ /POFF/
605  000157'    120  047117     000         .ASCIZ /PON/
606  000163'    122  052117  043117         .ASCIZ /ROTOFF/
     000170' 000106
607  000172' 047522  047524  000116         .ASCIZ /ROTON/
608  000200' 052522  000116                 .ASCIZ /RUN/
609  000204' 052522  046116     000         .ASCIZ /RUNL/
610  000211'    123  046105     000         .ASCIZ /SEL/
611  000215'    123  046525     000         .ASCIZ /SUM/
612  000221'    123  051127     000         .ASCIZ /SWR/
613  000225'    377                         .BYTE   -1
614
615                                 .EVEN
616
617          000001                 .END
```

```
ACSR   = 000102          CSRC   = 000102          ERRTYP= 000106          KMON   = ****** G        PDPLSI= 020000
ACTBIT= 004000          CTRLC  = 000003          EVNTBE= 000200          KPOFF  = ****** G        PDP60  = 004000
ADDR22= 001000          CTRLO  = 000017          EVNTHD= 000200          KPON   = ****** G        PDP70  = 010000
ADR    = 000006          CTRLU  = 000025          EVNTKT= 000203          KROTOF= ****** G        PRIO   = 000000
APTFER= 000004          DCEVNT= 000011          EVNTPE= 000202          KROTON= ****** G        PRI1   = 000040
APTPRE= 000200          DEFRTN= 000400          EVNTRE= 000201          KRUN   = ****** G        PRI4   = 000200
ASB    = 000106          DROPMO= 100000          FATERR= 100000          KRUNL  = ****** G        PRI5   = 000240
ASTAT  = 000104          DSEVNT= 000014          HRDCNT= 000044          KSEL   = ****** G        PRI6   = 000300
AUTO   = 000010          DT.ADD= 000042          HRDPAS= 000050          KSUM   = ****** G        PRI7   = 000340
AUTOST= 020000          DT.AP  = 000100          ICONT  = 000036          KSWR   = ****** G        PRO    = 000000
AWAS   = 000110          DT.APK= 000076          ICOUNT= 000040          KTERRO= 000040           PR4    = 000200
BIT0   = 000001          DT.BLS= 000034          IDNUM  = 000122          KTPRES= 000400           PR5    = 000240
BIT00  = 000001          DT.CF0= 000014          IE     = 000100          KTSTAT= 000020           PR6    = 000300
BIT01  = 000002          DT.CF1= 000016          INDPAR= 000040          KTXTND= 040000           PR7    = 000340
BIT02  = 000004          DT.ERR= 000020          INHDRP= 040000          LF     = 000012          PS     = 177776
BIT03  = 000010          DT.ESI= 000044          INHEPR= 020000          LPSTAT= 000001           PSW    = 177776
BIT04  = 000020          DT.EVN= 000000          INHREL= 001000          MAPSTA= 000200           RANNUM= 000054
BIT05  = 000040          DT.EXS= 000060          INHRRE= 000400          MED    = 075600          RBUFEA= 000130
BIT06  = 000100          DT.FCH= 000037          INIT   = 000030          MEMPAS= 040000          RBUFPA= 000126
BIT07  = 000200          DT.FCN= 000036          INTR   = 000120          MODEXH= 004000          RBUFSZ= 000132
BIT08  = 000400          DT.HMX= 000104          IOMOD  = 100000          MODHOL= 002000          RBUFVA= 000124
BIT09  = 001000          DT.KBE= 000024          IOMODP= 102000          MODSEL= 001000          RDSERV= 000101
BIT1   = 000002          DT.KBP= 000026          IOMODR= 112000          MSGCKD= 000010          RDWHMI= 000022
BIT10  = 002000          DT.KBR= 000022          IOMODX= 110000          MSGCKS= 000011          RELERR= 000020
BIT11  = 004000          DT.KBU= 000030          JACK   = 035060          MSGDER= 000005          RELMOD= 020000
BIT12  = 010000          DT.MLS= 000032          KCOFF  = ****** G        MSGDRP= 000017          RELTIM= 010000
BIT13  = 020000          DT.MTI= 000110          KCON   = ****** G        MSGECH= 177777          RES1   = 000056
BIT14  = 040000          DT.OFF= 000070          KDES   = ****** G        MSGEOP= 000013          RES2   = 000060
BIT15  = 100000          DT.PAS= 000074          KEXAM  = ****** G        MSGHDR= 000004          RICHAR= 031060
BIT2   = 000004          DT.PC  = 000002          KFILL  = ****** G        MSGHNG= 000022          RPTDAT= 002000
BIT3   = 000010          DT.PFL= 000062          KIPAR0= 172340          MSGHRD= 000007          RSTRT  = 000112
BIT4   = 000020          DT.PSW= 000004          KIPAR1= 172342          MSGMAP= 000021          RUBOUT= 000177
BIT5   = 000040          DT.PTA= 000064          KIPAR2= 172344          MSGNUL= 177775          RUNMOD= 100000
BIT6   = 000100          DT.RCS= 000102          KIPAR3= 172346          MSGPOP= 000002          R5VALU= 001740
BIT7   = 000200          DT.REL= 000040          KIPAR4= 172350          MSGPRM= 177776          SAM    = 075464
BIT8   = 000400          DT.SCT= 000066          KIPAR5= 172352          MSGRES= 000001          SBADR  = 000102
BIT9   = 001000          DT.SMX= 000106          KIPAR6= 172354          MSGSFT= 000006          SBKMOD= 000000
BKDEF  = 000002          DT.SP  = 000006          KIPAR7= 172356          MSGSKE= 000003          SBKSEL= 010000
BKMOD  = 000020          DT.SSI= 000046          KIPDR0= 172300          MSGSMB= 000015          SC.ADR= 000006
BKMODE= 040000          DT.ST0= 000010          KIPDR1= 172302          MSGSMH= 000014          SC.ALC= 000014
BKSLSH= 000134          DT.ST1= 000012          KIPDR2= 172304          MSGSMS= 000016          SC.APC= 000016
CAPRES= 000004          DT.SWR= 000056          KIPDR3= 172306          MSGSTD= 000000          SC.CKL= 000002
CASTAT= 000004          DT.SYP= 000072          KIPDR4= 172310          MSGSYS= 000012          SC.CKP= 000004
CDERCT= 000146          DT.WBU= 000050          KIPDR5= 172312          MSGVEC= 000020          SC.CLO= 000000
CDWDCT= 000144          DT.WHL= 000054          KIPDR6= 172314          NBKMOD= 001000          SC.HLD= 000010
CKTIM  = 100000          DT.WLL= 000052          KIPDR7= 172316          NCPUOP= 000020          SC.SCA= 000012
CLKPRE= 000001          DVID1  = 000014          KKTOFF= ****** G        NOAPTY= 000002          SENDLS= 177777
CM.DIS  000000RG        ECCMEM= 000100          KKTON  = ****** G        NULL   = 000000          SOFCNT= 000042
CM.TBL  000054RG        ECCSTA= 000010          KLPOFF= ****** G        OWEN   = 024020          SOFPAS= 000046
CONFIG= 000056          ENBEOP= 010000          KLPON  = ****** G        PAERR  = 000010          SPACE  = 000040
CQOVF  = 000001          ENBNUL= 000001          KMAP   = ****** G        PARPRE= 002000          SPOINT= 000032
CR     = 000015          ENDLST= 000000          KMOD   = ****** G        PARSTA= 000100          SPVALU= 002200
CSRA   = 000100          EOPBIT= 000001          KMOFF  = ****** G        PASCNT= 000034          SR0    = 177572
```

CMDTB1.MAC 28-JUL-78 09:13 SYMBOL TABLE

```
SR1    = 177574          SVR3   = 000070          UIPAR2= 177644          UIPDR4= 177610          WDFR   = 000116
SR2    = 177576          SVR4   = 000072          UIPAR3= 177646          UIPDR5= 177612          WDTO   = 000114
SR3    = 172516          SVR5   = 000074          UIPAR4= 177650          UIPDR6= 177614          WTINRE= 000352
STAT   = 000026          SVR6   = 000076          UIPAR5= 177652          UIPDR7= 177616          WTWHMI= 000222
STATBI= 064757           SYSCNT= 000052           UIPAR6= 177654          WASADR= 000104          XFLAG  = 000005
STAT1  = 000027          SYSERR= 000100           UIPAR7= 177656          WBSTAT= 000040          XOFF   = 000023
SUSPND= 000001           TMPIO  = 000002          UIPDR0= 177600          WBUFEA= 000136          XON    = 000021
SVR0   = 000062          TQOVF  = 000002          UIPDR1= 177602          WBUFPA= 000134          .      = 000226R
SVR1   = 000064          UIPAR0= 177640           UIPDR2= 177604          WBUFRQ= 000140
SVR2   = 000066          UIPAR1= 177642           UIPDR3= 177606          WBUFSZ= 000142

. ABS.  000000     000
        000226     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDTB1,DSKZ:CMDTB1=SPMAC/ML,EQUATE,CMDTB1
 RUN-TIME: .7 .6 .3 SECONDS
 RUN-TIME RATIO: 4/1=2.5
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE CMDTB2 - COMMAND TABLE 2
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CMDTB2
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTAINS SHORTENED VERSIONS OF THE VALID COMMAND
                        ;       AND COMMAND DISPATCH TABLES.
                        ;
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CM.DIS
                                .GLOBL  CM.TBL
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  KDES                    ;DESELECT ROUTINE
                                .GLOBL  KFILL                   ;FILL ROUTINE
                                .GLOBL  KMOD                    ;MODIFY ROUTINE
                                .GLOBL  KRUN                    ;RUN ROUTINE
                                .GLOBL  KSEL                    ;SELECT ROUTINE
                                .GLOBL  KSUM                    ;SUMMARY ROUTINE
                                .GLOBL  KSWR                    ;MODIFY SWITCH REGISTER ROUTINE
                                ;
                                ;****************
                                ;
000000'                         CM.DIS:                         ;COMMAND DISPATCH TABLE
000000' 000000G                         .WORD   KDES
000002' 000000G                         .WORD   KFILL
000004' 000000G                         .WORD   KMOD
000006' 000000G                         .WORD   KRUN
000010' 000000G                         .WORD   KSEL
000012' 000000G                         .WORD   KSUM
000014' 000000G                         .WORD   KSWR

000016'                         CM.TBL:                 ;VALID COMMAND TABLE
000016' 042504  000123                  .ASCIZ  /DES/
000022' 044506  046114  000             .ASCIZ  /FILL/
000027'    115  042117  000             .ASCIZ  /MOD/
000033'    122  047125  000             .ASCIZ  /RUN/
000037'    123  046105  000             .ASCIZ  /SEL/
000043'    123  046525  000             .ASCIZ  /SUM/
000047'    123  051127  000             .ASCIZ  /SWR/
000053'    377                          .BYTE   -1

                                .EVEN
000001                          .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CM.DIS  000000RG
CM.TBL  000016RG
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KDES   = ****** G
KFILL  = ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KMOD   = ****** G
KRUN   = ****** G
KSEL   = ****** G
KSUM   = ****** G
KSWR   = ****** G
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          UIPAR4= 177650          UIPDR3= 177606          WBUFEA= 000136          WTWHMI= 000222
TMPIO = 000002          UIPAR5= 177652          UIPDR4= 177610          WBUFPA= 000134          XFLAG = 000005
TQOVF = 000002          UIPAR6= 177654          UIPDR5= 177612          WBUFRQ= 000140          XOFF  = 000023
UIPAR0= 177640          UIPAR7= 177656          UIPDR6= 177614          WBUFSZ= 000142          XON   = 000021
UIPAR1= 177642          UIPDR0= 177600          UIPDR7= 177616          WDFR  = 000116          .     = 000054R
UIPAR2= 177644          UIPDR1= 177602          WASADR= 000104          WDTO  = 000114
UIPAR3= 177646          UIPDR2= 177604          WBSTAT= 000040          WTINRE= 000352

. ABS.  000000     000
        000054     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDTB2,DSKZ:CMDTB2=SPMAC/ML,EQUATE,CMDTB2
 RUN-TIME: .6 .6 .3 SECONDS
 RUN-TIME RATIO: 4/1=2.5
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE CMDTB3 - COMMAND TABLE 3
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CMDTB3
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTAINS THE VALID COMMAND AND COMMAND
                        ;       DISPATCH TABLES EXCEPT:
                        ;       CON, COFF, KTON, KTOFF, MON, MOFF, RUNL
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CM.DIS
                                .GLOBL  CM.TBL
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  KDES                    ;DESELECT ROUTINE
                                .GLOBL  KEXAM                   ;EXAMINE ROUTINE
                                .GLOBL  KFILL                   ;FILL ROUTINE
                                .GLOBL  KLPOFF                  ;LP OFF ROUTINE
                                .GLOBL  KLPON                   ;LP ON ROUTINE
                                .GLOBL  KMAP                    ;MAP ROUTINE
                                .GLOBL  KMOD                    ;MODIFY ROUTINE
                                .GLOBL  KPOFF                   ;PARITY OFF ROUTINE
                                .GLOBL  KPON                    ;PARITY ON ROUTINE
                                .GLOBL  KROTOF                  ;WRITE BUFFER ROTATION ROUTINE
                                .GLOBL  KROTON                  ;WRITE BUFFER ROTATION ON ROUTINE
                                .GLOBL  KRUN                    ;RUN ROUTINE
                                .GLOBL  KSEL                    ;SELECT ROUTINE
                                .GLOBL  KSUM                    ;SUMMARY ROUTINE
                                .GLOBL  KSWR                    ;MODIFY SWITCH REGISTER ROUTINE
                                ;
                                ;****************
                                ;
000000'                         CM.DIS:                         ;COMMAND DISPATCH TABLE
000000' 000000G                         .WORD   KDES
000002' 000000G                         .WORD   KEXAM
000004' 000000G                         .WORD   KFILL
000006' 000000G                         .WORD   KLPOFF
000010' 000000G                         .WORD   KLPON
000012' 000000G                         .WORD   KMAP
000014' 000000G                         .WORD   KMOD
000016' 000000G                         .WORD   KPOFF
000020' 000000G                         .WORD   KPON
000022' 000000G                         .WORD   KROTOF
000024' 000000G                         .WORD   KROTON
000026' 000000G                         .WORD   KRUN
000030' 000000G                         .WORD   KSEL
000032' 000000G                         .WORD   KSUM
000034' 000000G                         .WORD   KSWR

000036'                         CM.TBL:                         ;VALID COMMAND TABLE
000036' 042504  000123                  .ASCIZ /DES/
000042' 054105  046501     000          .ASCIZ /EXAM/
000047'    106  046111  000114          .ASCIZ /FILL/
000054' 050114  043117  000106          .ASCIZ /LPOFF/
000062' 050114  047117     000          .ASCIZ /LPON/
```

```
  583  000067'     115  050101     000          .ASCIZ /MAP/
  584  000073'     115  042117     000          .ASCIZ /MOD/
  585  000077'     120  043117  000106          .ASCIZ /POFF/
  586  000104' 047520  000116                    .ASCIZ /PON/
  587  000110' 047522  047524  043106          .ASCIZ /ROTOFF/
       000116'     000
  588  000117'     122  052117  047117          .ASCIZ /ROTON/
       000124'     000
  589  000125'     122  047125     000          .ASCIZ /RUN/
  590  000131'     123  046105     000          .ASCIZ /SEL/
  591  000135'     123  046525     000          .ASCIZ /SUM/
  592  000141'     123  051127     000          .ASCIZ /SWR/
  593  000145'     377                          .BYTE   -1
  594
  595                                  .EVEN
  596
  597          000001                  .END
```

```
ACSR   = 000102          CSRC   = 000102          ERRTYP= 000106          KSEL   = ****** G        PRI6   = 000300
ACTBIT= 004000          CTRLC  = 000003          EVNTBE= 000200          KSUM   = ****** G        PRI7   = 000340
ADDR22= 001000          CTRLO  = 000017          EVNTHD= 000200          KSWR   = ****** G        PR0    = 000000
ADR    = 000006          CTRLU  = 000025          EVNTKT= 000203          KTERRO= 000040          PR4    = 000200
APTFER= 000004          DCEVNT= 000011          EVNTPE= 000202          KTPRES= 000400          PR5    = 000240
APTPRE= 000200          DEFRTN= 000400          EVNTRE= 000201          KTSTAT= 000020          PR6    = 000300
ASB    = 000106          DROPMO= 100000          FATERR= 100000          KTXTND= 040000          PR7    = 000340
ASTAT  = 000104          DSEVNT= 000014          HRDCNT= 000044          LF     = 000012          PS     = 177776
AUTO   = 000010          DT.ADD= 000042          HRDPAS= 000050          LPSTAT= 000001          PSW    = 177776
AUTOST= 020000          DT.AP  = 000100          ICONT  = 000036          MAPSTA= 000200          RANNUM= 000054
AWAS   = 000110          DT.APK= 000076          ICOUNT= 000040          MED    = 076600          RBUFEA= 000130
BIT0   = 000001          DT.BLS= 000034          IDNUM  = 000122          MEMPAS= 040000          RBUFPA= 000126
BIT00  = 000001          DT.CF0= 000014          IE     = 000100          MODEXH= 004000          RBUFSZ= 000132
BIT01  = 000002          DT.CF1= 000016          INDPAR= 000040          MODHOL= 002000          RBUFVA= 000124
BIT02  = 000004          DT.ERR= 000020          INHDRP= 040000          MODSEL= 001000          RDSERV= 000101
BIT03  = 000010          DT.ESI= 000044          INHEPR= 020000          MSGCKD= 000010          RDWHMI= 000022
BIT04  = 000020          DT.EVN= 000000          INHREL= 001000          MSGCKS= 000011          RELERR= 000020
BIT05  = 000040          DT.EXS= 000060          INHRRE= 000400          MSGDER= 000005          RELMOD= 020000
BIT06  = 000100          DT.FCH= 000037          INIT   = 000030          MSGDRP= 000017          RELTIM= 010000
BIT07  = 000200          DT.FCN= 000036          INTR   = 000120          MSGECH= 177777          RES1   = 000056
BIT08  = 000400          DT.HMX= 000104          IOMOD  = 100000          MSGEOP= 000013          RES2   = 000060
BIT09  = 001000          DT.KBE= 000024          IOMODP= 102000          MSGHDR= 000004          RICHAR= 031060
BIT1   = 000002          DT.KBP= 000026          IOMODR= 112000          MSGHNG= 000022          RPTDAT= 002000
BIT10  = 002000          DT.KBR= 000022          IOMODX= 110000          MSGHRD= 000007          RSTRT  = 000112
BIT11  = 004000          DT.KBU= 000030          JACK   = 035060          MSGMAP= 000021          RUBOUT= 000177
BIT12  = 010000          DT.MLS= 000032          KDES   = ****** G        MSGNUL= 177775          RUNMOD= 100000
BIT13  = 020000          DT.MTI= 000110          KEXAM  = ****** G        MSGPOP= 000002          R5VALU= 001740
BIT14  = 040000          DT.OFF= 000070          KFILL  = ****** G        MSGPRM= 177776          SAM    = 075464
BIT15  = 100000          DT.PAS= 000074          KIPAR0= 172340          MSGRES= 000001          SBADR  = 000102
BIT2   = 000004          DT.PC  = 000002          KIPAR1= 172342          MSGSFT= 000006          SBKMOD= 000000
BIT3   = 000010          DT.PFL= 000062          KIPAR2= 172344          MSGSKE= 000003          SBKSEL= 010000
BIT4   = 000020          DT.PSW= 000004          KIPAR3= 172346          MSGSMB= 000015          SC.ADR= 000006
BIT5   = 000040          DT.PTA= 000064          KIPAR4= 172350          MSGSMH= 000014          SC.ALC= 000014
BIT6   = 000100          DT.RCS= 000102          KIPAR5= 172352          MSGSMS= 000016          SC.APC= 000016
BIT7   = 000200          DT.REL= 000040          KIPAR6= 172354          MSGSTD= 000000          SC.CKL= 000002
BIT8   = 000400          DT.SCT= 000066          KIPAR7= 172356          MSGSYS= 000012          SC.CKP= 000004
BIT9   = 001000          DT.SMX= 000106          KIPDR0= 172300          MSGVEC= 000020          SC.CLO= 000000
BKDEF  = 000002          DT.SP  = 000006          KIPDR1= 172302          NBKMOD= 001000          SC.HLD= 000010
BKMOD  = 000020          DT.SSI= 000046          KIPDR2= 172304          NCPUOP= 000020          SC.SCA= 000012
BKMODE= 040000          DT.ST0= 000010          KIPDR3= 172306          NOAPTY= 000002          SENDLS= 177777
BKSLSH= 000134          DT.ST1= 000012          KIPDR4= 172310          NULL   = 000000          SOFCNT= 000042
CAPRES= 000004          DT.SWR= 000056          KIPDR5= 172312          OWEN   = 024020          SOFPAS= 000046
CASTAT= 000004          DT.SYP= 000072          KIPDR6= 172314          PAERR  = 000010          SPACE  = 000040
CDERCT= 000146          DT.WBU= 000050          KIPDR7= 172316          PARPRE= 002000          SPOINT= 000032
CDWDCT= 000144          DT.WHL= 000054          KLPOFF= ****** G        PARSTA= 000100          SPVALU= 002200
CKTIM  = 100000          DT.WLL= 000052          KLPON  = ****** G        PASCNT= 000034          SR0    = 177572
CLKPRE= 000001          DVID1  = 000014          KMAP   = ****** G        PDPLSI= 020000          SR1    = 177574
CM.DIS  000000RG        ECCMEM= 000100          KMOD   = ****** G        PDP60  = 004000          SR2    = 177576
CM.TBL  000036RG        ECCSTA= 000010          KPOFF  = ****** G        PDP70  = 010000          SR3    = 172516
CONFIG= 000056          ENBEOP= 010000          KPON   = ****** G        PRI0   = 000000          STAT   = 000026
CQOVF  = 000001          ENBNUL= 000001          KROTOF= ****** G        PRI1   = 000040          STATBI= 064757
CR     = 000015          ENDLST= 000000          KROTON= ****** G        PRI4   = 000200          STAT1  = 000027
CSRA   = 000100          EOPBIT= 000001          KRUN   = ****** G        PRI5   = 000240          SUSPND= 000001
```

```
SVR0   = 000062        TMPIO  = 000002        UIPAR7= 177656        WASADR= 000104        WTWHMI= 000222
SVR1   = 000064        TQOVF  = 000002        UIPDR0= 177600        WBSTAT= 000040        XFLAG = 000005
SVR2   = 000066        UIPAR0= 177640        UIPDR1= 177602        WBUFEA= 000136        XOFF  = 000023
SVR3   = 000070        UIPAR1= 177642        UIPDR2= 177604        WBUFPA= 000134        XON   = 000021
SVR4   = 000072        UIPAR2= 177644        UIPDR3= 177606        WBUFRQ= 000140        .     = 000146R
SVR5   = 000074        UIPAR3= 177646        UIPDR4= 177610        WBUFSZ= 000142
SVR6   = 000076        UIPAR4= 177650        UIPDR5= 177612        WDFR  = 000116
SYSCNT= 000052         UIPAR5= 177652        UIPDR6= 177614        WDTO  = 000114
SYSERR= 000100         UIPAR6= 177654        UIPDR7= 177616        WTINRE= 000352

. ABS.  000000      000
        000146      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CMDTB3,DSKZ:CMDTB3=SPMAC/ML,EQUATE,CMDTB3
 RUN-TIME: .7 .6 .3 SECONDS
 RUN-TIME RATIO: 4/1=2.5
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE CTLQUE (CONTROL QUEUE MODULES)
                                .IDENT /V0.0/

                                ;++
                                ; MODULE PACKAGE NAME:
                                ;       CTLQUE
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS MODULE CONTAINS THREE MODULES. THE FIRST ONE
                                ;       INITIALIZES THE CONTROL QUEUE AND THE NEXT TWO ACTUALLY DO THE
                                ;       ENQUEING AND DEQUEING OF ITEMS FROM THE CONTROL QUEUE.
                                ;       THE DESCRIPTION IS GIVEN UNDER THE RESPECTIVE MODULES.
                                ;       THESE MODULES ARE:
                                ;
                                ;           1. CQINI - INITIALIZES THE CONTROL QUEUE.
                                ;           2. ENQCQ - ENQUES ITEMS ON THE CONTROL QUEUE.
                                ;           3. DEQCQ - DEQUES ITEMS FROM THE CONTROL QUEUE.
                                ;
                                ;       THE CONTROL QUEUE AND THE QUEUE POINTERS ARE CONTAINED IN
                                ;       THIS MODULE FOR USE BY THE ABOVE-NAMED MODULES LOCALLY.
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT    DATE        BY          REASON
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS & REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                ;
                                .GLOBL CQINI                    ;INITIALIZE CONTROL QUEUE
                                .GLOBL ENQCQ                    ;ENQUE ITEMS ON THE CONTROL QUEUE
                                .GLOBL DEQCQ                    ;DEQUE ITEMS FROM THE CONTROL QUEUE
                                ;
                                ; ****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL GETPSW                   ;GET CALLER'S PS WORD ROUTINE

                                .GLOBL OV.CQ                    ;CONTROL QUEUE BUFFER
                                .GLOBL  OV.HICQ
                                .GLOBL  OV.CQSIZ
                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 000000                  CQ.PSW: 0                       ;LOC. TO SAVE CALLER'S PS WORD
000002' 000000G                 CQ.INP: OV.CQ                   ;ENQUE POINTER
000004' 000000G                 CQ.OTP: OV.CQ                   ;DEQUE POINTER
000006' 000000                  CQ.CT:  0                       ;CURRENT ENTRY COUNT
000010' 000000G                 CQ.HI:  OV.HICQ                 ;QUEUE'S HIGH ADDR
000012' 000000G                 CQ.MAX: OV.CQSIZ                ;MAX NO. OF ENTRIES ALLOWED
000014' 000000G                 CQ.ICQ: OV.CQ                   ;START OF CQ

```

```
                        .SBTTL CQINI (FUNCTIONAL DESCRIPTION)

                        ;++
                        ; MODULE NAME:
                        ;       CQINI
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE INITIALIZES THE CONTROL QUEUE. IT RESETS THE ENQUE
                        ;       AND DEQUE POINTERS TO THE START OF THE QUEUE, AND SETS THE
                        ;       COUNT TO ZERO.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CQINI
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       ------  ---------   ----------- ----------------
                        ;
                        ;--
```

```
  616                                   .SBTTL CQINI (CODE)
  617
  618                                   ;****************
  619                                   ; ROUTINE TO INITIALIZE CONTROL QUEUE
  620
  621
  622                                   ;+
  623                                   ; INIT INPUT,OUTPUT AND COUNT
  624                                   ;-
  625
  626  000016'                          ROUTINE CQINI                                   CQINI:
  (2)  000016'                          LET CQ.INP := CQ.ICQ                                    MOV     CQ.ICQ,CQ.INP
  627  000016'
  (4)  000016' 016767  177772  177756
  628  000024'                          LET CQ.OTP := CQ.ICQ                                    MOV     CQ.ICQ,CQ.OTP
  (4)  000024' 016767  177764  177752
  629  000032'                          LET CQ.CT := #0                                         CLR     CQ.CT
  (4)  000032' 005067  177750
  630  000036'                          ENDRTN                                  50000$:
  (3)  000036'                                                                  50001$:
  (3)  000036'                                                                          RTS     PC
  (2)  000036' 000207
  631
```

```
                        .SBTTL ENQCQ (FUNCTIONAL DESCRIPTION)

                        ;++
                        ; MODULE NAME:
                        ;       ENQCQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE HANDLES QUEUEING OF ITEMS ON THE CONTROL
                        ;       QUEUE.
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE ADDRESS
                        ;       2. MODULE'S HEADER ADDRESS
                        ;       3. RETURN ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ERR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       GETPSW          ;GET CALLER'S PS WORD
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       ENQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
                        ;       SERVES AS A LOCK MECHANISM.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ENQCQ IN <DTADR,HDRADR,RETADR>
                        ;
                        ;               WHERE DTADR = DATA TABLE ADDRESS
                        ;                     HDRADR = HEADER ADDRESS
                        ;                     RETADR = RETURN ADDRESS
                        ;
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       ------  ---------   ----------  ----------------
                        ;
                        ;--
```

```
                                .SBTTL ENQCQ (CODE)

000040'                         ROUTINE ENQCQ <DT,HDR,RET>
(2) 000040'                                                                 ENQCQ:

                                ;+
                                ; SAVE R0
                                ;-

000040'                         PUSH R0
(2) 000040' 010046                                                                  MOV     R0,-(SP)

                                ;+
                                ; IF THE COUNT IS EQUAL TO MAXIMUM, THE
                                ; QUEUE IS FULL. SET ERROR FLAG IN THE
                                ; STATUS INDICATOR AND RETURN WITH ERROR.
                                ;-

000042'                         IF CQ.CT GE CQ.MAX THEN
(6) 000042' 026767  177740  177742                                                  CMP     CQ.CT,CQ.MAX
(9) 000050' 002410                                                                  BLT     50002$
000052'                             LET R0 := DT(R5)
(4) 000052' 016500  000000                                                          MOV     DT(R5),R0
000056'                             LET DT.ERR(R0) := DT.ERR(R0) SET.BY #CQOVF!FATERR
(6) 000056' 052760  100001  000020                                                  BIS     #CQOVF!FATERR,DT
000064'                             POP R0
(2) 000064' 012600                                                                  MOV     (SP)+,R0
000066'                             RETURN ERROR
(2) 000066' 000261                                                                  SEC
(4) 000070' 000443                                                                  BR      50001$
000072'                         ENDIF
(4) 000072'                                                                 50002$:

                                ;+
                                ; SAVE CALLER'S PS WORD AND CHANGE
                                ; PRIORITY LEVEL TO 7
                                ;-

000072'                         CALL GETPSW OUT <CQ.PSW>
(4) 000072' 162705  000002                                                          SUB     #1*2,R5
(3) 000076' 004767  000000G                                                         JSR     PC,GETPSW
(4) 000102' 012567  177672                                                          MOV     (R5)+,CQ.PSW
000106'                         PUSH #PR7
(2) 000106' 012746  000340                                                          MOV     #PR7,-(SP)
000112'                         PUSH #1$
(2) 000112' 012746  000120'                                                         MOV     #1$,-(SP)
000116'                         INLINE <RTI>
(2) 000116' 000002                                                                  RTI

                                ;+
                                ; IF THE ENQUE POINTER IS AT HIGH
                                ; ADDRESS, RE-INITIALIZE IT TO THE
                                ; START ADDRESS OF THE QUEUE.
                                ;-
000120'                         INLINE <1$:>
```

```
   (2)  000120'                                 IF CQ.INP EQ CQ.HI THEN                                 1$:
  723   000120'                                                                                         CMP     CQ.INP,CQ.HI
   (6)  000120' 026767  177656  177662                                                                 BNE     50003$
   (9)  000126' 001003
  724   000130'                                     LET CQ.INP := CQ.ICQ
   (4)  000130' 016767  177660  177644                                                                 MOV     CQ.ICQ,CQ.INP
  725   000136'                                 ENDIF
   (4)  000136'                                                                                 50003$:
  726
  727                                           ;+
  728                                           ; NOW ENQUE THE ENTRY.
  729                                           ;-
  730
  731   000136'                                 LET R0 := CQ.INP                                        MOV     CQ.INP,R0
   (4)  000136' 016700  177640
  732   000142'                                 LET (R0)+ := HDR(R5)                                    MOV     HDR(R5),(R0)+
   (4)  000142' 016520  000002
  733   000146'                                 LET (R0)+ := RET(R5)                                    MOV     RET(R5),(R0)+
   (4)  000146' 016520  000004
  734   000152'                                 LET CQ.INP := R0                                        MOV     R0,CQ.INP
   (4)  000152' 010067  177624
  735   000156'                                 LET CQ.CT := CQ.CT + #1                                 INC     CQ.CT
   (6)  000156' 005267  177624
  736
  737                                           ;+
  738                                           ; RESTORE CALLER'S PS WORD AND RETURN
  739                                           ;-
  740
  741   000162'                                 PUSH CQ.PSW                                             MOV     CQ.PSW,-(SP)
   (2)  000162' 016746  177612
  742   000166'                                 PUSH #2$                                                MOV     #2$,-(SP)
   (2)  000166' 012746  000174'
  743   000172'                                 INLINE <RTI>                                            RTI
   (2)  000172' 000002
  744   000174'                                 INLINE <2$:>                                            2$:
   (2)  000174'
  745   000174'                                 POP R0                                                  MOV     (SP)+,R0
   (2)  000174' 012600
  746   000176'                                 ENDRTN                                          50000$:
   (3)  000176'                                                                                         CLC
   (2)  000176' 000241                                                                          50001$:
   (3)  000200'                                                                                         RTS     PC
   (2)  000200' 000207
```

```
                        .SBTTL DEQCQ (FUNCTIONAL DESCRIPTION)


                        ;++
                        ; MODULE NAME:
                        ;       DEQCQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE HANDLES DEQUEUEING OF ITEMS FROM THE CONTROL
                        ;       QUEUE.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       1. MODULE'S HEADER ADDRESS
                        ;       2. RETURN ADDRESS
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       ERROR RETURN INDICATES QUEUE UNDERFLOW.
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       GETPSW
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       DEQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
                        ;       SERVES AS A LOCK MECHANISM.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL DEQCQ OUT <HDRADR,RETADR>
                        ;
                        ;               WHERE HDRADR = HEADER ADDRESS
                        ;                     RETADR = RETURN ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       ------  ---------   ----------  ----------------
                        ;
                        ;--
```

```
                                .SBTTL DEQCQ (CODE)

                                ROUTINE DEQCQ <HDR,RET>

000202'                         DEQCQ:
000202'

                                ;+
                                ; IF COUNT IS EQUAL TO ZERO, THE QUEUE
                                ; IS EMPTY. RETURN WITH ERROR.
                                ;-

                                IF CQ.CT LE #0 THEN

000202' 005767  177600                  TST     CQ.CT
000206' 003002                          BGT     50002$

                                    RETURN ERROR

000210'                                 SEC
000210' 000261                          BR      50001$
000212' 000444

                                ENDIF

000214'                         50002$:
000214'

                                ;+
                                ; SAVE CALLER'S PS WORD AND
                                ; CHANGE PRIORITY LEVEL TO 7
                                ;-

                                CALL GETPSW OUT <CQ.PSW>

000214'                                 SUB     #1*2,R5
000214' 162705  000002                  JSR     PC,GETPSW
000220' 004767  000000G                 MOV     (R5)+,CQ.PSW
000224' 012567  177550

                                PUSH #PR7

000230'                                 MOV     #PR7,-(SP)
000230' 012746  000340

                                PUSH #3$

000234'                                 MOV     #3$,-(SP)
000234' 012746  000242'

                                INLINE <RTI>

000240'                                 RTI
000240' 000002

                                ;+
                                ; IF DEQUE POINTER IS AT HIGH ADDRESS,
                                ; RE-INITIALIZE IT TO THE QUEUE'S
                                ; START ADDRESS.
                                ;-

                                INLINE <3$:>

000242'                         3$:
000242'

                                IF CQ.OTP EQ CQ.HI THEN

000242'                                 CMP     CQ.OTP,CQ.HI
000242' 026767  177536  177540          BNE     50003$
000250' 001003

                                    LET CQ.OTP := #QV.CQ

000252'                                 MOV     #QV.CQ,CQ.OTP
000252' 012767  000000G 177524

                                ENDIF

000260'                         50003$:
000260'

                                ;+
                                ; NOW DEQUEUE THE ENTRY.
                                ;-

                                PUSH R0

000260'
```

```
   (2)  000260' 010046                                                          MOV     R0,-(SP)
   836  000262'                      LET R0 := CQ.OTP
   (4)  000262' 016700  177516                                                  MOV     CQ.OTP,R0
   837  000266'                      LET HDR(R5) := (R0)+
   (4)  000266' 012065  000000                                                  MOV     (R0)+,HDR(R5)
   838  000272'                      LET RET(R5) := (R0)+
   (4)  000272' 012065  000002                                                  MOV     (R0)+,RET(R5)
   839  000276'                      LET CQ.OTP := R0
   (4)  000276' 010067  177502                                                  MOV     R0,CQ.OTP
   840  000302'                      LET CQ.CT := CQ.CT - #1
   (6)  000302' 005367  177500                                                  DEC     CQ.CT
   841
   842                               ;+
   843                               ; RESTORE CALLER'S PS WORD AND RETURN
   844                               ;-
   845
   846  000306'                      PUSH CQ.PSW
   (2)  000306' 016746  177466                                                  MOV     CQ.PSW,-(SP)
   847  000312'                      PUSH #4$
   (2)  000312' 012746  000320'                                                 MOV     #4$,-(SP)
   848  000316'                      INLINE <RTI>
   (2)  000316' 000002                                                          RTI
   849  000320'                      INLINE <4$:>
   (2)  000320'                                                                 4$:
   850  000320'                      POP R0
   (2)  000320' 012600                                                          MOV     (SP)+,R0
   851  000322'                      ENDRTN
   (3)  000322'                                                         50000$:
   (2)  000322' 000241                                                          CLC
   (3)  000324'                                                         50001$:
   (2)  000324' 000207                                                          RTS     PC
   852          000001               .END
```

```
ACSR   = 000102          CQ.ICQ  000014R          DT.WBU= 000050          KIPDR7= 172316          PRI5   = 000240
ACTBIT= 004000          CQ.INP  000002R          DT.WHL= 000054          KTERRO= 000040          PRI6   = 000300
ADDR22= 001000          CQ.MAX  000012R          DT.WLL= 000052          KTPRES= 000400          PRI7   = 000340
ADR    = 000006          CQ.OTP  000004R          DVID1  = 000014          KTSTAT= 000020          PR0    = 000000
APTFER= 000004          CQ.PSW  000000R          ECCMEM= 000100          KTXTND= 040000          PR4    = 000200
APTPRE= 000200          CR     = 000015          ECCSTA= 000010          LF     = 000012          PR5    = 000240
ASB    = 000106          CSRA   = 000100          ENBEOP= 010000          LPSTAT= 000001          PR6    = 000300
ASSEMB= 000010          CSRC   = 000102          ENBNUL= 000001          MAPSTA= 000200          PR7    = 000340
ASTAT  = 000104          CTRLC  = 000003          ENDLST= 000000          MED    = 076600          PS     = 177776
AUTO   = 000010          CTRLO  = 000017          ENQCQ   000040RG        MEMPAS= 040000          PSW    = 177776
AUTOST= 020000          CTRLU  = 000025          EOPBIT= 000001          MODEXH= 004000          RANNUM= 000054
AWAS   = 000110          DCEVNT= 000011          ERRTYP= 000106          MODHOL= 002000          RBUFEA= 000130
BIT0   = 000001          DEFRTN= 000400          EVNTBE= 000200          MODSEL= 001000          RBUFPA= 000126
BIT00  = 000001          DEQCQ   000202RG        EVNTHD= 000200          MSGCKD= 000010          RBUFSZ= 000132
BIT01  = 000002          DIAGMC= 000000          EVNTKT= 000203          MSGCKS= 000011          RBUFVA= 000124
BIT02  = 000004          DROPMO= 100000          EVNTPE= 000202          MSGDER= 000005          RDSERV= 000101
BIT03  = 000010          DSEVNT= 000014          EVNTRE= 000201          MSGDRP= 000017          RDWHMI= 000022
BIT04  = 000020          DT     = 000000          FATERR= 100000          MSGECH= 177777          RELERR= 000020
BIT05  = 000040          DT.ADD= 000042          GETPSW= ****** G        MSGEOP= 000013          RELMOD= 020000
BIT06  = 000100          DT.AP  = 000100          HDR    = 000000          MSGHDR= 000004          RELTIM= 010000
BIT07  = 000200          DT.APK= 000076          HRDCNT= 000044          MSGHNG= 000022          RES1   = 000056
BIT08  = 000400          DT.BLS= 000034          HRDPAS= 000050          MSGHRD= 000007          RES2   = 000060
BIT09  = 001000          DT.CF0= 000014          ICONT  = 000036          MSGMAP= 000021          RET    = 000002
BIT1   = 000002          DT.CF1= 000016          ICOUNT= 000040          MSGNUL= 177775          RICHAR= 031060
BIT10  = 002000          DT.ERR= 000020          IDNUM  = 000122          MSGPOP= 000002          RPTDAT= 002000
BIT11  = 004000          DT.ESI= 000044          IE     = 000100          MSGPRM= 177776          RSTRT  = 000112
BIT12  = 010000          DT.EVN= 000000          INDPAR= 000040          MSGRES= 000001          RUBOUT= 000177
BIT13  = 020000          DT.EXS= 000060          INHDRP= 040000          MSGSFT= 000006          RUNMOD= 100000
BIT14  = 040000          DT.FCH= 000037          INHEPR= 020000          MSGSKE= 000003          R5VALU= 001740
BIT15  = 100000          DT.FCN= 000036          INHREL= 001000          MSGSMB= 000015          SAM    = 075464
BIT2   = 000004          DT.HMX= 000104          INHRRE= 000400          MSGSMH= 000014          SBADR  = 000102
BIT3   = 000010          DT.KBE= 000024          INIT   = 000030          MSGSMS= 000016          SBKMOD= 000000
BIT4   = 000020          DT.KBP= 000026          INTR   = 000120          MSGSTD= 000000          SBKSEL= 010000
BIT5   = 000040          DT.KBR= 000022          IOMOD  = 100000          MSGSYS= 000012          SC.ADR= 000006
BIT6   = 000100          DT.KBU= 000030          IOMODP= 102000          MSGVEC= 000020          SC.ALC= 000014
BIT7   = 000200          DT.MLS= 000032          IOMODR= 112000          NBKMOD= 001000          SC.APC= 000016
BIT8   = 000400          DT.MTI= 000110          IOMODX= 110000          NCPUOP= 000020          SC.CKL= 000002
BIT9   = 001000          DT.OFF= 000070          JACK   = 035060          NOAPTY= 000002          SC.CKP= 000004
BKDEF  = 000002          DT.PAS= 000074          KIPAR0= 172340          NULL   = 000000          SC.CLO= 000000
BKMOD  = 000020          DT.PC  = 000002          KIPAR1= 172342          OV.CQ  = ****** G        SC.HLD= 000010
BKMODE= 040000          DT.PFL= 000062          KIPAR2= 172344          OV.CQS= ****** G        SC.SCA= 000012
BKSLSH= 000134          DT.PSW= 000004          KIPAR3= 172346          OV.HIC= ****** G        SENDLS= 177777
CAPRES= 000004          DT.PTA= 000064          KIPAR4= 172350          OWEN   = 024020          SOFCNT= 000042
CASTAT= 000004          DT.RCS= 000102          KIPAR5= 172352          PAERR  = 000010          SOFPAS= 000046
CDERCT= 000146          DT.REL= 000040          KIPAR6= 172354          PARPRE= 002000          SPACE  = 000040
CDWDCT= 000144          DT.SCT= 000066          KIPAR7= 172356          PARSTA= 000100          SPOINT= 000032
CKTIM  = 100000          DT.SMX= 000106          KIPDR0= 172300          PASCNT= 000034          SPVALU= 002200
CLKPRE= 000001          DT.SP  = 000006          KIPDR1= 172302          PDPLSI= 020000          SR0    = 177572
CONFIG= 000056          DT.SSI= 000046          KIPDR2= 172304          PDP60  = 004000          SR1    = 177574
CQINI   000016RG        DT.ST0= 000010          KIPDR3= 172306          PDP70  = 010000          SR2    = 177576
CQCVF  = 000001          DT.ST1= 000012          KIPDR4= 172310          PRI0   = 000000          SR3    = 172516
CQ.CT   000006R          DT.SWR= 000056          KIPDR5= 172312          PRI1   = 000040          STAT   = 000026
CQ.HI   000010R          DT.SYP= 000072          KIPDR6= 172314          PRI4   = 000200          STATBI= 064757
```

```
STAT1 = 000027          UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $$BYTE= 000403
SUSPND= 000001          UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $$CASE= 000000
SVR0  = 000062          UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $$DST = 000000
SVR1  = 000064          UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $$ELOC= 000402
SVR2  = 000066          UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $$ERFL= 000000
SVR3  = 000070          UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$FLAG= 000001
SVR4  = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$FROM= 000001
SVR5  = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$LOC = 000250R
SVR6  = 000076          UIPDR7= 177616          $F$DEC= 000220          $LOCTA= 177777          $$LOCN= 000000
SYSCNT= 000052          WASADR= 000104          $F$DO = 000340          $LSTIN= 000001          $$REG = 177777
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LSTTA= 000001          $$RETU= 000001
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $NESTL= 177777          $$RTN1= 050000
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $NSK0 = 000300          $$RTN2= 050001
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK1 = 000110          $$SRC = 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $SAVLE= 177777          $$TGSV= 000000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $TAGLE= 177777          $$TGS1= 000000
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $TAGNU= 050004          $$TGS2= 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $TEMP = 000300          $$TO  = 000000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TSK0 = 050003          $$$TAG= 050000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $$ARGC= 000004          .     = 000326R

. ABS.  000000     000
        000326     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CTLQUE,DSKZ:CTLQUE=SPMAC/ML,EQUATE,CTLQUE
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 40/19=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  CONOF   PROCESS THE 'CON' AND 'COFF' KEYBOARD COMMAND
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       CONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PACKAGE CONTAINS:
                        ;       1. KCON - PROCESS THE 'CON' KEYBOARD COMMAND
                        ;       2. KCOFF - PROCESS THE 'COFF' KEYBOARD COMMAND
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          BY          DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR CONOF

                                .MCALL  STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION i.1

        000001                  $LSTTAG=1
        000001                  $LSTIN=1
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KCON            ;KCON MODULE ENTRY POINT
                                .GLOBL  KCOFF           ;KCOFF MODULE ENTRY POINT
                                ;

                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CCNTRL          ;CONTROL WORD FOR CACHE REG.
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  CM.ARG          ;'INVALID ARGUMENT' MESSAGE
                                .GLOBL  KONTRL          ;CONTROL WORD
                                ;
                                ;****************
                                ;****************
                                ;
                                ;
                                ;****************
                                ;
                                ; LOCAL EQUATES:
                                ;
        000352                  CFLUSH=352              ;FLUSH 11/60 CACHE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
                                ;
000000' 041445  041501  042510  CO.BAD: .ASCIZ  /%CACHE ERROR - WON'T TURN ON%/
000006' 042440  051122  051117
000014' 026440  053440  047117
000022' 052047  052040  051125
000030' 020116  047117  000045
000036' 040503  044103  020105  CO.ON:  .ASCIZ  /CACHE ON%/
000044' 047117  000045
000050' 047516  041440  041501  CO.NUN: .ASCIZ  /NO CACHE%/
000056' 042510  000045
000062' 040503  044103  020105  CO.OFF: .ASCIZ  /CACHE OFF%/
000070' 043117  022506     000
        000076'                 .EVEN

```

```
577                             .SBTTL  KCON PROCESS THE 'CON'(CACHE ON) KEYBOARD COMMAND
578                             .IDENT /V0.0/
579
580                             ;++
581                             ; MODULE NAME:
582                             ;       KCON
583                             ;
584                             ; FUNCTIONAL DESCRIPTION:
585                             ;       THIS ROUTINE PROCESSES THE 'CON'(CACHE ON) KEYBOARD COMMAND. IT
586                             ;       WILL FIRST CHECK FOR THE PRESENCE
587                             ;       OF ILLEGAL ARGUMENTS. IF ANY, AN ERROR MESSAGE IS OUTPUTTED
588                             ;       AND IF NOT, AND IT'S AN 11/60, THE CACHE IS FIRST
589                             ;       CLEARED.
590                             ;       IF NO CACHE, AN ERROR MESSAGE IS STUFFED.
591                             ;
592                             ; INPUTS:
593                             ;       1. ADDRESS OF DATA TABLE
594                             ;       2. DECODE BUFFER PTR
595                             ;
596                             ; IMPLICIT INPUTS:
597                             ;       DT.CF0
598                             ;
599                             ; OUTPUTS:
600                             ;       NONE
601                             ;
602                             ; IMPLICIT OUTPUTS:
603                             ;       DT.KBRSP
604                             ;
605                             ; PATHOLOGICAL CONNECTIONS:
606                             ;       NONE
607                             ;
608                             ; SUBORDINATE ROUTINES CALLED:
609                             ;       1. ARGCHK - CHECK AN ARGUMENT
610                             ;       2. SAVREG - SAVE REGISTERS
611                             ;       3. RESREG - RESTORE REGISTERS
612                             ;
613                             ; FUNCTIONAL SIDE EFFECTS:
614                             ;       NONE
615                             ;
616                             ; CALLING SEQUENCE:
617                             ;       CALL KCON IN <DTADR,BUFPTR>
618                             ;       WHERE DTADR = ADDRESS OF DATA TABLE
619                             ;             BUFPTR = DECODE BUFFER POINTER
620                             ;
621                             ; VERSION:
622                             ;       0.0
623                             ;
624                             ;       EDIT            DATE            BY              REASON
625                             ;--
626
627
```

```
                                .SBTTL  KCON ROUTINE

000076'                         ROUTINE KCON <DTADR,BUFPTR>
(2) 000076'                                                                 KCON:

                                ;+
                                ; INIT AND SAVE DTABLE,BUFPTR
                                ;-

000076'                         CALL SAVREG
(3) 000076' 004767  000000G                                                 JSR     PC,SAVREG
000102'                         LET R1 := DTADR(R5)
(4) 000102' 016501  000000                                                  MOV     DTADR(R5),R1
000106'                         LET R2 := BUFPTR(R5)
(4) 000106' 016502  000002                                                  MOV     BUFPTR(R5),R2

                                ;+
                                ; CHECK FOR JUNK ARGUMENTS
                                ;-

000112'                         CALL ARGCHK IN <R2> OUT <R2>
(4) 000112' 162705  000002                                                  SUB     #1*2,R5
(3) 000116' 010546                                                          MOV     R5,-(SP)
(4) 000120' 010245                                                          MOV     R2,-(R5)
(3) 000122' 004767  000000G                                                 JSR     PC,ARGCHK
(3) 000126' 012605                                                          MOV     (SP)+,R5
(4) 000130' 012502                                                          MOV     (R5)+,R2

000132'                         IF.ERROR THEN
(6) 000132' 103045                                                          BCC     50002$

                                ;+
                                ;  NO JUNK ARGS...CONTINUE... SEE IF CACHE EXISTS AT ALL
                                ;-

000134'                                         IF #CAPRES SETIN DT.CF0(R1) THEN
(6) 000134' 032761  000004  000014                                          BIT     #CAPRES,DT.CF0(R
(9) 000142' 001435                                                          BEQ     50003$

                                ;+
                                ; THERE IS CACHE... SEE WHAT FLAVOR... PDP 11/60 OR OTHER
                                ;-

000144'                                                 IF #PDP60 SETIN DT.CF0(R1) THEN
(6) 000144' 032761  004000  000014                                          BIT     #PDP60,DT.CF0(R1
(9) 000152' 001404                                                          BEQ     50004$

                                ;+
                                ; IT'S AN 11/60 SO,USING THE 'MED' INSTRUCTION FEATURE OF IT, DO A
                                ; CACHE SWEEP. CODE OF 200 TO R0,352 TRAILING TO GET INIT REG(INTERNAL
                                ; MICROCODE). CLEAR THE FORCE MISS BITS....
                                ;-

000154'                                                         LET R0 := #200
(4) 000154' 012700  000200                                                  MOV     #200,R0
```

```
669  000160'                                                    INLINE <MED>
(2)  000160' 076600                                                                  MED
670  000162'                                                    INLINE <#CFLUSH>
(2)  000162' 000352                                                                  #CFLUSH
671  000164'                                                ENDIF
(4)  000164'                                                                    50004$:
672  000164'                                                LET KONTRL := KONTRL CLR.BY #14
(6)  000164' 042767  000014  000000G                                                BIC     #14,KONTRL
673  000172'                                                LET @CCNTRL := KONTRL
(4)  000172' 016777  000000G 000000G                                                MOV     KONTRL,@CCNTRL
674
675
676
677
678                                ;+
679                                ; SEE IF IT REALLY DID ENABLE... IF NOT STUFF ERROR MSG...
680                                ;-
681
682  000200'                                                IF #14 SETIN @CCNTRL THEN
(6)  000200' 032777  000014  000000G                                                BIT     #14,@CCNTRL
(9)  000206' 001404                                                                  BEQ     50005$
683  000210'                                                    LET DT.KBRSP(R1) := #CO.BAD
(4)  000210' 012761  000000' 000022                                                  MOV     #CO.BAD,DT.KBRSP
684  000216'                                                ELSE
(4)  000216' 000406                                                                  BR      50006$
(3)  000220'                                                                    50005$:
685
686                                ;+
687                                ; NOW STUFF THE CACHE ON MSG, SET THE CACHE ENABLE BIT IN STATUS WORD...
688                                ;-
689
690  000220'                                                    LET DT.KBRSP(R1) := #CO.ON
(4)  000220' 012761  000036' 000022                                                  MOV     #CO.ON,DT.KBRSP(
691  000226'                                                    LET DT.ST0(R1) := DT.ST0(R1) SET.BY #CASTAT
(6)  000226' 052761  000004  000010                                                  BIS     #CASTAT,DT.ST0(R
692
693  000234'                                                ENDIF
(4)  000234'                                                                    50006$:
694  000234'                                            ELSE
(4)  000234' 000403                                                                  BR      50007$
(3)  000236'                                                                    50003$:
695
696                                ;+
697                                ; THERE IS NO CACHE... TELL OPERATOR - STUFF THE NO CACHE ERROR MSG
698                                ;-
699
700  000236'                                                    LET DT.KBRSP(R1) := #CO.NUN
(4)  000236' 012761  000050' 000022                                                  MOV     #CO.NUN,DT.KBRSP
701  000244'                                            ENDIF
(4)  000244'                                                                    50007$:
702
703
704
705
706  000244'                           ELSE
(4)  000244' 000403                                                                  BR      50010$
```

```
(3)  000246'                                                                    50002$:
707
708                                 ;+
709                                 ; THERE WAS A JUNK ARGUMENT
710                                 ; DON'T HONOR COMMAND.... STUFF ERROR MSG
711                                 ;-
712
713  000246'                                LET DT.KBRSP(R1) := #CM.ARG
(4)  000246' 012761  000000G 000022                                             MOV     #CM.ARG,DT.KBRSP
714
715  000254'                        ENDIF
(4)  000254'                                                                    50010$:
716
717  000254'                        CALL RESREG
(3)  000254' 004767  000000G                                                    JSR     PC,RESREG
718
719  000260'                        ENDRTN
(3)  000260'                                                                    50000$:
(3)  000260'                                                                    50001$:
(2)  000260' 000207                                                             RTS     PC
720
```

```
                        .SBTTL  KCOFF PROCESS THE 'COFF' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KCOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'COFF'(CACHE OFF) KEYBOARD COMMAND. IT
                        ;       WILL FIRST CHECK FOR THE PRESENCE
                        ;       OF ILLEGAL ARGUMENTS. IF ANY, AN ERROR MESSAGE IS OUTPUTTED
                        ;       AND IF NOT THE FORCE MISS BITS IN THE CONTROL
                        ;       REGISTER ARE SET.
                        ;       IF NO CACHE AN ERROR MESSAGE IS STUFFED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. DECODE BUFFER PTR
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK - CHECK AN ARGUMENT
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KCOFF IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             BUFPTR = DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  KCOFF PROCESS THE KEYBOARD COMMANDS 'COFF'

000262'                         ROUTINE KCOFF <DTADR,BUFPTR>
(2) 000262'                                                                     KCOFF:

                                ;+
                                ; INITIALIZE AND SAVE DATA TABLE ADDRESS AND BUFPTR
                                ;-

000262'                         PUSH R0,R1
(2) 000262' 010046                                                              MOV     R0,-(SP)
(3) 000264' 010146                                                              MOV     R1,-(SP)
000266'                         LET R0 := DTADR(R5)
(4) 000266' 016500  000000                                                      MOV     DTADR(R5),R0
000272'                         LET R1 := BUFPTR(R5)
(4) 000272' 016501  000002                                                      MOV     BUFPTR(R5),R1

                                ;+
                                ; SEE IF ANY JUNK ARGUMENTS
                                ;-

000276'                         CALL ARGCHK IN <R1> OUT <R1>
(4) 000276' 162705  000002                                                      SUB     #1*2,R5
(3) 000302' 010546                                                              MOV     R5,-(SP)
(4) 000304' 010145                                                              MOV     R1,-(R5)
(3) 000306' 004767  000000G                                                     JSR     PC,ARGCHK
(3) 000312' 012605                                                              MOV     (SP)+,R5
(4) 000314' 012501                                                              MOV     (R5)+,R1

000316'                         IF.ERROR THEN
(6) 000316' 103025                                                              BCC     50002$

                                ;+
                                ; NO JUNK ARGUMENTS FOUND SO SEE IF CACHE EXISTS AT ALL
                                ;-


000320'                                 IF #CAPRES SETIN DT.CF0(R0) THEN
(6) 000320' 032760  000004  000014                                              BIT     #CAPRES,DT.CF0(R
(9) 000326' 001415                                                              BEQ     50003$


                                ;+
                                ; CACHE DOES EXIST SO TURN OFF CACHE AND STUFF MSG
                                ;-
000330'                                         LET KONTRL := KONTRL SET.BY #14
(6) 000330' 052767  000014  000000G                                            BIS     #14,KONTRL
000336'                                         LET @CCNTRL := KONTRL
(4) 000336' 016777  000000G 000000G                                            MOV     KONTRL,@CCNTRL
000344'                                         LET DT.KBRSP(R0) := #C0.OFF
(4) 000344' 012760  000062' 000022                                              MOV     #C0.OFF,DT.KBRSP
000352'                                         LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #CASTAT
(6) 000352' 042760  000004  000010                                              BIC     #CASTAT,DT.ST0(R

000360'                                 ELSE
```

```
(4)  000360' 000403                                                                    BR      50004$
(3)  000362'                                                                50003$:
810
811                             ;+
812                             ; NO CACHE ON SYSTEM - GO INSULT OPERATOR....
813                             ; STUFF ERROR MSG...
814                             ;-
815  000362'                                    LET DT.KBRSP(R0) := #CO.NUN
(4)  000362' 012760  000050' 000022                                                    MOV     #CO.NUN,DT.KBRSP
816
817  000370'                            ENDIF
(4)  000370'                                                                50004$:
818  000370'                        ELSE
(4)  000370' 000403                                                                    BR      50005$
(3)  000372'                                                                50002$:
819
820                             ;+
821                             ; THERE ARE JUNK ARGUMENTS - STUFF ERROR MSG AND LEAVE
822                             ;-
823
824  000372'                            LET DT.KBRSP(R0) := #CM.ARG
(4)  000372' 012760  000000G 000022                                                    MOV     #CM.ARG,DT.KBRSP
825
826  000400'                        ENDIF
(4)  000400'                                                                50005$:
827
828                             ;+
829                             ; CLEAN UP AND SCRAM
830                             ;-
831
832  000400'                        POP R1,R0
(2)  000400' 012601                                                                    MOV     (SP)+,R1
(3)  000402' 012600                                                                    MOV     (SP)+,R0
833
834  000404'                        ENDRTN
(3)  000404'                                                                50000$:
(3)  000404'                                                                50001$:
(2)  000404' 000207                                                                    RTS     PC
835
836          000001                .END
```

```
ACSR   = 000102          CONFIG= 000056          DT.WBU= 000050          KONTRL= ****** G        PR0    = 000000
ACTBIT= 004000          CO.BAD  000000R         DT.WHL= 000054          KTERRO= 000040          PR4    = 000200
ADDR22= 001000          CO.NUN  000050R         DT.WLL= 000052          KTPRES= 000400          PR5    = 000240
ADR    = 000006          CO.OFF  000062R         DVID1 = 000014          KTSTAT= 000020          PR6    = 000300
APTFER= 000004          CO.ON   000036R         ECCMEM= 000100          KTXTND= 040000          PR7    = 000340
APTPRE= 000200          CQOVF = 000001          ECCSTA= 000010          LF     = 000012          PS     = 177776
ARGCHK= ****** G        CR     = 000015          ENBEOP= 010000          LPSTAT= 000001          PSW    = 177776
ASB    = 000106          CSRA   = 000100          ENBNUL= 000001          MAPSTA= 000200          RANNUM= 000054
ASSEMB= 000010          CSRC   = 000102          ENDLST= 000000          MED    = 076600          RBUFEA= 000130
ASTAT  = 000104          CTRLC = 000003          EOPBIT= 000001          MEMPAS= 040000          RBUFPA= 000126
AUTO   = 000010          CTRLO = 000017          ERRTYP= 000106          MODEXH= 004000          RBUFSZ= 000132
AUTOST= 020000          CTRLU = 000025          EVNTBE= 000200          MODHOL= 002000          RBUFVA= 000124
AWAS   = 000110          DCEVNT= 000011          EVNTHD= 000200          MODSEL= 001000          RDSERV= 000101
BIT0   = 000001          DEFRTN= 000400          EVNTKT= 000203          MSGCKD= 000010          RDWHMI= 000022
BIT00  = 000001          DIAGMC= 000000          EVNTPE= 000202          MSGCKS= 000011          RELERR= 000020
BIT01  = 000002          DROPMO= 100000          EVNTRE= 000201          MSGDER= 000005          RELMOD= 020000
BIT02  = 000004          DSEVNT= 000014          FATERR= 100000          MSGDRP= 000017          RELTIM= 010000
BIT03  = 000010          DTADR  = 000000          HRDCNT= 000044          MSGECH= 177777          RESREG= ****** G
BIT04  = 000020          DT.ADD= 000042          HRDPAS= 000050          MSGEOP= 000013          RES1   = 000056
BIT05  = 000040          DT.AP  = 000100          ICONT  = 000036          MSGHDR= 000004          RES2   = 000060
BIT06  = 000100          DT.APK= 000076          ICOUNT= 000040          MSGHNG= 000022          RICHAR= 031060
BIT07  = 000200          DT.BLS= 000034          IDNUM  = 000122          MSGHRD= 000007          RPTDAT= 002000
BIT08  = 000400          DT.CF0= 000014          IE     = 000100          MSGMAP= 000021          RSTRT  = 000112
BIT09  = 001000          DT.CF1= 000016          INDPAR= 000040          MSGNUL= 177775          RUBOUT= 000177
BIT1   = 000002          DT.ERR= 000020          INHDRP= 040000          MSGPOP= 000002          RUNMOD= 100000
BIT10  = 002000          DT.ESI= 000044          INHEPR= 020000          MSGPRM= 177776          R5VALU= 001740
BIT11  = 004000          DT.EVN= 000000          INHREL= 001000          MSGRES= 000001          SAM    = 075464
BIT12  = 010000          DT.EXS= 000060          INHRRE= 000400          MSGSFT= 000006          SAVREG= ****** G
BIT13  = 020000          DT.FCH= 000037          INIT   = 000030          MSGSKE= 000003          SBADR  = 000102
BIT14  = 040000          DT.FCN= 000036          INTR   = 000120          MSGSMB= 000015          SBKMOD= 000000
BIT15  = 100000          DT.HMX= 000104          IOMOD  = 100000          MSGSMH= 000014          SBKSEL= 010000
BIT2   = 000004          DT.KBE= 000024          IOMODP= 102000          MSGSMS= 000016          SC.ADR= 000006
BIT3   = 000010          DT.KBP= 000026          IOMODR= 112000          MSGSTD= 000000          SC.ALC= 000014
BIT4   = 000020          DT.KBR= 000022          IOMODX= 110000          MSGSYS= 000012          SC.APC= 000016
BIT5   = 000040          DT.KBU= 000030          JACK   = 035060          MSGVEC= 000020          SC.CKL= 000002
BIT6   = 000100          DT.MLS= 000032          KCOFF    000262RG       NBKMOD= 001000          SC.CKP= 000004
BIT7   = 000200          DT.MTI= 000110          KCON     000076RG       NCPUOP= 000020          SC.CLO= 000000
BIT8   = 000400          DT.OFF= 000070          KIPAR0= 172340          NOAPTY= 000002          SC.HLD= 000010
BIT9   = 001000          DT.PAS= 000074          KIPAR1= 172342          NULL   = 000000          SC.SCA= 000012
BKDEF  = 000002          DT.PC  = 000002          KIPAR2= 172344          OWEN   = 024020          SENDLS= 177777
BKMOD  = 000020          DT.PFL= 000062          KIPAR3= 172346          PAERR  = 000010          SOFCNT= 000042
BKMODE= 040000          DT.PSW= 000004          KIPAR4= 172350          PARPRE= 002000          SOFPAS= 000046
BKSLSH= 000134          DT.PTA= 000064          KIPAR5= 172352          PARSTA= 000100          SPACE  = 000040
BUFPTR= 000002          DT.RCS= 000102          KIPAR6= 172354          PASCNT= 000034          SPOINT= 000032
CAPRES= 000004          DT.REL= 000040          KIPAR7= 172356          PDPLSI= 020000          SPVALU= 002200
CASTAT= 000004          DT.SCT= 000066          KIPDR0= 172300          PDP60  = 004000          SR0    = 177572
CCNTRL= ****** G        DT.SMX= 000106          KIPDR1= 172302          PDP70  = 010000          SR1    = 177574
CDERCT= 000146          DT.SP  = 000006          KIPDR2= 172304          PRI0   = 000000          SR2    = 177576
CDWDCT= 000144          DT.SSI= 000046          KIPDR3= 172306          PRI1   = 000040          SR3    = 172516
CFLUSH= 000352          DT.ST0= 000010          KIPDR4= 172310          PRI4   = 000200          STAT   = 000026
CKTIM  = 100000          DT.ST1= 000012          KIPDR5= 172312          PRI5   = 000240          STATBI= 064757
CLKPRE= 000C01          DT.SWR= 000056          KIPDR6= 172314          PRI6   = 000300          STAT1  = 000027
CM.ARG= ****** G        DT.SYP= 000072          KIPDR7= 172316          PRI7   = 000340          SUSPND= 000001
```

```
SVR0  = 000062        UIPDR2= 177604        $F$AND= 000310        $F$YES= 000402        $$BYTE= 000403
SVR1  = 000064        UIPDR3= 177606        $F$BAD= 000401        $IFLEV= 177777        $$CASE= 000000
SVR2  = 000066        UIPDR4= 177610        $F$BLA= 000170        $ISK0 = 000001        $$DST = 000000
SVR3  = 000070        UIPDR5= 177612        $F$CAS= 000150        $ISK1 = 000001        $$ELOC= 000402
SVR4  = 000072        UIPDR6= 177614        $F$DEC= 000220        $ISK2 = 000001        $$ERFL= 000000
SVR5  = 000074        UIPDR7= 177616        $F$DO = 000340        $LOCTA= 177777        $$FLAG= 000001
SVR6  = 000076        WASADR= 000104        $F$FAL= 000405        $LSTIN= 000001        $$FROM= 000001
SYSCNT= 000052        WBSTAT= 000040        $F$GOO= 000400        $LSTTA= 000001        $$LOC = 000326R
SYSERR= 000100        WBUFEA= 000136        $F$IF = 000110        $NESTL= 177777        $$LOCN= 000000
TMPIO = 000002        WBUFPA= 000134        $F$INC= 000210        $NSK0 = 000300        $$REG = 177777
TQOVF = 000002        WBUFRQ= 000140        $F$LOO= 000200        $NSK1 = 000110        $$RETU= 000000
UIPAR0= 177640        WBUFSZ= 000142        $F$NAM= 000160        $NSK2 = 000110        $$RTN1= 050000
UIPAR1= 177642        WDFR  = 000116        $F$NO = 000403        $NSK3 = 000110        $$RTN2= 050001
UIPAR2= 177644        WDTO  = 000114        $F$OR = 000320        $SAVLE= 177777        $$SRC = 000000
UIPAR3= 177646        WTINRE= 000352        $F$RTI= 000350        $TAGLE= 177777        $$TGSV= 000000
UIPAR4= 177650        WTWHMI= 000222        $F$RTN= 000300        $TAGNU= 050006        $$TGS1= 000000
UIPAR5= 177652        XFLAG = 000005        $F$SEL= 000140        $TEMP = 000300        $$TGS2= 000000
UIPAR6= 177654        XOFF  = 000023        $F$THE= 000330        $TSK0 = 050005        $$TO  = 000001
UIPAR7= 177656        XON   = 000021        $F$TRU= 000404        $TSK1 = 050004        $$$TAG= 050000
UIPDR0= 177600        $BGNLE= 177777        $F$UNT= 000130        $TSK2 = 050006        .     = 000406R
UIPDR1= 177602        $ERFLG= 000400        $F$WHI= 000120        $$ARGC= 000004

. ABS.  000000    000
        000406    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:CONOF,DSKZ:CONOF=SPMAC/ML,EQUATE,CONOF
 RUN-TIME: 15 5 .4 SECONDS
 RUN-TIME RATIO: 42/21=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE DPEOP - XXDP/ACT-11 END-OF-PASS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       DPEOP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTROLS RETURN TO AND RESTART FROM XXDP/ACT-11
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ESIZ, DT.ADDR, DT.BLST, DT.SSIZ, DT.ST1
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       MD.COD                          ;MESSAGE CODE IN MSGDEQ
                        ;       DX.R5                           ;R5 STACK POINTER
                        ;       DX.SP                           ;R6 STACK POINTER
                        ;       BA.MPTR                         ;BACKGROUND MODULE LIST POINTER
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       RSTRCY                          ;RECOVER FROM RESET
                        ;       MSGDEQ                          ;MESSAGE DEQUEUER
                        ;       DX.INI                          ;MONITOR INIT. ROUTINE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       CAUSES PROGRAM CONTROL TO LEAVE THE EXERCISER
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL DPEOP IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  DPEOP                           ;MODULE ENTRY POINT
                                .GLOBL  $ENDAD                          ;XXDP/ACT-11 "HOOK"
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  RSTRCY                          ;RECOVER FROM RESET
                                .GLOBL  DX.INI                          ;MONITOR INIT. ROUTINE
                                .GLOBL  DX.SP                           ;POINTER TO MONITOR'S R6 STACK
                                .GLOBL  DX.R5                           ;POINTER TO MONITOR'S R5 STACK
                                .GLOBL  DX.DEQ                          ;MONITOR'S DEQUEUING ROUTINE
                                .GLOBL  BA.MPTR                         ;BACKGROUND MODULE LIST POINTER
                                .GLOBL  MSGDEQ                          ;MESSAGE DEQUEUER
                                .GLOBL  MD.COD                          ;MESSAGE TYPE CODE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  DP.R0:  .WORD 0                         ;R0 STORAGE
000002' 000000                  DP.R1:  .WORD 0                         ;R1 STORAGE
000004' 000000                  DP.R2:  .WORD 0                         ;R2 STORAGE
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL DPEOP ROUTINE

000006'                         ROUTINE DPEOP <TABL>
(2) 000006'                                                                     DPEOP:


                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

000006'                         LET R0 := TABL(R5)
(4) 000006' 016500  000000                                                              MOV     TABL(R5),R0

                                ;+
                                ; WAIT FOR THE TYPE QUEUE TO EMPTY.
                                ;-

000012'                         REPEAT
(3) 000012'                                                                     50002$:
000012'                                 CALL MSGDEQ IN <R0>
(3) 000012' 010546                                                                      MOV     R5,-(SP)
(4) 000014' 010045                                                                      MOV     R0,-(R5)
(3) 000016' 004767  000000G                                                             JSR     PC,MSGDEQ
(3) 000022' 012605                                                                      MOV     (SP)+,R5
000024'                         UNTIL MD.COD EQ #MSGNUL
(3) 000024' 026727  000000G 177775                                                      CMP     MD.COD,#MSGNUL
(6) 000032' 001367                                                                      BNE     50002$

                                ;+
                                ; MOVE THE XXDP MONITOR (EVEN IF IT DOESN'T EXIST, BECAUSE WE DON'T KNOW
                                ; IF WE HAVE XXDP OF ACT-11) TO THE TOP 1.5K OF MEMORY (BUT NOT HIGHER
                                ; THAN 28K SINCE XXDP CAN'T USE KT).
                                ;-

000034'                         IF DT.SSIZ(R0) HIS #1600 THEN
(6) 000034' 026027  000046  001600                                                      CMP     DT.SSIZ(R0),#160
(9) 000042' 103403                                                                      BLO     50003$
000044'                                 LET R2 := #160000
(4) 000044' 012702  160000                                                              MOV     #160000,R2
000050'                         ELSE
(4) 000050' 000410                                                                      BR      50004$
(3) 000052'                                                                     50003$:
000052'                                 LET R2 := DT.SSIZ(R0) SHIFT #6
(4) 000052' 016002  000046                                                              MOV     DT.SSIZ(R0),R2
(7) 000056' 006302                                                                      ASL     R2
(7) 000060' 006302                                                                      ASL     R2
(7) 000062' 006302                                                                      ASL     R2
(7) 000064' 006302                                                                      ASL     R2
(7) 000066' 006302                                                                      ASL     R2
(7) 000070' 006302                                                                      ASL     R2
000072'                         ENDIF
(4) 000072'                                                                     50004$:


```

```
                                ;+
                                ; SET R1 TO  THE 1ST ADDRESS ABOVE THE EXERCISER, SET R3 TO
                                ; 1.5K, AND MOVE THE CODE.
                                ; (DON'T FORGET THAT DT.ESIZ IS THE SIZE OF THE EXERCISER PLUS THE XXDP
                                ; MONITOR.)
                                ;-

000072'                         LET R1 := DT.ESIZ(R0) + #2
(4) 000072' 016001  000044                                                      MOV     DT.ESIZ(R0),R1
(6) 000076' 062701  000002                                                      ADD     #2,R1
000102'                         LET R3 := #^D1536
(4) 000102' 012703  003000                                                      MOV     #^D1536,R3

000106'                         WHILE R3 NE #0 DO
(4) 000106'                                                             50005$:
(6) 000106' 005703                                                              TST     R3
(9) 000110' 001403                                                              BEQ     50006$
000112'                                 LET -(R2) := -(R1)
(4) 000112' 014142                                                              MOV     -(R1),-(R2)
000114'                                 LET R3 := R3 - #1
(6) 000114' 005303                                                              DEC     R3
000116'                         ENDDO
(4) 000116' 000773                                                              BR      50005$
(3) 000120'                                                             50006$:


                                ;+
                                ; SAVE REGISTERS FOR AFTER THE RETURN FROM XXDP.
                                ;-

000120'                         LET DP.R0 := R0
(4) 000120' 010067  177654                                                      MOV     R0,DP.R0
000124'                         LET DP.R1 := R1
(4) 000124' 010167  177652                                                      MOV     R1,DP.R1
000130'                         LET DP.R2 := R2
(4) 000130' 010267  177650                                                      MOV     R2,DP.R2


                                ;+
                                ;****************
                                ; HERE IS THE ACTUAL XXDP/ACT-11 STUFF.
                                ;-

000134'                         INLINE <        RESET>
(2) 000134' 000005                                                              RESET
000136'                         INLINE <        MOV     @#42,R0>
(2) 000136' 013700  000042                                                      MOV     @#42,R0
000142'                         INLINE <$ENDAD: JSR     PC,(R0)>
(2) 000142' 004710                                                      $ENDAD: JSR     PC,(R0)
000144'                         INLINE <        .WORD NOP,NOP,NOP>
(2) 000144' 000240  000240  000240                                              .WORD NOP,NOP,NO
                                ;+
                                ;****************
                                ;-


```

```
                                ;+
                                ; RESTORE THE REGISTERS WE WERE USING.
                                ;-

000152'                         LET R0 := DP.R0
(4) 000152' 016700  177622                                                      MOV     DP.R0,R0
000156'                         LET R1 := DP.R1
(4) 000156' 016701  177620                                                      MOV     DP.R1,R1
000162'                         LET R2 := DP.R2
(4) 000162' 016702  177616                                                      MOV     DP.R2,R2


                                ;+
                                ; NOW MOVE THE XXDP MONITOR BACK TO THE TOP OF THE EXERCISER.
                                ;-

000166'                         LET R3 := #^D1536
(4) 000166' 012703  003000                                                      MOV     #^D1536,R3
000172'                         WHILE R3 NE #0 DO
(4) 000172'                                                             50007$:
(6) 000172' 005703                                                              TST     R3
(9) 000174' 001403                                                              BEQ     50010$
000176'                                 LET (R1)+ := (R2)+
(4) 000176' 012221                                                              MOV     (R2)+,(R1)+
000200'                                 LET R3 := R3 - #1
(6) 000200' 005303                                                              DEC     R3
000202'                         ENDDO
(4) 000202' 000773                                                              BR      50007$
(3) 000204'                                                             50010$:


                                ;+
                                ; RE-INITIALIZE THE STACKS.
                                ;-

000204'                         LET SP := #SPVALUE
(4) 000204' 012706  002200                                                      MOV     #SPVALUE,SP
000210'                         LET DX.SP := SP
(4) 000210' 010667  000000G                                                     MOV     SP,DX.SP
000214'                         LET R5 := #R5VALUE
(4) 000214' 012705  001740                                                      MOV     #R5VALUE,R5
000220'                         LET DX.R5 := R5
(4) 000220' 010567  000000G                                                     MOV     R5,DX.R5


                                ;+
                                ; RE-INITIALIZE THE BACKGROUND LIST POINTER
                                ;-

000224'                         LET BA.MPTR := DT.BLST(R0)
(4) 000224' 016067  000034  000000G                                             MOV     DT.BLST(R0),BA.M


                                ;+
                                ; SINCE ACT-11 REQUIRES AT LEAST 200 MILLISECONDS BETWEEN RESETS,
                                ; GO INTO A WAIT LOOP.
```

```
  708                                   ;-
  709
  710  000232'                          LET R4 := #10
  (4)  000232' 012704  000010                                                          MOV     #10,R4
  711  000236'                          WHILE R4 GT #0 DO
  (4)  000236'                                                                 50011$:
  (6)  000236' 005704                                                                  TST     R4
  (9)  000240' 003410                                                                  BLE     50012$
  712  000242'                                  LET R3 := #177777
  (4)  000242' 012703  177777                                                          MOV     #177777,R3
  713  000246'                                  WHILE R3 NE #0 DO
  (4)  000246'                                                                 50013$:
  (6)  000246' 005703                                                                  TST     R3
  (9)  000250' 001402                                                                  BEQ     50014$
  714  000252'                                          LET R3 := R3 - #1
  (6)  000252' 005303                                                                  DEC     R3
  715  000254'                                  ENDDO
  (4)  000254' 000774                                                                  BR      50013$
  (3)  000256'                                                                 50014$:
  716  000256'                                  LET R4 := R4 - #1
  (6)  000256' 005304                                                                  DEC     R4
  717  000260'                          ENDDO
  (4)  000260' 000766                                                                  BR      50011$
  (3)  000262'                                                                 50012$:
  718
  719
  720                                   ;+
  721                                   ; NOW WE CAN DO A RESET.
  722                                   ;-
  723
  724  000262'                          INLINE <RESET>
  (2)  000262' 000005                                                                  RESET
  725
  726
  727                                   ;+
  728                                   ; RESTORE ALL PROCESSOR OPTIONS TO THE STATE THEY WERE IN BEFORE
  729                                   ; WE BEGAN THIS NONSENSE.
  730                                   ;-
  731
  732  000264'                          CALL RSTRCY IN <R0>
  (3)  000264' 010546                                                                  MOV     R5,-(SP)
  (4)  000266' 010045                                                                  MOV     R0,-(R5)
  (3)  000270' 004767  000000G                                                        JSR     PC,RSTRCY
  (3)  000274' 012605                                                                  MOV     (SP)+,R5
  733
  734
  735                                   ;+
  736                                   ; CLEAR MEMPAS.
  737                                   ;-
  738
  739  000276'                          LET DT.ST1(R0) := DT.ST1(R0) CLR.BY #MEMPAS
  (6)  000276' 042760  040000  000012                                                  BIC     #MEMPAS,DT.ST1(R
  740
  741
  742                                   ;+
  743                                   ; RE-INITALIZE THE MONITOR.
```

```
  744                                   ;-
  745
  746  000304'                          CALL DX.INI
  (3)  000304' 004767  000000G                                                  JSR     PC,DX.INI
  747
  748
  749                                   ;+
  750                                   ; NOW WE CAN CONTINUE WHERE WE LEFT OFF BY RETURNING TO THE MONITOR'S
  751                                   ; DE-QUEUEING MECHANISM.
  752                                   ;-
  753
  754  000310'                          INLINE <JMP @#DX.DEQ>
  (2)  000310' 000137  000000G                                                  JMP @#DX.DEQ
  755
  756
  757  000314'                          ENDRTN
  (3)  000314'                                                          50000$:
  (3)  000314'                                                          50001$:
  (2)  000314' 000207                                                           RTS     PC
  758          000001                   .END
```

```
ACSR  = 000102          CSRC  = 000102          DX.INI= ****** G        KTXTND= 040000          PR5   = 000240
ACTBIT= 004000          CTRLC = 000003          DX.R5 = ****** G        LF    = 000012          PR6   = 000300
ADDR22= 001000          CTRLO = 000017          DX.SP = ****** G        LPSTAT= 000001          PR7   = 000340
ADR   = 000006          CTRLU = 000025          ECCMEM= 000100          MAPSTA= 000200          PS    = 177776
APTFER= 000004          DCEVNT= 000011          ECCSTA= 000010          MD.COD= ****** G        PSW   = 177776
APTPRE= 000200          DEFRTN= 000400          ENBEOP= 010000          MED   = 076600          RANNUM= 000054
ASB   = 000106          DIAGMC= 000000          ENBNUL= 000001          MEMPAS= 040000          RBUFEA= 000130
ASSEMB= 000010          DPEOP   000006RG        ENDLST= 000000          MODEXH= 004000          RBUFPA= 000126
ASTAT = 000104          DP.R0   000000R         EOPBIT= 000001          MODHOL= 002000          RBUFSZ= 000132
AUTO  = 000010          DP.R1   000002R         ERRTYP= 000106          MODSEL= 001000          RBUFVA= 000124
AUTOST= 020000          DP.R2   000004R         EVNTBE= 000200          MSGCKD= 000010          RDSERV= 000101
AWAS  = 000110          DROPMO= 100000          EVNTHD= 000200          MSGCKS= 000011          RDWHMI= 000022
BA.MPT= ****** G        DSEVNT= 000014          EVNTKT= 000203          MSGDEQ= ****** G        RELERR= 000020
BIT0  = 000001          DT.ADD= 000042          EVNTPE= 000202          MSGDER= 000005          RELMOD= 020000
BIT00 = 000001          DT.AP = 000100          EVNTRE= 000201          MSGDRP= 000017          RELTIM= 010000
BIT01 = 000002          DT.APK= 000076          FATERR= 100000          MSGECH= 177777          RES1  = 000056
BIT02 = 000004          DT.BLS= 000034          HRDCNT= 000044          MSGEOP= 000013          RES2  = 000060
BIT03 = 000010          DT.CF0= 000014          HRDPAS= 000050          MSGHDR= 000004          RICHAR= 031060
BIT04 = 000020          DT.CF1= 000016          ICONT = 000036          MSGHNG= 000022          RPTDAT= 002000
BIT05 = 000040          DT.ERR= 000020          ICOUNT= 000040          MSGHRD= 000007          RSTRCY= ****** G
BIT06 = 000100          DT.ESI= 000044          IDNUM = 000122          MSGMAP= 000021          RSTRT = 000112
BIT07 = 000200          DT.EVN= 000000          IE    = 000100          MSGNUL= 177775          RUBOUT= 000177
BIT08 = 000400          DT.EXS= 000060          INDPAR= 000040          MSGPOP= 000002          RUNMOD= 100000
BIT09 = 001000          DT.FCH= 000037          INHDRP= 040000          MSGPRM= 177776          R5VALU= 001740
BIT1  = 000002          DT.FCN= 000036          INHEPR= 020000          MSGRES= 000001          SAM   = 075464
BIT10 = 002000          DT.HMX= 000104          INHREL= 001000          MSGSFT= 000006          SBADR = 000102
BIT11 = 004000          DT.KBE= 000024          INHRRE= 000400          MSGSKE= 000003          SBKMOD= 000000
BIT12 = 010000          DT.KBP= 000026          INIT  = 000030          MSGSMB= 000015          SBKSEL= 010000
BIT13 = 020000          DT.KBR= 000022          INTR  = 000120          MSGSMH= 000014          SC.ADR= 000006
BIT14 = 040000          DT.KBU= 000030          IOMOD = 100000          MSGSMS= 000016          SC.ALC= 000014
BIT15 = 100000          DT.MLS= 000032          IOMODP= 102000          MSGSTD= 000000          SC.APC= 000016
BIT2  = 000004          DT.MTI= 000110          IOMODR= 112000          MSGSYS= 000012          SC.CKL= 000002
BIT3  = 000010          DT.OFF= 000070          IOMODX= 110000          MSGVEC= 000020          SC.CKP= 000004
BIT4  = 000020          DT.PAS= 000074          JACK  = 035060          NBKMOD= 001000          SC.CLO= 000000
BIT5  = 000040          DT.PC = 000002          KIPAR0= 172340          NCPUOP= 000020          SC.HLD= 000010
BIT6  = 000100          DT.PFL= 000062          KIPAR1= 172342          NOAPTY= 000002          SC.SCA= 000012
BIT7  = 000200          DT.PSW= 000004          KIPAR2= 172344          NULL  = 000000          SENDLS= 177777
BIT8  = 000400          DT.PTA= 000064          KIPAR3= 172346          OWEN  = 024020          SOFCNT= 000042
BIT9  = 001000          DT.RCS= 000102          KIPAR4= 172350          PAERR = 000010          SOFPAS= 000046
BKDEF = 000002          DT.REL= 000040          KIPAR5= 172352          PARPRE= 002000          SPACE = 000040
BKMOD = 000020          DT.SCT= 000066          KIPAR6= 172354          PARSTA= 000100          SPOINT= 000032
BKMODE= 040000          DT.SMX= 000106          KIPAR7= 172356          PASCNT= 000034          SPVALU= 002200
BKSLSH= 000134          DT.SP = 000006          KIPDR0= 172300          PDPLSI= 020000          SR0   = 177572
CAPRES= 000004          DT.SSI= 000046          KIPDR1= 172302          PDP60 = 004000          SR1   = 177574
CASTAT= 000004          DT.ST0= 000010          KIPDR2= 172304          PDP70 = 010000          SR2   = 177576
CDERCT= 000146          DT.ST1= 000012          KIPDR3= 172306          PRI0  = 000000          SR3   = 172516
CDWDCT= 000144          DT.SWR= 000056          KIPDR4= 172310          PRI1  = 000040          STAT  = 000026
CKTIM = 100000          DT.SYP= 000072          KIPDR5= 172312          PRI4  = 000200          STATBI= 064757
CLKPRE= 000001          DT.WBU= 000050          KIPDR6= 172314          PRI5  = 000240          STAT1 = 000027
CONFIG= 000056          DT.WHL= 000054          KIPDR7= 172316          PRI6  = 000300          SUSPND= 000001
CQOVF = 000001          DT.WLL= 000052          KTERR0= 000040          PRI7  = 000340          SVR0  = 000062
CR    = 000015          DVID1 = 000014          KTPRES= 000400          PR0   = 000000          SVR1  = 000064
CSRA  = 000100          DX.DEQ= ****** G        KTSTAT= 000020          PR4   = 000200          SVR2  = 000066
```

```
SVR3   = 000070          UIPDR4= 177610          $F$BAD= 000401          $IFLEV= 177777          $$DST  = 000000
SVR4   = 000072          UIPDR5= 177612          $F$BLA= 000170          $ISK0 = 000001          $$ELOC= 000402
SVR5   = 000074          UIPDR6= 177614          $F$CAS= 000150          $LOCTA= 177777          $$ERFL= 000000
SVR6   = 000076          UIPDR7= 177616          $F$DEC= 000220          $LSTIN= 000001          $$FLAG= 000340
SYSCNT= 000052           WASADR= 000104          $F$DO  = 000340         $LSTTA= 000001          $$FROM= 000000
SYSERR= 000100           WBSTAT= 000040          $F$FAL= 000405          $NESTL= 177777          $$LOC  = 000250R
TABL   = 000000          WBUFEA= 000136          $F$GOO= 000400          $NSK0 = 000300          $$LOCN= 000000
TMPIO  = 000002          WBUFPA= 000134          $F$IF  = 000110         $NSK1 = 000120          $$REG  = 177777
TQOVF  = 000002          WBUFRQ= 000140          $F$INC= 000210          $NSK2 = 000120          $$RETU= 000000
UIPAR0= 177640           WBUFSZ= 000142          $F$LOO= 000200          $SAVLE= 177777          $$RTN1= 050000
UIPAR1= 177642           WDFR   = 000116         $F$NAM= 000160          $SSK0 = 050012          $$RTN2= 050001
UIPAR2= 177644           WDTO   = 000114         $F$NO  = 000403         $TAGLE= 177777          $$SRC  = 000000
UIPAR3= 177646           WTINRE= 000352          $F$OR  = 000320         $TAGNU= 050015          $$TGSV= 000000
UIPAR4= 177650           WTWHMI= 000222          $F$RTI= 000350          $TEMP  = 000300         $$TGS1= 000000
UIPAR5= 177652           XFLAG  = 000005         $F$RTN= 000300          $TSK0 = 050011          $$TGS2= 000000
UIPAR6= 177654           XOFF   = 000023         $F$SEL= 000140          $TSK1 = 050012          $$TO   = 000000
UIPAR7= 177656           XON    = 000021         $F$THE= 000330          $TSK2 = 050013          $$$TAG= 050000
UIPDR0= 177600           $BGNLE= 177777          $F$TRU= 000404          $TSK3 = 050014          .      = 000316R
UIPDR1= 177602           $ENDAD  000142RG        $F$UNT= 000130          $$ARGC= 000002
UIPDR2= 177604           $ERFLG= 000400          $F$WHI= 000120          $$BYTE= 000403
UIPDR3= 177606           $F$AND= 000310          $F$YES= 000402          $$CASE= 000000

. ABS.  000000     000
        000316     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DPEOP,DSKZ:DPEOP=SPMAC/ML,EQUATE,DPEOP
 RUN-TIME: 16 5 .4 SECONDS
 RUN-TIME RATIO: 45/22=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE DPSTRT - XXDP/ACT-11 START-UP
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       DPSTRT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       MOVES XXDP MONITOR AND FAKES A "RUN" COMMAND
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ESIZ,DT.SSIZ,DT.REL
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ESIZ,DT.REL
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                  ;SAVE REGISTERS
                        ;       RESREG                  ;RESTORE REGISTERS
                        ;       KRUN                    ;PROCESS "RUN" COMMAND
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       STARTS THE EXERCISING OF THE SYSTEM
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL DPSTRT IN <A>
                        ;               A=DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  DPSTRT                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  KRUN                    ;PROCESS "RUN" COMMAND
                                ;
                                ;***************
                                ;

```

```
                                .SBTTL DPSTRT ROUTINE

000000'                         ROUTINE DPSTRT <TABL>
(2) 000000'                                                             DPSTRT:


                                ;+
                                ; SAVE REGISTERS
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                                     JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE.
                                ;-

000004'                         LET R0 := TABL(R5)
(4) 000004' 016500  000000                                                      MOV     TABL(R5),R0

                                ;+
                                ; MOVE THE XXDP MONITOR (EVEN IF IT DOESN'T EXIST, SINCE WE DON'T KNOW
                                ; IF WE'RE UNDER XXDP OR ACT-11) FROM ITS LOAD AREA (HIGHEST NON-KT
                                ; ADDRESSABLE MEMORY IN THE SYSTEM) TO JUST ABOVE THE EXERCISER.
                                ; (R1 IS INITIALIZED TO THE 1ST FREE ADDRESS ABOVE THE EXERCISER, AND R2 IS
                                ; INITIALIZED TO THE LOWEST ADDRESS OF XXDP.)
                                ;-

000010'                         IF DT.SSIZ(R0) HIS #1600 THEN
(6) 000010' 026027  000046  001600                                              CMP     DT.SSIZ(R0),#160
(9) 000016' 103403                                                              BLO     50002$
000020'                                 LET R3 := #160000
(4) 000020' 012703  160000                                                      MOV     #160000,R3
000024'                         ELSE
(4) 000024' 000410                                                              BR      50003$
(3) 000026'                                                             50002$:
000026'                                 LET R3 := DT.SSIZ(R0) SHIFT #+6
(4) 000026' 016003  000046                                                      MOV     DT.SSIZ(R0),R3
(7) 000032' 006303                                                              ASL     R3
(7) 000034' 006303                                                              ASL     R3
(7) 000036' 006303                                                              ASL     R3
(7) 000040' 006303                                                              ASL     R3
(7) 000042' 006303                                                              ASL     R3
(7) 000044' 006303                                                              ASL     R3
000046'                         ENDIF
(4) 000046'                                                             50003$:
000046'                         LET R2 := R3 - #^D3074
(4) 000046' 010302                                                              MOV     R3,R2
(6) 000050' 162702  006002                                                      SUB     #^D3074,R2

000054'                         LET R1 := DT.ESIZ(R0) + #2
(4) 000054' 016001  000044                                                      MOV     DT.ESIZ(R0),R1
(6) 000060' 062701  000002                                                      ADD     #2,R1
000064'                         WHILE R2 LO R3 DO
```

```
(4)  000064'                                                                    50004$:
(6)  000064' 020203                                                                     CMP     R2,R3
(9)  000066' 103002                                                                     BHIS    50005$
614  000070'                            LET (R1)+ := (R2)+
(4)  000070' 012221                                                                     MOV     (R2)+,(R1)+
615  000072'                        ENDDO
(4)  000072' 000774                                                                     BR      50004$
(3)  000074'                                                                    50005$:
616
617
618                                 ;+
619                                 ; ADJUST THE EXERCISER SIZE TO INCLUDE THE XXDP MONITOR.
620                                 ;-
621
622  000074'                        LET DT.ESIZ(R0) := R1 - #2
(4)  000074' 010160  000044                                                             MOV     R1,DT.ESIZ(R0)
(6)  000100' 162760  000002  000044                                                     SUB     #2,DT.ESIZ(R0)
623
624
625                                 ;+
626                                 ; INCREASE THE RELOCATION CONSTANT TO 28K.
627                                 ;-
628
629  000106'                        LET DT.REL(R0) := #1600
(4)  000106' 012760  001600  000040                                                     MOV     #1600,DT.REL(R0)
630
631
632                                 ;+
633                                 ; NOW PRETEND A "RUN" COMMAND HAS BEEN TYPED.
634                                 ;-
635
636  000114'                        CALL KRUN IN <R0,#0>
(3)  000114' 010546                                                                     MOV     R5,-(SP)
(5)  000116' 012745  000000                                                             MOV     #0,-(R5)
(4)  000122' 010045                                                                     MOV     R0,-(R5)
(3)  000124' 004767  000000G                                                            JSR     PC,KRUN
(3)  000130' 012605                                                                     MOV     (SP)+,R5
637
638
639                                 ;+
640                                 ; RESTORE REGISTERS AND RETURN.
641                                 ;-
642
643  000132'                        CALL RESREG
(3)  000132' 004767  000000G                                                            JSR     PC,RESREG
644
645  000136'                        ENDRTN
(3)  000136'                                                                    50000$:
(3)  000136'                                                                    50001$:
(2)  000136' 000207                                                                     RTS     PC
646
647          000001                .END
```

```
ACSR   = 000102   CTRLC  = 000003   EOPBIT= 000001   MODHOL= 002000   RBUFVA= 000124
ACTBIT= 004000   CTRLO  = 000017   ERRTYP= 000106   MODSEL= 001000   RDSERV= 000101
ADDR22= 001000   CTRLU  = 000025   EVNTBE= 000200   MSGCKD= 000010   RDWHMI= 000022
ADR    = 000006   DCEVNT= 000011   EVNTHD= 000200   MSGCKS= 000011   RELERR= 000020
APTFER= 000004   DEFRTN= 000400   EVNTKT= 000203   MSGDER= 000005   RELMOD= 020000
APTPRE= 000200   DIAGMC= 000000   EVNTPE= 000202   MSGDRP= 000017   RELTIM= 010000
ASB    = 000106   DPSTRT  000000RG  EVNTRE= 000201   MSGECH= 177777   RESREG= ****** G
ASSEMB= 000010   DROPMO= 100000   FATERR= 100000   MSGEOP= 000013   RES1   = 000056
ASTAT  = 000104   DSEVNT= 000014   HRDCNT= 000044   MSGHDR= 000004   RES2   = 000060
AUTO   = 000010   DT.ADD= 000042   HRDPAS= 000050   MSGHNG= 000022   RICHAR= 031060
AUTOST= 020000   DT.AP  = 000100   ICONT  = 000036   MSGHRD= 000007   RPTDAT= 002000
AWAS   = 000110   DT.APK= 000076   ICOUNT= 000040   MSGMAP= 000021   RSTRT  = 000112
BIT0   = 000001   DT.BLS= 000034   IDNUM  = 000122   MSGNUL= 177775   RUBOUT= 000177
BIT00  = 000001   DT.CF0= 000014   IE     = 000100   MSGPOP= 000002   RUNMOD= 100000
BIT01  = 000002   DT.CF1= 000016   INDPAR= 000040   MSGPRM= 177776   R5VALU= 001740
BIT02  = 000004   DT.ERR= 000020   INHDRP= 040000   MSGRES= 000001   SAM    = 075464
BIT03  = 000010   DT.ESI= 000044   INHEPR= 020000   MSGSFT= 000006   SAVREG= ****** G
BIT04  = 000020   DT.EVN= 000000   INHREL= 001000   MSGSKE= 000003   SBADR  = 000102
BIT05  = 000040   DT.EXS= 000060   INHRRE= 000400   MSGSMB= 000015   SBKMOD= 000000
BIT06  = 000100   DT.FCH= 000037   INIT   = 000030   MSGSMH= 000014   SBKSEL= 010000
BIT07  = 000200   DT.FCN= 000036   INTR   = 000120   MSGSMS= 000016   SC.ADR= 000006
BIT08  = 000400   DT.HMX= 000104   IOMOD  = 100000   MSGSTD= 000000   SC.ALC= 000014
BIT09  = 001000   DT.KBE= 000024   IOMODP= 102000   MSGSYS= 000012   SC.APC= 000016
BIT1   = 000002   DT.KBP= 000026   IOMODR= 112000   MSGVEC= 000020   SC.CKL= 000002
BIT10  = 002000   DT.KBR= 000022   IOMODX= 110000   NBKMOD= 001000   SC.CKP= 000004
BIT11  = 004000   DT.KBU= 000030   JACK   = 035060   NCPUOP= 000020   SC.CLO= 000000
BIT12  = 010000   DT.MLS= 000032   KIPAR0= 172340   NDAPTY= 000002   SC.HLD= 000010
BIT13  = 020000   DT.MTI= 000110   KIPAR1= 172342   NULL   = 000000   SC.SCA= 000012
BIT14  = 040000   DT.OFF= 000070   KIPAR2= 172344   OWEN   = 024020   SENDLS= 177777
BIT15  = 100000   DT.PAS= 000074   KIPAR3= 172346   PAERR  = 000010   SOFCNT= 000042
BIT2   = 000004   DT.PC  = 000002   KIPAR4= 172350   PARPRE= 002000   SOFPAS= 000046
BIT3   = 000010   DT.PFL= 000062   KIPAR5= 172352   PARSTA= 000100   SPACE  = 000040
BIT4   = 000020   DT.PSW= 000004   KIPAR6= 172354   PASCNT= 000034   SPOINT= 000032
BIT5   = 000040   DT.PTA= 000064   KIPAR7= 172356   PDPLSI= 020000   SPVALU= 002200
BIT6   = 000100   DT.RCS= 000102   KIPDR0= 172300   PDP60  = 004000   SR0    = 177572
BIT7   = 000200   DT.REL= 000040   KIPDR1= 172302   PDP70  = 010000   SR1    = 177574
BIT8   = 000400   DT.SCT= 000066   KIPDR2= 172304   PRI0   = 000000   SR2    = 177576
BIT9   = 001000   DT.SMX= 000106   KIPDR3= 172306   PRI1   = 000040   SR3    = 172516
BKDEF  = 000002   DT.SP  = 000006   KIPDR4= 172310   PRI4   = 000200   STAT   = 000026
BKMOD  = 000020   DT.SSI= 000046   KIPDR5= 172312   PRI5   = 000240   STATBI= 064757
BKMODE= 040000   DT.ST0= 000010   KIPDR6= 172314   PRI6   = 000300   STAT1  = 000027
BKSLSH= 000134   DT.ST1= 000012   KIPDR7= 172316   PRI7   = 000340   SUSPND= 000001
CAPRES= 000004   DT.SWR= 000056   KRUN   = ****** G PR0    = 000000   SVR0   = 000062
CASTAT= 000004   DT.SYP= 000072   KTERRO= 000040   PR4    = 000200   SVR1   = 000064
CDERCT= 000146   DT.WBU= 000050   KTPRES= 000400   PR5    = 000240   SVR2   = 000066
CDWDCT= 000144   DT.WHL= 000054   KTSTAT= 000020   PR6    = 000300   SVR3   = 000070
CKTIM  = 100000   DT.WLL= 000052   KTXTND= 040000   PR7    = 000340   SVR4   = 000072
CLKPRE= 000001   DVID1  = 000014   LF     = 000012   PS     = 177776   SVR5   = 000074
CONFIG= 000056   ECCMEM= 000100   LPSTAT= 000001   PSW    = 177776   SVR6   = 000076
CQOVF  = 000001   ECCSTA= 000010   MAPSTA= 000200   RANNUM= 000054   SYSCNT= 000052
CR     = 000015   ENBEOP= 010000   MED    = 076600   RBUFEA= 000130   SYSERR= 000100
CSRA   = 000100   ENBNUL= 000001   MEMPAS= 040000   RBUFPA= 000126   TABL   = 000000
CSRC   = 000102   ENDLST= 000000   MODEXH= 004000   RBUFSZ= 000132   TMPIO  = 000002
```

```
TQOVF = 000002          WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$DST = 000000
UIPAR0= 177640          WBUFEA= 000136          $F$DO = 000340          $ISK0 = 000001          $$ELOC= 000402
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LOCTA= 177777          $$ERFL= 000000
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000340
UIPAR3= 177646          WBUFSZ= 000142          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR4= 177650          WDFR  = 000116          $F$INC= 000210          $NESTL= 177777          $$LOC = 000066R
UIPAR5= 177652          WDTO  = 000114          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $NSK1 = 000120          $$REG = 177777
UIPAR7= 177656          WTWHMI= 000222          $F$NO = 000403          $SAVLE= 177777          $$RETU= 000000
UIPDR0= 177600          XFLAG = 000005          $F$OR = 000320          $SSK0 = 050005          $$RTN1= 050000
UIPDR1= 177602          XOFF  = 000023          $F$RTI= 000350          $TAGLE= 177777          $$RTN2= 050001
UIPDR2= 177604          XON   = 000021          $F$RTN= 000300          $TAGNU= 050006          $$SRC = 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TEMP = 000300          $$TGSV= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $TSK0 = 050004          $$TGS1= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $TSK1 = 050005          $$TGS2= 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$ARGC= 000002          $$TO  = 000000
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$BYTE= 000403          $$$TAG= 050000
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$CASE= 000000          .     = 000140R


. ABS.  000000    000
        000140    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DPSTRT,DSKZ:DPSTRT=SPMAC/ML,EQUATE,DPSTRT
 RUN-TIME: 13 2 .3 SECONDS
 RUN-TIME RATIO: 34/16=2.1
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                                .TITLE  DRPMOD DROP A MODULE FROM AN EXERCISER
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       DRPMOD
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS ROUTINE WILL DROP A MODULE FROM AN EXERCISER. IT WILL
                                ;       FIRST INSURE THAT THE MODULE BEING DROPPED HAS NOT ALREADY
                                ;       BEEN DROPPED. IF THE MODULE IS AN SBK OR NBK, IT WILL CLEAR
                                ;       THE MODHOLD BIT IN THE STATUS INDICATOR. THEN CLEAR THE
                                ;       MODULES ACTIVE BIT IN THE STATUS WORD AND SET THE DROPPED BIT
                                ;       IN THE MODULE'S STATUS WORD. THEN, SET THE CKTIM BIT.
                                ;       IF THE MODULE BEING DROPPED IS THE SYSTEM CLOCK MODULE, THE
                                ;       CLOCK PRESENT BIT IN THE CONFIGURATION WORD WILL BE CLEARED
                                ;       AND THE LOCATION SC.CLO WILL BE CLEARED.
                                ;
                                ; INPUTS:
                                ;       1. DATA TABLE ADDRESS
                                ;       2. MODULE HEADER ADDRESS
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.ST0
                                ;       DT.ST1
                                ;       DT.PC
                                ;       DT.ADDR
                                ;       DT.SCT
                                ;       DT.CF0
                                ;       DT.MLST
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       1. DT.MACT - NUMBER OF MODULES ACTIVE
                                ;       2. DT.ST0 - STATUS INDICATOR
                                ;       3. DT.MEOP - MODULES END OF PASS COUNT
                                ;       4. DT.CF0 - CONFIGURATION WORD 0
                                ;       5. DT.ST1 - STATUS INDICATOR 1
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       1. SAVREG
                                ;       2. RESREG
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL DRPMOD IN <DTADR,MODHDR>
                                ;       WHERE
                                ;       1. DTADR = ADDRESS OF DATA TABLE
                                ;       2. MODHDR - MODULE BEING DROPPED HEADER ADDRESS
```

```
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR DRPMOD

                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1

                000001          $LSTTAG=1
                000001          $LSTIN=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  DRPMOD          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  ENQTQ           ;ENQUEUE A MESSAGE
                                .GLOBL  DUNCHK          ;SEE IF ANY OTHER MODULES CAN RUN
                                .GLOBL  SAVREG          ;SAVE THE RECISTERS
                                .GLOBL  RESREG          ;RESTORE THE REGISTERS
                                ;
                                ;****************
                                ;
                                ;****************
```

```
                                .SBTTL  DRPMOD ROUTINE

                                ROUTINE DRPMOD <DTADR,MODHDR>
000000'                                                                 DRPMOD:
(2)  000000'

                                ;+
                                ; INIT AND SAVE DATA TABLE ADDRESS AND MODULE HEADER ADDRESS
                                ;-

                                CALL SAVREG
000000'                                                                 JSR     PC,SAVREG
(3)  000000' 004767  000000G
                                LET R0 := DTADR(R5)
000004'                                                                 MOV     DTADR(R5),R0
(4)  000004' 016500  000000
                                LET R1 := MODHDR(R5)
000010'                                                                 MOV     MODHDR(R5),R1
(4)  000010' 016501  000002


                                ;+
                                ; SET CKTIM BIT TO INDICATE THAT MODULE HAS BEEN
                                ; DROPPED. SET DROPPED BIT IN MODULE STAT WORD
                                ;-

                                LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
000014'                                                                 BIS     #CKTIM,DT.ST1(R0
(6)  000014' 052760  100000  000012


                                LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
000022'                                                                 BIC     #ACTBIT,STAT(R1)
(6)  000022' 042761  004000  000026
                                LET STAT(R1) := STAT(R1) SET.BY #BIT13
000030'                                                                 BIS     #BIT13,STAT(R1)
(6)  000030' 052761  020000  000026

                                ;+
                                ; GET A COPY OF THE STATUS WORD AND SEE IF NBK OR SBK MOD
                                ;-

                                LET R2 := STAT(R1)
000036'                                                                 MOV     STAT(R1),R2
(4)  000036' 016102  000026
                                LET R2 := R2 CLR.BY #STATBITS
000042'                                                                 BIC     #STATBITS,R2
(6)  000042' 042702  064757


                                IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
000046'                                                                 CMP     R2,#SBKMOD
(6)  000046' 020227  000000                                             BEQ     50002$
(8)  000052' 001403                                                     CMP     R2,#NBKMOD
(6)  000054' 020227  001000                                             BNE     50003$
(9)  000060' 001003
(6)  000062'                                                    50002$:


                                ;+
                                ; IT'S AN SBK OR NBK SO CLEAR THE MODHOLD BIT IN DT.ST0
                                ;-

                                        LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #MODHOLD
000062'                                                                 BIC     #MODHOLD,DT.ST0(
(6)  000062' 042760  002000  000010
```

```
                                ENDIF                                                   50003$:
                                ;+
                                ; GET ADDRESS OF SYS CLOCK TABLE
                                ;-
                                LET R2 := DT.SCT(R0)                                            MOV     DT.SCT(R0),R2
                                ;+
                                ; IF SYS CLOCK IS THE ONE BEING DROPPED - CLEAR OUT SC.CLO
                                ; AND CLOCK PRESENT BIT IN DT.CF0
                                ;-
                                IF R1 EQ SC.SCA(R2) THEN                                        CMP     R1,SC.SCA(R2)
                                                                                                BNE     50004$
                                ;+
                                ; YUP - IT'S THE SYS CLOCK
                                ;-
                                        LET SC.CLO(R2) := #0                                    CLR     SC.CLO(R2)
                                        LET DT.CF0(R0) := DT.CF0(R0) CLR.BY #CLKPRES            BIC     #CLKPRES,DT.CF0(
                                ENDIF                                                   50004$:
                                ;+
                                ; SEE IF ANY MORE MODULES CAN RUN
                                ;-
                                CALL DUNCHK IN <R0,R1>                                          MOV     R5,-(SP)
                                                                                                MOV     R1,-(R5)
                                                                                                MOV     R0,-(R5)
                                                                                                JSR     PC,DUNCHK
                                                                                                MOV     (SP)+,R5
                                ;+
                                ; ENQUE THE MESSAGE
                                ;-
                                LET R3 := DT.PC(R0) - #2                                        MOV     DT.PC(R0),R3
                                                                                                SUB     #2,R3
                                CALL ENQTQ IN <R0,#MSGDRP,R3,R1,#0>                             MOV     R5,-(SP)
                                                                                                MOV     #0,-(R5)
                                                                                                MOV     R1,-(R5)
```

```
(6)  000150' 010345                                                          MOV     R3,-(R5)
(5)  000152' 012745  000017                                                  MOV     #MSGDRP,-(R5)
(4)  000156' 010045                                                          MOV     R0,-(R5)
(3)  000160' 004767  000000G                                                 JSR     PC,ENQTQ
(3)  000164' 012605                                                          MOV     (SP)+,R5
679
680
681                                  ;+
682                                  ; CLEAN UP
683                                  ;-
684
685  000166'                         CALL RESREG
(3)  000166' 004767  000000G                                                 JSR     PC,RESREG
686
687  000172'                         ENDRTN
(3)  000172'                                                         50000$:
(3)  000172'                                                         50001$:
(2)  000172' 000207                                                          RTS     PC
688
689
690          000001                 .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD  000000RG
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DUNCHK= ****** G
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHDR= 000002
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$DST  = 000000
TMPIO = 000002          WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$ELOC= 000402
TQOVF = 000002          WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$ERFL= 000000
UIPAR0= 177640          WBUFEA= 000136          $F$DO = 000340          $ISK0 = 000001          $$FLAG= 000001
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $LOCTA= 177777          $$FROM= 000000
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LSTIN= 000001          $$LOC = 000100R
UIPAR3= 177646          WBUFSZ= 000142          $F$IF = 000110          $LSTTA= 000001          $$LOCN= 000000
UIPAR4= 177650          WDFR  = 000116          $F$INC= 000210          $NESTL= 177777          $$REG = 177777
UIPAR5= 177652          WDTO  = 000114          $F$LOO= 000200          $NSK0 = 000300          $$RETU= 000000
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $NSK1 = 000110          $$RTN1= 050000
UIPAR7= 177656          WTWHMI= 000222          $F$NO = 000403          $SAVLE= 177777          $$RTN2= 050001
UIPDR0= 177600          XFLAG = 000005          $F$OR = 000320          $TAGLE= 177777          $$SRC = 000000
UIPDR1= 177602          XOFF  = 000023          $F$RTI= 000350          $TAGNU= 050005          $$TGSV= 000000
UIPDR2= 177604          XON   = 000021          $F$RTN= 000300          $TEMP = 000300          $$TGS1= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TSK0 = 050004          $$TGS2= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $$ARGC= 000004          $$TO  = 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $$BYTE= 000403          $$$TAG= 050000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $$CASE= 000000          .     = 000174R

. ABS.  000000    000
        000174    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DRPMOD,DSKZ:DRPMOD=SPMAC/ML,EQUATE,DRPMOD
 RUN-TIME: 13 3 .4 SECONDS
 RUN-TIME RATIO: 38/17=2.1
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  DUNCHK - SEE IF ANY MORE MODULES CAN RUN
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       DUNCHK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE CHECKS ALL OPTION MODULES TO SEE IF ANY OPTION
                        ;       MODULE CAN POSSIBLY RUN. IF NOT, AND THE EXERCISER IS RELOCATED,
                        ;       A RELOCATION BACK TO LOWEST MEMORY OCCURS AND A RETURN TO CMD
                        ;       MODE TAKES PLACE. A MESSAGE INDICATING THAT THE
                        ;       RUN IS OVER AND THE LAST MODULE IS DROPPED IS OUTPUTTED.
                        ;       IF MORE MODULES CAN RUN, A RETURN SIMPLY OCCURS.
                        ;
                        ; INPUTS:
                        ;       1. DTABLE
                        ;       2. MODULE HEADER ADDRESS (OF MODULE BEING DROPPED)
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.MLST
                        ;       2. DT.ADDR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.STO
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. PRRLOC
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;       4. MSGDHOOK
                        ;       5. APTKIL
                        ;       6. APTDED
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL DUNCHK IN <DT,AD> WHERE DT = DATA TABLE ADDRESS
                        ;       AND AD = MODULE HEADER ADDRESS.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR DUNCHK

                                .MCALL  STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTTAG=1
        000001                  $LSTIN=1


                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  DUNCHK
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                .GLOBL  PRRLOC          ;RELOCATE CODE
                                .GLOBL  MSGDHOOK        ;O/P A MESSAGE
                                .GLOBL  DX.RSTRT        ;RESTART ADDRESS
                                .GLOBL  APTDED          ;PUT APT IN THE COFFIN
                                .GLOBL  APTKIL          ;KILL APT FATALLY
                                ;
                                ;
                                ;***************
                                ;

                                ;***************
                                ;

                                ; LOCAL STORAGE:
                                ;
000000'   045                   DR.WOW: .ASCII  /%/
000001' 000005                  DR.NAM: .BLKB   5
000006' 042040  047522  050120          .ASCII  / DROPPED/
000014' 042105
000016' 020445  020041  052522          .ASCIZ  /%!! RUN OVER - ALL MODULES DROPPED !!%/
000024' 020116  053117  051105
000032' 026440  040440  046114
000040' 046440  042117  046125
000046' 051505  042040  047522
000054' 050120  042105  020440
000062' 022441     000
        000066'                 .EVEN
                                ;
```

```
607  000066'                          ROUTINE DUNCHK <DTB,MOD>
(2)  000066'                                                                DUNCHK:
608  000066'                          CALL SAVREG
(3)  000066' 004767  000000G                                                JSR     PC,SAVREG
609  000072'                          LET R0 := DTB(R5)
(4)  000072' 016500  000000                                                 MOV     DTB(R5),R0
610  000076'                          LET R1 := MOD(R5)
(4)  000076' 016501  000002                                                 MOV     MOD(R5),R1
611                                   ;+
612                                   ; IF UNDER APT AND WE GOT HERE (TO DUNCHK) IT WAS PROBABLY THE CLOCK
613                                   ; THAT DID IT - SO CALL THE APTDED ROUTINE WHICH WILL PUT THE
614                                   ; DROPPED MODULE NAME INTO APT MAILBOX....THEN CALL APTKIL,
615                                   ; WHICH WILL HALT THE EXERCISER FATALLY......
616                                   ;-
617
618
619  000102'                          IF #APTPRES SETIN DT.CF0(R0) THEN
(6)  000102' 032760  000200  000014                                         BIT     #APTPRES,DT.CF0(
(9)  000110' 001413                                                         BEQ     50002$
620  000112'                                 CALL APTDED IN <R0,R1>
(3)  000112' 010546                                                         MOV     R5,-(SP)
(5)  000114' 010145                                                         MOV     R1,-(R5)
(4)  000116' 010045                                                         MOV     R0,-(R5)
(3)  000120' 004767  000000G                                                JSR     PC,APTDED
(3)  000124' 012605                                                         MOV     (SP)+,R5
621  000126'                                 CALL APTKIL IN <R0>
(3)  000126' 010546                                                         MOV     R5,-(SP)
(4)  000130' 010045                                                         MOV     R0,-(R5)
(3)  000132' 004767  000000G                                                JSR     PC,APTKIL
(3)  000136' 012605                                                         MOV     (SP)+,R5
622  000140'                          ENDIF
(4)  000140'                                                                50002$:
623
624                                   ;+
625                                   ; SEE IF THERE ARE ANY POSSIBLE MODULES LEFT TO RUN...
626                                   ;-
627
628  000140'                          LET R2 := DT.MLST(R0)
(4)  000140' 016002  000032                                                 MOV     DT.MLST(R0),R2
629  000144'                          LET R4 := #0
(4)  000144' 005004                                                         CLR     R4
630  000146'                          WHILE (R2) NE #0 DO
(4)  000146'                                                                50003$:
(6)  000146' 005712                                                         TST     (R2)
(9)  000150' 001413                                                         BEQ     50004$
631  000152'                                 LET R3 := (R2)+
(4)  000152' 012203                                                         MOV     (R2)+,R3
632  000154'                                 IF #BIT14 SETIN STAT(R3) AND #BIT13 NOTSETIN STAT(R3) THEN
(6)  000154' 032763  040000  000026                                         BIT     #BIT14,STAT(R3)
(9)  000162' 001405                                                         BEQ     50005$
(6)  000164' 032763  020000  000026                                         BIT     #BIT13,STAT(R3)
(9)  000172' 001001                                                         BNE     50005$
633  000174'                                         LET R4 := R4 + #1
(6)  000174' 005204                                                         INC     R4
634  000176'                                 ENDIF
(4)  000176'                                                                50005$:
```

```
635  000176'                          ENDDO
(4)  000176' 000763                                                          BR      50003$
(3)  000200'                                                          50004$:
636
637
638
639                                   ;+
640                                   ; IF NO MORE CAN RUN, GET THE MODULE NAME INTO THE
641                                   ; MESSAGE THAT SAYS !! RUN OVER !! ETC. AND THEN O/P MESSAGE.
642                                   ;-
643
644  000200'                          IF R4 EQ #0 THEN
(6)  000200' 005704                                                          TST     R4
(9)  000202' 001046                                                          BNE     50006$
645  000204'                                 LET R2 := #DR.NAM
(4)  000204' 012702  000001'                                                 MOV     #DR.NAM,R2
646  000210'                                 LET R3 := #5
(4)  000210' 012703  000005                                                  MOV     #5,R3
647  000214'                                 WHILE R3 NE #0 DO
(4)  000214'                                                          50007$:
(6)  000214' 005703                                                          TST     R3
(9)  000216' 001403                                                          BEQ     50010$
648  000220'                                         LET (R2)+ :B= (R1)+
(4)  000220' 112122                                                          MOVB    (R1)+,(R2)+
649  000222'                                         LET R3 := R3 - #1
(6)  000222' 005303                                                          DEC     R3
650  000224'                                 ENDDO
(4)  000224' 000773                                                          BR      50007$
(3)  000226'                                                          50010$:
651  000226'                                 CALL MSGDHOOK IN <R0,#MSGPOP,#DR.WOW,#443$>
(3)  000226' 010546                                                          MOV     R5,-(SP)
(7)  000230' 012745  000260'                                                 MOV     #443$,-(R5)
(6)  000234' 012745  000000'                                                 MOV     #DR.WOW,-(R5)
(5)  000240' 012745  000002                                                  MOV     #MSGPOP,-(R5)
(4)  000244' 010045                                                          MOV     R0,-(R5)
(3)  000246' 004767  000000G                                                 JSR     PC,MSGDHOOK
(3)  000252' 012605                                                          MOV     (SP)+,R5
652  000254'                                 INLINE <1$:     NOP>
(2)  000254' 000240                                                   1$:    NOP
653  000256'                                 INLINE <BR      1$>
(2)  000256' 000776                                                          BR      1$
654  000260'                                 INLINE <443$:>
(2)  000260'                                                          443$:
655
656
657                                   ;+
658                                   ; IF RELOCATED, RELOCATE BACK DOWN TO ZERO
659                                   ;-
660
661  000260'                                 IF DT.ADDR(R0) NE #200 THEN
(6)  000260' 026027  000042  000200                                          CMP     DT.ADDR(R0),#200
(9)  000266' 001407                                                          BEQ     50011$
662  000270'                                         CALL PRRLOC IN <R0,#200>
(3)  000270' 010546                                                          MOV     R5,-(SP)
(5)  000272' 012745  000200                                                  MOV     #200,-(R5)
(4)  000276' 010045                                                          MOV     R0,-(R5)
```

```
(3)  000300' 004767  000000G                                                    JSR     PC,PRRLOC
(3)  000304' 012605                                                             MOV     (SP)+,R5
663  000306'                        ENDIF
(4)  000306'                                                                50011$:
664
665  000306'                        LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RUNMODE
(6)  000306' 042760  100000  000010                                            BIC     #RUNMODE,DT.ST0(
666
667  000314'                        INLINE < JMP    DX.RSTRT>
(2)  000314' 000167  000000G                                                    JMP     DX.RSTRT
668
669  000320'                    ENDIF
(4)  000320'                                                                50006$:
670
671
672
673
674                             ;+
675                             ; RESTORE REGISTERS
676                             ;-
677
678  000320'                    CALL RESREG
(3)  000320' 004767  000000G                                                    JSR     PC,RESREG
679  000324'                    ENDRTN
(3)  000324'                                                                50000$:
(3)  000324'                                                                50001$:
(2)  000324' 000207                                                             RTS     PC
680          000001            .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTDED= ****** G
APTFER= 000004
APTKIL= ****** G
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DR.NAM  000001R
DR.WOW  000000R
DSEVNT= 000014
DTB   = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DUNCHK  000066RG
DVID1 = 000014
DX.RST= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MOD   = 000002
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRRLOC= ****** G
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
```

```
SVR3  = 000070          UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$BYTE= 000403
SVR4  = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$CASE= 000000
SVR5  = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$DST = 000000
SVR6  = 000076          UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$ELOC= 000402
SYSCNT= 000052          WASADR= 000104          $F$DO = 000340          $LOCTA= 177777          $SERFL= 000000
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000001
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $NESTL= 177777          $$LOC = 000266R
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK1 = 000110          $$REG = 177777
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK2 = 000110          $$RETU= 000000
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $SAVLE= 177777          $$RTN1= 050000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $SSK0 = 050010          $$RTN2= 050001
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TAGLE= 177777          $$SRC = 000000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGNU= 050012          $$TGSV= 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TEMP = 000300          $$TGS1= 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TSK0 = 050006          $$TGS2= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK1 = 050011          $$TO  = 000000
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $TSK2 = 050010          $$$TAG= 050000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $$ARGC= 000004          .     = 000326R

. ABS.  000000    000
        000326    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:DUNCHK,DSKZ:DUNCHK=SPMAC/ML,EQUATE,DUNCHK
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 41/20=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE ERREC - ERROR RECOVERY
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       ERREC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       ON FATAL ERRORS, THE CURRENT RUN OF THE EXERCISER IS
                        ;       ABORTED.  IF THE ERROR IS NOT FATAL, THE EXERCISER IS
                        ;       RESTARTED.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ERR, DT.ADDR, DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ST0
                        ;
                        ; PATHOLGICAL CONNECTIONS:
                        ;       BUSY FLAG IN MSGDEQ (MD.BSY)
                        ;       MESSAGE CODE IN MSGDEQ (MD.COD)
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       MSGDHOOK                        ;HOOK MESSAGE TO PROPER DRIVER
                        ;       PRRLOC                          ;PROCESS RELOCATION
                        ;       WBFLIM                          ;DETERMINE WRITE BUFFER LIMITS
                        ;       UNIMAP                          ;LOAD UNIBUS MAP
                        ;       MSGDEQ                          ;MESSAGE DEQUEUER
                        ;       CLKON                           ;TURN ON SYSTEM CLOCK
                        ;       KBINI                           ;INITIALIZE KEYBOARD
                        ;       APTKIL                          ;KILL AN APT RUN
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ERREC IN <A>
                        ;               A=DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINTIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  ERREC                           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  MSGDHOOK
                                .GLOBL  APTKIL
                                .GLOBL  PRRLOC
                                .GLOBL  WBFLIM
                                .GLOBL  UNIMAP
                                .GLOBL  MSGDEQ
                                .GLOBL  CLKON
                                .GLOBL  KBINI
                                .GLOBL  DX.RSTST                        ;MONITOR'S RESTART LOCATION
                                .GLOBL  MD.BSY                          ;MSGDEQ BUSY FLAG
                                .GLOBL  MD.COD                          ;MESSAGE CODE IN MSGDEQ
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000'                         ER.TBL:                                 ;MESSAGE ADDRESS TABLE
000000' 000072'                         .WORD ER.CQO
000002' 000121'                         .WORD ER.TQO
000004' 000145'                         .WORD ER.APT
000006' 000151'                         .WORD ER.PAR
000010' 000167'                         .WORD ER.REL
000012' 000202'                         .WORD ER.KT
000014' 000205'                         .WORD ER.SYS

                                ; - MESSAGES -
000016' 043045  052101  046101  ER.MG1: .ASCIZ /%FATAL ERROR%TOO MANY /
000024' 042440  051122  051117
000032' 052045  047517  046440
000040' 047101  020131     000
000045'    040  051105  047522  ER.MG2: .ASCIZ / ERRORS%RUN ABORTED%/
000052' 051522  051045  047125
000060' 040440  047502  052122
000066' 042105  000045

```

```
                                    ;+
                                    ; THE FOLLOWING MESSAGES MUST BE IN THE SAME ORDER AS THEY ARE
                                    ; IN THE ERROR WORD OF THE DATA TABLE, I. E., THE ERROR
                                    ; MESSAGE ASSOCIATED WITH BIT 0 MUST BE FIRST, THEN THE ERROR MESSAGE
                                    ; ASSOCIATED WITH BIT 1, ETC.
                                    ;-

000072' 047503  052116  047522  ER.CQO: .ASCIZ /CONTROL QUEUE OVERFLOW/
000100' 020114  052521  052505
000106' 020105  053117  051105
000114' 046106  053517     000
000121'    124  050131  020105  ER.TQO: .ASCIZ /TYPE QUEUE OVERFLOW/
000126' 052521  052505  020105
000134' 053117  051105  046106
000142' 053517     000
000145'    101  052120     000  ER.APT: .ASCIZ /APT/
000151'    120  051101  052111  ER.PAR: .ASCIZ /PARITY OR ECC/
000156' 020131  051117  042440
000164' 041503     000
000167'    122  046105  041517  ER.REL: .ASCIZ /RELOCATION/
000174' 052101  047511  000116
000202' 052113     000          ER.KT:  .ASCIZ /KT/
000205'    123  051531  042524  ER.SYS: .ASCIZ /SYSTEM/
000212' 000115

                                    .EVEN
```

```
                                .SBTTL ERREC ROUTINE

                                ROUTINE ERREC <TABL>
000214'                                                         ERREC:
(2)  000214'

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
000214'                                                                 JSR     PC,SAVREG
(3)  000214' 004767  000000G

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE.
                                ;-

                                LET R0 := TABL(R5)
000220'                                                                 MOV     TABL(R5),R0
(4)  000220' 016500  000000

                                ;+
                                ; SEE IF THE ERROR WAS FATAL.
                                ;-

                                IF #FATERR SETIN DT.ERR(R0) THEN
000224'                                                                 BIT     #FATERR,DT.ERR(R
(6)  000224' 032760  100000  000020                                    BEQ     50002$
(9)  000232' 001532

                                ;+
                                ; THE ERROR WAS FATAL.
                                ;-


                                ;+
                                ; DETERMINE THE TYPE OF ERROR.  THIS IS DONE BY FIRST COPYING THE CONTENTS
                                ; OF THE ERROR WORD (EXCEPT THE FATAL ERROR BIT) INTO TEMPORARY
                                ; STORAGE (R2).  THEN THE BIT NUMBER (R4) OF THE SET ERROR INDICATOR
                                ; IS DETERMINED.
                                ;-

                                        LET R2 := DT.ERR(R0) CLR.BY #100000
000234'                                                                 MOV     DT.ERR(R0),R2
(4)  000234' 016002  000020                                             BIC     #100000,R2
(6)  000240' 042702  100000
                                        LET R3 := #000001
000244'                                                                 MOV     #000001,R3
(4)  000244' 012703  000001
                                        LET R4 := #-1
000250'                                                                 MOV     #-1,R4
(4)  000250' 012704  177777
                                        REPEAT
000254'                                                         50003$:
(3)  000254'
                                                LET R2 := R2 CLR.BY R3
000254'                                                                 BIC     R3,R2
(6)  000254' 040302
                                                LET R4 := R4 + #1
000256'                                                                 INC     R4
(6)  000256' 005204
```

```
672  000260'                                    LET R3 := R3 SHIFT #+1                    ASL     R3
(7)  000260' 006303
673  000262'                                  UNTIL R2 EQ #0                              TST     R2
(3)  000262' 005702                                                                      BNE     50003$
(6)  000264' 001373
674
675
676                                           ;+
677                                           ; NOW DOUBLE THE ERROR BIT NUMBER (R4) AND USE IT AS AN OFFSET INTO
678                                           ; THE MESSAGE ADDRESS TABLE, THEN OUTPUT THE PROPER MESSAGE.
679                                           ;-
680
681  000266'                                    LET R4 := R4 SHIFT #+1                    ASL     R4
(7)  000266' 006304
682
683  000270'                                    LET R1 := #ER.TBL + R4                    MOV     #ER.TBL,R1
(4)  000270' 012701  000000'                                                             ADD     R4,R1
(6)  000274' 060401
684
685  000276'                                    CALL MSGDHOOK IN <R0,#MSGPOP,#ER.MG1,#2$>
(3)  000276' 010546                                                                      MOV     R5,-(SP)
(7)  000300' 012745  000330'                                                             MOV     #2$,-(R5)
(6)  000304' 012745  000016'                                                             MOV     #ER.MG1,-(R5)
(5)  000310' 012745  000002                                                              MOV     #MSGPOP,-(R5)
(4)  000314' 010045                                                                      MOV     R0,-(R5)
(3)  000316' 004767  000000G                                                             JSR     PC,MSGDHOOK
(3)  000322' 012605                                                                      MOV     (SP)+,R5
686  000324'                                    INLINE <1$: NOP>                          1$: NOP
(2)  000324' 000240
687  000326'                                    INLINE <BR 1$>                            BR 1$
(2)  000326' 000776
688  000330'                                    INLINE <2$:>                              2$:
(2)  000330'
689  000330'                                    CALL MSGDHOOK IN <R0,#MSGPOP,(R1),#4$>
(3)  000330' 010546                                                                      MOV     R5,-(SP)
(7)  000332' 012745  000360'                                                             MOV     #4$,-(R5)
(6)  000336' 011145                                                                      MOV     (R1),-(R5)
(5)  000340' 012745  000002                                                              MOV     #MSGPOP,-(R5)
(4)  000344' 010045                                                                      MOV     R0,-(R5)
(3)  000346' 004767  000000G                                                             JSR     PC,MSGDHOOK
(3)  000352' 012605                                                                      MOV     (SP)+,R5
690  000354'                                    INLINE <3$: NOP>                          3$: NOP
(2)  000354' 000240
691  000356'                                    INLINE <BR 3$>                            BR 3$
(2)  000356' 000776
692  000360'                                    INLINE <4$:>                              4$:
(2)  000360'
693  000360'                                    CALL MSGDHOOK IN <R0,#MSGPOP,#ER.MG2,#6$>
(3)  000360' 010546                                                                      MOV     R5,-(SP)
(7)  000362' 012745  000412'                                                             MOV     #6$,-(R5)
(6)  000366' 012745  000045'                                                             MOV     #ER.MG2,-(R5)
(5)  000372' 012745  000002                                                              MOV     #MSGPOP,-(R5)
(4)  000376' 010045                                                                      MOV     R0,-(R5)
(3)  000400' 004767  000000G                                                             JSR     PC,MSGDHOOK
(3)  000404' 012605                                                                      MOV     (SP)+,R5
694  000406'                                    INLINE <5$: NOP>
```

```
   (2)  000406' 000240                                                                  5$: NOP
   695  000410'                              INLINE <BR 5$>
   (2)  000410' 000776                                                                  BR 5$
   696
   697  000412'                         INLINE <6$:>
   (2)  000412'                                                                         6$:
   698
   699
   700                                  ;+
   701                                  ; IF THE EXERCISER IS NOT AT THE LOWEST PART OF MEMORY, MOVE IT, RESET
   702                                  ; THE WRITE BUFFER, AND RELOAD THE UNIBUS MAP IF IT EXISTS.
   703                                  ;-
   704
   705  000412'                              IF DT.ADDR(R0) NE #200 THEN
   (6)  000412' 026027  000042  000200                                                 CMP     DT.ADDR(R0),#200
   (9)  000420' 001425                                                                  BEQ     50004$
   706  000422'                                  CALL PRRLOC IN <R0,#200>
   (3)  000422' 010546                                                                  MOV     R5,-(SP)
   (5)  000424' 012745  000200                                                          MOV     #200,-(R5)
   (4)  000430' 010045                                                                  MOV     R0,-(R5)
   (3)  000432' 004767  000000G                                                        JSR     PC,PRRLOC
   (3)  000436' 012605                                                                  MOV     (SP)+,R5
   707  000440'                                  CALL WBFLIM IN <R0>
   (3)  000440' 010546                                                                  MOV     R5,-(SP)
   (4)  000442' 010045                                                                  MOV     R0,-(R5)
   (3)  000444' 004767  000000G                                                        JSR     PC,WBFLIM
   (3)  000450' 012605                                                                  MOV     (SP)+,R5
   708  000452'                                  IF #ADDR22 SETIN DT.CF0(R0) THEN
   (6)  000452' 032760  001000  000014                                                 BIT     #ADDR22,DT.CF0(R
   (9)  000460' 001405                                                                  BEQ     50005$
   709  000462'                                      CALL UNIMAP IN <R0>
   (3)  000462' 010546                                                                  MOV     R5,-(SP)
   (4)  000464' 010045                                                                  MOV     R0,-(R5)
   (3)  000466' 004767  000000G                                                        JSR     PC,UNIMAP
   (3)  000472' 012605                                                                  MOV     (SP)+,R5
   710  000474'                                  ENDIF
   (4)  000474'                                                                 50005$:
   711  000474'                              ENDIF
   (4)  000474'                                                                 50004$:
   712
   713
   714                                  ;+
   715                                  ; CLEAR THE RUN MODE BIT.
   716                                  ;-
   717
   718  000474'                              LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RUNMODE
   (6)  000474' 042760  100000  000010                                                 BIC     #RUNMODE,DT.ST0(
   719
   720                                  ;+
   721                                  ; IF AN APT FATAL ERROR ... CALL THE APT KILL ROUTINE
   722                                  ;-
   723
   724
   725
   726
   727  000502'                              IF #APTFER SETIN DT.ERR(R0) THEN
```

```
(6)  000502' 032760  000004  000020                                          BIT     #APTFER,DT.ERR(R
(9)  000510' 001402                                                          BEQ     50006$
728  000512'                                  CALL APTKIL
(3)  000512' 004767  000000G                                                JSR     PC,APTKIL
729  000516'                              ENDIF
(4)  000516'                                                         50006$:
730
731  000516'                          ELSE                                   BR      50007$
(4)  000516' 000431                                                  50002$:
(3)  000520'
732
733                                   ;+
734                                   ; THE ERROR WAS NOT FATAL.
735                                   ;-
736
737
738                                   ;+
739                                   ; ENABLE THE KEYBOARD INPUT.
740                                   ;-
741
742
743  000520'                                  CALL KBINI IN <R0>
(3)  000520' 010546                                                          MOV     R5,-(SP)
(4)  000522' 010045                                                          MOV     R0,-(R5)
(3)  000524' 004767  000000G                                                JSR     PC,KBINI
(3)  000530' 012605                                                          MOV     (SP)+,R5
744
745
746                                   ;+
747                                   ; WAIT FOT THE TYPE QUEUE TO EMPTY.
748                                   ;-
749
750  000532'                                  LET MD.BSY := #0
(4)  000532' 005067  000000G                                                CLR     MD.BSY
751  000536'                                  REPEAT
(3)  000536'                                                         50010$:
752  000536'                                      CALL MSGDEQ IN <R0>
(3)  000536' 010546                                                          MOV     R5,-(SP)
(4)  000540' 010045                                                          MOV     R0,-(R5)
(3)  000542' 004767  000000G                                                JSR     PC,MSGDEQ
(3)  000546' 012605                                                          MOV     (SP)+,R5
753  000550'                                      UNTIL MD.COD EQ #MSGNUL
(3)  000550' 026727  000000G 177775                                          CMP     MD.COD,#MSGNUL
(6)  000556' 001367                                                          BNE     50010$
754
755
756                                   ;+
757                                   ; TURN ON THE SYSTEM CLOCK IF IT EXISTS.
758                                   ;-
759
760  000560'                                  IF #CLKPRES SETIN DT.CF0(R0) THEN
(6)  000560' 032760  000001  000014                                          BIT     #CLKPRES,DT.CF0(
(9)  000566' 001405                                                          BEQ     50011$
761  000570'                                      CALL CLKON IN <R0>
(3)  000570' 010546                                                          MOV     R5,-(SP)
(4)  000572' 010045                                                          MOV     R0,-(R5)
```

```
   (3)  000574' 004767  000000G                                                    JSR     PC,CLKON
   (3)  000600' 012605                                                             MOV     (SP)+,R5
  762   000602'                                ENDIF
   (4)  000602'                                                          50011$:
  763
  764
  765   000602'                               ENDIF
   (4)  000602'                                                          50007$:
  766
  767                                        ;+
  768                                        ; RESTORE REGISTERS
  769                                        ;-
  770
  771   000602'                              CALL RESREG
   (3)  000602' 004767  000000G                                                    JSR     PC,RESREG
  772
  773
  774                                        ;+
  775                                        ; RESTART THE MONITOR.
  776                                        ;-
  777
  778   000606'                              INLINE <JMP DX.RSTRT>
   (2)  000606' 000167  000000G                                                    JMP DX.RSTRT
  779
  780
  781   000612'                              ENDRTN
   (3)  000612'                                                          50000$:
   (3)  000612'                                                          50001$:
   (2)  000612' 000207                                                             RTS     PC
  782           000001                      .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTKIL= ****** G
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKON = ****** G
CLKPRE= 000001
CONFIG= 000056
CQCVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.RST= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERREC   000214RG
ERRTYP= 000106
ER.APT  000145R
ER.CQO  000072R
ER.KT   000202R
ER.MG1  000016R
ER.MG2  000045R
ER.PAR  000151R
ER.REL  000167R
ER.SYS  000205R
ER.TBL  000000R
ER.TQO  000121R
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KBINI = ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.BSY= ****** G
MD.COD= ****** G
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDHO= ****** G
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRIO  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRRLOC= ****** G
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
```

```
SR1    = 177574          UIPAR4= 177650          WTWHMI= 000222          $F$THE= 000330          $TSK2 = 050005
SR2    = 177576          UIPAR5= 177652          XFLAG = 000005          $F$TRU= 000404          $$ARGC= 000002
SR3    = 172516          UIPAR6= 177654          XOFF  = 000023          $F$UNT= 000130          $$BYTE= 000403
STAT   = 000026          UIPAR7= 177656          XON   = 000021          $F$WHI= 000120          $$CASE= 000000
STATBI= 064757           UIPDR0= 177600          $BGNLE= 177777          $F$YES= 000402          $$DST = 000000
STAT1  = 000027          UIPDR1= 177602          $ERFLG= 000400          $IFLEV= 177777          $$ELOC= 000402
SUSPND= 000001           UIPDR2= 177604          $F$AND= 000310          $ISK0 = 000001          $$ERFL= 000000
SVR0   = 000062          UIPDR3= 177606          $F$BAD= 000401          $ISK1 = 000001          $$FLAG= 000001
SVR1   = 000064          UIPDR4= 177610          $F$BLA= 000170          $ISK2 = 000001          $$FROM= 000000
SVR2   = 000066          UIPDR5= 177612          $F$CAS= 000150          $LOCTA= 177777          $$LOC = 000566R
SVR3   = 000070          UIPDR6= 177614          $F$DEC= 000220          $LSTIN= 000001          $$LOCN= 000000
SVR4   = 000072          UIPDR7= 177616          $F$DO = 000340          $LSTTA= 000001          $$REG = 177777
SVR5   = 000074          UNIMAP= ****** G        $F$FAL= 000405          $NESTL= 177777          $$RETU= 000000
SVR6   = 000076          WASADR= 000104          $F$GDO= 000400          $NSK0 = 000300          $$RTN1= 050000
SYSCNT= 000052           WBFLIM= ****** G        $F$IF = 000110          $NSK1 = 000110          $$RTN2= 050001
SYSERR= 000100           WBSTAT= 000040          $F$INC= 000210          $NSK2 = 000110          $$SRC = 000000
TABL   = 000000          WBUFEA= 000136          $F$LOG= 000200          $NSK3 = 000110          $$TGSV= 000000
TMPIO  = 000002          WBUFPA= 000134          $F$NAM= 000160          $SAVLE= 177777          $$TGS1= 000000
TQOVF  = 000002          WBUFRQ= 000140          $F$NO = 000403          $TAGLE= 177777          $$TGS2= 000000
UIPAR0= 177640           WBUFSZ= 000142          $F$OR = 000320          $TAGNU= 050012          $$TO  = 000000
UIPAR1= 177642           WDFR   = 000116         $F$RTI= 000350          $TEMP = 000300          $$$TAG= 050000
UIPAR2= 177644           WDTO   = 000114         $F$RTN= 000300          $TSK0 = 050007          .     = 000614R
UIPAR3= 177646           WTINRE= 000352          $F$SEL= 000140          $TSK1 = 050011

. ABS.  000000     000
        000614     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:ERREC,DSKZ:ERREC=SPMAC/ML,EQUATE,ERREC
 RUN-TIME: 15 5 .4 SECONDS
 RUN-TIME RATIO: 44/22=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       FILLMSG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE IS RESPONSIBLE FOR FIILLING IN THE BLANK FIELDS
                        ;       OF THE MONITOR GENERATED SKELETAL MESSAGES.
                        ;       THE TYPES OF SKELETAL MESSAGES SUPPORTED ARE:
                        ;               1. END OF PASS                  2. HARD ERROR
                        ;               3. SUM COMMAND                  4. SOFT ERROR
                        ;               5. COMMON HEADER                6. DROP MODULE
                        ;               7. DATA ERROR                   8. BAD VECTOR
                        ;               9. MAP SUMMARY                  10. CKDATA CALL
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE
                        ;       2. MSG TYPE CODE
                        ;       3. MSG ADDR/SPECIAL VALUE
                        ;       4. MODULE'S HDR ADDR
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       1. ADDRESS OF FILLED-IN MESSAGE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. BDACNV               ;BINARY TO DECIMAL ASCII CONVERSION
                        ;       2. BOAC                 ;BINARY TO OCTAL-ASCII CONVERSION
                        ;       3. GPA                  ;GET PHYSICAL ADDRESS
                        ;       4. SAVREG               ;
                        ;       5. RESREG               ;
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL FILLMSG IN <DTABLE,TYPCODE,MSGADDR,HDRADDR> OUT <FILMSGADR>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
000000'                                 $LSTIN = 1
                000001                  $LSTTAG = 1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  FILLMSG                 ;MODULE'S ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  BDACNV                  ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
                                .GLOBL  BOA16                   ;BINARY TO OCTAL-ASCII CONVERSION ROUTINE
                                .GLOBL  BOAC                    ;BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  CKHUNG                  ;CHECK FOR HUNG MODULES ROUTINE
                                .GLOBL  GPA                     ;GET PHYSICAL ADDRESS ROUTINE
                                .GLOBL SAVREG                   ;
                                .GLOBL RESREG                   ;
                                ;
                                ;***************
                                ;
                                ; LOCAL EQUATES:
                                ;
                000010          FLDMAX  = ^D<8>                 ;MAX FIELDS PER LINE
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  FLDCNT: .WORD   0               ;FIELD COUNTER
000002' 000000                  LINECNT: .WORD  0               ;LINE COUNTER
000004'                         TMP1:                           ;STORAGE WORD
000004' 000000                  TMPVA:  .WORD   0               ;STORAGE/VIRTUAL ADDRESS
000006' 000000                  TMPPA:  .WORD   0               ;STORAGE/PHYSICAL ADDRESS
000010' 000000                  TMPEA:  .WORD   0               ;STORAGE/EA BITS
000012' 000000                  FM.PA:  .WORD   0               ;STORAGE - PHYS. ADDR.
000014' 000000                  FM.EA:  .WORD   0               ;STORAGE - EA BITS
                                ;
                                ;***************
                                ;
```

```
                                .SBTTL  MESSAGE - END-OF-PASS


000016'    045                  EOPMSG: .ASCII /%/
000017' 000005                  EOPNAM: .BLKB   ^D<5>
000024' 042440  042116  050040          .ASCII  / END PASS #/
000032' 051501  020123     043
000037' 000005                  EOPPAS: .BLKB   ^D<5>
000044' 000045                          .ASCIZ  /%/
```

```
                        .SBTTL  MESSAGE - SUM COMMAND HEADER LINE (JUST A CR FOR THIS MONITOR)

                        
000046' 000045          RNHMSG: .ASCIZ  /%/
```

```
                                .SBTTL  MESSAGE - SUM COMMAND MODULE LINE

000050'    045                  RNBMSG: .ASCII /%/
000051' 000005                  RNBNAM: .BLKB   ^D<5>
000056' 040440  020124  040526          .ASCII  / AT VA: /
000064' 020072
000066' 000006                  RNBPC:  .BLKB   ^D<6>
000074' 051440  040524  020124          .ASCII  / STAT /
000102' 000006                  RNBSTAT: .BLKB  ^D<6>
000110' 050040  051501  020123          .ASCII  / PASS #/
000116'    043
000117' 000005                  RNBPAS: .BLKB   ^D<5>
000124' 044040  042122  051105          .ASCII  / HRDERRS /
000132' 051522     040
000135' 000005                  RNBHRD: .BLKB   ^D<5>
000142' 051440  052106  051105          .ASCII  / SFTERRS /
000150' 051522     040
000153' 000005                  RNBSFT: .BLKB   ^D<5>
000160'    000                          .BYTE 0
```

```
                                        .SBTTL  MESSAGE - SUM COMMAND SUMMARY LINE

000161'     045  054523  052123  RNSMSG: .ASCII  /%SYSTEM ERRORS: /
000166' 046505  042440  051122
000174' 051117  035123     040
000201' 000005                  RNSERR: .BLKB   ^D<5>
000206' 020040  020040  050040          .ASCII  /     POWER FAILS: /
000214' 053517  051105  043040
000222' 044501  051514  020072
000230' 000005                  RNSPWR: .BLKB   ^D<5>
000235'     045     000                .ASCIZ /%/
```

```
                                .SBTTL  MESSAGE - COMMON HEADER

000237'     045                 HDRMSG: .ASCII /%/
000240' 000005                  HDRNAM: .BLKB   ^D<5>
000245'     040  040520  020072         .ASCII  / PA: /
000252' 000010                  HDRPA:  .BLKB   ^D<8>
000262' 040440  041520  020072         .ASCII / APC: /
000270' 000006                  HDRPC:  .BLKB 6
000276' 050040  051501  020123         .ASCII  / PASS #/
000304'     043
000305' 000005                  HDRPAS: .BLKB   ^D<5>
000312' 000045                          .ASCIZ /%/

```

```
                                .SBTTL  MESSAGE - HRDERR,SOFERR

000314' 045                     DATAMSG:        .ASCII /%/
000315' 000005                  DATNAM: .BLKB   ^D<5>
000322' 050040  035101  040             .ASCII  / PA: /
000327' 000010                  DATPC:  .BLKB   ^D<8>
000337' 040     050101  035103          .ASCII  / APC: /
000344' 040
000345' 000006                  DATAPC: .BLKB   ^D<6>
000353' 040     040520  051523          .ASCII  / PASS #/
000360' 021440
000362' 000005                  DATPAS: .BLKB   ^D<5>
000367' 040                             .ASCII  / /
000370' 000004                  DATTYP: .BLKB   ^D<4>                   ;DATA, HARD, SOFT
000374' 042440  051122  043             .ASCII  / ERR#/
000401' 000005                  DATNUM: .BLKB   ^D<5>
000406' 041445  051123  035101          .ASCII /%CSRA: /
000414' 040
000415' 000006                  DATADD: .BLKB   ^D<6>
000423' 040     051503  041522          .ASCII  / CSRC: /
000430' 020072
000432' 000006                  DATCON: .BLKB   ^D<6>
000440' 040440  052123  052101          .ASCII  / ASTAT: /
000446' 020072
000450' 000006                  DATSTAT: .BLKB  ^D<6>
000456' 042440  051122  054524          .ASCII  / ERRTYP: /
000464' 035120  040
000467' 000006                  DATERR: .BLKB   ^D<6>
000475' 045                             .ASCII /%/
000476' 000                     DATEND: .BYTE   0                       ;END OF MESSAGE INDICATOR

```

```
                                .SBTTL  MESSAGE - HRDERR,SOFERR,EXTENSION

000477'                         EXTMSG:
000477' 000344                          .BLKB   ^D<228>
001043'    000                  EXTEND: .BYTE 0

```

```
                                .SBTTL  MESSAGE - DROP MODULE

001044' 045                     DRPMSG: .ASCII /%/
001045' 000005                  DRPNAM: .BLKB   ^D<5>
001052' 042040  047522  050120          .ASCII / DROPPED AT APC: /
001060' 042105  040440  020124
001066' 050101  035103     040
001073' 000006                  DRPPC:  .BLKB   ^D<6>
001101'    045     000                  .ASCIZ /%/

```

```
                                .SBTTL  MESSAGE - CKDATA,DATACK,DATERR

001103'    045                  CKDMSG: .ASCII /%/
001104' 000005                  CKDNAM: .BLKB   ^D<5>
001111'    040  040520  020072          .ASCII  / PA: /
001116' 000010                  CKDPA:  .BLKB   ^D<8>
001126' 040440  041520  020072          .ASCII  / APC: /
001134' 000006                  CKDVA:  .BLKB   ^D<6>
001142' 050040  051501  020123          .ASCII  / PASS #/
001150'    043
001151' 000005                  CKDPAS: .BLKB   ^D<5>
001156' 042440  051122  021440          .ASCII  / ERR #/
001164' 000005                  CKDNUM: .BLKB   ^D<5>
001171'    045  051503  040522          .ASCII  /%CSRA: /
001176' 020072
001200' 000006                  CKDADD: .BLKB   ^D<6>
001206' 051440  041057  020072          .ASCII  " S/B: "
001214' 000006                  CKDSB:  .BLKB   ^D<6>
001222' 053440  051501  020072          .ASCII  / WAS: /
001230' 000006                  CKDWAS: .BLKB   ^D<6>
001236' 053440  040522  051104          .ASCII  / WRADR /
001244'    040
001245' 000010                  CKDWRA: .BLKB   ^D<8>
001255'    040  042122  042101          .ASCII  / RDADR /
001262' 020122
001264' 000010                  CKDRDA: .BLKB   ^D<8>
001274' 000045                          .ASCIZ /%/
```

```
                                .SBTTL  MESSAGE - CKDATA/DATACK SUMMARY LINE

001276'                         CKDSMSG:
001276' 000005                  CKDSNAM: .BLKB  ^D<5>
001303'    040  040510  020104          .ASCII / HAD /
001310' 000005                  CKDSCNT: .BLKB ^D<5>
001315'    040  051105  047522          .ASCII / ERRORS OUT OF /
001322' 051522  047440  052125
001330' 047440  020106
001334' 000005                  CKDSRD: .BLKB ^D<5>
001341'    040  047527  042122          .ASCIZ / WORDS READ%/
001346' 020123  042522  042101
001354' 000045

                                        .EVEN
```

```
                                .SBTTL  FILLMSG - FILL-IN THE BLANKS ROUTINE


                                ROUTINE FILLMSG <DTABLE,TYPCODE, MSGADR, HDRADR,FLMSADR>
001356'                                                                         FILLMSG:


                                ;+
                                ; SAVE REGISTERS AND SAVE DTABLE,ADR,MSGADR AND TYPCODE
                                ;-

                                CALL SAVREG
001356' 004767  000000G                                                 JSR     PC,SAVREG
001362'                         LET R0 := DTABLE(R5)
001362' 016500  000000                                                  MOV     DTABLE(R5),R0
001366'                         LET R1 := HDRADR(R5)
001366' 016501  000006                                                  MOV     HDRADR(R5),R1
001372'                         LET R2 := MSGADR(R5)
001372' 016502  000004                                                  MOV     MSGADR(R5),R2
001376'                         LET R4 := TYPCODE(R5)
001376' 016504  000002                                                  MOV     TYPCODE(R5),R4
```

```
                                .SBTTL  PROCESS END-OF-PASS TYPE MESSAGE



                                ;+
                                ; IS IT AN END-OF-PASS TYPE? IF SO, FILL IN THE BLANK FIELDS,
                                ; UPDATE THE MODULE'S PASS TIME, AND CHECK FOR ANY "HUNG" MODULES
                                ;-

001402'                         IF R4 EQ #MSGEOP THEN
(6)  001402' 020427  000013                                             CMP     R4,#MSGEOP
(9)  001406' 001024                                                     BNE     50002$

                                ;+
                                ; GET MODULE NAME
                                ;-

001410'                                 CALL GETNAM IN <R1, #EOPNAM>
(3)  001410' 010546                                                     MOV     R5,-(SP)
(5)  001412' 012745  000017'                                            MOV     #EOPNAM,-(R5)
(4)  001416' 010145                                                     MOV     R1,-(R5)
(3)  001420' 004767  002032                                             JSR     PC,GETNAM
(3)  001424' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASS COUNT TO DECIMAL ASCII
                                ;-
001426'                                 CALL BDACNV IN <PASCNT(R1), #EOPPAS>
(3)  001426' 010546                                                     MOV     R5,-(SP)
(5)  001430' 012745  000037'                                            MOV     #EOPPAS,-(R5)
(4)  001434' 016145  000034                                             MOV     PASCNT(R1),-(R5)
(3)  001440' 004767  000000G                                            JSR     PC,BDACNV
(3)  001444' 012605                                                     MOV     (SP)+,R5

001446'                         LET FLMSADR(R5) := #EOPMSG
(4)  001446' 012765  000016' 000010                                     MOV     #EOPMSG,FLMSADR(
001454'                         INLINE <JMP 1$>
(2)  001454' 000167  001770                                             JMP 1$
001460'                         ENDIF
(4)  001460'                                                    50002$:
```

```
                                .SBTTL  PROCESS COMMON HEADER TYPE MESSAGE

                                ;+
                                ; IS IT A COMMON HEADER TYPE MESSAGE?
                                ;-

001460'                         IF R4 EQ #MSGHDR THEN
(6)  001460' 020427  000004                                             CMP     R4,#MSGHDR
(9)  001464' 001045                                                     BNE     50003$

                                ;+
                                ; GET MODULE NAME
                                ;-
001466'                                 CALL GETNAM IN <R1,#HDRNAM>
(3)  001466' 010546                                                     MOV     R5,-(SP)
(5)  001470' 012745  000240'                                            MOV     #HDRNAM,-(R5)
(4)  001474' 010145                                                     MOV     R1,-(R5)
(3)  001476' 004767  001754                                             JSR     PC,GETNAM
(3)  001502' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; GET PHYSICAL ADDRESS
                                ;-
001504'                                 CALL GETAPA IN <R0,R2,#HDRPA>
(3)  001504' 010546                                                     MOV     R5,-(SP)
(6)  001506' 012745  000252'                                            MOV     #HDRPA,-(R5)
(5)  001512' 010245                                                     MOV     R2,-(R5)
(4)  001514' 010045                                                     MOV     R0,-(R5)
(3)  001516' 004767  001774                                             JSR     PC,GETAPA
(3)  001522' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

001524'                                 LET R3 := R2 - R1
(4)  001524' 010203                                                     MOV     R2,R3
(6)  001526' 160103                                                     SUB     R1,R3

                                ;+
                                ; AND CONVERT IT TO ASCII
                                ;-

001530'                                 CALL BOA16 IN <R3, #HDRPC>
(3)  001530' 010546                                                     MOV     R5,-(SP)
(5)  001532' 012745  000270'                                            MOV     #HDRPC,-(R5)
(4)  001536' 010345                                                     MOV     R3,-(R5)
(3)  001540' 004767  000000G                                            JSR     PC,BOA16
(3)  001544' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASCNT TO ASCII
                                ;-

001546'                                 CALL BDACNV IN <PASCNT(R1), #HDRPAS>
(3)  001546' 010546                                                     MOV     R5,-(SP)
```

```
(5)  001550' 012745  000305'                                                    MOV     #HDRPAS,-(R5)
(4)  001554' 016145  000034                                                     MOV     PASCNT(R1),-(R5)
(3)  001560' 004767  000000G                                                    JSR     PC,BDACNV
(3)  001564' 012605                                                             MOV     (SP)+,R5
831  001566'                            LET FLMSADR(R5) := #HDRMSG
(4)  001566' 012765  000237' 000010                                             MOV     #HDRMSG,FLMSADR(
832  001574'                            INLINE <JMP 1$>
(2)  001574' 000167  001650                                                     JMP 1$
833  001600'                        ENDIF
(4)  001600'                                                                50003$:
834
```

```
                                .SBTTL  PROCESS HRDERR,SOFERR

                                ;+
                                ; IF NOT A HARD OR SOFT ERROR, CONTINUE CHECKING
                                ;-

001600'                         IF R4 NE #MSGSFT AND R4 NE #MSGHRD THEN
(6) 001600' 020427  000006                                              CMP     R4,#MSGSFT
(9) 001604' 001405                                                      BEQ     50004$
(6) 001606' 020427  000007                                              CMP     R4,#MSGHRD
(9) 001612' 001402                                                      BEQ     50004$
001614'                                 INLINE <JMP 100$>
(2) 001614' 000167  000474                                              JMP 100$
001620'                         ENDIF
(4) 001620'                                                         50004$:

                                ;+
                                ; INSERT THE END OF MESSAGE CHARACTER AND ASSUME THAT THERE IS NO  EXTENDED PRINTOUT
                                ;-

001620'                         LET DATEND :B= #0
(4) 001620' 105067  176652                                              CLRB    DATEND

                                ;+
                                ; GET MODULE NAME
                                ;-

001624'                         CALL GETNAM IN <R1, #DATNAM>
(3) 001624' 010546                                                      MOV     R5,-(SP)
(5) 001626' 012745  000315'                                             MOV     #DATNAM,-(R5)
(4) 001632' 010145                                                      MOV     R1,-(R5)
(3) 001634' 004767  001616                                              JSR     PC,GETNAM
(3) 001640' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; GET P.A.
                                ;-

001642'                         CALL GETAPA IN <R0,R2,#DATPC>
(3) 001642' 010546                                                      MOV     R5,-(SP)
(6) 001644' 012745  000327'                                             MOV     #DATPC,-(R5)
(5) 001650' 010245                                                      MOV     R2,-(R5)
(4) 001652' 010045                                                      MOV     R0,-(R5)
(3) 001654' 004767  001636                                              JSR     PC,GETAPA
(3) 001660' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

001662'                         LET R3 := R2 - R1
(4) 001662' 010203                                                      MOV     R2,R3
(6) 001664' 160103                                                      SUB     R1,R3

                                ;+
```

```
                        PROCESS HRDERR,SOFERR

                        ; CONVERT IT TO ASCII
                        ;-

001666'                 CALL BOA16 IN <R3, #DATAPC>
(3) 001666' 010546                                              MOV     R5,-(SP)
(5) 001670' 012745 000345'                                      MOV     #DATAPC,-(R5)
(4) 001674' 010345                                              MOV     R3,-(R5)
(3) 001676' 004767 000000G                                      JSR     PC,BOA16
(3) 001702' 012605                                              MOV     (SP)+,R5

                        ;+
                        ; CONVERT PASS COUNT TO ASCII
                        ;-

001704'                 CALL BDACNV IN <PASCNT(R1), #DATPAS>
(3) 001704' 010546                                              MOV     R5,-(SP)
(5) 001706' 012745 000362'                                      MOV     #DATPAS,-(R5)
(4) 001712' 016145 000034                                       MOV     PASCNT(R1),-(R5)
(3) 001716' 004767 000000G                                      JSR     PC,BDACNV
(3) 001722' 012605                                              MOV     (SP)+,R5

001724'                 LET R3 := #DATTYP
(4) 001724' 012703 000370'                                      MOV     #DATTYP,R3

                        ;+
                        ; IF SOFT, FILL IN WORD 'SOFT'. IF HARD, FILL IN WORD 'HARD'.
                        ;-
001730'                 IF R4 EQ #MSGSFT THEN
(6) 001730' 020427 000006                                       CMP     R4,#MSGSFT
(9) 001734' 001013                                              BNE     50005$
001736'                         LET (R3)+ :B= #'S
(4) 001736' 112723 000123                                       MOVB    #'S,(R3)+
001742'                         LET (R3)+ :B= #'O
(4) 001742' 112723 000117                                       MOVB    #'O,(R3)+
001746'                         LET (R3)+ :B= #'F
(4) 001746' 112723 000106                                       MOVB    #'F,(R3)+
001752'                         LET (R3)+ :B= #'T
(4) 001752' 112723 000124                                       MOVB    #'T,(R3)+
001756'                         LET R0 := SOFCNT(R1)
(4) 001756' 016100 000042                                       MOV     SOFCNT(R1),R0
001762'                 ELSE
(4) 001762' 000412                                              BR      50006$
(3) 001764'                                                 50005$:
001764'                         LET (R3)+ :B= #'H
(4) 001764' 112723 000110                                       MOVB    #'H,(R3)+
001770'                         LET (R3)+ :B= #'A
(4) 001770' 112723 000101                                       MOVB    #'A,(R3)+
001774'                         LET (R3)+ :B= #'R
(4) 001774' 112723 000122                                       MOVB    #'R,(R3)+
002000'                         LET (R3)+ :B= #'D
(4) 002000' 112723 000104                                       MOVB    #'D,(R3)+
002004'                         LET R0 := HRDCNT(R1)
(4) 002004' 016100 000044                                       MOV     HRDCNT(R1),R0
002010'                 ENDIF
(4) 002010'                                                 50006$:

```

```
                                ;+
                                ; CONVERT HARD OR SOFT TO ASCII
                                ;-

002010'                         CALL BDACNV IN <R0,#DATNUM>
(3)  002010' 010546                                          MOV     R5,-(SP)
(5)  002012' 012745  000401'                                 MOV     #DATNUM,-(R5)
(4)  002016' 010045                                          MOV     R0,-(R5)
(3)  002020' 004767  000000G                                 JSR     PC,BDACNV
(3)  002024' 012605                                          MOV     (SP)+,R5

                                ;+
                                ; CONVERT CSRA,CSRC,ASTAT,ERRTYP TO ASCII
                                ;-

002026'                         CALL BOA16 IN <CSRA(R1), #DATADD>
(3)  002026' 010546                                          MOV     R5,-(SP)
(5)  002030' 012745  000415'                                 MOV     #DATADD,-(R5)
(4)  002034' 016145  000100                                  MOV     CSRA(R1),-(R5)
(3)  002040' 004767  000000G                                 JSR     PC,BOA16
(3)  002044' 012605                                          MOV     (SP)+,R5
002046'                         CALL BOA16 IN <CSRC(R1), #DATCON>
(3)  002046' 010546                                          MOV     R5,-(SP)
(5)  002050' 012745  000432'                                 MOV     #DATCON,-(R5)
(4)  002054' 016145  000102                                  MOV     CSRC(R1),-(R5)
(3)  002060' 004767  000000G                                 JSR     PC,BOA16
(3)  002064' 012605                                          MOV     (SP)+,R5
002066'                         CALL BOA16 IN <ASTAT(R1),#DATSTAT>
(3)  002066' 010546                                          MOV     R5,-(SP)
(5)  002070' 012745  000450'                                 MOV     #DATSTAT,-(R5)
(4)  002074' 016145  000104                                  MOV     ASTAT(R1).-(R5)
(3)  002100' 004767  000000G                                 JSR     PC,BOA16
(3)  002104' 012605                                          MOV     (SP)+,R5
002106'                         CALL BOA16 IN <ERRTYP(R1),#DATERR>
(3)  002106' 010546                                          MOV     R5,-(SP)
(5)  002110' 012745  000467'                                 MOV     #DATERR,-(R5)
(4)  002114' 016145  000106                                  MOV     ERRTYP(R1),-(R5)
(3)  002120' 004767  000000G                                 JSR     PC,BOA16
(3)  002124' 012605                                          MOV     (SP)+,R5
```

```
                                .SBTTL  PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?



                                ;+
                                ; IS THERE A HRDERR/SOFERR EXTENDED PRINTOUT ASSOCIATED WITH THIS MESSAGE?
                                ;-

002126'                         IF R4 EQ #MSGSFT OR R4 EQ #MSGHRD THEN
(6) 002126' 020427  000006                                              CMP     R4,#MSGSFT
(8) 002132' 001403                                                      BEQ     50007$
(6) 002134' 020427  000007                                              CMP     R4,#MSGHRD
(9) 002140' 001065                                                      BNE     50010$
(6) 002142'                                                         50007$:
002142'                                 LET R2 := R2 + #4
(6) 002142' 062702  000004                                              ADD     #4,R2
002146'                                 LET R2 := (R2)
(4) 002146' 011202                                                      MOV     (R2),R2
002150'                                 IF R2 NE #0 THEN
(6) 002150' 005702                                                      TST     R2
(9) 002152' 001453                                                      BEQ     50011$
002154'                                         LET DATEND :B= #<CR>
(4) 002154' 112767  000015  176314                                      MOVB    #<CR>,DATEND

                                ;+
                                ; GO THROUGH THE MESSAGE TABLE AND, FOR EACH ADDRESS IN THE TABLE, FIND
                                ; THE CONTENTS OF THAT ADDRESS, CONVERT IT TO ASCII, STORE THE ASCII IN THE
                                ; EXTENDED PRINTOUT TABLE (EXTMSG) FOLLOWED BY A SPACE AND, IF THE MAXIMUM
                                ; NUMBER OF ENTRIES PER LINE HAS BEEN REACHED, LOAD A CR, LF (%) INTO THE
                                ; TABLE.  CONTINUE THIS UNTIL EITHER THE LAST ENTRY IN THE MESSAGE
                                ; TABLE OR THE END OF THE PRINTOUT STORAGE AREA (EXTEND)
                                ; IS REACHED.
                                ;-



002162'                                 LET R3 := #EXTMSG
(4) 002162' 012703  000477'                                             MOV     #EXTMSG,R3
002166'                                 LET FLDCNT := #0
(4) 002166' 005067  175606                                              CLR     FLDCNT
002172'                                 WHILE (R2) NE #-1 DO
(4) 002172'                                                         50012$:
(6) 002172' 021227  177777                                              CMP     (R2),#-1
(9) 002176' 001436                                                      BEQ     50013$
002200'                                         WHILE R3 LO #EXTEND DO
(4) 002200'                                                         50014$:
(6) 002200' 020327  001043'                                             CMP     R3,#EXTEND
(9) 002204' 103032                                                      BHIS    50015$
002206'                                                 WHILE FLDCNT LT #FLDMAX DO
(4) 002206'                                                         50016$:
(6) 002206' 026727  175566  000010                                      CMP     FLDCNT,#FLDMAX
(9) 002214' 002021                                                      BGE     50017$
002216'                                                         CALL BOA16 IN <@(R2)+,R3>
(3) 002216' 010546                                                      MOV     R5,-(SP)
(5) 002220' 010345                                                      MOV     R3,-(R5)
(4) 002222' 013245                                                      MOV     @(R2)+,-(R5)
```

```
 (3)  002224' 004767  000000G                                                                    JSR     PC,BOA16
 (3)  002230' 012605                                                                             MOV     (SP)+,R5
      002232'                                                      LET R3 := R3 + #6
 (6)  002232' 062703  000006                                                                     ADD     #6,R3
      002236'                                                      LET (R3)+ :B= #<SPACE>
 (4)  002236' 112723  000040                                                                     MOVB    #<SPACE>,(R3)+
      002242'                                                      IF (R2) EQ #-1 THEN
 (6)  002242' 021227  177777                                                                     CMP     (R2),#-1
 (9)  002246' 001001                                                                             BNE     50020$
      002250'                                                            INLINE <BR              1000$>
 (2)  002250' 000411                                                                             BR      1000$
      002252'                                                      ENDIF
 (4)  002252'                                                                        50020$:
      002252'                                                      LET FLDCNT := FLDCNT + #1
 (6)  002252' 005267  175522                                                                     INC     FLDCNT
      002256'                                                ENDDO
 (4)  002256' 000753                                                                             BR      50016$
 (3)  002260'                                                                        50017$:
      002260'                                          LET (R3)+ :B= #'%
 (4)  002260' 112723  000045                                                                     MOVB    #'%,(R3)+
      002264'                                          LET FLDCNT := #0
 (4)  002264' 005067  175510                                                                     CLR     FLDCNT
      002270'                                          ENDDO
 (4)  002270' 000743                                                                             BR      50014$
 (3)  002272'                                                                        50015$:
      002272'                                    ENDDO
 (4)  002272' 000737                                                                             BR      50012$
 (3)  002274'                                                                        50013$:
      002274'                        INLINE <1000$:>
 (2)  002274'                                                                                    1000$:
      002274'                                    LET (R3)+ :B= #'%
 (4)  002274' 112723  000045                                                                     MOVB    #'%,(R3)+
      002300'                                    LET (R3)+ :B= #0
 (4)  002300' 105023                                                                             CLRB    (R3)+


      002302'                              ENDIF
 (4)  002302'                                                                        50011$:
      002302'                              LET FLMSADR(R5) := #DATAMS
 (4)  002302' 012765  000314' 000010                                                             MOV     #DATAMS,FLMSADR(
      002310'                              INLINE  <JMP     1$>
 (2)  002310' 000167  001134                                                                     JMP     1$
      002314'                        ENDIF
 (4)  002314'                                                                        50010$:
      002314'                        INLINE <100$:>
 (2)  002314'                                                                                    100$:
```

```
                                .SBTTL  PROCESS SUM COMMAND TYPE MESSAGES


                                ;+
                                ; IS IT A SUM COMMAND HEADER LINE TYPE MESSAGE?
                                ;-

002314'                         IF R4 EQ #MSGSMH THEN                                   CMP     R4,#MSGSMH
(6)  002314' 020427  000014                                                             BNE     50021$
(9)  002320' 001005
002322'                                 LET FLMSADR(R5) := #RNHMSG
(4)  002322' 012765  000046' 000010                                                     MOV     #RNHMSG,FLMSADR(
002330'                                 INLINE <JMP 1$>
(2)  002330' 000167  001114                                                             JMP 1$
002334'                         ENDIF
(4)  002334'                                                                    50021$:
```

```
                                .SBTTL  PROCESS SUM COMMAND MODULE LINE MESSAGE


                                ;+
                                ; IS IT A "SUM COMMAND-MODULE-LINE" MESSAGE?
                                ;-

                                IF R4 EQ #MSGSMB THEN
002334'                                                                 CMP     R4,#MSGSMB
(6)  002334' 020427  000015                                             BNE     50022$
(9)  002340' 001063
                                ;+
                                ; GET MODULE NAME
                                ;-

                                        CALL GETNAM IN <R1, #RNBNAM>
002342'                                                                 MOV     R5,-(SP)
(3)  002342' 010546                                                     MOV     #RNBNAM,-(R5)
(5)  002344' 012745  000051'                                            MOV     R1,-(R5)
(4)  002350' 010145                                                     JSR     PC,GETNAM
(3)  002352' 004767  001100                                             MOV     (SP)+,R5
(3)  002356' 012605

                                ;+
                                ; CONVERT PC,STAT,PASS COUNT,HARD ERORRS AND SOFT ERRORS
                                ; TO ASCII
                                ;-
                                        CALL BOA16 IN <R1,#RNBPC>
002360'                                                                 MOV     R5,-(SP)
(3)  002360' 010546                                                     MOV     #RNBPC,-(R5)
(5)  002362' 012745  000066'                                            MOV     R1,-(R5)
(4)  002366' 010145                                                     JSR     PC,BOA16
(3)  002370' 004767  000000G                                            MOV     (SP)+,R5
(3)  002374' 012605
                                        CALL BOA16 IN <STAT(R1), #RNBSTAT>
002376'                                                                 MOV     R5,-(SP)
(3)  002376' 010546                                                     MOV     #RNBSTAT,-(R5)
(5)  002400' 012745  000102'                                            MOV     STAT(R1),-(R5)
(4)  002404' 016145  000026                                             JSR     PC,BOA16
(3)  002410' 004767  000000G                                            MOV     (SP)+,R5
(3)  002414' 012605
                                        CALL BDACNV IN <PASCNT(R1), #RNBPAS>
002416'                                                                 MOV     R5,-(SP)
(3)  002416' 010546                                                     MOV     #RNBPAS,-(R5)
(5)  002420' 012745  000117'                                            MOV     PASCNT(R1),-(R5)
(4)  002424' 016145  000034                                             JSR     PC,BDACNV
(3)  002430' 004767  000000G                                            MOV     (SP)+,R5
(3)  002434' 012605
                                        CALL BDACNV IN <HRDCNT(R1), #RNBHRD>
002436'                                                                 MOV     R5,-(SP)
(3)  002436' 010546                                                     MOV     #RNBHRD,-(R5)
(5)  002440' 012745  000135'                                            MOV     HRDCNT(R1),-(R5)
(4)  002444' 016145  000044                                             JSR     PC,BDACNV
(3)  002450' 004767  000000G                                            MOV     (SP)+,R5
(3)  002454' 012605
                                        CALL BDACNV IN <SOFCNT(R1), #RNBSFT>
002456'                                                                 MOV     R5,-(SP)
(3)  002456' 010546                                                     MOV     #RNBSFT,-(R5)
(5)  002460' 012745  000153'                                            MOV     SOFCNT(R1),-(R5)
(4)  002464' 016145  000042                                             JSR     PC,BDACNV
(3)  002470' 004767  000000G
```

```
   (3)  002474' 012605                                                          MOV     (SP)+,R5
  1011  002476'                          LET FLMSADR(R5) := #RNBMSG
   (4)  002476' 012765  000050' 000010                                         MOV     #RNBMSG,FLMSADR(
  1012  002504'                          INLINE <JMP 1$>
   (2)  002504' 000167  000740                                                 JMP 1$
  1013  002510'                          ENDIF
   (4)  002510'                                                         50022$:
  1014
```

```
                                .SBTTL  PROCESS SUM COMMAND SUMMARY LINE MESSAGE


                                ;+
                                ; IS IT A SUM COMMAND SUMMARY LINE MESSAGE?
                                ;-

1024  002510'                   IF R4 EQ #MSGSMS THEN                                 CMP     R4,#MSGSMS
 (6)  002510' 020427  000016                                                         BNE     50023$
 (9)  002514' 001025
1025
                                ;+
                                ; CONVERT SYSTEM ERRORS AND POWER FAILS TO ASCII
                                ;-
1029  002516'                           CALL BDACNV IN <DT.EXS(R0), #RNSERR>          MOV     R5,-(SP)
 (3)  002516' 010546                                                                 MOV     #RNSERR,-(R5)
 (5)  002520' 012745  000201'                                                        MOV     DT.EXS(R0),-(R5)
 (4)  002524' 016045  000060                                                         JSR     PC,BDACNV
 (3)  002530' 004767  000000G                                                        MOV     (SP)+,R5
 (3)  002534' 012605
1030  002536'                           CALL BDACNV IN <DT.PFL(R0), #RNSPWR>          MOV     R5,-(SP)
 (3)  002536' 010546                                                                 MOV     #RNSPWR,-(R5)
 (5)  002540' 012745  000230'                                                        MOV     DT.PFL(R0),-(R5)
 (4)  002544' 016045  000062                                                         JSR     PC,BDACNV
 (3)  002550' 004767  000000G                                                        MOV     (SP)+,R5
 (3)  002554' 012605
1031  002556'                           LET FLMSADR(R5) := #RNSMSG                    MOV     #RNSMSG,FLMSADR(
 (4)  002556' 012765  000161' 000010
1032  002564'                           INLINE <JMP 1$>                               JMP 1$
 (2)  002564' 000167  000660
1033  002570'                   ENDIF                                         50023$:
 (4)  002570'
1034
```

```
                                .SBTTL  PROCESS DROP MODULE TYPE MESSAGE


                                ;+
                                ; IS IT A DROP MODULE MESSAGE?
                                ;-

002570'                         IF R4 EQ #MSGDRP THEN
(6)  002570' 020427  000017                                             CMP     R4,#MSGDRP
(9)  002574' 001025                                                     BNE     50024$

                                ;+
                                ; GET MODULE NAME
                                ;-
002576'                                 CALL GETNAM IN <R1,#DRPNAM>
(3)  002576' 010546                                                     MOV     R5,-(SP)
(5)  002600' 012745  001045'                                            MOV     #DRPNAM,-(R5)
(4)  002604' 010145                                                     MOV     R1,-(R5)
(3)  002606' 004767  000644                                             JSR     PC,GETNAM
(3)  002612' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

002614'                                 LET R3 := R2 - R1
(4)  002614' 010203                                                     MOV     R2,R3
(6)  002616' 160103                                                     SUB     R1,R3

                                ;+
                                ; CONVERT DROP PC TO ASCII
                                ;-

002620'                                 CALL BOA16 IN <R3, #DRPPC>
(3)  002620' 010546                                                     MOV     R5,-(SP)
(5)  002622' 012745  001073'                                            MOV     #DRPPC,-(R5)
(4)  002626' 010345                                                     MOV     R3,-(R5)
(3)  002630' 004767  000000G                                            JSR     PC,BOA16
(3)  002634' 012605                                                     MOV     (SP)+,R5
002636'                                 LET FLMSADR(R5) := #DRPMSG
(4)  002636' 012765  001044' 000010                                     MOV     #DRPMSG,FLMSADR(
002644'                                 INLINE <JMP 1$>
(2)  002644' 000167  000600                                             JMP 1$
002650'                         ENDIF
(4)  002650'                                                    50024$:
```

```
                                .SBTTL  PROCESS CKDATA/DATACK/DATERR


                                ;+
                                ; IS IT A CKDATA/DATACK/DATERR TYPE MESSAGE?
                                ;-
1074  002650'                   IF R4 NE #MSGCKD AND R4 NE #MSGDER THEN
 (6)  002650' 020427  000010                                                    CMP     R4,#MSGCKD
 (9)  002654' 001405                                                            BEQ     50025$
 (6)  002656' 020427  000005                                                    CMP     R4,#MSGDER
 (9)  002662' 001402                                                            BEQ     50025$
1075  002664'                           INLINE <JMP     339$>
 (2)  002664' 000167  000460                                                    JMP     339$
1076  002670'                   ENDIF
 (4)  002670'                                                           50025$:

1078                            ;+
1079                            ; GET MODULE NAME
1080                            ;-
1081
1082  002670'                   CALL GETNAM IN <R1, #CKDNAM>
 (3)  002670' 010546                                                            MOV     R5,-(SP)
 (5)  002672' 012745  001104'                                                   MOV     #CKDNAM,-(R5)
 (4)  002676' 010145                                                            MOV     R1,-(R5)
 (3)  002700' 004767  000552                                                    JSR     PC,GETNAM
 (3)  002704' 012605                                                            MOV     (SP)+,R5
1083
1084                            ;+
1085                            ; GET A PA
1086                            ;-
1087
1088  002706'                   CALL GETAPA IN <R0, R2, #CKDPA>
 (3)  002706' 010546                                                            MOV     R5,-(SP)
 (6)  002710' 012745  001116'                                                   MOV     #CKDPA,-(R5)
 (5)  002714' 010245                                                            MOV     R2,-(R5)
 (4)  002716' 010045                                                            MOV     R0,-(R5)
 (3)  002720' 004767  000572                                                    JSR     PC,GETAPA
 (3)  002724' 012605                                                            MOV     (SP)+,R5
1089
1090                            ;+
1091                            ; FORM APC
1092                            ;-
1093
1094  002726'                   LET R3 := R2 - R1
 (4)  002726' 010203                                                            MOV     R2,R3
 (6)  002730' 160103                                                            SUB     R1,R3
1095
1096                            ;+
1097                            ; CONVERT VA TO ASCII
1098                            ;-
1099
1100  002732'                   CALL BOA16 IN <R3, #CKDVA>
 (3)  002732' 010546                                                            MOV     R5,-(SP)
 (5)  002734' 012745  001134'                                                   MOV     #CKDVA,-(R5)
 (4)  002740' 010345                                                            MOV     R3,-(R5)
 (3)  002742' 004767  000000G                                                   JSR     PC,BOA16
```

```
 (3)  002746' 012605                                                    MOV     (SP)+,R5
1101
1102                                    ;+
1103                                    ; CONVERT PASS COUNT TO ASCII
1104                                    ;-
1105
1106  002750'                           CALL BDACNV IN <PASCNT(R1), #CKDPAS>
 (3)  002750' 010546                                                    MOV     R5,-(SP)
 (5)  002752' 012745  001151'                                           MOV     #CKDPAS,-(R5)
 (4)  002756' 016145  000034                                            MOV     PASCNT(R1),-(R5)
 (3)  002762' 004767  000000G                                           JSR     PC,BDACNV
 (3)  002766' 012605                                                    MOV     (SP)+,R5
1107
1108                                    ;+
1109                                    ; CONVERT SOFCNT TO ASCII
1110                                    ;-
1111
1112  002770'                           CALL BDACNV IN <SOFCNT(R1), #CKDNUM>
 (3)  002770' 010546                                                    MOV     R5,-(SP)
 (5)  002772' 012745  001164'                                           MOV     #CKDNUM,-(R5)
 (4)  002776' 016145  000042                                            MOV     SOFCNT(R1),-(R5)
 (3)  003002' 004767  000000G                                           JSR     PC,BDACNV
 (3)  003006' 012605                                                    MOV     (SP)+,R5
1113
1114                                    ;+
1115                                    ; CONVERT CSRA,ASB,AWAS TO ASCII
1116                                    ;-
1117
1118  003010'                           CALL BOA16 IN <CSRA(R1), #CKDADD>
 (3)  003010' 010546                                                    MOV     R5,-(SP)
 (5)  003012' 012745  001200'                                           MOV     #CKDADD,-(R5)
 (4)  003016' 016145  000100                                            MOV     CSRA(R1),-(R5)
 (3)  003022' 004767  000000G                                           JSR     PC,BOA16
 (3)  003026' 012605                                                    MOV     (SP)+,R5
1119  003030'                           CALL BOA16 IN <ASB(R1), #CKDSB>
 (3)  003030' 010546                                                    MOV     R5,-(SP)
 (5)  003032' 012745  001214'                                           MOV     #CKDSB,-(R5)
 (4)  003036' 016145  000106                                            MOV     ASB(R1),-(R5)
 (3)  003042' 004767  000000G                                           JSR     PC,BOA16
 (3)  003046' 012605                                                    MOV     (SP)+,R5
1120  003050'                           CALL BOA16 IN <AWAS(R1), #CKDWAS>
 (3)  003050' 010546                                                    MOV     R5,-(SP)
 (5)  003052' 012745  001230'                                           MOV     #CKDWAS,-(R5)
 (4)  003056' 016145  000110                                            MOV     AWAS(R1),-(R5)
 (3)  003062' 004767  000000G                                           JSR     PC,BOA16
 (3)  003066' 012605                                                    MOV     (SP)+,R5
1121
1122                                    ;+
1123                                    ; IF NOT A DATERR,
1124                                    ; GO BUILD THE PHYSICAL ADDRESSES
1125                                    ;-
1126  003070'                           IF R4 NE #MSGDER THEN
 (6)  003070' 020427  000005                                            CMP     R4,#MSGDER
 (9)  003074' 001476                                                    BEQ     50026$
1127
1128
```

```
                                ;+
                                ; CLEAR MSB OF WORD COUNT
                                ;-

                                        LET R3 := CDWDCT(R1) CLR.BY #BIT15
003076'                                                                         MOV     CDWDCT(R1),R3
003076' 016103  000144                                                          BIC     #BIT15,R3

                                ;+
                                ; FORM WRITE BUFFER FAILING ADDRESS
                                ;-

                                        LET R3 := R3 - #1
003106'                                                                         DEC     R3
                                        LET R3 := R3 SHIFT 1
003110'                                                                         ASL     R3
                                        LET R4 := R3 + WBUFPA(R1)
003112'                                                                         MOV     R3,R4
003112' 010304                                                                  ADD     WBUFPA(R1),R4
003114' 066104  000134

                                        LET TMPPA := R4
003120'                                                                         MOV     R4,TMPPA
003120' 010467  174662
                                        LET TMPEA := WBUFEA(R1)
003124'                                                                         MOV     WBUFEA(R1),TMPEA
003124' 016167  000136  174656
                                        LET FM.PA := TMPPA
003132'                                                                         MOV     TMPPA,FM.PA
003132' 016767  174650  174652
                                        LET FM.EA := TMPEA
003140'                                                                         MOV     TMPEA,FM.EA
003140' 016767  174644  174646


                                ;+
                                ; CONVERT WRITE ADDRESS TO ASCII
                                ;-

                                        CALL BOAC IN <R0,FM.PA,FM.EA, #CKDWRA>
003146'                                                                         MOV     R5,-(SP)
003146' 010546                                                                  MOV     #CKDWRA,-(R5)
003150' 012745  001245'                                                         MOV     FM.EA,-(R5)
003154' 016745  174634                                                          MOV     FM.PA,-(R5)
003160' 016745  174626                                                          MOV     R0,-(R5)
003164' 010045                                                                  JSR     PC,BOAC
003166' 004767  000000G                                                         MOV     (SP)+,R5
003172' 012605
                                        LET R4 := R2 + #4
003174'                                                                         MOV     R2,R4
003174' 010204                                                                  ADD     #4,R4
003176' 062704  000004
                                        LET R3 := R3 + @(R4)
003202'                                                                         ADD     @(R4),R3
003202' 067403  000000
                                        LET R4 := (R4) + #2
003206'                                                                         MOV     (R4),R4
003206' 011404                                                                  ADD     #2,R4
003210' 062704  000002
                                        LET R4 := (R4)
003214'                                                                         MOV     (R4),R4
003214' 011404

                                        LET TMPPA := R3
003216'                                                                         MOV     R3,TMPPA
003216' 010367  174564
```

```
1161  003222'                               LET TMPEA := R4
 (4)  003222' 010467  174562                                                    MOV     R4,TMPEA
1162
1163  003226'                               LET FM.PA := TMPPA
 (4)  003226' 016767  174554  174556                                           MOV     TMPPA,FM.PA
1164  003234'                               LET FM.EA := TMPEA
 (4)  003234' 016767  174550  174552                                           MOV     TMPEA,FM.EA
1165
1166                                       ;+
1167                                       ; CONVERT READ ADDRESS TO ASCII
1168                                       ;-
1169
1170  003242'                               CALL BOAC IN <R0, FM.PA,FM.EA, #CKDRDA>
 (3)  003242' 010546                                                           MOV     R5,-(SP)
 (7)  003244' 012745  001264'                                                  MOV     #CKDRDA,-(R5)
 (6)  003250' 016745  174540                                                   MOV     FM.EA,-(R5)
 (5)  003254' 016745  174532                                                   MOV     FM.PA,-(R5)
 (4)  003260' 010045                                                           MOV     R0,-(R5)
 (3)  003262' 004767  000000G                                                  JSR     PC,BOAC
 (3)  003266' 012605                                                           MOV     (SP)+,R5
1171
1172                                       ;+
1173                                       ; IT'S A DATERR, SO GET SBADR AND WASADR
1174                                       ;-
1175
1176  003270'                              ELSE
 (4)  003270' 000422                                                           BR      50027$
 (3)  003272'                                                          50026$:
1177  003272'                               CALL GETAPA IN <R0,SBADR(R1),#CKDWRA>
 (3)  003272' 010546                                                           MOV     R5,-(SP)
 (6)  003274' 012745  001245'                                                  MOV     #CKDWRA,-(R5)
 (5)  003300' 016145  000102                                                   MOV     SBADR(R1),-(R5)
 (4)  003304' 010045                                                           MOV     R0,-(R5)
 (3)  003306' 004767  000204                                                   JSR     PC,GETAPA
 (3)  003312' 012605                                                           MOV     (SP)+,R5
1178  003314'                               CALL GETAPA IN <R0,WASADR(R1),#CKDRDA>
 (3)  003314' 010546                                                           MOV     R5,-(SP)
 (6)  003316' 012745  001264'                                                  MOV     #CKDRDA,-(R5)
 (5)  003322' 016145  000104                                                   MOV     WASADR(R1),-(R5)
 (4)  003326' 010045                                                           MOV     R0,-(R5)
 (3)  003330' 004767  000162                                                   JSR     PC,GETAPA
 (3)  003334' 012605                                                           MOV     (SP)+,R5
1179
1180
1181  003336'                              ENDIF
 (4)  003336'                                                          50027$:
1182
1183
1184  003336'                               LET FLMSADR(R5) := #CKDMSG
 (4)  003336' 012765  001103' 000010                                           MOV     #CKDMSG,FLMSADR(
1185  003344'                               INLINE <JMP 1$>
 (2)  003344' 000167  000100                                                   JMP 1$
1186  003350'                               INLINE <339$:>
 (2)  003350'                                                                  339$:
1187
```

```
                                .SBTTL  PROCESS CKDATA/DATACK SUMMARY LINE


                                ;+
                                ; IS IT THE CKDATA/DATACK SUMMARY LINE?
                                ;-

003350'                         IF R4 EQ #MSGCKS THEN
 (6) 003350' 020427  000011                                             CMP     R4,#MSGCKS
 (9) 003354' 001035                                                     BNE     50030$
003356'                                 LET R3 := CDERCT(R1) CLR.BY #BIT15
 (4) 003356' 016103  000146                                             MOV     CDERCT(R1),R3
 (6) 003362' 042703  100000                                             BIC     #BIT15,R3

                                ;+
                                ; GET THE MODULE NAME
                                ;-

003366'                                 CALL GETNAM IN <R1,#CKDSNAM>
 (3) 003366' 010546                                                     MOV     R5,-(SP)
 (5) 003370' 012745  001276'                                            MOV     #CKDSNAM,-(R5)
 (4) 003374' 010145                                                     MOV     R1,-(R5)
 (3) 003376' 004767  000054                                             JSR     PC,GETNAM
 (3) 003402' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT SUMMARY COUNT TO ASCII
                                ;-

003404'                                 CALL BDACNV IN <R3, #CKDSCNT>
 (3) 003404' 010546                                                     MOV     R5,-(SP)
 (5) 003406' 012745  001310'                                            MOV     #CKDSCNT,-(R5)
 (4) 003412' 010345                                                     MOV     R3,-(R5)
 (3) 003414' 004767  000000G                                            JSR     PC,BDACNV
 (3) 003420' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT NUMBER OF LOCATIONS READ TO ASCII
                                ;-

003422'                                 CALL BDACNV IN <RBUFSZ(R1),#CKDSRD>
 (3) 003422' 010546                                                     MOV     R5,-(SP)
 (5) 003424' 012745  001334'                                            MOV     #CKDSRD,-(R5)
 (4) 003430' 016145  000132                                             MOV     RBUFSZ(R1),-(R5)
 (3) 003434' 004767  000000G                                            JSR     PC,BDACNV
 (3) 003440' 012605                                                     MOV     (SP)+,R5
003442'                                 LET FLMSADR(R5) := #CKDSMSG
 (4) 003442' 012765  001276' 000010                                     MOV     #CKDSMSG,FLMSADR
003450'                         ENDIF
 (4) 003450'                                                    50030$:
```

```
                                .SBTTL  DO THE NECESSARY CLEAN-UP AND THEN RETURN
1218
1219
1220
1221
1222
1223
1224
1225                            ;+
1226                            ; NOW THAT WE'VE FILLED-IN THE BLANK FIELDS, DO THE CLEAN UP AND RETURN
1227                            ;-
1228
1229  003450'                   INLINE <1$:>                                    1$:
 (2)  003450'
1230  003450'                   CALL RESREG                                     JSR     PC,RESREG
 (3)  003450' 004767  000000G
1231  003454'                   ENDRTN                                50000$:
 (3)  003454'                                                       50001$:
 (3)  003454'                                                               RTS     PC
 (2)  003454' 000207
1232
```

```
                                .SBTTL  GET MODULE NAME

                                ;+
                                ; THIS ROUTINE RETRIEVES THE MODULE NAME FROM THE MODULE'S HEADER AND
                                ; STORES IT IN THE CALLER SUPPLIED STORAGE AREA
                                ;-

003456'                         ROUTINE GETNAM <HDRADR, RSLTADR>
003456'                                                                 GETNAM:

                                ;+
                                ; SAVE REGISTERS
                                ;-

003456'                         CALL SAVREG
003456' 004767  000000G                                                 JSR     PC,SAVREG

                                ;+
                                ; GET HEADER ADDR., MOVE 5 CHARS TO RESULT ADDR.
                                ;-

003462'                         LET R0 := HDRADR(R5)
003462' 016500  000000                                                  MOV     HDRADR(R5),R0
003466'                         LET R1 := RSLTADR(R5)
003466' 016501  000002                                                  MOV     RSLTADR(R5),R1
003472'                         LET R2 := #^D<5>
003472' 012702  000005                                                  MOV     #^D<5>,R2
003476'                         WHILE R2 GT #0 DO
003476'                                                                 50002$:
003476' 005702                                                          TST     R2
003500' 003403                                                          BLE     50003$
003502'                                 LET (R1)+ :B= (R0)+
003502' 112021                                                          MOVB    (R0)+,(R1)+
003504'                                 LET R2 := R2 - #1
003504' 005302                                                          DEC     R2
003506'                         ENDDO
003506' 000773                                                          BR      50002$
003510'                                                                 50003$:

                                ;+
                                ; RESTORE REGISTERS
                                ;-

003510'                         CALL RESREG
003510' 004767  000000G                                                 JSR     PC,RESREG
003514'                         ENDRTN
003514'                                                                 50000$:
003514'                                                                 50001$:
003514' 000207                                                          RTS     PC
```

```
                                .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII


                                ;+
                                ; THIS ROUTINE ACCEPTS A 16-BIT VIRTUAL ADDRESS AND RETURNS AN 8-BYTE
                                ; (22 BIT) PHYSICAL ADDRESS IN THE CALLER SUPPLIER STORAGE AREA
                                ;-

003516'                         ROUTINE GETAPA <DTABLE, VA, RSLTADR>
003516'                                                                 GETAPA:


                                ;+
                                ; SAVE REGISTERS
                                ;-

003516'                         CALL SAVREG
003516' 004767  000000G                                                 JSR     PC,SAVREG
                                ;+
                                ; SAVE DTABLE ADDR,RESULT ADDR
                                ;-

003522'                         LET R0 := DTABLE(R5)
003522' 016500  000000                                                  MOV     DTABLE(R5),R0
003526'                         LET R1 := RSLTADR(R5)
003526' 016501  000004                                                  MOV     RSLTADR(R5),R1
003532'                         LET TMPVA := VA(R5)
003532' 016567  000002  174244                                          MOV     VA(R5),TMPVA
                                ;+
                                ; CONVERT TO PA AND THEN TO ASCII
                                ;-
003540'                         CALL GPA IN <R0, #TMPVA>
003540' 010546                                                          MOV     R5,-(SP)
003542' 012745  000004'                                                 MOV     #TMPVA,-(R5)
003546' 010045                                                          MOV     R0,-(R5)
003550' 004767  000000G                                                 JSR     PC,GPA
003554' 012605                                                          MOV     (SP)+,R5
003556'                         CALL BOAC IN <R0, TMPPA, TMPEA, R1>
003556' 010546                                                          MOV     R5,-(SP)
003560' 010145                                                          MOV     R1,-(R5)
003562' 016745  174222                                                  MOV     TMPEA,-(R5)
003566' 016745  174214                                                  MOV     TMPPA,-(R5)
003572' 010045                                                          MOV     R0,-(R5)
003574' 004767  000000G                                                 JSR     PC,BOAC
003600' 012605                                                          MOV     (SP)+,R5


                                ;+
                                ; RESTORE REGISTERS
                                ;-
003602'                         CALL RESREG
003602' 004767  000000G                                                 JSR     PC,RESREG
003606'                         ENDRTN
003606'                                                         50000$:
003606'                                                         50001$:
003606' 000207                                                          RTS     PC
```

```
1304        000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BDACNV= ****** G
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BOAC   = ****** G
BOA16  = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKDADD  001200R
CKDMSG  001103R
CKDNAM  001104R
CKDNUM  001164R
CKDPA   001116R
CKDPAS  001151R
CKDRDA  001264R
CKDSB   001214R
CKDSCN  001310R
CKDSMS  001276R
CKDSNA  001276R
CKDSRD  001334R
CKDVA   001134R
CKDWAS  001230R
CKDWRA  001245R
CKHUNG= ****** G
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DATADD  000415R
DATAMS  000314R
DATAPC  000345R
DATCON  000432R
DATEND  000476R
DATERR  000467R
DATNAM  000315R
DATNUM  000401R
DATPAS  000362R
DATPC   000327R
DATSTA  000450R
DATTYP  000370R
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMSG  001044R
DRPNAM  001045R
DRPPC   001073R
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
EOPMSG  000016R
EOPNAM  000017R
EOPPAS  000037R
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
EXTEND  001043R
EXTMSG  000477R
FATERR= 100000
FILLMS  001356RG
FLDCNT  000000R
FLDMAX= 000010
FLMSAD= 000010
FM.EA   000014R
FM.PA   000012R
GETAPA  003516R
GETNAM  003456R
GPA    = ****** G
HDRADR= 000000
HDRMSG  000237R
HDRNAM  000240R
HDRPA   000252R
HDRPAS  000305R
HDRPC   000270R
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LINECN  000002R
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGADR= 000004
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NDAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
```

```
RBUFPA= 000126          SC.APC= 000016          UIPAR0= 177640          $F$DEC= 000220          $TAGNU= 050002
RBUFSZ= 000132          SC.CKL= 000002          UIPAR1= 177642          $F$DO = 000340          $TEMP = 000300
RBUFVA= 000124          SC.CKP= 000004          UIPAR2= 177644          $F$FAL= 000405          $TSK0 = 050002
RDSERV= 000101          SC.CLO= 000000          UIPAR3= 177646          $F$GOO= 000400          $TSK1 = 050003
RDWHMI= 000022          SC.HLD= 000010          UIPAR4= 177650          $F$IF = 000110          $TSK10= 050020
RELERR= 000020          SC.SCA= 000012          UIPAR5= 177652          $F$INC= 000210          $TSK2 = 050012
RELMOD= 020000          SENDLS= 177777          UIPAR6= 177654          $F$LOO= 000200          $TSK3 = 050013
RELTIM= 010000          SOFCNT= 000042          UIPAR7= 177656          $F$NAM= 000160          $TSK4 = 050014
RESREG= ****** G        SOFPAS= 000046          UIPDR0= 177600          $F$NO = 000403          $TSK5 = 050015
RES1  = 000056          SPACE = 000040          UIPDR1= 177602          $F$OR = 000320          $TSK6 = 050016
RES2  = 000060          SPOINT= 000032          UIPDR2= 177604          $F$RTI= 000350          $TSK7 = 050017
RICHAR= 031060          SPVALU= 002200          UIPDR3= 177606          $F$RTN= 000300          $$ARGC= 000006
RNBHRD  000135R         SR0   = 177572          UIPDR4= 177610          $F$SEL= 000140          $$BYTE= 000403
RNBMSG  000050R         SR1   = 177574          UIPDR5= 177612          $F$THE= 000330          $$CASE= 000000
RNBNAM  000051R         SR2   = 177576          UIPDR6= 177614          $F$TRU= 000404          $$DST = 000000
RNBPAS  000117R         SR3   = 172516          UIPDR7= 177616          $F$UNT= 000130          $$ELOC= 000402
RNBPC   000066R         STAT  = 000026          VA    = 000002          $F$WHI= 000120          $$ERFL= 000000
RNBSFT  000153R         STATBI= 064757          WASADR= 000104          $F$YES= 000402          $$FLAG= 000340
RNBSTA  000102R         STAT1 = 000027          WBSTAT= 000040          $IFLEV= 177777          $$FROM= 000000
RNHMSG  000046R         SUSPND= 000001          WBUFEA= 000136          $ISK0 = 000001          $$LOC = 003500R
RNSERR  000201R         SVR0  = 000062          WBUFPA= 000134          $ISK1 = 000001          $$LOCN= 000000
RNSMSG  000161R         SVR1  = 000064          WBUFRQ= 000140          $ISK2 = 000001          $$REG = 177777
RNSPWR  000230R         SVR2  = 000066          WBUFSZ= 000142          $LOCTA= 177777          $$RETU= 000000
RPTDAT= 002000          SVR3  = 000070          WDFR  = 000116          $LSTIN= 000001          $$RTN1= 050000
RSLTAD= 000004          SVR4  = 000072          WDTO  = 000114          $LSTTA= 000001          $$RTN2= 050001
RSTRT = 000112          SVR5  = 000074          WTINRE= 000352          $NESTL= 177777          $$SRC = 000000
RUBOUT= 000177          SVR6  = 000076          WTWHMI= 000222          $NSK0 = 000300          $$TGSV= 000000
RUNMOD= 100000          SYSCNT= 000052          XFLAG = 000005          $NSK1 = 000120          $$TGS1= 000000
R5VALU= 001740          SYSERR= 000100          XOFF  = 000023          $NSK2 = 000110          $$TGS2= 000000
SAM   = 075464          TMPEA   000010R         XON   = 000021          $NSK3 = 000120          $$TO  = 000000
SAVREG= ****** G        TMPIO = 000002          $BGNLE= 177777          $NSK4 = 000120          $$$TAG= 050000
SBADR = 000102          TMPPA   000006R         $ERFLG= 000400          $NSK5 = 000120          .     = 003610R
SBKMOD= 000000          TMPVA   000004R         $F$AND= 000310          $NSK6 = 000110
SBKSEL= 010000          TMP1    000004R         $F$BAD= 000401          $SAVLE= 177777
SC.ADR= 000006          TQOVF = 000002          $F$BLA= 000170          $SSK0 = 050003
SC.ALC= 000014          TYPCOD= 000002          $F$CAS= 000150          $TAGLE= 177777

. ABS.  000000     000
        003610     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:FILLML,DSKZ:FILLML=SPMAC/ML,EQUATE,FILLML
 RUN-TIME: 30 20 .5 SECONDS
 RUN-TIME RATIO: 89/51=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE FILLMS - MODULE TO FILL IN BLANK FIELDS IN SKELETAL MESSAGES
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       FILLMSG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE IS RESPONSIBLE FOR FIILLING IN THE BLANK FIELDS
                        ;       OF THE MONITOR GENERATED SKELETAL MESSAGES.
                        ;       THE TYPES OF SKELETAL MESSAGES SUPPORTED ARE:
                        ;               1. END OF PASS                  2. HARD ERROR
                        ;               3. SUM COMMAND                  4. SOFT ERROR
                        ;               5. COMMON HEADER                6. DROP MODULE
                        ;               7. DATA ERROR                   8. BAD VECTOR
                        ;               9. MAP SUMMARY                  10. CKDATA CALL
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE
                        ;       2. MSG TYPE CODE
                        ;       3. MSG ADDR/SPECIAL VALUE
                        ;       4. MODULE'S HDR ADDR
                        ;
                        ; IMPLICIT INPUTS:
                        ;       STATUS INDICATOR WORD 0
                        ;
                        ; OUTPUTS:
                        ;       1. ADDRESS OF FILLED-IN MESSAGE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. BDACNV               ;BINARY TO DECIMAL ASCII CONVERSION
                        ;       2. UPDTIM               ;UPDATE MODULE'S END-OF-PASS TIME
                        ;       3. CKHUNG               ;CHECK FOR HUNG OPTION MODULES
                        ;       4. BOAC                 ;BINARY TO OCTAL-ASCII CONVERSION
                        ;       5. GPA                  ;GET PHYSICAL ADDRESS
                        ;       6. SAVREG               ;
                        ;       7. RESREG               ;
                        ;       8. UNIPA                ;GET UNIBUS PHYSICAL ADDRESS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL FILLMSG IN <DTABLE,TYPCODE,MSGADDR,HDRADDR> OUT <FILMSGADR>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
(1) 000000'                             $LSTIN = 1
000001                                  $LSTTAG = 1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  FILLMSG                 ;MODULE'S ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  BDACNV                  ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
                                .GLOBL  BOA16                   ;BINARY TO OCTAL-ASCII CONVERSION ROUTINE
                                .GLOBL  BOAC                    ;BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  CKHUNG                  ;CHECK FOR HUNG MODULES ROUTINE
                                .GLOBL  GPA                     ;GET PHYSICAL ADDRESS ROUTINE
                                .GLOBL  PRHMS                   ;GET TIME ROUTINE
                                .GLOBL  PRPSCN                  ;GET PASS TIME ROUTINE
                                .GLOBL  UPDTIM                  ;UPDATE MODULES PASS TIME ROUTINE
                                .GLOBL SAVREG                   ;
                                .GLOBL RESREG                   ;
                                .GLOBL  UNIPA                   ;GET UNIBUS PHYSICAL ADDRESS
                                ;
                                ;***************
                                ;
                                ; LOCAL EQUATES:
                                ;
000010                          FLDMAX  = ^D<8>                 ;MAX FIELDS PER LINE
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  FLDCNT: .WORD   0               ;FIELD COUNTER
000002' 000000                  LINECNT: .WORD  0               ;LINE COUNTER
000004'                         TMP1:                           ;STORAGE WORD
000004' 000000                  TMPVA:  .WORD   0               ;STORAGE/VIRTUAL ADDRESS
000006' 000000                  TMPPA:  .WORD   0               ;STORAGE/PHYSICAL ADDRESS
000010' 000000                  TMPEA:  .WORD   0               ;STORAGE/EA BITS
000012' 000000                  FM.PA:  .WORD   0               ;STORAGE - PHYS. ADDR.
000014' 000000                  FM.EA:  .WORD   0               ;STORAGE - EA BITS
                                ;
                                ;***************
                                ;
```

```
                                .SBTTL  MESSAGE - END-OF-PASS


000016'    045                  EOPMSG: .ASCII /%/
000017' 000005                  EOPNAM: .BLKB   ^D<5>
000024' 042440  042116  050040          .ASCII  / END PASS #/
000032' 051501  020123     043
000037' 000005                  EOPPAS: .BLKB   ^D<5>
000044' 020056  052522  052116          .ASCII  /. RUNTIME: /
000052' 046511  035105     040
000057' 000011                  EOPTIM: .BLKB   ^D<9>
000070' 050040  052123  046511          .ASCII  / PSTIME: /
000076' 035105     040
000101' 000011                  EOPPST: .BLKB   ^D<9>
000112' 000045                          .ASCIZ  /%/
```

```
                                .SBTTL  MESSAGE - SUM COMMAND HEADER LINE

000114' 051445  046525  040515  RNHMSG: .ASCII  /%SUMMARY AT RUNTIME: /
000122' 054522  040440  020124
000130' 052522  052116  046511
000136' 035105     040
000141' 000011          RNHTIM: .BLKB   ^D<9>
000152'    000                  .BYTE   0                       ;ASCII TERMINATOR
```

```
                                .SBTTL  MESSAGE - SUM COMMAND MODULE LINE

000153'    045                  RNBMSG: .ASCII /%/
000154' 000005                  RNBNAM: .BLKB   ^D<5>
000161'    040  052101  053040          .ASCII  / AT VA: /
000166' 035101     040
000171' 000006                  RNBPC:  .BLKB   ^D<6>
000177'    040  052123  052101          .ASCII  / STAT /
000204'    040
000205' 000006                  RNBSTAT: .BLKB  ^D<6>
000213'    040  040520  051523          .ASCII  / PASS #/
000220' 021440
000222' 000005                  RNBPAS: .BLKB   ^D<5>
000227'    040  051110  042504          .ASCII  / HRDERRS /
000234' 051122  020123
000240' 000005                  RNBHRD: .BLKB   ^D<5>
000245'    040  043123  042524          .ASCII  / SFTERRS /
000252' 051122  020123
000256' 000005                  RNBSFT: .BLKB   ^D<5>
000263'    000                          .BYTE 0
```

```
                                        .SBTTL  MESSAGE - SUM COMMAND SUMMARY LINE

000264' 051445  051531  042524  RNSMSG: .ASCII  /%SYSTEM ERRORS: /
000272' 020115  051105  047522
000300' 051522  020072
000304' 000005                  RNSERR: .BLKB   ^D<5>
000311'    040  020040  020040          .ASCII  /     POWER FAILS: /
000316' 047520  042527  020122
000324' 040506  046111  035123
000332'    040
000333' 000005                  RNSPWR: .BLKB   ^D<5>
000340' 000045                          .ASCIZ /%/
```

```
                                .SBTTL  MESSAGE - COMMON HEADER

000342'     045                 HDRMSG: .ASCII /%/
000343' 000005                  HDRNAM: .BLKB   ^D<5>
000350' 050040  035101      040         .ASCII  / PA: /
000355' 000010                  HDRPA:  .BLKB   ^D<8>
000365'     040  050101  035103         .ASCII / APC: /
000372'     040
000373' 000006                  HDRPC:  .BLKB 6
000401'     040  040520  051523         .ASCII  / PASS #/
000406' 021440
000410' 000005                  HDRPAS: .BLKB   ^D<5>
000415'     045      000                .ASCIZ /%/

```

```
                                .SBTTL  MESSAGE - HRDERR,SOFERR

                        DATAMSG:        .ASCII /%/
                        DATNAM: .BLKB   ^D<5>
                                .ASCII  / PA: /
                        DATPC:  .BLKB   ^D<8>
                                .ASCII  / APC: /
                        DATAPC: .BLKB   ^D<6>
                                .ASCII  / PASS #/

                        DATPAS: .BLKB   ^D<5>
                                .ASCII  / /
                        DATTYP: .BLKB   ^D<4>                   ;DATA, HARD, SOFT
                                .ASCII  / ERR#/
                        DATNUM: .BLKB   ^D<5>
                                .ASCII /%CSRA: /

                        DATADD: .BLKB   ^D<6>
                                .ASCII  / CSRC: /

                        DATCON: .BLKB   ^D<6>
                                .ASCII  / ASTAT: /

                        DATSTAT: .BLKB  ^D<6>
                                .ASCII  / ERRTYP: /

                        DATERR: .BLKB   ^D<6>
                                .ASCII /%/
                        DATEND: .BYTE   0                       ;END OF MESSAGE INDICATOR
```

```
                                .SBTTL  MESSAGE - HRDERR,SOFERR,EXTENSION

000602'                         EXTMSG:
000602' 000344                          .BLKB   ^D<228>
001146'    000                  EXTEND: .BYTE 0
```

```
                                .SBTTL  MESSAGE - DROP MODULE

                                DRPMSG: .ASCII /%/
                                DRPNAM: .BLKB   ^D<5>
001147'     045                         .ASCII / DROPPED AT APC: /
001150' 000005
001155'     040 051104  050117
001162' 042520  020104  052101
001170' 040440  041520  020072
001176' 000006                  DRPPC:  .BLKB   ^D<6>
001204' 000045                          .ASCIZ /%/
```

```
                                    .SBTTL  MESSAGE - BAD VECTOR

001206' 041045  042101  053040  BADVMSG: .ASCII /%BAD VECTOR: /
001214' 041505  047524  035122
001222'    040
001223' 000006                  BADVCT: .BLKB   ^D<6>
001231'    045     000                  .ASCIZ /%/
```

```
                                .SBTTL  MESSAGE - MAP COMMAND

001233'    045                  MAPMSG: .ASCII /%/
001234' 000005                  MAPNAM: .BLKB   ^D<5>
001241'    040  052101  053040          .ASCII  / AT VA: /
001246' 035101     040
001251' 000006                  MAPPC:  .BLKB   ^D<6>
001257'    040  052123  052101          .ASCII  / STAT: /
001264' 020072
001266' 000006                  MAPSTT: .BLKB   ^D<6>
001274'    000                          .BYTE 0

```

```
                                .SBTTL  MESSAGE - HUNG MODULE

                                HNGMSG: .ASCII /%/
                                HNGNAM: .BLKB   ^D<5>
001303' 040 051511 044040               .ASCIZ  ' IS HUNG/DROPPED%'
001310' 047125 027507 051104
001316' 050117 042520 022504
001324' 000
```

```
                                        .SBTTL  MESSAGE - CKDATA,DATACK,DATERR

001325'    045                  CKDMSG: .ASCII /%/
001326' 000005                  CKDNAM: .BLKB   ^D<5>
001333'    040 040520 020072            .ASCII  / PA: /
001340' 000010                  CKDPA:  .BLKB   ^D<8>
001350' 040440 041520 020072            .ASCII  / APC: /
001356' 000006                  CKDVA:  .BLKB   ^D<6>
001364' 050040 051501 020123            .ASCII  / PASS #/
001372'    043
001373' 000005                  CKDPAS: .BLKB   ^D<5>
001400' 042440 051122 021440            .ASCII  / ERR #/
001406' 000005                  CKDNUM: .BLKB   ^D<5>
001413'    045 051503 040522            .ASCII  /%CSRA: /
001420' 020072
001422' 000006                  CKDADD: .BLKB   ^D<6>
001430' 051440 041057 020072            .ASCII  " S/B: "
001436' 000006                  CKDSB:  .BLKB   ^D<6>
001444' 053440 051501 020072            .ASCII  / WAS: /
001452' 000006                  CKDWAS: .BLKB   ^D<6>
001460' 053440 040522 051104            .ASCII  / WRADR /
001466'    040
001467' 000010                  CKDWRA: .BLKB   ^D<8>
001477'    040 042122 042101            .ASCII  / RDADR /
001504' 020122
001506' 000010                  CKDRDA: .BLKB   ^D<8>
001516' 000045                          .ASCIZ /%/
```

```
                                        .SBTTL  MESSAGE - CKDATA/DATACK SUMMARY LINE

                                        CKDSMSG:
001520'                                 CKDSNAM: .BLKB  ^D<5>
001520' 000005
001525'     040  040510  020104                  .ASCII / HAD /
001532' 000005                          CKDSCNT: .BLKB ^D<5>
001537'     040  051105  047522                  .ASCII / ERRORS OUT OF /
001544' 051522  047440  052125
001552' 047440  020106
001556' 000005                          CKDSRD: .BLKB ^D<5>
001563'     040  047527  042122                  .ASCIZ / WORDS READ%/
001570' 020123  042522  042101
001576' 000045

                                                .EVEN
```

```
                                .SBTTL  FILLMSG - FILL-IN THE BLANKS ROUTINE


001600'                         ROUTINE FILLMSG <DTABLE,TYPCODE, MSGADR, HDRADR,FLMSADR>
001600'                                                                          FILLMSG:


                                ;+
                                ; SAVE REGISTERS AND SAVE DTABLE,ADR,MSGADR AND TYPCODE
                                ;-

001600'                         CALL SAVREG
001600' 004767  000000G                                                          JSR     PC,SAVREG
001604'                         LET R0 := DTABLE(R5)
001604' 016500  000000                                                           MOV     DTABLE(R5),R0
001610'                         LET R1 := HDRADR(R5)
001610' 016501  000006                                                           MOV     HDRADR(R5),R1
001614'                         LET R2 := MSGADR(R5)
001614' 016502  000004                                                           MOV     MSGADR(R5),R2
001620'                         LET R4 := TYPCODE(R5)
001620' 016504  000002                                                           MOV     TYPCODE(R5),R4
```

```
                                .SBTTL  PROCESS END-OF-PASS TYPE MESSAGE



                                ;+
                                ; IS IT AN END-OF-PASS TYPE? IF SO, FILL IN THE BLANK FIELDS,
                                ; UPDATE THE MODULE'S PASS TIME, AND CHECK FOR ANY "HUNG" MODULES
                                ;-


001624'                         IF R4 EQ #MSGEOP THEN
(6)  001624' 020427  000013                                             CMP     R4,#MSGEOP
(9)  001630' 001057                                                     BNE     50002$

                                ;+
                                ; GET MODULE NAME
                                ;-

001632'                                 CALL GETNAM IN <R1, #EOPNAM>
(3)  001632' 010546                                                     MOV     R5,-(SP)
(5)  001634' 012745  000017'                                            MOV     #EOPNAM,-(R5)
(4)  001640' 010145                                                     MOV     R1,-(R5)
(3)  001642' 004767  002406                                             JSR     PC,GETNAM
(3)  001646' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASS COUNT TO DECIMAL ASCII
                                ;-
001650'                                 CALL BDACNV IN <PASCNT(R1), #EOPPAS>
(3)  001650' 010546                                                     MOV     R5,-(SP)
(5)  001652' 012745  000037'                                            MOV     #EOPPAS,-(R5)
(4)  001656' 016145  000034                                             MOV     PASCNT(R1),-(R5)
(3)  001662' 004767  000000G                                            JSR     PC,BDACNV
(3)  001666' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; GET ELAPSED TIME CONVERTED TO ASCII
                                ;-

001670'                                 CALL GETETIM IN <R0, #EOPTIM>
(3)  001670' 010546                                                     MOV     R5,-(SP)
(5)  001672' 012745  000057'                                            MOV     #EOPTIM,-(R5)
(4)  001676' 010045                                                     MOV     R0,-(R5)
(3)  001700' 004767  002410                                             JSR     PC,GETETIM
(3)  001704' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASS TIME TO ASCII
                                ;-
001706'                                 CALL GETPTIM IN <R0, R1, #EOPPST>
(3)  001706' 010546                                                     MOV     R5,-(SP)
(6)  001710' 012745  000101'                                            MOV     #EOPPST,-(R5)
(5)  001714' 010145                                                     MOV     R1,-(R5)
(4)  001716' 010045                                                     MOV     R0,-(R5)
(3)  001720' 004767  002460                                             JSR     PC,GETPTIM
(3)  001724' 012605                                                     MOV     (SP)+,R5
```

```
                                ;+
                                ; IF A CLOCK... UPDATE TIME
                                ;-

001726'                         IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 001726' 032760 000001 000014                                        BIT     #CLKPRES,DT.CF0(
(9) 001734' 001410                                                      BEQ     50003$
                                        CALL UPDTIM IN <R0,R1>
001736'                                                                 MOV     R5,-(SP)
(3) 001736' 010546                                                      MOV     R1,-(R5)
(5) 001740' 010145                                                      MOV     R0,-(R5)
(4) 001742' 010045                                                      JSR     PC,UPDTIM
(3) 001744' 004767 000000G                                              MOV     (SP)+,R5
(3) 001750' 012605

                                ;+
                                ; SEE IF ANY MODULE IS HUNG
                                ;-

                                        CALL CKHUNG
001752'                                                                 JSR     PC,CKHUNG
(3) 001752' 004767 000000G
                                ENDIF
001756'                                                         50003$:
(4) 001756'
                                LET FLMSADR(R5) := #EOPMSG
001756'                                                                 MOV     #EOPMSG,FLMSADR(
(4) 001756' 012765 000016' 000010
                                INLINE <JMP 1$>
001764'                                                                 JMP 1$
(2) 001764' 000167 002256
                                ENDIF
001770'                                                         50002$:
(4) 001770'
```

```
                                .SBTTL  PROCESS COMMON HEADER TYPE MESSAGE

                                ;+
                                ; IS IT A COMMON HEADER TYPE MESSAGE?
                                ;-

001770'                         IF R4 EQ #MSGHDR THEN
(6)  001770' 020427  000004                                             CMP     R4,#MSGHDR
(9)  001774' 001045                                                     BNE     50004$

                                ;+
                                ; GET MODULE NAME
                                ;-
001776'                                 CALL GETNAM IN <R1,#HDRNAM>
(3)  001776' 010546                                                     MOV     R5,-(SP)
(5)  002000' 012745  000343'                                            MOV     #HDRNAM,-(R5)
(4)  002004' 010145                                                     MOV     R1,-(R5)
(3)  002006' 004767  002242                                             JSR     PC,GETNAM
(3)  002012' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; GET PHYSICAL ADDRESS
                                ;-
002014'                                 CALL GETAPA IN <R0,R2,#HDRPA>
(3)  002014' 010546                                                     MOV     R5,-(SP)
(6)  002016' 012745  000355'                                            MOV     #HDRPA,-(R5)
(5)  002022' 010245                                                     MOV     R2,-(R5)
(4)  002024' 010045                                                     MOV     R0,-(R5)
(3)  002026' 004767  002450                                             JSR     PC,GETAPA
(3)  002032' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

002034'                                 LET R3 := R2 - R1
(4)  002034' 010203                                                     MOV     R2,R3
(6)  002036' 160103                                                     SUB     R1,R3

                                ;+
                                ; AND CONVERT IT TO ASCII
                                ;-

002040'                                 CALL BOA16 IN <R3, #HDRPC>
(3)  002040' 010546                                                     MOV     R5,-(SP)
(5)  002042' 012745  000373'                                            MOV     #HDRPC,-(R5)
(4)  002046' 010345                                                     MOV     R3,-(R5)
(3)  002050' 004767  000000G                                            JSR     PC,BOA16
(3)  002054' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASCNT TO ASCII
                                ;-

002056'                                 CALL BDACNV IN <PASCNT(R1), #HDRPAS>
(3)  002056' 010546                                                     MOV     R5,-(SP)
```

```
(5)  002060' 012745  000410'                                                    MOV     #HDRPAS,-(R5)
(4)  002064' 016145  000034                                                     MOV     PASCNT(R1),-(R5)
(3)  002070' 004767  000000G                                                    JSR     PC,BDACNV
(3)  002074' 012605                                                             MOV     (SP)+,R5
895  002076'                        LET FLMSADR(R5) := #HDRMSG
(4)  002076' 012765  000342' 000010                                             MOV     #HDRMSG,FLMSADR(
896  002104'                        INLINE <JMP 1$>
(2)  002104' 000167  002136                                                     JMP 1$
897  002110'                    ENDIF
(4)  002110'                                                                50004$:
898
```

```
                                .SBTTL  PROCESS HRDERR,SOFERR


                                ;+
                                ; IF NOT A HARD OR SOFT ERROR, CONTINUE CHECKING
                                ;-


002110'                         IF R4 NE #MSGSFT AND R4 NE #MSGHRD THEN
(6)  002110' 020427  000006                                             CMP     R4,#MSGSFT
(9)  002114' 001405                                                     BEQ     50005$
(6)  002116' 020427  000007                                             CMP     R4,#MSGHRD
(9)  002122' 001402                                                     BEQ     50005$
002124'                                 INLINE <JMP 100$>               JMP 100$
(2)  002124' 000167  000474
002130'                         ENDIF                                 50005$:
(4)  002130'
                                ;+
                                ; INSERT THE END OF MESSAGE CHARACTER AND ASSUME THAT THERE IS NO  EXTENDED PRINTOUT
                                ;-


002130'                         LET DATEND :B= #0                       CLRB    DATEND
(4)  002130' 105067  176445


                                ;+
                                ; GET MODULE NAME
                                ;-


002134'                         CALL GETNAM IN <R1, #DATNAM>            MOV     R5,-(SP)
(3)  002134' 010546                                                     MOV     #DATNAM,-(R5)
(5)  002136' 012745  000420'                                            MOV     R1,-(R5)
(4)  002142' 010145                                                     JSR     PC,GETNAM
(3)  002144' 004767  002104                                             MOV     (SP)+,R5
(3)  002150' 012605


                                ;+
                                ; GET P.A.
                                ;-


002152'                         CALL GETAPA IN <R0,R2,#DATPC>           MOV     R5,-(SP)
(3)  002152' 010546                                                     MOV     #DATPC,-(R5)
(6)  002154' 012745  000432'                                            MOV     R2,-(R5)
(5)  002160' 010245                                                     MOV     R0,-(R5)
(4)  002162' 010045                                                     JSR     PC,GETAPA
(3)  002164' 004767  002312                                             MOV     (SP)+,R5
(3)  002170' 012605


                                ;+
                                ; FORM APC
                                ;-


002172'                         LET R3 := R2 - R1                       MOV     R2,R3
(4)  002172' 010203                                                     SUB     R1,R3
(6)  002174' 160103


                                ;+
```

```
                                ; CONVERT IT TO ASCII
                                ;-

002176'                         CALL BOA16 IN <R3, #DATAPC>
(3)  002176' 010546                                                     MOV     R5,-(SP)
(5)  002200' 012745  000450'                                            MOV     #DATAPC,-(R5)
(4)  002204' 010345                                                     MOV     R3,-(R5)
(3)  002206' 004767  000000G                                           JSR     PC,BOA16
(3)  002212' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; CONVERT PASS COUNT TO ASCII
                                ;-

002214'                         CALL BDACNV IN <PASCNT(R1), #DATPAS>
(3)  002214' 010546                                                     MOV     R5,-(SP)
(5)  002216' 012745  000465'                                            MOV     #DATPAS,-(R5)
(4)  002222' 016145  000034                                             MOV     PASCNT(R1),-(R5)
(3)  002226' 004767  000000G                                           JSR     PC,BDACNV
(3)  002232' 012605                                                     MOV     (SP)+,R5

002234'                         LET R3 := #DATTYP
(4)  002234' 012703  000473'                                            MOV     #DATTYP,R3

                                ;+
                                ; IF SOFT, FILL IN WORD 'SOFT'. IF HARD, FILL IN WORD 'HARD'.
                                ;-
002240'                         IF R4 EQ #MSGSFT THEN
(6)  002240' 020427  000006                                             CMP     R4,#MSGSFT
(9)  002244' 001013                                                     BNE     50006$
002246'                                 LET (R3)+ :B= #'S
(4)  002246' 112723  000123                                             MOVB    #'S,(R3)+
002252'                                 LET (R3)+ :B= #'O
(4)  002252' 112723  000117                                             MOVB    #'O,(R3)+
002256'                                 LET (R3)+ :B= #'F
(4)  002256' 112723  000106                                             MOVB    #'F,(R3)+
002262'                                 LET (R3)+ :B= #'T
(4)  002262' 112723  000124                                             MOVB    #'T,(R3)+
002266'                                 LET R0 := SOFCNT(R1)
(4)  002266' 016100  000042                                             MOV     SOFCNT(R1),R0
002272'                         ELSE
(4)  002272' 000412                                                     BR      50007$
(3)  002274'                                                    50006$:
002274'                                 LET (R3)+ :B= #'H
(4)  002274' 112723  000110                                             MOVB    #'H,(R3)+
002300'                                 LET (R3)+ :B= #'A
(4)  002300' 112723  000101                                             MOVB    #'A,(R3)+
002304'                                 LET (R3)+ :B= #'R
(4)  002304' 112723  000122                                             MOVB    #'R,(R3)+
002310'                                 LET (R3)+ :B= #'D
(4)  002310' 112723  000104                                             MOVB    #'D,(R3)+
002314'                                 LET R0 := HRDCNT(R1)
(4)  002314' 016100  000044                                             MOV     HRDCNT(R1),R0
002320'                         ENDIF
(4)  002320'                                                    50007$:
```

```
                                ;+
                                ; CONVERT HARD OR SOFT TO ASCII
                                ;-

002320'                         CALL BDACNV IN <R0,#DATNUM>
(3)  002320' 010546                                          MOV     R5,-(SP)
(5)  002322' 012745  000504'                                 MOV     #DATNUM,-(R5)
(4)  002326' 010045                                          MOV     R0,-(R5)
(3)  002330' 004767  000000G                                 JSR     PC,BDACNV
(3)  002334' 012605                                          MOV     (SP)+,R5

                                ;+
                                ; CONVERT CSRA,CSRC,ASTAT,ERRTYP TO ASCII
                                ;-

002336'                         CALL BOA16 IN <CSRA(R1), #DATADD>
(3)  002336' 010546                                          MOV     R5,-(SP)
(5)  002340' 012745  000520'                                 MOV     #DATADD,-(R5)
(4)  002344' 016145  000100                                  MOV     CSRA(R1),-(R5)
(3)  002350' 004767  000000G                                 JSR     PC,BOA16
(3)  002354' 012605                                          MOV     (SP)+,R5
002356'                         CALL BOA16 IN <CSRC(R1), #DATCON>
(3)  002356' 010546                                          MOV     R5,-(SP)
(5)  002360' 012745  000535'                                 MOV     #DATCON,-(R5)
(4)  002364' 016145  000102                                  MOV     CSRC(R1),-(R5)
(3)  002370' 004767  000000G                                 JSR     PC,BOA16
(3)  002374' 012605                                          MOV     (SP)+,R5
002376'                         CALL BOA16 IN <ASTAT(R1),#DATSTAT>
(3)  002376' 010546                                          MOV     R5,-(SP)
(5)  002400' 012745  000553'                                 MOV     #DATSTAT,-(R5)
(4)  002404' 016145  000104                                  MOV     ASTAT(R1),-(R5)
(3)  002410' 004767  000000G                                 JSR     PC,BOA16
(3)  002414' 012605                                          MOV     (SP)+,R5
002416'                         CALL BOA16 IN <ERRTYP(R1),#DATERR>
(3)  002416' 010546                                          MOV     R5,-(SP)
(5)  002420' 012745  000572'                                 MOV     #DATERR,-(R5)
(4)  002424' 016145  000106                                  MOV     ERRTYP(R1),-(R5)
(3)  002430' 004767  000000G                                 JSR     PC,BOA16
(3)  002434' 012605                                          MOV     (SP)+,R5

```

```
                        .SBTTL  PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?


                        ;+
                        ; IS THERE A HRDERR/SOFERR EXTENDED PRINTOUT ASSOCIATED WITH THIS MESSAGE?
                        ;-

002436'                 IF R4 EQ #MSGSFT OR R4 EQ #MSGHRD THEN
(6) 002436' 020427 000006                                       CMP     R4,#MSGSFT
(8) 002442' 001403                                              BEQ     50010$
(6) 002444' 020427 000007                                       CMP     R4,#MSGHRD
(9) 002450' 001065                                              BNE     50011$
(6) 002452'                                                  50010$:
002452'                         LET R2 := R2 + #4
(6) 002452' 062702 000004                                       ADD     #4,R2
002456'                         LET R2 := (R2)
(4) 002456' 011202                                              MOV     (R2),R2
002460'                         IF R2 NE #0 THEN
(6) 002460' 005702                                              TST     R2
(9) 002462' 001453                                              BEQ     50012$
002464'                                 LET DATEND :B= #<CR>
(4) 002464' 112767 000015 176107                                MOVB    #<CR>,DATEND

                        ;+
                        ; GO THROUGH THE MESSAGE TABLE AND, FOR EACH ADDRESS IN THE TABLE, FIND
                        ; THE CONTENTS OF THAT ADDRESS, CONVERT IT TO ASCII, STORE THE ASCII IN THE
                        ; EXTENDED PRINTOUT TABLE (EXTMSG) FOLLOWED BY A SPACE AND, IF THE MAXIMUM
                        ; NUMBER OF ENTRIES PER LINE HAS BEEN REACHED, LOAD A CR, LF (%) INTO THE
                        ; TABLE.  CONTINUE THIS UNTIL EITHER THE LAST ENTRY IN THE MESSAGE
                        ; TABLE OR THE END OF THE PRINTOUT STORAGE AREA (EXTEND)
                        ; IS REACHED.
                        ;-


002472'                         LET R3 := #EXTMSG
(4) 002472' 012703 000602'                                      MOV     #EXTMSG,R3
002476'                         LET FLDCNT := #0
(4) 002476' 005067 175276                                       CLR     FLDCNT
002502'                         WHILE (R2) NE #-1 DO
(4) 002502'                                                  50013$:
(6) 002502' 021227 177777                                       CMP     (R2),#-1
(9) 002506' 001436                                              BEQ     50014$
002510'                                 WHILE R3 LO #EXTEND DO
(4) 002510'                                                  50015$:
(6) 002510' 020327 001146'                                      CMP     R3,#EXTEND
(9) 002514' 103032                                              BHIS    50016$
002516'                                         WHILE FLDCNT LT #FLDMAX DO
(4) 002516'                                                  50017$:
(6) 002516' 026727 175256 000010                                CMP     FLDCNT,#FLDMAX
(9) 002524' 002021                                              BGE     50020$
002526'                                                 CALL BOA16 IN <@(R2)+,R3>
(3) 002526' 010546                                              MOV     R5,-(SP)
(5) 002530' 010345                                              MOV     R3,-(R5)
(4) 002532' 013245                                              MOV     @(R2)+,-(R5)
```

```
                                                                                                                    JSR     PC,BOA16
  (3)  002534' 004767  000000G                                                                                      MOV     (SP)+,R5
  (3)  002540' 012605
1015   002542'                                                                                 LET R3 := R3 + #6
  (6)  002542' 062703  000006                                                                                       ADD     #6,R3
1016   002546'                                                                                 LET (R3)+ :B= #<SPACE>
  (4)  002546' 112723  000040                                                                                       MOVB    #<SPACE>,(R3)+
1017   002552'                                                                                 IF (R2) EQ #-1 THEN
  (6)  002552' 021227  177777                                                                                       CMP     (R2),#-1
  (9)  002556' 001001                                                                                               BNE     50021$
1018   002560'                                                                                          INLINE <BR       1000$>
  (2)  002560' 000411                                                                                               BR      1000$
1019   002562'                                                                                 ENDIF
  (4)  002562'                                                                                                  50021$:
1020   002562'                                                                                 LET FLDCNT := FLDCNT + #1
  (6)  002562' 005267  175212                                                                                       INC     FLDCNT
1021   002566'                                                                      ENDDO
  (4)  002566' 000753                                                                                               BR      50017$
  (3)  002570'                                                                                                  50020$:
1022   002570'                                                              LET (R3)+ :B= #'%
  (4)  002570' 112723  000045                                                                                       MOVB    #'%,(R3)+
1023   002574'                                                              LET FLDCNT := #0
  (4)  002574' 005067  175200                                                                                       CLR     FLDCNT
1024   002600'                                                              ENDDO
  (4)  002600' 000743                                                                                               BR      50015$
  (3)  002602'                                                                                                  50016$:
1025   002602'                                                  ENDDO
  (4)  002602' 000737                                                                                               BR      50013$
  (3)  002604'                                                                                                  50014$:
1026   002604'                                          INLINE <1000$:>
  (2)  002604'                                                                                                          1000$:
1027   002604'                                                  LET (R3)+ :B= #'%
  (4)  002604' 112723  000045                                                                                       MOVB    #'%,(R3)+
1028   002610'                                                  LET (R3)+ :B= #0
  (4)  002610' 105023                                                                                               CLRB    (R3)+
1029
1030
1031   002612'                                                          ENDIF
  (4)  002612'                                                                                                  50012$:
1032   002612'                                                          LET FLMSADR(R5) := #DATAMS
  (4)  002612' 012765  000417' 000010                                                                               MOV     #DATAMS,FLMSADR(
1033   002620'                                                          INLINE  <JMP    1$>
  (2)  002620' 000167  001422                                                                                       JMP     1$
1034   002624'                                          ENDIF
  (4)  002624'                                                                                                  50011$:
1035   002624'                                          INLINE <100$:>
  (2)  002624'                                                                                                          100$:
```

```
1037                                    .SBTTL  PROCESS SUM COMMAND TYPE MESSAGES
1038
1039
1040
1041                                    ;+
1042                                    ; IS IT A SUM COMMAND HEADER LINE TYPE MESSAGE?
1043                                    ;-
1044
1045  002624'                           IF R4 EQ #MSGSMH THEN
 (6)  002624' 020427  000014                                                        CMP     R4,#MSGSMH
 (9)  002630' 001014                                                                BNE     50022$
1046  002632'                                   CALL GETETIM IN <R0, #RNHTIM>
 (3)  002632' 010546                                                                MOV     R5,-(SP)
 (5)  002634' 012745  000141'                                                       MOV     #RNHTIM,-(R5)
 (4)  002640' 010045                                                                MOV     R0,-(R5)
 (3)  002642' 004767  001446                                                        JSR     PC,GETETIM
 (3)  002646' 012605                                                                MOV     (SP)+,R5
1047  002650'                                   LET FLMSADR(R5) := #RNHMSG
 (4)  002650' 012765  000114' 000010                                               MOV     #RNHMSG,FLMSADR(
1048  002656'                                   INLINE <JMP 1$>
 (2)  002656' 000167  001364                                                       JMP 1$
1049  002662'                           ENDIF
 (4)  002662'                                                                  50022$:
```

```
                                .SBTTL  PROCESS SUM COMMAND MODULE LINE MESSAGE



                                ;+
                                ; IS IT A "SUM COMMAND-MODULE-LINE" MESSAGE?
                                ;-

002662'                         IF R4 EQ #MSGSMB THEN                           CMP     R4,#MSGSMB
 (6) 002662' 020427  000015                                                     BNE     50023$
 (9) 002666' 001063


                                ;+
                                ; GET MODULE NAME
                                ;-

002670'                                 CALL GETNAM IN <R1, #RNBNAM>            MOV     R5,-(SP)
 (3) 002670' 010546                                                             MOV     #RNBNAM,-(R5)
 (5) 002672' 012745  000154'                                                    MOV     R1,-(R5)
 (4) 002676' 010145                                                             JSR     PC,GETNAM
 (3) 002700' 004767  001350                                                     MOV     (SP)+,R5
 (3) 002704' 012605


                                ;+
                                ; CONVERT PC,STAT,PASS COUNT,HARD ERORRS AND SOFT ERRORS
                                ; TO ASCII
                                ;-
002706'                                 CALL BOA16 IN <R1,#RNBPC>               MOV     R5,-(SP)
 (3) 002706' 010546                                                             MOV     #RNBPC,-(R5)
 (5) 002710' 012745  000171'                                                    MOV     R1,-(R5)
 (4) 002714' 010145                                                             JSR     PC,BOA16
 (3) 002716' 004767  000000G                                                    MOV     (SP)+,R5
 (3) 002722' 012605
002724'                                 CALL BOA16 IN <STAT(R1), #RNBSTAT>      MOV     R5,-(SP)
 (3) 002724' 010546                                                             MOV     #RNBSTAT,-(R5)
 (5) 002726' 012745  000205'                                                    MOV     STAT(R1),-(R5)
 (4) 002732' 016145  000026                                                     JSR     PC,BOA16
 (3) 002736' 004767  000000G                                                    MOV     (SP)+,R5
 (3) 002742' 012605
002744'                                 CALL BDACNV IN <PASCNT(R1), #RNBPAS>    MOV     R5,-(SP)
 (3) 002744' 010546                                                             MOV     #RNBPAS,-(R5)
 (5) 002746' 012745  000222'                                                    MOV     PASCNT(R1),-(R5)
 (4) 002752' 016145  000034                                                     JSR     PC,BDACNV
 (3) 002756' 004767  000000G                                                    MOV     (SP)+,R5
 (3) 002762' 012605
002764'                                 CALL BDACNV IN <HRDCNT(R1), #RNBHRD>    MOV     R5,-(SP)
 (3) 002764' 010546                                                             MOV     #RNBHRD,-(R5)
 (5) 002766' 012745  000240'                                                    MOV     HRDCNT(R1),-(R5)
 (4) 002772' 016145  000044                                                     JSR     PC,BDACNV
 (3) 002776' 004767  000000G                                                    MOV     (SP)+,R5
 (3) 003002' 012605
003004'                                 CALL BDACNV IN <SOFCNT(R1), #RNBSFT>    MOV     R5,-(SP)
 (3) 003004' 010546                                                             MOV     #RNBSFT,-(R5)
 (5) 003006' 012745  000256'                                                    MOV     SOFCNT(R1),-(R5)
 (4) 003012' 016145  000042                                                     JSR     PC,BDACNV
 (3) 003016' 004767  000000G
```

```
   (3)  003022' 012605                                                                  MOV     (SP)+,R5
  1076  003024'                        LET FLMSADR(R5) := #RNBMSG
   (4)  003024' 012765  000153' 000010                                                 MOV     #RNBMSG,FLMSADR(
  1077  003032'                        INLINE <JMP 1$>
   (2)  003032' 000167  001210                                                         JMP 1$
  1078  003036'                        ENDIF
   (4)  003036'                                                                 50023$:
  1079
```

```
                                .SBTTL  PROCESS SUM COMMAND SUMMARY LINE MESSAGE

                                ;+
                                ; IS IT A SUM COMMAND SUMMARY LINE MESSAGE?
                                ;-

003036'                         IF R4 EQ #MSGSMS THEN
(6)  003036' 020427  000016                                             CMP     R4,#MSGSMS
(9)  003042' 001025                                                     BNE     50024$

                                ;+
                                ; CONVERT SYSTEM ERRORS AND POWER FAILS TO ASCII
                                ;-
003044'                                 CALL BDACNV IN <DT.EXS(R0), #RNSERR>
(3)  003044' 010546                                                     MOV     R5,-(SP)
(5)  003046' 012745  000304'                                            MOV     #RNSERR,-(R5)
(4)  003052' 016045  000060                                             MOV     DT.EXS(R0),-(R5)
(3)  003056' 004767  000000G                                            JSR     PC,BDACNV
(3)  003062' 012605                                                     MOV     (SP)+,R5
003064'                                 CALL BDACNV IN <DT.PFL(R0), #RNSPWR>
(3)  003064' 010546                                                     MOV     R5,-(SP)
(5)  003066' 012745  000333'                                            MOV     #RNSPWR,-(R5)
(4)  003072' 016045  000062                                             MOV     DT.PFL(R0),-(R5)
(3)  003076' 004767  000000G                                            JSR     PC,BDACNV
(3)  003102' 012605                                                     MOV     (SP)+,R5
003104'                                 LET FLMSADR(R5) := #RNSMSG
(4)  003104' 012765  000264' 000010                                     MOV     #RNSMSG,FLMSADR(
                                        INLINE <JMP 1$>
003112'                                                                 JMP 1$
(2)  003112' 000167  001130
003116'                         ENDIF
(4)  003116'                                                    50024$:
                                
```

```
                                .SBTTL  PROCESS DROP MODULE TYPE MESSAGE


                                ;+
                                ; IS IT A DROP MODULE MESSAGE?
                                ;-

003116'                         IF R4 EQ #MSGDRP THEN
 (6) 003116' 020427  000017                                             CMP     R4,#MSGDRP
 (9) 003122' 001025                                                     BNE     50025$

                                ;+
                                ; GET MODULE NAME
                                ;-
003124'                                 CALL GETNAM IN <R1,#DRPNAM>
 (3) 003124' 010546                                                     MOV     R5,-(SP)
 (5) 003126' 012745  001150'                                            MOV     #DRPNAM,-(R5)
 (4) 003132' 010145                                                     MOV     R1,-(R5)
 (3) 003134' 004767  001114                                             JSR     PC,GETNAM
 (3) 003140' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

003142'                                 LET R3 := R2 - R1
 (4) 003142' 010203                                                     MOV     R2,R3
 (6) 003144' 160103                                                     SUB     R1,R3

                                ;+
                                ; CONVERT DROP PC TO ASCII
                                ;-

003146'                                 CALL BOA16 IN <R3, #DRPPC>
 (3) 003146' 010546                                                     MOV     R5,-(SP)
 (5) 003150' 012745  001176'                                            MOV     #DRPPC,-(R5)
 (4) 003154' 010345                                                     MOV     R3,-(R5)
 (3) 003156' 004767  000000G                                            JSR     PC,BOA16
 (3) 003162' 012605                                                     MOV     (SP)+,R5
003164'                                 LET FLMSADR(R5) := #DRPMSG
 (4) 003164' 012765  001147' 000010                                     MOV     #DRPMSG,FLMSADR(
003172'                                 INLINE <JMP 1$>
 (2) 003172' 000167  001050                                             JMP 1$
003176'                         ENDIF
 (4) 003176'                                                    50025$:
```

```
                                .SBTTL  PROCESS HUNG MODULE MESSAGE


                                ;+
                                ; IS IT A HUNG MODULE TYPE MESSAGE?
                                ;-

003176'                         IF R4 EQ #MSGHNG THEN                           CMP     R4,#MSGHNG
 (6)  003176' 020427  000022                                                    BNE     50026$
 (9)  003202' 001014
003204'                                 CALL GETNAM IN <R1, #HNGNAM>            MOV     R5,-(SP)
 (3)  003204' 010546                                                            MOV     #HNGNAM,-(R5)
 (5)  003206' 012745  001276'                                                   MOV     R1,-(R5)
 (4)  003212' 010145                                                            JSR     PC,GETNAM
 (3)  003214' 004767  001034                                                    MOV     (SP)+,R5
 (3)  003220' 012605
003222'                                 LET FLMSADR(R5) := #HNGMSG              MOV     #HNGMSG,FLMSADR(
 (4)  003222' 012765  001275' 000010
003230'                                 INLINE <JMP 1$>                         JMP 1$
 (2)  003230' 000167  001012
003234'                         ENDIF                                   50026$:
 (4)  003234'
```

```
                                .SBTTL  PROCESS BAD VECTOR TYPE MESSAGE


                                ;+
                                ; IS IT A BAD VECTOR MESSAGE?
                                ;-

003234'                         IF R4 EQ #MSGVEC THEN                           CMP     R4,#MSGVEC
(6)  003234' 020427  000020                                                     BNE     50027$
(9)  003240' 001016
                                ;+
                                ; SAVE BAD VECTOR ADDRESS IN R2
                                ;-

003242'                                 LET R2 := MSGADR(R5)                    MOV     MSGADR(R5),R2
(4)  003242' 016502  000004
                                ;+
                                ; CONVERT BAD VECTOR TO ASCII
                                ;-

003246'                                 CALL BOA16 IN <R2, #BADVCT>             MOV     R5,-(SP)
(3)  003246' 010546                                                             MOV     #BADVCT,-(R5)
(5)  003250' 012745  001223'                                                    MOV     R2,-(R5)
(4)  003254' 010245                                                             JSR     PC,BOA16
(3)  003256' 004767  000000G                                                    MOV     (SP)+,R5
(3)  003262' 012605
003264'                                 LET FLMSADR(R5) := #BADVMSG             MOV     #BADVMSG,FLMSADR
(4)  003264' 012765  001206' 000010
003272'                                 INLINE <JMP 1$>                         JMP 1$
(2)  003272' 000167  000750
003276'                         ENDIF                                   50027$:
(4)  003276'
```

```
1173                                    .SBTTL  PROCESS MAP COMMAND TYPE MESSAGE
1174
1175
1176
1177
1178                                    ;+
1179                                    ; IS IT A MAP MESSAGE?
1180                                    ;-
1181
1182  003276'                           IF R4 EQ #MSGMAP THEN
 (6)  003276' 020427  000021                                                    CMP     R4,#MSGMAP
 (9)  003302' 001033                                                            BNE     50030$
1183
1184                                    ;+
1185                                    ; GET MODULE NAME
1186                                    ;-
1187
1188  003304'                                   CALL GETNAM IN <R1, #MAPNAM>
 (3)  003304' 010546                                                            MOV     R5,-(SP)
 (5)  003306' 012745  001234'                                                   MOV     #MAPNAM,-(R5)
 (4)  003312' 010145                                                            MOV     R1,-(R5)
 (3)  003314' 004767  000734                                                    JSR     PC,GETNAM
 (3)  003320' 012605                                                            MOV     (SP)+,R5
1189
1190                                    ;+
1191                                    ; CONVERT PC AND STATUS TO ASCII
1192                                    ;-
1193
1194  003322'                                   CALL BOA16 IN <R1, #MAPPC>
 (3)  003322' 010546                                                            MOV     R5,-(SP)
 (5)  003324' 012745  001251'                                                   MOV     #MAPPC,-(R5)
 (4)  003330' 010145                                                            MOV     R1,-(R5)
 (3)  003332' 004767  000000G                                                   JSR     PC,BOA16
 (3)  003336' 012605                                                            MOV     (SP)+,R5
1195  003340'                                   CALL BOA16 IN <STAT(R1), #MAPSTT>
 (3)  003340' 010546                                                            MOV     R5,-(SP)
 (5)  003342' 012745  001266'                                                   MOV     #MAPSTT,-(R5)
 (4)  003346' 016145  000026                                                    MOV     STAT(R1),-(R5)
 (3)  003352' 004767  000000G                                                   JSR     PC,BOA16
 (3)  003356' 012605                                                            MOV     (SP)+,R5
1196  003360'                                   LET FLMSADR(R5) := #MAPMSG
 (4)  003360' 012765  001233' 000010                                            MOV     #MAPMSG,FLMSADR(
1197  003366'                                   INLINE <JMP 1$>
 (2)  003366' 000167  000654                                                    JMP 1$
1198  003372'                           ENDIF
 (4)  003372'                                                           50030$:
1199
```

```
                                .SBTTL  PROCESS CKDATA/DATACK/DATERR


                                ;+
                                ; IS IT A CKDATA/DATACK/DATERR TYPE MESSAGE?
                                ;-
003372'                         IF R4 NE #MSGCKD AND R4 NE #MSGDER THEN
(6)  003372' 020427  000010                                             CMP     R4,#MSGCKD
(9)  003376' 001405                                                     BEQ     50031$
(6)  003400' 020427  000005                                             CMP     R4,#MSGDER
(9)  003404' 001402                                                     BEQ     50031$
003406'                                 INLINE <JMP     339$>
(2)  003406' 000167  000534                                             JMP     339$
003412'                         ENDIF
(4)  003412'                                                          50031$:

                                ;+
                                ; GET MODULE NAME
                                ;-

003412'                         CALL GETNAM IN <R1, #CKDNAM>
(3)  003412' 010546                                                     MOV     R5,-(SP)
(5)  003414' 012745  001326'                                            MOV     #CKDNAM,-(R5)
(4)  003420' 010145                                                     MOV     R1,-(R5)
(3)  003422' 004767  000626                                             JSR     PC,GETNAM
(3)  003426' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; GET A PA
                                ;-

003430'                         CALL GETAPA IN <R0, R2, #CKDPA>
(3)  003430' 010546                                                     MOV     R5,-(SP)
(6)  003432' 012745  001340'                                            MOV     #CKDPA,-(R5)
(5)  003436' 010245                                                     MOV     R2,-(R5)
(4)  003440' 010045                                                     MOV     R0,-(R5)
(3)  003442' 004767  001034                                             JSR     PC,GETAPA
(3)  003446' 012605                                                     MOV     (SP)+,R5

                                ;+
                                ; FORM APC
                                ;-

003450'                         LET R3 := R2 - R1
(4)  003450' 010203                                                     MOV     R2,R3
(6)  003452' 160103                                                     SUB     R1,R3

                                ;+
                                ; CONVERT VA TO ASCII
                                ;-

003454'                         CALL BOA16 IN <R3, #CKDVA>
(3)  003454' 010546                                                     MOV     R5,-(SP)
(5)  003456' 012745  001356'                                            MOV     #CKDVA,-(R5)
(4)  003462' 010345                                                     MOV     R3,-(R5)
(3)  003464' 004767  000000G                                            JSR     PC,BOA16
```

```
                                                                        MOV     (SP)+,R5
  (3)  003470' 012605
 1234
 1235                                ;+
 1236                                ; CONVERT PASS COUNT TO ASCII
 1237                                ;-
 1238
 1239  003472'                       CALL BDACNV IN <PASCNT(R1), #CKDPAS>
                                                                        MOV     R5,-(SP)
  (3)  003472' 010546                                                   MOV     #CKDPAS,-(R5)
  (5)  003474' 012745  001373'                                          MOV     PASCNT(R1),-(R5)
  (4)  003500' 016145  000034                                           JSR     PC,BDACNV
  (3)  003504' 004767  000000G                                         MOV     (SP)+,R5
  (3)  003510' 012605
 1240
 1241                                ;+
 1242                                ; CONVERT SOFCNT TO ASCII
 1243                                ;-
 1244
 1245  003512'                       CALL BDACNV IN <SOFCNT(R1), #CKDNUM>
                                                                        MOV     R5,-(SP)
  (3)  003512' 010546                                                   MOV     #CKDNUM,-(R5)
  (5)  003514' 012745  001406'                                          MOV     SOFCNT(R1),-(R5)
  (4)  003520' 016145  000042                                           JSR     PC,BDACNV
  (3)  003524' 004767  000000G                                         MOV     (SP)+,R5
  (3)  003530' 012605
 1246
 1247                                ;+
 1248                                ; CONVERT CSRA,ASB,AWAS TO ASCII
 1249                                ;-
 1250
 1251  003532'                       CALL BOA16 IN <CSRA(R1), #CKDADD>
                                                                        MOV     R5,-(SP)
  (3)  003532' 010546                                                   MOV     #CKDADD,-(R5)
  (5)  003534' 012745  001422'                                          MOV     CSRA(R1),-(R5)
  (4)  003540' 016145  000100                                           JSR     PC,BOA16
  (3)  003544' 004767  000000G                                         MOV     (SP)+,R5
  (3)  003550' 012605
 1252  003552'                       CALL BOA16 IN <ASB(R1), #CKDSB>
                                                                        MOV     R5,-(SP)
  (3)  003552' 010546                                                   MOV     #CKDSB,-(R5)
  (5)  003554' 012745  001436'                                          MOV     ASB(R1),-(R5)
  (4)  003560' 016145  000106                                           JSR     PC,BOA16
  (3)  003564' 004767  000000G                                         MOV     (SP)+,R5
  (3)  003570' 012605
 1253  003572'                       CALL BOA16 IN <AWAS(R1), #CKDWAS>
                                                                        MOV     R5,-(SP)
  (3)  003572' 010546                                                   MOV     #CKDWAS,-(R5)
  (5)  003574' 012745  001452'                                          MOV     AWAS(R1),-(R5)
  (4)  003600' 016145  000110                                           JSR     PC,BOA16
  (3)  003604' 004767  000000G                                         MOV     (SP)+,R5
  (3)  003610' 012605
 1254
 1255                                ;+
 1256                                ; IF NOT A DATERR,
 1257                                ; GO BUILD THE PHYSICAL ADDRESSES
 1258                                ;-
 1259  003612'                       IF R4 NE #MSGDER THEN
                                                                        CMP     R4,#MSGDER
  (6)  003612' 020427  000005                                           BEQ     50032$
  (9)  003616' 001524
 1260
 1261
```

```
                                ;+
                                ; CLEAR MSB OF WORD COUNT
                                ;-

                                        LET R3 := CDWDCT(R1) CLR.BY #BIT15
003620'                                                                 MOV     CDWDCT(R1),R3
003620' 016103  000144                                                  BIC     #BIT15,R3
003624' 042703  100000

                                ;+
                                ; FORM WRITE BUFFER FAILING ADDRESS
                                ;-

                                        LET R3 := R3 - #1
003630'                                                                 DEC     R3
003630' 005303
                                        LET R3 := R3 SHIFT 1
003632'                                                                 ASL     R3
003632' 006303
                                        LET R4 := R3 + WBUFPA(R1)
003634'                                                                 MOV     R3,R4
003634' 010304                                                          ADD     WBUFPA(R1),R4
003636' 066104  000134

                                        LET TMPPA := R4
003642'                                                                 MOV     R4,TMPPA
003642' 010467  174140
                                        LET TMPEA := WBUFEA(R1)
003646'                                                                 MOV     WBUFEA(R1),TMPEA
003646' 016167  000136  174134
                                ;+
                                ; IF MAPPING ON, CALL UNIPA
                                ;-

                                        IF #MAPSTAT SETIN DT.STO(R0) THEN
003654'                                                                 BIT     #MAPSTAT,DT.STO(
003654' 032760  000200  000010                                          BEQ     50033$
003662' 001407
                                                CALL UNIPA IN <#TMPPA>
003664'                                                                 MOV     R5,-(SP)
003664' 010546                                                          MOV     #TMPPA,-(R5)
003666' 012745  000006'                                                 JSR     PC,UNIPA
003672' 004767  000000G                                                 MOV     (SP)+,R5
003676' 012605
                                        ELSE
003700'                                                                 BR      50034$
003700' 000406                                                  50033$:
003702'
                                                LET FM.PA := TMPPA
003702'                                                                 MOV     TMPPA,FM.PA
003702' 016767  174100  174102
                                                LET FM.EA := TMPEA
003710'                                                                 MOV     TMPEA,FM.EA
003710' 016767  174074  174076
                                        ENDIF
003716'                                                         50034$:
003716'


                                ;+
                                ; CONVERT WRITE ADDRESS TO ASCII
                                ;-

                                        CALL BOAC IN <R0,FM.PA,FM.EA, #CKDWRA>
003716'                                                                 MOV     R5,-(SP)
003716' 010546                                                          MOV     #CKDWRA,-(R5)
003720' 012745  001467'                                                 MOV     FM.EA,-(R5)
003724' 016745  174064
```

```
                                                                        MOV     FM.PA,-(R5)
 (5)  003730' 016745  174056                                            MOV     R0,-(R5)
 (4)  003734' 010045                                                    JSR     PC,BOAC
 (3)  003736' 004767  000000G                                           MOV     (SP)+,R5
 (3)  003742' 012605
1296  003744'                    LET R4 := R2 + #4                      MOV     R2,R4
 (4)  003744' 010204                                                    ADD     #4,R4
 (6)  003746' 062704  000004
1297  003752'                    LET R3 := R3 + @(R4)                   ADD     @(R4),R3
 (6)  003752' 067403  000000
1298  003756'                    LET R4 := (R4) + #2                    MOV     (R4),R4
 (4)  003756' 011404                                                    ADD     #2,R4
 (6)  003760' 062704  000002
1299  003764'                    LET R4 := (R4)                         MOV     (R4),R4
 (4)  003764' 011404
1300
1301  003766'                    LET TMPPA := R3                        MOV     R3,TMPPA
 (4)  003766' 010367  174014
1302  003772'                    LET TMPEA := R4                        MOV     R4,TMPEA
 (4)  003772' 010467  174012
1303
1304                            ;+
1305                            ; IF MAPPING ON CALL UNIPA
1306                            ;-
1307
1308  003776'                           IF #MAPSTAT SETIN DT.STO(R0) THEN  BIT     #MAPSTAT,DT.STO(
 (6)  003776' 032760  000200  000010                                      BEQ     50035$
 (9)  004004' 001407
1309  004006'                                 CALL UNIPA IN <#TMPPA>     MOV     R5,-(SP)
 (3)  004006' 010546                                                    MOV     #TMPPA,-(R5)
 (4)  004010' 012745  000006'                                           JSR     PC,UNIPA
 (3)  004014' 004767  000000G                                           MOV     (SP)+,R5
 (3)  004020' 012605
1310  004022'                           ELSE                           BR      50036$
 (4)  004022' 000406                                              50035$:
 (3)  004024'
1311  004024'                                 LET FM.PA := TMPPA         MOV     TMPPA,FM.PA
 (4)  004024' 016767  173756  173760
1312  004032'                                 LET FM.EA := TMPEA         MOV     TMPEA,FM.EA
 (4)  004032' 016767  173752  173754
1313  004040'                           ENDIF                          50036$:
 (4)  004040'
1314
1315                            ;+
1316                            ; CONVERT READ ADDRESS TO ASCII
1317                            ;-
1318
1319  004040'                   CALL BOAC IN <R0, FM.PA,FM.EA, #CKDRDA>  MOV     R5,-(SP)
 (3)  004040' 010546                                                    MOV     #CKDRDA,-(R5)
 (7)  004042' 012745  001506'                                           MOV     FM.EA,-(R5)
 (6)  004046' 016745  173742                                            MOV     FM.PA,-(R5)
 (5)  004052' 016745  173734                                            MOV     R0,-(R5)
 (4)  004056' 010045                                                    JSR     PC,BOAC
 (3)  004060' 004767  000000G                                           MOV     (SP)+,R5
 (3)  004064' 012605
1320  004066'                   ELSE                                   BR      50037$
 (4)  004066' 000422
```

```
                                                                                50032$:
  (3)  004070'
 1321
 1322
 1323                                   ;+
 1324                                   ; GET A P.A.
 1325                                   ;-
 1326
 1327  004070'                                 CALL GETAPA IN <R0,SBADR(R1),#CKDWRA>          MOV     R5,-(SP)
  (3)  004070' 010546                                                                        MOV     #CKDWRA,-(R5)
  (6)  004072' 012745  001467'                                                               MOV     SBADR(R1),-(R5)
  (5)  004076' 016145  000102                                                                MOV     R0,-(R5)
  (4)  004102' 010045                                                                        JSR     PC,GETAPA
  (3)  004104' 004767  000372                                                                MOV     (SP)+,R5
  (3)  004110' 012605
 1328  004112'                                 CALL GETAPA IN <R0,WASADR(R1),#CKDRDA>          MOV     R5,-(SP)
  (3)  004112' 010546                                                                        MOV     #CKDRDA,-(R5)
  (6)  004114' 012745  001506'                                                               MOV     WASADR(R1),-(R5)
  (5)  004120' 016145  000104                                                                MOV     R0,-(R5)
  (4)  004124' 010045                                                                        JSR     PC,GETAPA
  (3)  004126' 004767  000350                                                                MOV     (SP)+,R5
  (3)  004132' 012605
 1329  004134'                           ENDIF                                          50037$:
  (4)  004134'
 1330  004134'                           LET FLMSADR(R5) := #CKDMSG                             MOV     #CKDMSG,FLMSADR(
  (4)  004134' 012765  00132E' 000010
 1331  004142'                           INLINE <JMP 1$>                                        JMP 1$
  (2)  004142' 000167  000100
 1332  004146'                           INLINE <339$:>                                         339$:
  (2)  004146'
 1333
```

```
                                .SBTTL  PROCESS CKDATA/DATACK SUMMARY LINE


                                ;+
                                ; IS IT THE CKDATA/DATACK SUMMARY LINE?
                                ;-

004146'                         IF R4 EQ #MSGCKS THEN
(6) 004146' 020427  000011                                              CMP     R4,#MSGCKS
(9) 004152' 001035                                                      BNE     50040$
004154'                                 LET R3 := CDERCT(R1) CLR.BY #BIT15
(4) 004154' 016103  000146                                              MOV     CDERCT(R1),R3
(6) 004160' 042703  100000                                              BIC     #BIT15,R3

                                ;+
                                ; GET THE MODULE NAME
                                ;-

004164'                                 CALL GETNAM IN <R1,#CKDSNAM>
(3) 004164' 010546                                                      MOV     R5,-(SP)
(5) 004166' 012745  001520'                                             MOV     #CKDSNAM,-(R5)
(4) 004172' 010145                                                      MOV     R1,-(R5)
(3) 004174' 004767  000054                                              JSR     PC,GETNAM
(3) 004200' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; CONVERT SUMMARY COUNT TO ASCII
                                ;-

004202'                                 CALL BDACNV IN <R3, #CKDSCNT>
(3) 004202' 010546                                                      MOV     R5,-(SP)
(5) 004204' 012745  001532'                                             MOV     #CKDSCNT,-(R5)
(4) 004210' 010345                                                      MOV     R3,-(R5)
(3) 004212' 004767  000000G                                             JSR     PC,BDACNV
(3) 004216' 012605                                                      MOV     (SP)+,R5

                                ;+
                                ; CONVERT NUMBER OF LOCATIONS READ TO ASCII
                                ;-

004220'                                 CALL BDACNV IN <RBUFSZ(R1),#CKDSRD>
(3) 004220' 010546                                                      MOV     R5,-(SP)
(5) 004222' 012745  001556'                                             MOV     #CKDSRD,-(R5)
(4) 004226' 016145  000132                                              MOV     RBUFSZ(R1),-(R5)
(3) 004232' 004767  000000G                                             JSR     PC,BDACNV
(3) 004236' 012605                                                      MOV     (SP)+,R5
004240'                                 LET FLMSADR(R5) := #CKDSMSG
(4) 004240' 012765  001520' 000010                                      MOV     #CKDSMSG,FLMSADR
004246'                         ENDIF
(4) 004246'                                                     50040$:
```

```
                                .SBTTL  DO THE NECESSARY CLEAN-UP AND THEN RETURN


                                ;+
                                ; NOW THAT WE'VE FILLED-IN THE BLANK FIELDS, DO THE CLEAN UP AND RETURN
                                ;-

004246'                         INLINE <1$:>
004246'                                                                 1$:
004246'                         CALL RESREG
004246' 004767  000000G                                                 JSR     PC,RESREG
004252'                         ENDRTN
004252'                                                         50000$:
004252'                                                         50001$:
004252' 000207                                                  RTS     PC

```

```
                                .SBTTL  GET MODULE NAME

                                ;+
                                ; THIS ROUTINE RETRIEVES THE MODULE NAME FROM THE MODULE'S HEADER AND
                                ; STORES IT IN THE CALLER SUPPLIED STORAGE AREA
                                ;-

004254'                         ROUTINE GETNAM <HDRADR, RSLTADR>
(2)  004254'                                                            GETNAM:

                                ;+
                                ; SAVE REGISTERS
                                ;-

004254'                         CALL SAVREG
(3)  004254' 004767  000000G                                                   JSR     PC,SAVREG

                                ;+
                                ; GET HEADER ADDR., MOVE 5 CHARS TO RESULT ADDR.
                                ;-

004260'                         LET R0 := HDRADR(R5)
(4)  004260' 016500  000000                                                    MOV     HDRADR(R5),R0
004264'                         LET R1 := RSLTADR(R5)
(4)  004264' 016501  000002                                                    MOV     RSLTADR(R5),R1
004270'                         LET R2 := #^D<5>
(4)  004270' 012702  000005                                                    MOV     #^D<5>,R2
004274'                         WHILE R2 GT #0 DO
(4)  004274'                                                            50002$:
(6)  004274' 005702                                                            TST     R2
(9)  004276' 003403                                                            BLE     50003$
004300'                                 LET (R1)+ :B= (R0)+
(4)  004300' 112021                                                            MOVB    (R0)+,(R1)+
004302'                                 LET R2 := R2 - #1
(6)  004302' 005302                                                            DEC     R2
004304'                         ENDDO
(4)  004304' 000773                                                            BR      50002$
(3)  004306'                                                            50003$:

                                ;+
                                ; RESTORE REGISTERS
                                ;-

004306'                         CALL RESREG
(3)  004306' 004767  000000G                                                   JSR     PC,RESREG
004312'                         ENDRTN
(3)  004312'                                                            50000$:
(3)  004312'                                                            50001$:
(2)  004312' 000207                                                            RTS     PC
```

```
                                .SBTTL  GET ELASPED TIME


                                ;+
                                ; THIS ROUTINE RETRIEVES THE EXERCISER'S ELASPED TIME AND RETURNS IT IN
                                ; THE 9-BYTE FORMAT: "HHH:MM:SS"
                                ;-

004314'                         ROUTINE GETETIM <DTABLE, RSLTADR>
004314'                                                                 GETETIM:


                                ;+
                                ; SAVE REGISTERS
                                ;-
004314'                         CALL SAVREG
004314' 004767  000000G                                                         JSR     PC,SAVREG

                                ;+
                                ; SAVE DTABLE ADDR.,RESULT ADDR.
                                ;-

004320'                         LET R0 := DTABLE(R5)
004320' 016500  000000                                                          MOV     DTABLE(R5),R0
004324'                         LET R1 := RSLTADR(R5)
004324' 016501  000002                                                          MOV     RSLTADR(R5),R1


                                ;+
                                ; IF A CLOCK ON SYSTEM, CONVERT ELAPSED TIME INTO HRS,
                                ; MIN., SECS.
                                ;-

004330'                         IF #CLKPRES SETIN DT.CF0(R0) THEN
004330' 032760  000001  000014                                                  BIT     #CLKPRES,DT.CF0(
004336' 001407                                                                  BEQ     50002$
004340'                                 CALL PRHMS IN <R0, R1>
004340' 010546                                                                  MOV     R5,-(SP)
004342' 010145                                                                  MOV     R1,-(R5)
004344' 010045                                                                  MOV     R0,-(R5)
004346' 004767  000000G                                                         JSR     PC,PRHMS
004352' 012605                                                                  MOV     (SP)+,R5
004354'                         ELSE
004354' 000410                                                                  BR      50003$
004356'                                                                 50002$:

                                ;+
                                ; IF NOT, FILL IN BLANKS WITH SPACES
                                ;-

004356'                                 LET R2 := #^D<9>
004356' 012702  000011                                                          MOV     #^D<9>,R2
004362'                                 WHILE R2 GT #0 DO
004362'                                                                 50004$:
004362' 005702                                                                  TST     R2
004364' 003404                                                                  BLE     50005$
```

```
                                        LET (R1)+ :B= #<SPACE>                          MOVB    #<SPACE>,(R1)+

                                        LET R2 := R2 - #1

                                                                                        DEC     R2

                                ENDDO                                                   BR      50004$
                                                                                50005$:
                                                                                50003$:
                                ENDIF

                                ;+
                                ; RESTORE REGISTERS
                                ;-

                                CALL RESREG                                             JSR     PC,RESREG

                                ENDRTN                                          50000$:
                                                                                50001$:
                                                                                        RTS     PC
```

```
                                .SBTTL  GET MODULE'S PASS TIME


                                ;+
                                ; THIS ROUTINE GETS THE OPTION MODULE'S PASS TIME AND RETURNS IT IN THE
                                ; 9-BYTE FORMAT: "HHH:MM:SS"
                                ;-

004404'                         ROUTINE GETPTIM <DTABLE,HDRADR,RSLTADR>
004404'                                                                 GETPTIM:

                                ;+
                                ; SAVE REGISTERS
                                ;-

004404'                         CALL SAVREG
004404' 004767  000000G                                                 JSR     PC,SAVREG

                                ;+
                                ; SAVE DTABLE ADDR,HEADER ADDR,RESULT ADDR.
                                ;-

004410'                         LET R0 := DTABLE(R5)
004410' 016500  000000                                                  MOV     DTABLE(R5),R0
004414'                         LET R1 := HDRADR(R5)
004414' 016501  000002                                                  MOV     HDRADR(R5),R1
004420'                         LET R2 := RSLTADR(R5)
004420' 016502  000004                                                  MOV     RSLTADR(R5),R2

                                ;+
                                ; IF A CLOCK ON SYSTEM, UPDATE MODULE'S PASS TIME
                                ;-

004424'                         IF #CLKPRES SETIN DT.CF0(R0) THEN
004424' 032760  000001  000014                                          BIT     #CLKPRES,DT.CF0(
004432' 001410                                                          BEQ     50002$
004434'                                 CALL PRPSCNT IN <R0,R1,R2>
004434' 010546                                                          MOV     R5,-(SP)
004436' 010245                                                          MOV     R2,-(R5)
004440' 010145                                                          MOV     R1,-(R5)
004442' 010045                                                          MOV     R0,-(R5)
004444' 004767  000000G                                                 JSR     PC,PRPSCNT
004450' 012605                                                          MOV     (SP)+,R5
004452'                         ELSE
004452' 000410                                                          BR      50003$
004454'                                                         50002$:

                                ;+
                                ; INSERT SPACES
                                ;-

004454'                                 LET R3 := #^D<9>
004454' 012703  000011                                                  MOV     #^D<9>,R3
004460'                                 WHILE R3 GT #0 DO
004460'                                                         50004$:
004460' 005703                                                          TST     R3
```

```
   (9)  004462' 003404                                                          BLE     50005$
 1504   004464'                          LET (R2)+ :B= #<SPACE>                MOVB    #<SPACE>,(R2)+
   (4)  004464' 112722  000040
 1505   004470'                          LET R3 := R3 - #1                     DEC     R3
   (6)  004470' 005303
 1506   004472'                      ENDDO                                     BR      50004$
   (4)  004472' 000772                                                  50005$:
   (3)  004474'
 1507   004474'                      ENDIF                               50003$:
   (4)  004474'
```

```
1509
1510                                    ;+
1511                                    ; RESTORE REGISTERS
1512                                    ;-
1513
1514  004474'                           CALL RESREG                                     JSR     PC,RESREG
 (3)  004474' 004767  000000G
1515  004500'                           ENDRTN
 (3)  004500'                                                           50000$:
 (3)  004500'                                                           50001$:
 (2)  004500' 000207                                                            RTS     PC
1516
1517
1518
```

```
                                .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII


                                ;+
                                ; THIS ROUTINE ACCEPTS A 16-BIT VIRTUAL ADDRESS AND RETURNS AN 8-BYTE
                                ; (22 BIT) PHYSICAL ADDRESS IN THE CALLER SUPPLIER STORAGE AREA
                                ;-

004502'                         ROUTINE GETAPA <DTABLE, VA, RSLTADR>
004502'                                                                 GETAPA:


                                ;+
                                ; SAVE REGISTERS
                                ;-

004502'                         CALL SAVREG                                     JSR     PC,SAVREG
004502' 004767  000000G
                                ;+
                                ; SAVE DTABLE ADDR,RESULT ADDR
                                ;-

004506'                         LET R0 := DTABLE(R5)                            MOV     DTABLE(R5),R0
004506' 016500  000000
004512'                         LET R1 := RSLTADR(R5)                           MOV     RSLTADR(R5),R1
004512' 016501  000004
004516'                         LET TMPVA := VA(R5)                             MOV     VA(R5),TMPVA
004516' 016567  000002  173260
                                ;+
                                ; CONVERT TO PA AND THEN TO ASCII
                                ;-
004524'                         CALL GPA IN <R0, #TMPVA>                        MOV     R5,-(SP)
004524' 010546                                                                  MOV     #TMPVA,-(R5)
004526' 012745  000004'                                                         MOV     R0,-(R5)
004532' 010045                                                                  JSR     PC,GPA
004534' 004767  000000G                                                         MOV     (SP)+,R5
004540' 012605
004542'                         CALL BOAC IN <R0, TMPPA, TMPEA, R1>             MOV     R5,-(SP)
004542' 010546                                                                  MOV     R1,-(R5)
004544' 010145                                                                  MOV     TMPEA,-(R5)
004546' 016745  173236                                                          MOV     TMPPA,-(R5)
004552' 016745  173230                                                          MOV     R0,-(R5)
004556' 010045                                                                  JSR     PC,BOAC
004560' 004767  000000G                                                         MOV     (SP)+,R5
004564' 012605


                                ;+
                                ; RESTORE REGISTERS
                                ;-
004566'                         CALL RESREG                                     JSR     PC,RESREG
004566' 004767  000000G
004572'                         ENDRTN                                  50000$:
004572'                                                                 50001$:
004572'                                                                         RTS     PC
004572' 000207
```

```
 1555          000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BADVCT  001223R
BADVMS  001206R
BDACNV= ****** G
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BOAC   = ****** G
BOA16  = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKDADD  001422R
CKDMSG  001325R
CKDNAM  001326R
CKDNUM  001406R
CKDPA   001340R
CKDPAS  001373R
CKDRDA  001506R
CKDSB   001436R
CKDSCN  001532R
CKDSMS  001520R
CKDSNA  001520R
CKDSRD  001556R
CKDVA   001356R
CKDWAS  001452R
CKDWRA  001467R
CKHUNG= ****** G
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DATADD  000520R
DATAMS  000417R
DATAPC  000450R
DATCON  000535R
DATEND  000601R
DATERR  000572R
DATNAM  000420R
DATNUM  000504R
DATPAS  000465R
DATPC   000432R
DATSTA  000553R
DATTYP  000473R
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMSG  001147R
DRPNAM  001150R
DRPPC   001176R
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
EOPMSG  000016R
EOPNAM  000017R
EOPPAS  000037R
EOPPST  000101R
EOPTIM  000057R
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
EXTEND  001146R
EXTMSG  000602R
FATERR= 100000
FILLMS  001600RG
FLDCNT  000000R
FLDMAX= 000010
FLMSAD= 000010
FM.EA   000014R
FM.PA   000012R
GETAPA  004502R
GETETI  004314R
GETNAM  004254R
GETPTI  004404R
GPA    = ****** G
HDRADR= 000002
HDRMSG  000342R
HDRNAM  000343R
HDRPA   000355R
HDRPAS  000410R
HDRPC   000373R
HNGMSG  001275R
HNGNAM  001276R
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LINECN  000002R
LPSTAT= 000001
MAPMSG  001233R
MAPNAM  001234R
MAPPC   001251R
MAPSTA= 000200
MAPSTT  001266R
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGADR= 000004
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRHMS  = ****** G
PRI0   = 000000
PRI1   = 000040
```

```
PRI4  = 000200          RSLTAD= 000004          SYSCNT= 000052          XOFF  = 000023          $NSK5 = 000120
PRI5  = 000240          RSTRT = 000112          SYSERR= 000100          XON   = 000021          $NSK6 = 000110
PRI6  = 000300          RUBOUT= 000177          TMPEA   000010R         $BGNLE= 177777          $SAVLE= 177777
PRI7  = 000340          RUNMOD= 100000          TMPIO = 000002          $ERFLG= 000400          $SSK0 = 050005
PRPSCN= ****** G        R5VALU= 001740          TMPPA   000006R         $F$AND= 000310          $TAGLE= 177777
PR0   = 000000          SAM   = 075464          TMPVA   000004R         $F$BAD= 000401          $TAGNU= 050002
PR4   = 000200          SAVREG= ****** G        TMP1    000004R         $F$BLA= 000170          $TEMP = 000300
PR5   = 000240          SBADR = 000102          TQOVF = 000002          $F$CAS= 000150          $TSK0 = 050003
PR6   = 000300          SBKMOD= 000000          TYPCOD= 000002          $F$DEC= 000220          $TSK1 = 050004
PR7   = 000340          SBKSEL= 010000          UIPAR0= 177640          $F$DO = 000340          $TSK10= 050021
PS    = 177776          SC.ADR= 000006          UIPAR1= 177642          $F$FAL= 000405          $TSK2 = 050005
PSW   = 177776          SC.ALC= 000014          UIPAR2= 177644          $F$GOO= 000400          $TSK3 = 050014
RANNUM= 000054          SC.APC= 000016          UIPAR3= 177646          $F$IF = 000110          $TSK4 = 050015
RBUFEA= 000130          SC.CKL= 000002          UIPAR4= 177650          $F$INC= 000210          $TSK5 = 050016
RBUFPA= 000126          SC.CKP= 000004          UIPAR5= 177652          $F$LOO= 000200          $TSK6 = 050017
RBUFSZ= 000132          SC.CLO= 000000          UIPAR6= 177654          $F$NAM= 000160          $TSK7 = 050020
RBUFVA= 000124          SC.HLD= 000010          UIPAR7= 177656          $F$NO = 000403          $$ARGC= 000006
RDSERV= 000101          SC.SCA= 000012          UIPDR0= 177600          $F$OR = 000320          $$BYTE= 000403
RDWHMI= 000022          SENDLS= 177777          UIPDR1= 177602          $F$RTI= 000350          $$CASE= 000000
RELERR= 000020          SOFCNT= 000042          UIPDR2= 177604          $F$RTN= 000300          $$DST = 000000
RELMOD= 020000          SOFPAS= 000046          UIPDR3= 177606          $F$SEL= 000140          $$ELOC= 000402
RELTIM= 010000          SPACE = 000040          UIPDR4= 177610          $F$THE= 000330          $$ERFL= 000000
RESREG= ****** G        SPOINT= 000032          UIPDR5= 177612          $F$TRU= 000404          $$FLAG= 000001
RES1  = 000056          SPVALU= 002200          UIPDR6= 177614          $F$UNT= 000130          $$FROM= 000000
RES2  = 000060          SR0   = 177572          UIPDR7= 177616          $F$WHI= 000120          $$LOC = 004462R
RICHAR= 031060          SR1   = 177574          UNIPA = ****** G        $F$YES= 000402          $$LOCN= 000000
RNBHRD  000240R         SR2   = 177576          UPDTIM= ****** G        $IFLEV= 177777          $$REG = 177777
RNBMSG  000153R         SR3   = 172516          VA    = 000002          $ISK0 = 000001          $$RETU= 000000
RNBNAM  000154R         STAT  = 000026          WASADR= 000104          $ISK1 = 000001          $$RTN1= 050000
RNBPAS  000222R         STATBI= 064757          WBSTAT= 000040          $ISK2 = 000001          $$RTN2= 050001
RNBPC   000171R         STAT1 = 000027          WBUFEA= 000136          $LOCTA= 177777          $$SRC = 000000
RNBSFT  000256R         SUSPND= 000001          WBUFPA= 000134          $LSTIN= 000001          $$TGSV= 000000
RNBSTA  000205R         SVR0  = 000062          WBUFRQ= 000140          $LSTTA= 000001          $$TGS1= 000000
RNHMSG  000114R         SVR1  = 000064          WBUFSZ= 000142          $NESTL= 177777          $$TGS2= 000000
RNHTIM  000141R         SVR2  = 000066          WDFR  = 000116          $NSK0 = 000300          $$TO  = 000000
RNSERR  000304R         SVR3  = 000070          WDTO  = 000114          $NSK1 = 000110          $$$TAG= 050000
RNSMSG  000264R         SVR4  = 000072          WTINRE= 000352          $NSK2 = 000120          .     = 004574R
RNSPWR  000333R         SVR5  = 000074          WTWHMI= 000222          $NSK3 = 000120
RPTDAT= 002000          SVR6  = 000076          XFLAG = 000005          $NSK4 = 000120

. ABS.  000000     000
        004574     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:FILLMS,DSKZ:FILLMS=SPMAC/ML,EQUATE,FILLMS
 RUN-TIME: 37 27 .4 SECONDS
 RUN-TIME RATIO: 105/65=1.6
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
508                             .TITLE GETPSW (GET CALLER'S PS WORD)
509                             .IDENT /V0.0/
510
511                             ;++
512                             ; MODULE NAME:
513                             ;       GETPSW
514                             ;
515                             ; FUNCTIONAL DESCRIPTION:
516                             ;       THIS MODULE RETURNS THE CALLER'S PROCESSOR STATUS WORD.
517                             ;
518                             ; INPUTS:
519                             ;       NONE
520                             ;
521                             ; IMPLICIT INPUTS:
522                             ;       NONE
523                             ;
524                             ; OUTPUTS:
525                             ;       PROCESSOR STATUS WORD
526                             ;
527                             ; IMPLICIT OUTPUTS:
528                             ;       NONE
529                             ;
530                             ; PATHOLOGICAL CONNECTIONS:
531                             ;       NONE
532                             ;
533                             ; SUBORDINATE MODULES CALLED:
534                             ;       NONE
535                             ;
536                             ; FUNCTIONAL SIDE EFFECTS:
537                             ;       NONE
538                             ;
539                             ; CALLING SEQUENCE:
540                             ;       CALL GETPSW OUT <AA>
541                             ;           WHERE AA = PS WORD
542                             ;
543                             ; VERSION:
544                             ;       0.0
545                             ;
546                             ;       EDIT    DATE        BY          REASON
547                             ;       -----   --------    ----------  ------------------
548                             ;
549                             ;--
550
```

```
                                .SBTTL GETPSW (COMMON DEFINITIONS & REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                    000001      $LSTIN=1
                    000001      $LSTTAG=1

                                ;***************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL GETPSW           ;MODULE'S ENTRY POINT
```

```
                                .SBTTL GETPSW (CODE)

                                ROUTINE GETPSW <AA>
                                                                        GETPSW:

                                ;+
                                ; SAVE VECTOR 10; LOAD ADDRESS TO HANDLE ILLEGAL INSTRUCTION
                                ; TRAP. EXECUTE AN ILLEGAL INSTRUCTION. THE TRAP WILL PUSH PSW
                                ; AND PC+ ON THE STACK. IN THE TRAP HANDLER, GET THE PSW AND
                                ; RETURN TO PC+ BY DOING AN RTI. NOW RESTORE THE ORIGINAL
                                ; TRAP HANDLER AND RETURN.
                                ;-

000000'                         PUSH @#10
000000' 013746  000010                                                  MOV     @#10,-(SP)
000004'                         LET @#10 := #1$
000004' 012737  000022' 000010                                          MOV     #1$,@#10
000012'                         INLINE <#77>
000012' 000077                                                          #77
000014'                         POP @#10
000014' 012637  000010                                                  MOV     (SP)+,@#10
000020'                         ENDRTN
000020'                                                         50000$:
000020'                                                         50001$:
000020' 000207                                                          RTS     PC

000022'                         INLINE <1$:>
000022'                                                                 1$:
000022'                         LET AA(R5) := 2(SP)
000022' 016665  000002  000000                                          MOV     2(SP),AA(R5)
000030'                         INLINE <RTI>            ;RETURN
000030' 000002                                                          RTI

000001                          .END
```

```
AA     = 000000
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GETPSW  000000RG
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
UIPAR1= 177642
```

```
UIPAR2= 177644          WBUFPA= 000134          $F$DO  = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF  = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSKO = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000150          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO  = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR  = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP  = 000402          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST  = 000000          .     = 000032R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000

. ABS.  000000     000
        000032     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:GETPSW,DSKZ:GETPSW=SPMAC/ML,EQUATE,GETPSW
 RUN-TIME: 10 .9 .3 SECONDS
 RUN-TIME RATIO: 26/12=2.1
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE HRDADR HRDADRCHK - HARDWARE ADDRESS CHECK
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       HRDADR
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       MODULE WILL DETERMINE IF AN ADDRESS IS AVAILABLE BY TRYING TO READ THE
                        ;       LOCATION.
                        ;       LOCATION 4 WILL BE MODIFIED TO FIELD A NON-EXISTENT MEMORY
                        ;       TRAP. AN ERROR INDICATION WILL BE RETURNED IF THE ADDRESS
                        ;       DOES NOT RESPOND.
                        ;
                        ; INPUTS:
                        ;       ADDRESS TO BE CHECKED
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       ERROR INDICATOR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL HRDADR IN <A>
                        ;               A= ADDRESS TO BE CHECKED
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN  =1
        000001                  $LSTTAG =1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  HRDADR          ;MODULE ENTRY POINT
```

```
                                .SBTTL  HRDADRCHK ROUTINE

                                ROUTINE HRDADRCHK <ADRCHK>
                                                                        HRDADRCHK:

                                ;+
                                ; SAVE REGISTERS AND RETRIEVE ADDRESS
                                ; FROM R5 STACK
                                ;-

                                PUSH    R0
                                                                        MOV     R0,-(SP)
                                LET R0 := ADRCHK(R5)
                                                                        MOV     ADRCHK(R5),R0

                                ;+
                                ; SAVE CURRENT CONTENTS OF LOC. 4 ON STACK
                                ;-

                                PUSH    @#4
                                                                        MOV     @#4,-(SP)

                                ;+
                                ; LOAD LOC. 4 WITH THE TEMPORARY NON-EXISTENT
                                ; MEMORY TRAP ROUTINE AT LABEL 1$
                                ;-

                                LET @#4 := #1$
                                                                        MOV     #1$,@#4

                                ;+
                                ; CHECK TO SEE IF ADDRESS IS A VALID ADDRESS
                                ;-

                                LET (R0) := (R0) + #0
                                                                        ADD     #0,(R0)

                                ;+
                                ; RESTORE LOC. 4 AND R0, AND RETURN NO ERROR INDICATOR
                                ;-

                                POP     @#4,R0
                                                                        MOV     (SP)+,@#4
                                                                        MOV     (SP)+,R0

                                RETURN NO.ERROR
                                                                        BR      50000$
```

```
                                ;+
                                ; NON-EXISTENT MEMORY TRAP TOOK PLACE,
                                ; THEREFORE ADDRESS TESTED WAS ILLEGAL,
                                ; REPLACE RETURN ADDRESS ON STACK WITH 2$
                                ; AND DO A RTI WHICH WILL SEND YOU TO 2$.
                                ;-

000034'                         INLINE <1$:>
000034'                                                                 1$:
000034'                         LET  (SP) := #2$
000034' 012716  000042'                                                 MOV     #2$,(SP)
000040'                         INLINE <RTI>
000040' 000002                                                          RTI

                                ;+
                                ; RETURN HERE IF TRAP OCCURRED, CLEAN-UP STACK AND RETURN ERROR
                                ;-

000042'                         INLINE <2$:>
000042'                                                                 2$:

000042'                         POP     @#4,R0
000042' 012637  000004                                                  MOV     (SP)+,@#4
000046' 012600                                                          MOV     (SP)+,R0

000050'                         RETURN ERROR
000050' 000261                                                          SEC
000052' 000401                                                          BR      50001$

                                ;+
                                ;RETURN TO CALLER
                                ;-

000054'                         ENDRTN
000054'                                                         50000$:
000054' 000241                                                          CLC
000056'                                                         50001$:
000056' 000207                                                          RTS     PC

        000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
ADRCHK= 000000
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR  000000RG
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
```

```
UIPAR2= 177644          WBUFPA= 000134          $F$DO  = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC  = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR   = 000116          $F$IF  = 000110          $LSTTA= 000001          $$REG  = 177777
UIPAR6= 177654          WDTO   = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000001
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0  = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG  = 000005          $F$NO  = 000403          $TAGLE= 177777          $$SRC  = 000000
UIPDR2= 177604          XOFF   = 000023          $F$OR  = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON    = 000021          $F$RTI= 000350          $TEMP  = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000000          $F$SEL= 000140          $$BYTE= 000000          $$TO   = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST  = 000000          .      = 000060R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000403
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000

. ABS.  000000     000
        000060     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:HRDADR,DSKZ:HRDADR=SPMAC/ML,EQUATE,HRDADR
 RUN-TIME: 11  1  .3 SECONDS
 RUN-TIME RATIO: 27/13=2.1
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE HTRAP - FIELD ERROR TRAPS
                        .IDENT /V0.0/


                        ;++
                        ; MODULE NAME:
                        ;       HTRAP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS MODULE IS ENTERED AS A RESULT OF A HARDWARE TRAP.
                        ;       THE CODE SIMPLY PUSHES AN EVENT CODE ON THE STACK AND EXITS.
                        ;       THERE ARE SEVERAL ENTRY POINTS. THE ENTRY POINT IS DETERMINED
                        ;       BY THE HARDWARE TRAP VECTOR.
                        ;
                        ; INPUTS:
                        ;       NONE.
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE.
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE.
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       NONE.
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALLED BY HARDWARE ONLY.
                        ;
                        ; VERSION:
                        ;       0.0
                        ; EDIT          BY              DATE            REASON
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES
                        ;
                                .MCALL  STRUCT
                                STRUCT
000000'                 .PRINT  ;SPMAC: VERSION 1.1
(1) 000000'
                        000001  $LSTIN = 1
                        000001  $LSTTAG = 1
                        ;
                        ;
                        ;
                        ;****************
                        ;
                        ;REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  HT.EXT          ;ADDRESS TO EXIT THIS MODULE THROUGH
                        .GLOBL  HTBUSS          ;ENTRY POINT FOR BUSS ERROR TRAP
                        .GLOBL  HTREIN          ;ENTRY POINT FOR RESERVED INSTRUCTION TRAP
                        .GLOBL  HTPAER          ;ENTRY POINT FOR PARITY ERROR TRAP
                        .GLOBL  HTKT            ;ENTRY POINT FOR KT TRAP
                        ;
                        ;****************
                        ;
                        ;LOCAL STORAGE:
                        ;
000000' 000000          HT.EXT: .WORD   0       ;ADDRESS TO EXIT THIS MODULE THRU
                        ;
```

```
                        .SBTTL HTRAP ROUTINE
                        ;
                        ;+
                        ;ENTER HERE ON BUSS ERROR TRAP
                        ;-
000002'                 INLINE <HTBUSS:>
000002'                                                                 HTBUSS:
000002'                         PUSH    #EVNTBE                         MOV     #EVNTBE,-(SP)
000002' 012746  000200
000006' 000177  177766          JMP     @HT.EXT         ;EXIT
                        ;+
                        ;ENTER HERE ON RESERVED INSTR. TRAP
                        ;-
000012'                 INLINE <HTREIN:>
000012'                                                                 HTREIN:
000012'                         PUSH    #EVNTRE                         MOV     #EVNTRE,-(SP)
000012' 012746  000201
000016' 000177  177756          JMP     @HT.EXT         ;EXIT
                        ;+
                        ;ENTER HERE ON PARITY ERROR TRAP
                        ;-
000022'                 INLINE <HTPAER:>
000022'                                                                 HTPAER:
000022'                         PUSH    #EVNTPE                         MOV     #EVNTPE,-(SP)
000022' 012746  000202
000026' 000177  177746          JMP     @HT.EXT         ;EXIT
                        ;+
                        ;ENTER HERE ON MEMORY MANAGEMENT ERROR
                        ;
000032'                 INLINE <HTKT:>
000032'                                                                 HTKT:
000032'                         PUSH    #EVNTKT                         MOV     #EVNTKT,-(SP)
000032' 012746  000203
000036' 000177  177736          JMP     @HT.EXT         ;EXIT

        000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQCVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
HTBUSS  000002RG
HTKT    000032RG
HTPAER  000022RG
HTREIN  000012RG
HT.EXT  000000RG
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RJBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
```

```
TQDVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616

WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000000
$F$AND= 000310
$F$BAD= 000401

$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404

$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050000
$$ARGC= 000000
$$BYTE= 000000
$$CASE= 000000
$$DST = 000000
$$ELOC= 000000
$$ERFL= 000000

$$FLAG= 000000
$$FROM= 000000
$$LOC = 000000
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 000000
$$RTN2= 000000
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000042R

. ABS.  000000  000
        000042  001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:HTRAP,DSKZ:HTRAP=SPMAC/ML,EQUATE,HTRAP
RUN-TIME: 10 .7 .3 SECONDS
RUN-TIME RATIO: 26/11=2.2
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  ICSROO INDIRECT ECC CSRS ON/OFF
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       ICSROO
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       ROUTINE WILL TURN ON/OFF ECC MEMORY IF CSRS ARE LOCATED ON THE MAIN
                        ;       MEMORY BUSS.
                        ;       THIS IS ACCOMPLISHED AS FOLLOWS:
                        ;                 - NO NPR ACTIVITY
                        ;                 - TURN OFF CACHE
                        ;                 - MAP KIPAR6 TO 177400
                        ;                 - ENABLE KT AND 22 BIT ADDRESSING
                        ;                 - SET UP MAP REG 36 TO 17760000
                        ;                 - TURN MAP BOX ON
                        ;                 - ACCESS CSR REGISTERS LISTED IN PARITY TABLE AND
                        ;                   TAKE APPROPRIATE ACTION. (ON/OFF)
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;       ON/OFF FLAG
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PTA
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ICSROO IN <DTABLE,FLAG>
                        ;                 DTABLE - ADDRESS OF DTABLE
                        ;                 FLAG - ECC ON/OFF FLAG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN = 1
            000001              $LSTTAG = 1



                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  ICSROO                  ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  CCNTRL                  ;CACHE CONTROL REGISTER

                                ;****************
                                ; LOCAL EQUATES
                                ;
            152100              IN.PCSR = 152100                ;START OF PARITY CSRS BY INDIRECT MODE
            152136              IN.PCLS = 152136                ;LAST PARITY CSR BY INDIRECT MODE
            170370              IN.M36  = 170370                ;MAP REG 36 LOWER 16 BITS
            170372              IN.M37  = 170372                ;MAP REG 36 UPPER 6 BITS
```

```
                        .SBTTL INDIRECT ECC CSR ON/OFF ROUTINE

000000'                 ROUTINE ICSROO <DTABLE,FLAG>
(2) 000000'                                                     ICSROO:


                        ;+
                        ; SAVE REGISTERS AND GET ADDRESS OF PARITY TABLE
                        ;-

000000'                 PUSH R0,R1
(2) 000000' 010046                                              MOV     R0,-(SP)
(3) 000002' 010146                                              MOV     R1,-(SP)
000004'                 LET R0 := DTABLE(R5)
(4) 000004' 016500 000000                                       MOV     DTABLE(R5),R0
000010'                 LET R1 := DT.PTA(R0)
(4) 000010' 016001 000064                                       MOV     DT.PTA(R0),R1

                        ;+
                        ; SAVE STATUS OF CACHE, KT AND UNIBUS MAP REGISTERS
                        ;-

000014'                 PUSH @CCNTRL
(2) 000014' 017746 000000G                                      MOV     @CCNTRL,-(SP)
000020'                 PUSH @#SR0
(2) 000020' 013746 177572                                       MOV     @#SR0,-(SP)
000024'                 PUSH @#SR3
(2) 000024' 013746 172516                                       MOV     @#SR3,-(SP)
000030'                 PUSH @#IN.M36
(2) 000030' 013746 170370                                       MOV     @#IN.M36,-(SP)
000034'                 PUSH @#IN.M37
(2) 000034' 013746 170372                                       MOV     @#IN.M37,-(SP)
000040'                 PUSH @#KIPAR6
(2) 000040' 013746 172354                                       MOV     @#KIPAR6,-(SP)

                        ;+
                        ; TURN CACHE OFF, TURN KT ON, SETUP KIPAR6 TO POINT TO
                        ; MAP REG 36 AND SETUP MAP REGISTER 36 TO LOOK AT 128K I/O PAGE SHADOW AREA
                        ;-

000044'                 LET @CCNTRL := @CCNTRL SET.BY #14
(6) 000044' 052777 000014 000000G                               BIS     #14,@CCNTRL
000052'                 LET @#KIPAR6 := #177400
(4) 000052' 012737 177400 172354                                MOV     #177400,@#KIPAR6
000060'                 LET @#SR0 := @#SR0 SET.BY #BIT00
(6) 000060' 052737 000001 177572                                BIS     #BIT00,@#SR0
000066'                 LET @#SR3 := @#SR3 SET.BY #60
(6) 000066' 052737 000060 172516                                BIS     #60,@#SR3
000074'                 LET @#IN.M36 := #160000
(4) 000074' 012737 160000 170370                                MOV     #160000,@#IN.M36
000102'                 LET @#IN.M37 := #77
(4) 000102' 012737 000077 170372                                MOV     #77,@#IN.M37

                        ;+
                        ; WHILE ENTRY IN PARITY TABLE TAKE THE APPROPRIATE ACTION OF TURNING ON/OFF ECC MEMORY.
                        ;-
```

```
                        INDIRECT ECC CSR ON/OFF ROUTINE

                        WHILE (R1) NE #ENDLST DO
                                                                50002$:
                                                                        CMP     (R1),#ENDLST
                                                                        BEQ     50003$

                                IF FLAG(R5) EQ #0 THEN
                                                                        TST     FLAG(R5)
                                                                        BNE     50004$
                                        LET @(R1)+ := @(R1)+ SET.BY #3
                                                                        BIS     #3,@(R1)+
                                ELSE
                                                                        BR      50005$
                                                                50004$:
                                        LET @(R1)+ := @(R1)+ CLR.BY #3
                                                                        BIC     #3,@(R1)+
                                ENDIF
                                                                50005$:

                        ENDDO
                                                                        BR      50002$
                                                                50003$:

                        ;+
                        ; NOW RESTORE STATUS TO CACHE, KT AND MAP REGISTERS
                        ;-

                        POP @#KIPAR6                                    MOV     (SP)+,@#KIPAR6

                        POP @#IN.M37                                    MOV     (SP)+,@#IN.M37

                        POP @#IN.M36                                    MOV     (SP)+,@#IN.M36

                        POP @#SR3                                       MOV     (SP)+,@#SR3

                        POP @#SR0                                       MOV     (SP)+,@#SR0

                        POP @CCNTRL                                     MOV     (SP)+,@CCNTRL

                        ;+
                        ; NOW RESTORE REGISTERS AND RETURN
                        ;-

                        POP R1,R0                                       MOV     (SP)+,R1
                                                                        MOV     (SP)+,R0

                        ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC

                        .END
```

```
ACSR  = 000102        CSRC  = 000102        ENDLST= 000000        LF    = 000012        PS    = 177776
ACTBIT= 004000        CTRLC = 000003        EOPBIT= 000001        LPSTAT= 000001        PSW   = 177776
ADDR22= 001000        CTRLO = 000017        ERRTYP= 000106        MAPSTA= 000200        RANNUM= 000054
ADR   = 000006        CTRLU = 000025        EVNTBE= 000200        MED   = 076600        RBUFEA= 000130
APTFER= 000004        DCEVNT= 000011        EVNTHD= 000200        MEMPAS= 040000        RBUFPA= 000126
APTPRE= 000200        DEFRTN= 000400        EVNTKT= 000203        MODEXH= 004000        RBUFSZ= 000132
ASB   = 000106        DIAGMC= 000000        EVNTPE= 000202        MODHOL= 002000        RBUFVA= 000124
ASSEMB= 000010        DROPMO= 100000        EVNTRE= 000201        MODSEL= 001000        RDSERV= 000101
ASTAT = 000104        DSEVNT= 000014        FATERR= 100000        MSGCKD= 000010        RDWHMI= 000022
AUTO  = 000010        DTABLE= 000000        FLAG  = 000002        MSGCKS= 000011        RELERR= 000020
AUTOST= 020000        DT.ADD= 000042        HRDCNT= 000044        MSGDER= 000005        RELMOD= 020000
AWAS  = 000110        DT.AP = 000100        HRDPAS= 000050        MSGDRP= 000017        RELTIM= 010000
BIT0  = 000001        DT.APK= 000076        ICONT = 000036        MSGECH= 177777        RES1  = 000056
BIT00 = 000001        DT.BLS= 000034        ICOUNT= 000040        MSGEOP= 000013        RES2  = 000060
BIT01 = 000002        DT.CF0= 000014        ICSR00  000000RG      MSGHDR= 000004        RICHAR= 031060
BIT02 = 000004        DT.CF1= 000016        IDNUM = 000122        MSGHNG= 000022        RPTDAT= 002000
BIT03 = 000010        DT.ERR= 000020        IE    = 000100        MSGHRD= 000007        RSTRT = 000112
BIT04 = 000020        DT.ESI= 000044        INDPAR= 000040        MSGMAP= 000021        RUBOUT= 000177
BIT05 = 000040        DT.EVN= 000000        INHDRP= 040000        MSGNUL= 177775        RUNMOD= 100000
BIT06 = 000100        DT.EXS= 000060        INHEPR= 020000        MSGPOP= 000002        R5VALU= 001740
BIT07 = 000200        DT.FCH= 000037        INHREL= 001000        MSGPRM= 177776        SAM   = 075464
BIT08 = 000400        DT.FCN= 000036        INHRRE= 000400        MSGRES= 000001        SBADR = 000102
BIT09 = 001000        DT.HMX= 000104        INIT  = 000030        MSGSFT= 000006        SBKMOD= 000000
BIT1  = 000002        DT.KBE= 000024        INTR  = 000120        MSGSKE= 000003        SBKSEL= 010000
BIT10 = 002000        DT.KBP= 000026        IN.M36= 170370        MSGSMB= 000015        SC.ADR= 000006
BIT11 = 004000        DT.KBR= 000022        IN.M37= 170372        MSGSMH= 000014        SC.ALC= 000014
BIT12 = 010000        DT.KBU= 000030        IN.PCL= 152136        MSGSMS= 000016        SC.APC= 000016
BIT13 = 020000        DT.MLS= 000032        IN.PCS= 152100        MSGSTD= 000000        SC.CKL= 000002
BIT14 = 040000        DT.MTI= 000110        IOMOD = 100000        MSGSYS= 000012        SC.CKP= 000004
BIT15 = 100000        DT.OFF= 000070        IOMODP= 102000        MSGVEC= 000020        SC.CLD= 000000
BIT2  = 000004        DT.PAS= 000074        IOMODR= 112000        NBKMOD= 001000        SC.HLD= 000010
BIT3  = 000010        DT.PC = 000002        IOMODX= 110000        NCPUOP= 000020        SC.SCA= 000012
BIT4  = 000020        DT.PFL= 000062        JACK  = 035060        NOAPTY= 000002        SENDLS= 177777
BIT5  = 000040        DT.PSW= 000004        KIPAR0= 172340        NULL  = 000000        SOFCNT= 000042
BIT6  = 000100        DT.PTA= 000064        KIPAR1= 172342        OWEN  = 024020        SOFPAS= 000046
BIT7  = 000200        DT.RCS= 000102        KIPAR2= 172344        PAERR = 000010        SPACE = 000040
BIT8  = 000400        DT.REL= 000040        KIPAR3= 172346        PARPRE= 002000        SPOINT= 000032
BIT9  = 001000        DT.SCT= 000066        KIPAR4= 172350        PARSTA= 000100        SPVALU= 002200
BKDEF = 000002        DT.SMX= 000106        KIPAR5= 172352        PASCNT= 000034        SR0   = 177572
BKMOD = 000020        DT.SP = 000006        KIPAR6= 172354        PDPLSI= 020000        SR1   = 177574
BKMODE= 040000        DT.SSI= 000046        KIPAR7= 172356        PDP60 = 004000        SR2   = 177576
BKSLSH= 000134        DT.ST0= 000010        KIPDR0= 172300        PDP70 = 010000        SR3   = 172516
CAPRES= 000004        DT.ST1= 000012        KIPDR1= 172302        PRI0  = 000000        STAT  = 000026
CASTAT= 000004        DT.SWR= 000056        KIPDR2= 172304        PRI1  = 000040        STATBI= 064757
CCNTRL= ****** G      DT.SYP= 000072        KIPDR3= 172306        PRI4  = 000200        STAT1 = 000027
CDERCT= 000146        DT.WBU= 000050        KIPDR4= 172310        PRI5  = 000240        SUSPND= 000001
CDWDCT= 000144        DT.WHL= 000054        KIPDR5= 172312        PRI6  = 000300        SVR0  = 000062
CKTIM = 100000        DT.WLL= 000052        KIPDR6= 172314        PRI7  = 000340        SVR1  = 000064
CLKPRE= 000001        DVID1 = 000014        KIPDR7= 172316        PR0   = 000000        SVR2  = 000066
CONFIG= 000056        ECCMEM= 000100        KTERRO= 000040        PR4   = 000200        SVR3  = 000070
CQOVF = 000001        ECCSTA= 000010        KTPRES= 000400        PR5   = 000240        SVR4  = 000072
CR    = 000015        ENBEOP= 010000        KTSTAT= 000020        PR6   = 000300        SVR5  = 000074
CSRA  = 000100        ENBNUL= 000001        KTXTND= 040000        PR7   = 000340        SVR6  = 000076
```

```
SYSCNT= 000052          UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$CASE= 000000
SYSERR= 000100          WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$DST = 000000
TMPIO = 000002          WBSTAT= 000040          $F$DO = 000340          $LOCTA= 177777          $$ELOC= 000402
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LSTIN= 000001          $$ERFL= 000000
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LSTTA= 000001          $$FLAG= 000001
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $NESTL= 177777          $$FROM= 000000
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NSK0 = 000300          $$LOC = 000122R
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK1 = 000120          $$LOCN= 000000
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK2 = 000110          $$REG = 177777
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $SAVLE= 177777          $$RETU= 000000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $SSK0 = 050003          $$RTN1= 050000
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $TAGLE= 177777          $$RTN2= 050001
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $TAGNU= 050006          $$SRC = 000000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TEMP = 000300          $$TGSV= 000000
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TSK0 = 050002          $$TGS1= 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TSK1 = 050003          $$TGS2= 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $TSK2 = 050005          $$TO  = 000000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$ARGC= 000004          $$$TAG= 050000
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$BYTE= 000403          .     = 000176R

. ABS.  000000    000
        000176    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:ICSROO,DSKZ:ICSROO=SPMAC/ML,EQUATE,ICSROO
 RUN-TIME: 12 2 .3 SECONDS
 RUN-TIME RATIO: 31/16=1.9
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  ICSRSZ INDIRECT ECC MEMORY CSR SIZING ROUTINE
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       ICSRSZ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL DETERMINE IF ECC MEMORY CSRS ARE HIDDEN ON
                        ;       THE MAIN MEMORY BUS OF THE 11/70 SYSTEM.
                        ;       THE MANNER IN WHICH THIS IS ACCOMPLISHED IS BY
                        ;       THE FOLLOWING:
                        ;                 - NO NPR ACTIVITY
                        ;                 - TURN CACHE OFF
                        ;                 - MAP KIPAR6 TO 177400
                        ;                 - ENABLE MEMORY MANAGEMENT IN 22 BIT MODE
                        ;                 - MAP UNIBUS MAP REGISTER 36 TO 17760000
                        ;                 - TURN ON UNIBUS MAP BOX
                        ;                 - ACCESS VIRTUAL ADDRESSES 152100 THROUGH 152136
                        ;                   AS ECC MEMORY CSRS
                        ;       IF ECC FOUND THEN THE CSR ADDRESS WILL BE ENTERED INTO THE PARITY TABLE.
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PTA
                        ;       DT.CF0
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       KTSET - SET UP KT REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ICSRSZ IN <DTABLE>
                        ;                 DTABLE - ADDRESS OF DTABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
```

```
   564                                  ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN = 1
        000001                  $LSTTAG = 1


                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  ICSRSZ                  ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SARE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  KTSET                   ;SET UP KT
                                .GLOBL  HRDADRCHK               ;ILLEGAL ADDRESS CHECK
                                .GLOBL  CCNTRL                  ;CACHE CONTROL REGISTER
                                ;****************
                                ; LOCAL EQUATES
                                ;
        152100                  IC.PCSR = 152100                ;START OF PARITY CSRS BY INDIRECT MODE
        152136                  IC.PCLS = 152136                ;LAST PARITY CSR BY INDIRECT MODE
        170370                  IC.M36  = 170370                ;MAP REG 36 LOWER 16 BITS
        170372                  IC.M37  = 170372                ;MAP REG 36 UPPER 6 BITS
```

```
                                .SBTTL  INDIRECT ECC MEMORY CSR SIZING ROUTINE

                                ROUTINE ICSRSZ <DTABLE>
                                                                        ICSRSZ:

                                ;+
                                ; SAVE REGISTERS AND GET ADDRESS OF DTABLE
                                ;-

                                CALL SAVREG
                                                                        JSR     PC,SAVREG
                                LET R0 := DTABLE(R5)
                                                                        MOV     DTABLE(R5),R0

                                ;+
                                ; SAVE CONTENTS OF CACHE CSR, KT STATUS, KIPAR6, AND MAP REGISTER
                                ;-

                                PUSH @CCNTRL
                                                                        MOV     @CCNTRL,-(SP)
                                PUSH @#SR0
                                                                        MOV     @#SR0,-(SP)
                                PUSH @#SR3
                                                                        MOV     @#SR3,-(SP)
                                PUSH @#IC.M36
                                                                        MOV     @#IC.M36,-(SP)
                                PUSH @#IC.M37
                                                                        MOV     @#IC.M37,-(SP)
                                PUSH @#KIPAR6
                                                                        MOV     @#KIPAR6,-(SP)

                                ;+
                                ; TURN CACHE OFF, SETUP KT, SETUP KIPAR6 TO POINT TO MAP REG 36,
                                ; AND MAP REGISTER 36 TO LOOK AT 128K I/O SHADOW AREA.
                                ;-

                                LET @CCNTRL := @CCNTRL SET.BY #14
                                                                        BIS     #14,@CCNTRL
                                CALL KTSET IN <R0>
                                                                        MOV     R5,-(SP)
                                                                        MOV     R0,-(R5)
                                                                        JSR     PC,KTSET
                                                                        MOV     (SP)+,R5
                                LET @#KIPAR6 := #177400
                                                                        MOV     #177400,@#KIPAR6
                                LET @#SR0 := @#SR0 SET.BY #BIT00
                                                                        BIS     #BIT00,@#SR0
                                LET @#SR3 := @#SR3 SET.BY #60
                                                                        BIS     #60,@#SR3
                                LET R3 := #IC.PCSR
                                                                        MOV     #IC.PCSR,R3
                                LET @#IC.M36 := #160000
                                                                        MOV     #160000,@#IC.M36
                                LET @#IC.M37 := #77
                                                                        MOV     #77,@#IC.M37
```

```
                                ;+
                                ; GET ADDRESS OF PARITY TABLE
                                ;-

000122'                         LET R1 := DT.PTA(R0)
(4) 000122' 016001  000064                                              MOV     DT.PTA(R0),R1

                                ;+
                                ; NOW DETERMINE IF ANY PARITY CSRS EXIST INDIRECTLY, IF SO LOAD ADDRESS INTO
                                ; PARITY TABLE AND SET ECC MEMORY PRESENT AND ALSO INDIRECT PARITY REG BIT.
                                ;-

000126'                         WHILE R3 LOS #IC.PCLS DO
(4) 000126'                                                         50002$:
(6) 000126' 020327  152136                                              CMP     R3,#IC.PCLS
(9) 000132' 101021                                                      BHI     50003$
000134'                                 CALL HRDADRCHK IN <R3>
(3) 000134' 010546                                                      MOV     R5,-(SP)
(4) 000136' 010345                                                      MOV     R3,-(R5)
(3) 000140' 004767  000000G                                             JSR     PC,HRDADRCHK
(3) 000144' 012605                                                      MOV     (SP)+,R5
000146'                                 IF.NO.ERROR THEN
(6) 000146' 103410                                                      BCS     50004$
000150'                                         LET (R1)+ := R3
(4) 000150' 010321                                                      MOV     R3,(R1)+
000152'                                         LET DT.CF0(R0) := DT.CF0(R0) SET.BY #ECCMEM
(6) 000152' 052760  000100  000014                                      BIS     #ECCMEM,DT.CF0(R
000160'                                         LET DT.CF0(R0) := DT.CF0(R0) SET.BY #INDPAR
(6) 000160' 052760  000040  000014                                      BIS     #INDPAR,DT.CF0(R

                                ;+
                                ; CLEAR OUT CSR, THUS TURNING OFF ECC
                                ;-

000166'                                 LET (R3) := #0
(4) 000166' 005013                                                      CLR     (R3)
000170'                                 ENDIF
(4) 000170'                                                         50004$:

                                ;+
                                ; UPDATE CSR ADDRESS AND GO LOOK AGAIN
                                ;-

000170'                                 LET R3 := R3 + #4
(6) 000170' 062703  000004                                              ADD     #4,R3
000174'                         ENDDO
(4) 000174' 000754                                                      BR      50002$
000176'                                                             50003$:

                                ;+
                                ; NOW RESTORE CACHE, KT AND MAP REGISTERS
                                ;-

000176'                         POP @#KIPAR6
(2) 000176' 012637  172354                                              MOV     (SP)+,@#KIPAR6
```

```
  669  000202'                        POP @#IC.M37
  (2)  000202' 012637  170372                                          MOV     (SP)+,@#IC.M37
  670  000206'                        POP @#IC.M36
  (2)  000206' 012637  170370                                          MOV     (SP)+,@#IC.M36
  671  000212'                        POP @#SR3
  (2)  000212' 012637  172516                                          MOV     (SP)+,@#SR3
  672  000216'                        POP @#SR0
  (2)  000216' 012637  177572                                          MOV     (SP)+,@#SR0
  673  000222'                        POP @CCNTRL
  (2)  000222' 012677  000000G                                        MOV     (SP)+,@CCNTRL
  674
  675                                 ;+
  676                                 ; RESTORE REGISTERS AND RETURN TO CALLER
  677                                 ;-
  678
  679  000226'                        CALL RESREG
  (3)  000226' 004767  000000G                                        JSR     PC,RESREG
  680
  681  000232'                        ENDRTN
  (3)  000232'                                                     50000$:
  (3)  000232'                                                     50001$:
  (2)  000232' 000207                                                  RTS     PC
  682          000001                 .END
```

```
ACSR  = 000102          CSRC  = 000102          ENDLST= 000000          KTXTND= 040000          PR7   = 000340
ACTBIT= 004000          CTRLC = 000003          EOPBIT= 000001          LF    = 000012          PS    = 177776
ADDR22= 001000          CTRLO = 000017          ERRTYP= 000106          LPSTAT= 000001          PSW   = 177776
ADR   = 000006          CTRLU = 000025          EVNTBE= 000200          MAPSTA= 000200          RANNUM= 000054
APTFER= 000004          DCEVNT= 000011          EVNTHD= 000200          MED   = 076600          RBUFEA= 000130
APTPRE= 000200          DEFRTN= 000400          EVNTKT= 000203          MEMPAS= 040000          RBUFPA= 000126
ASB   = 000106          DIAGMC= 000000          EVNTPE= 000202          MODEXH= 004000          RBUFSZ= 000132
ASSEMB= 000010          DROPMO= 100000          EVNTRE= 000201          MODHOL= 002000          RBUFVA= 000124
ASTAT = 000104          DSEVNT= 000014          FATERR= 100000          MODSEL= 001000          RDSERV= 000101
AUTO  = 000010          DTABLE= 000000          HRDADR= ****** G        MSGCKD= 000010          RDWHMI= 000022
AUTOST= 020000          DT.ADD= 000042          HRDCNT= 000044          MSGCKS= 000011          RELERR= 000020
AWAS  = 000110          DT.AP = 000100          HRDPAS= 000050          MSGDER= 000005          RELMOD= 020000
BIT0  = 000001          DT.APK= 000076          ICONT = 000036          MSGDRP= 000017          RELTIM= 010000
BIT00 = 000001          DT.BLS= 000034          ICOUNT= 000040          MSGECH= 177777          RESREG= ****** G
BIT01 = 000002          DT.CF0= 000014          ICSRSZ  000000RG        MSGEOP= 000013          RES1  = 000056
BIT02 = 000004          DT.CF1= 000016          IC.M36= 170370          MSGHDR= 000004          RES2  = 000060
BIT03 = 000010          DT.ERR= 000020          IC.M37= 170372          MSGHNG= 000022          RICHAR= 031060
BIT04 = 000020          DT.ESI= 000044          IC.PCL= 152136          MSGHRD= 000007          RPTDAT= 002000
BIT05 = 000040          DT.EVN= 000000          IC.PCS= 152100          MSGMAP= 000021          RSTRT = 000112
BIT06 = 000100          DT.EXS= 000060          IDNUM = 000122          MSGNUL= 177775          RUBOUT= 000177
BIT07 = 000200          DT.FCH= 000037          IE    = 000100          MSGPOP= 000002          RUNMOD= 100000
BIT08 = 000400          DT.FCN= 000036          INDPAR= 000040          MSGPRM= 177776          R5VALU= 001740
BIT09 = 001000          DT.HMX= 000104          INHDRP= 040000          MSGRES= 000001          SAM   = 075464
BIT1  = 000002          DT.KBE= 000024          INHEPR= 020000          MSGSFT= 000006          SAVREG= ****** G
BIT10 = 002000          DT.KBP= 000026          INHREL= 001000          MSGSKE= 000003          SBADR = 000102
BIT11 = 004000          DT.KBR= 000022          INHRRE= 000400          MSGSMB= 000015          SBKMOD= 000000
BIT12 = 010000          DT.KBU= 000030          INIT  = 000030          MSGSMH= 000014          SBKSEL= 010000
BIT13 = 020000          DT.MLS= 000032          INTR  = 000120          MSGSMS= 000016          SC.ADR= 000006
BIT14 = 040000          DT.MTI= 000110          IOMOD = 100000          MSGSTD= 000000          SC.ALC= 000014
BIT15 = 100000          DT.OFF= 000070          IOMODP= 102000          MSGSYS= 000012          SC.APC= 000016
BIT2  = 000004          DT.PAS= 000074          IOMODR= 112000          MSGVEC= 000020          SC.CKL= 000002
BIT3  = 000010          DT.PC = 000002          IOMODX= 110000          NBKMOD= 001000          SC.CKP= 000004
BIT4  = 000020          DT.PFL= 000062          JACK  = 035060          NCPUOP= 000020          SC.CLO= 000000
BIT5  = 000040          DT.PSW= 000004          KIPAR0= 172340          NDAPTY= 000002          SC.HLD= 000010
BIT6  = 000100          DT.PTA= 000064          KIPAR1= 172342          NULL  = 000000          SC.SCA= 000012
BIT7  = 000200          DT.RCS= 000102          KIPAR2= 172344          OWEN  = 024020          SENDLS= 177777
BIT8  = 000400          DT.REL= 000040          KIPAR3= 172346          PAERR = 000010          SOFCNT= 000042
BIT9  = 001000          DT.SCT= 000066          KIPAR4= 172350          PARPRE= 002000          SOFPAS= 000046
BKDEF = 000002          DT.SMX= 000106          KIPAR5= 172352          PARSTA= 000100          SPACE = 000040
BKMOD = 000020          DT.SP = 000006          KIPAR6= 172354          PASCNT= 000034          SPOINT= 000032
BKMODE= 040000          DT.SSI= 000046          KIPAR7= 172356          PDPLSI= 020000          SPVALU= 002200
BKSLSH= 000134          DT.ST0= 000010          KIPDR0= 172300          PDP60 = 004000          SR0   = 177572
CAPRES= 000004          DT.ST1= 000012          KIPDR1= 172302          PDP70 = 010000          SR1   = 177574
CASTAT= 000004          DT.SWR= 000056          KIPDR2= 172304          PRI0  = 000000          SR2   = 177576
CCNTRL= ****** G        DT.SYP= 000072          KIPDR3= 172306          PRI1  = 000040          SR3   = 172516
CDERCT= 000146          DT.WBU= 000050          KIPDR4= 172310          PRI4  = 000200          STAT  = 000026
CDWDCT= 000144          DT.WHL= 000054          KIPDR5= 172312          PRI5  = 000240          STATBI= 064757
CKTIM = 100000          DT.WLL= 000052          KIPDR6= 172314          PRI6  = 000300          STAT1 = 000027
CLKPRE= 000001          DVID1 = 000014          KIPDR7= 172316          PRI7  = 000340          SUSPND= 000001
CONFIG= 000056          ECCMEM= 000100          KTERRO= 000040          PR0   = 000000          SVR0  = 000062
CQOVF = 000001          ECCSTA= 000010          KTPRES= 000400          PR4   = 000200          SVR1  = 000064
CR    = 000015          ENBEOP= 010000          KTSET = ****** G        PR5   = 000240          SVR2  = 000066
CSRA  = 000100          ENBNUL= 000001          KTSTAT= 000020          PR6   = 000300          SVR3  = 000070
```

```
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR   = 000116
WDTO   = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG  = 000005
XOFF   = 000023
XON    = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO  = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF  = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO  = 000403
$F$OR  = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0  = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0  = 000300
$NSK1  = 000120
$NSK2  = 000110
$SAVLE= 177777
$SSK0  = 050003
$TAGLE= 177777
$TAGNU= 050005
$TEMP  = 000300
$TSK0  = 050002
$TSK1  = 050003
$TSK2  = 050004
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST  = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC  = 000146R
$$LOCN= 000000
$$REG  = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC  = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO   = 000000
$$$TAG= 050000
.      = 000234R

. ABS.  000000     000
        000234     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:ICSRSZ,DSKZ:ICSRSZ=SPMAC/ML,EQUATE,ICSRSZ
 RUN-TIME: 13 3 .3 SECONDS
 RUN-TIME RATIO: 32/17=1.9
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KEXAM  PROCESS THE 'EXAM' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KEXAM
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'EXAM' KEYBOARD COMMAND. IT WILL
                        ;       DETERMINE IF AN ARGUMENT IS INCLUDED IN THE COMMAND STRING.
                        ;       IF NO ARGUMENTS ARE INCLUDED, THE LAST LOCATION EXAMINED
                        ;       WILL BE OUTPUTTED. IF ARGUMENTS ARE INCLUDED, THE MODULE
                        ;       NAME AND/OR THE OCTAL ADDRESS WILL BE VERIFIED AND IF
                        ;       NO ERRORS, THE APPROPRIATE LOCATION AND
                        ;       ITS CONTENTS WILL BE OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.STO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.BADNAME,CM.ADR,CM.ODD,CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK - CHECK FOR ARGUMENTS
                        ;       SAVREG
                        ;       RESREG
                        ;       BOA16
                        ;       NUMCHK
                        ;       NAMCHK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KEXAM IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;               BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KEXAM

                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1

          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KE.LOC          ;INITIALIZED TO 0 IF RUN CMD IS GIVEN
                                .GLOBL  KEXAM           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONV ROUTINE
                                .GLOBL  ARGCHK          ;ROUTINE TO CHECK AN ARGUMENT
                                .GLOBL  CM.BADNAME      ;"INVALID MODULE NAME" MESSAGE
                                .GLOBL  CM.ADR          ;"BAD ADDRESS" MESSAGE
                                .GLOBL  NUMCHK          ;VERIFY A NUMBER IS OCTAL, &
                                                        ;16 BITS OR LESS IN LENGTH
                                .GLOBL  CM.ODD          ;'NOT AN EVEN NUMBER' MSG ADDRESS
                                .GLOBL  NAMCHK          ;VERIFY A MODULE NAME
                                .GLOBL  CM.ARG          ;"INVALID ARGUMENT" MESSAGE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  KE.LOC: .WORD   0       ;HOLD LOCATION TO BE EXAMINED
000002'                         KE.MG1:
000002' 000006                  KE.MG2: .BLKB   6       ;HOLD ASCII OF ADDRESS TO BE EXAMINED
000010' 020057                          .ASCII  '/ '    ;SEPARATOR
000012' 000006                  KE.MG3: .BLKB   6       ;HOLD ASCII CONTENTS OF ADDRESS
000020' 000045                          .ASCIZ  /%/
                                .EVEN
                                ;
                                ;****************
```

```
                                .SBTTL  KEXAM ROUTINE

000022'                         ROUTINE KEXAM <DTADR,CMDBUF>
(2)  000022'                                                                KEXAM:

                                ;+
                                ; SAVE REGISTERS AND SAVE ADDRESS OF DTABLE & ADDRESS OF DECODE BUFFER
                                ;-

000022'                         CALL SAVREG
(3)  000022' 004767  000000G                                                JSR     PC,SAVREG

000026'                         LET R0 := DTADR(R5)
(4)  000026' 016500  000000                                                 MOV     DTADR(R5),R0
000032'                         LET R1 := CMDBUF(R5)
(4)  000032' 016501  000002                                                 MOV     CMDBUF(R5),R1

                                ;+
                                ; INIT R2 FOR LATER USE
                                ;-

000036'                         LET R2 := #0
(4)  000036' 005002                                                         CLR     R2

                                ;+
                                ; GET TO FIRST ARGUMENT
                                ;-

000040'                         CALL ARGCHK IN <R1> OUT <R1>
(4)  000040' 162705  000002                                                 SUB     #1*2,R5
(3)  000044' 010546                                                         MOV     R5,-(SP)
(4)  000046' 010145                                                         MOV     R1,-(R5)
(3)  000050' 004767  000000G                                                JSR     PC,ARGCHK
(3)  000054' 012605                                                         MOV     (SP)+,R5
(4)  000056' 012501                                                         MOV     (R5)+,R1

                                ;+
                                ; IF THERE IS A CR AS FIRST ARGUMENT - CONVERT LAST EXAMINED ADDRESS
                                ; TO ASCII, OUTPUT IT, AND RETURN
                                ;-
000060'                         IF.ERROR THEN
(6)  000060' 103003                                                         BCC     50002$

000062'                                 CALL CONV
(3)  000062' 004767  000264                                                 JSR     PC,CONV



000066'                         ELSE
(4)  000066' 000526                                                         BR      50003$
(3)  000070'                                                        50002$:

                                ;+
                                ; THERE IS AN ARGUMENT - FIND OUT WHETHER ALPHA OR NUMERIC
```

```
                                ;-
                                ;
                                        IFB (R1) LT #'0 ORB (R1) GT #'9 THEN
000070'                                                                         CMPB    (R1),#'0
(6)  000070' 121127  000060                                                     BLT     50004$
(8)  000074' 002403                                                             CMPB    (R1),#'9
(6)  000076' 121127  000071                                                     BLE     50005$
(9)  000102' 003440                                                     50004$:
(6)  000104'
                                ;+
                                ; IT'S ALPHABETIC - SO VERIFY MODULE NAME
                                ;-
                                ;
                                                CALL NAMCHK IN <R0,R1> OUT <R2,R1>
000104'                                                                         SUB     #2*2,R5
(4)  000104' 162705  000004                                                     MOV     R5,-(SP)
(3)  000110' 010546                                                             MOV     R1,-(R5)
(5)  000112' 010145                                                             MOV     R0,-(R5)
(4)  000114' 010045                                                             JSR     PC,NAMCHK
(3)  000116' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000122' 012605                                                             MOV     (R5)+,R2
(4)  000124' 012502                                                             MOV     (R5)+,R1
(4)  000126' 012501
                                ;+
                                ; IF THE MODULE NAME IS BAD - STUFF ERROR MSG AND LEAVE
                                ;-
                                ;
                                                IF.ERROR THEN
000130'                                                                         BCC     50006$
(6)  000130' 103006
                                ;
                                                        LET DT.KBRSP(R0) := #CM.BADNAME
000132'                                                                         MOV     #CM.BADNAME,DT.K
(4)  000132' 012760  000000G 000022                     CALL RESREG
000140'                                                                         JSR     PC,RESREG
(3)  000140' 004767  000000G                            RETURN
000144'                                                                         BR      50000$
(4)  000144' 000501
                                ;
                                                ELSE
000146'                                                                 50006$:
(3)  000146'
                                ;+
                                ; OTHERWISE MODULE NAME IS O.K. - ADVANCE TO NEXT ARGUMENT
                                ;-
                                ;
                                                CALL ARGCHK IN <R1> OUT <R1>
000146'                                                                         SUB     #1*2,R5
(4)  000146' 162705  000002                                                     MOV     R5,-(SP)
(3)  000152' 010546                                                             MOV     R1,-(R5)
(4)  000154' 010145                                                             JSR     PC,ARGCHK
(3)  000156' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000162' 012605                                                             MOV     (R5)+,R1
(4)  000164' 012501
                                ;+
                                ; IF THERE IS NO ADDITIONAL ARGUMENT - YOU CANNOT HAVE A MODULE
                                ; NAME WITHOUT NUMBER.... TELL OPERATOR HE MADE MISTAKE (STUFF
                                ; ERROR MSG)...
```

```
                                ;-
                                        IF.ERROR THEN
(6)  000166' 103006                                                     BCC     50010$
                                                LET DT.KBRSP(R0) := #CM.ARG
(4)  000170' 012760  000000G 000022                                    MOV     #CM.ARG,DT.KBRSP
                                                CALL RESREG
(3)  000176' 004767  000000G                                           JSR     PC,RESREG
                                                RETURN
(4)  000202' 000462                                                    BR      50000$
                                        ENDIF
(4)  000204'                                                    50010$:
                                ENDIF
(4)  000204'                                                    50007$:

                        ENDIF
(4)  000204'                                                    50005$:


                                ;+
                                ; CHECK OUT THE NUMBER
                                ;-

                                CALL NUMCHK IN <R0,R1> OUT <R1,R3>
(4)  000204' 162705  000004                                            SUB     #2*2,R5
(3)  000210' 010546                                                    MOV     R5,-(SP)
(5)  000212' 010145                                                    MOV     R1,-(R5)
(4)  000214' 010045                                                    MOV     R0,-(R5)
(3)  000216' 004767  000000G                                           JSR     PC,NUMCHK
(3)  000222' 012605                                                    MOV     (SP)+,R5
(4)  000224' 012501                                                    MOV     (R5)+,R1
(4)  000226' 012503                                                    MOV     (R5)+,R3

                                IF.NO.ERROR THEN
(6)  000230' 103445                                                    BCS     50011$


                                ;+
                                ; IF VALID VIRTUAL ADDRESS THEN
                                ; SHIFT NUMBER ONE PLACE TO RIGHT. IF THE C BIT SETS IT IS AN ODD
                                ; NUMBER... NOT ALLOWED !!!. STUFF ERROR MSG AND RETURN....
                                ; IF THE NUMBER IS OK,RESTORE IT AND CONTINUE.
                                ;-

                                   LET R3 := R3 ROTATE -1
(7)  000232' 006003                                                    ROR     R3
                                   IFCOND CS THEN
(6)  000234' 103004                                                    BCC     50012$
                                     LET DT.KBRSP(R0) := #CM.ODD
(4)  000236' 012760  000000G 000022                                    MOV     #CM.ODD,DT.KBRSP
                                   ELSE
(4)  000244' 000437                                                    BR      50013$
(3)  000246'                                                    50012$:
```

```
                                                LET R3 := R3 ROTATE 1
(7)  000246' 006103                                                     ROL     R3
                                ;+
                                ; THE NUMBER IS OK...CHECK FOR ANY JUNK ARGUMENTS..
                                ;-
                                                    CALL ARGCHK IN <R1> OUT <R1>
(4)  000250' 162705  000002                                             SUB     #1*2,R5
(3)  000254' 010546                                                     MOV     R5,-(SP)
(4)  000256' 010145                                                     MOV     R1,-(R5)
(3)  000260' 004767  000000G                                            JSR     PC,ARGCHK
(3)  000264' 012605                                                     MOV     (SP)+,R5
(4)  000266' 012501                                                     MOV     (R5)+,R1
                                ;+
                                ; IF IT'S NOT A CR... STUFF ERROR MSG AND LEAVE
                                ;-
                                                    IF.NO.ERROR THEN
(6)  000270' 103404                                                     BCS     50014$
                                ;-
                                                        LET DT.KBRSP(R0) := #CM.ARG
(4)  000272' 012760  000000G 000022                                     MOV     #CM.ARG,DT.KBRSP
                                ;+
                                ; ELSE THERE ARE NO JUNK ARGUMENTS... NOW SEE IF THE NUMBER IS IN
                                ; THE EXERCISER RANGE... AND NOT > THAN 32 K
                                ;-
                                                    ELSE
(4)  000300' 000421                                                     BR      50015$
(3)  000302'                                                        50014$:
                                ;+
                                ; FORM THE SUM (MODULE HEADER ADDRESS + NUMBER
                                ;-
                                                        LET R3 := R3 + R2
(6)  000302' 060203                                                     ADD     R2,R3
                                                        IFCOND CS THEN
(6)  000304' 103004                                                     BCC     50016$
                                                            LET DT.KBRSP(R0) := #CM.ADR
(4)  000306' 012760  000000G 000022                                     MOV     #CM.ADR,DT.KBRSP
                                                        ELSE
(4)  000314' 000413                                                     BR      50017$
(3)  000316'                                                        50016$:
                                ;+
```

```
                                ; IF THE NUMBER IS TOO BIG... TELL OPERATOR AND LEAVE...
                                ;-
000316'                                                 IF R3 HI DT.ESIZ(R0) THEN
(6)  000316' 020360  000044                                                             CMP     R3,DT.ESIZ(R0)
(9)  000322' 101404                                                                     BLOS    50020$
                                                        LET DT.KBRSP(R0) := #CM.ADR
(4)  000324' 012760  000000G 000022                                                     MOV     #CM.ADR,DT.KBRSP

                                ;+
                                ; ELSE THE NUMBER IS OK.... GO OUTPUT IT...
                                ;-
000332'                                                 ELSE
(4)  000332' 000404                                                                     BR      50021$
(3)  000334'                                                                    50020$:
                                                        LET KE.LOC := R3
(4)  000334' 010367  177440                                                             MOV     R3,KE.LOC
                                                        CALL CONV
(3)  000340' 004767  000006                                                             JSR     PC,CONV
000344'                                                 ENDIF
(4)  000344'                                                                    50021$:
                                                    ENDIF
(4)  000344'                                                                    50017$:
                                                ENDIF
(4)  000344'                                                                    50015$:
                                            ENDIF
(4)  000344'                                                                    50013$:
                                        ENDIF
(4)  000344'                                                                    50011$:
                                ENDIF
(4)  000344'                                                                    50003$:

                                ;+
                                ;CLEAN UP AND LEAVE
                                ;-

                                CALL RESREG
(3)  000344' 004767  000000G                                                           JSR     PC,RESREG

000350'                         ENDRTN
(3)  000350'                                                                    50000$:
(3)  000350'                                                                    50001$:
(2)  000350' 000207                                                                     RTS     PC

000352'                         ROUTINE CONV
(2)  000352'                                                                    CONV:

                                ;+
                                ; THIS ROUTINE CONVERTS THE LOCATION TO BE EXAMINED AND ITS CONTENTS
                                ; TO ASCII
                                ;-
```

```
797
798  000352'                        CALL BOA16 IN <KE.LOC,#KE.MG2>
(3)  000352' 010546                                                    MOV     R5,-(SP)
(5)  000354' 012745  000002'                                           MOV     #KE.MG2,-(R5)
(4)  000360' 016745  177414                                            MOV     KE.LOC,-(R5)
(3)  000364' 004767  000000G                                           JSR     PC,BOA16
(3)  000370' 012605                                                    MOV     (SP)+,R5
799  000372'                        CALL BOA16 IN <@KE.LOC,#KE.MG3>
(3)  000372' 010546                                                    MOV     R5,-(SP)
(5)  000374' 012745  000012'                                           MOV     #KE.MG3,-(R5)
(4)  000400' 017745  177374                                            MOV     @KE.LOC,-(R5)
(3)  000404' 004767  000000G                                           JSR     PC,BOA16
(3)  000410' 012605                                                    MOV     (SP)+,R5
800  000412'                        LET DT.KBRSP(R0) := #KE.MG1
(4)  000412' 012760  000002' 000022                                    MOV     #KE.MG1,DT.KBRSP
801  000420'                        ENDRTN
(3)  000420'                                                   50000$:
(3)  000420'                                                   50001$:
(2)  000420' 000207                                                    RTS     PC
802          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16 = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CMDBUF= 000002
CM.ADR= ****** G
CM.ARG= ****** G
CM.BAD= ****** G
CM.ODD= ****** G
CONFIG= 000056
CONV    000352R
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KEXAM   000022RG
KE.LOC  000000RG
KE.MG1  000002R
KE.MG2  000002R
KE.MG3  000012R
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK= ****** G
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
NUMCHK= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
```

KEXAM.MAC 19-SEP-78 16:11 SYMBOL TABLE

```
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$ISK3 = 000001
$ISK4 = 000001
$ISK5 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000110
$NSK2 = 000110
$NSK3 = 000110
$NSK4 = 000110
$NSK5 = 000110
$NSK6 = 000110
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050002
$TEMP = 000300
$TSK0 = 050003
$TSK1 = 050011
$TSK2 = 050013
$TSK3 = 050015
$TSK4 = 050017
$TSK5 = 050021
$$ARGC= 000000
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000322R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000002
$$STAG= 050000
.     = 000422R

. ABS.  000000     000
        000422     001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:KEXAM,DSKZ:KEXAM=SPMAC/ML,EQUATE,KEXAM
RUN-TIME: 15 5 .4 SECONDS
RUN-TIME RATIO: 38/21=1.7
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KFILL   PROCESS THE 'FILL' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KFILL
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'FILL' KEYBOARD COMMAND.
                        ;       IF NO ARGUMENT IS FOUND IN THE DECODE BUFFER, THE CONTENTS
                        ;       OF THE FILL COUNT/FILL CHAR ARE OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.KBRSP
                        ;       2. DT.FCNT
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK - VERIFY AN ARGUMENT
                        ;       2. BOA16 - BINARY TO OCTAL ASCII CONVERSION
                        ;       3. SAVREG
                        ;       4. RESREG
                        ;       5. NUMCHK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KFILL IN <DTADR,BUFPTR>
                        ;       WHERE:  DTADR = ADDRESS OF DTABLE
                        ;               BUFPTR = CMD DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KFILL

                                .MCALL  STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ; REFERENCES BY OTHER MODULES:
                                ;
                                .GLOBL  KFILL           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  NUMCHK          ;CHECK AN OCTAL ASCII NUMBER
                                .GLOBL  CM.ARG          ;"INVALID OR MISSING ARGUMENT" MESSAGE
                                ;
                                ;****************
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 043045  046111  027514  KF.MG1: .ASCII  '%FILL/ '        ;FILL MESSAGE
000006'    040
000007' 000006                  KF.MG2: .BLKB   6               ;HOLD ASCII DT.FLCNT CONTENTS
000015'    045     000                  .ASCIZ  /%/
        000020'                 .EVEN
                                ;****************
```

```
                                .SBTTL  KFILL ROUTINE


                                ROUTINE KFILL <DTADR,BUFPTR>                                    KFILL:

                                ;+
                                ; SAVE REGISTERS AND SAVE DATA TABLE ADDRESS, AND DECODE BUFFER POINTER
                                ;-

                                CALL SAVREG                                                     JSR     PC,SAVREG

                                LET R0 := DTADR(R5)                                             MOV     DTADR(R5),R0

                                LET R1 := BUFPTR(R5)                                            MOV     BUFPTR(R5),R1


                                ;+
                                ; R3 GETS ADDRESS OF FILL COUNT
                                ;-

                                LET R3 :=  #DT.FCNT + R0                                        MOV     #DT.FCNT,R3
                                                                                                ADD     R0,R3
                                ;+
                                ; GET TO FIRST ARGUMENT
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>                                    SUB     #1*2,R5
                                                                                                MOV     R5,-(SP)
                                                                                                MOV     R1,-(R5)
                                                                                                JSR     PC,ARGCHK
                                                                                                MOV     (SP)+,R5
                                                                                                MOV     (R5)+,R1
                                ;+
                                ; IF IT'S NOT A CARRIAGE RETURN, VERIFY THE NUMBER
                                ;-

                                IF.NO.ERROR THEN                                                BCS     50002$


                                        CALL NUMCHK IN  <R0,R1> OUT <R1,R2>                     SUB     #2*2,R5
                                                                                                MOV     R5,-(SP)
                                                                                                MOV     R1,-(R5)
                                                                                                MOV     R0,-(R5)
                                                                                                JSR     PC,NUMCHK
                                                                                                MOV     (SP)+,R5
                                                                                                MOV     (R5)+,R1
                                                                                                MOV     (R5)+,R2

                                ;+
                                ; IF THERE IS AN ERROR IN THE NUMBER, LEAVE
```

```
                                ;-
000110'                                 IF.ERROR THEN
(6)  000110' 103001                                                     BCC     50003$

                                                INLINE <BR      1$>
000112'                                                                 BR      1$
(2)  000112' 000463

000114'                                 ENDIF
(4)  000114'                                                    50003$:

                                ;+
                                ; GET NEXT ARGUMENT
                                ;-

000114'                                 CALL ARGCHK IN <R1> OUT <R1>
(4)  000114' 162705  000002                                             SUB     #1*2,R5
(3)  000120' 010546                                                     MOV     R5,-(SP)
(4)  000122' 010145                                                     MOV     R1,-(R5)
(3)  000124' 004767  000000G                                            JSR     PC,ARGCHK
(3)  000130' 012605                                                     MOV     (SP)+,R5
(4)  000132' 012501                                                     MOV     (R5)+,R1

                                ;+
                                ; IF AN ERROR... HE DIDN'T GIVE ENOUGH ARGUMENTS
                                ;-

000134'                                 IF.ERROR THEN
(6)  000134' 103004                                                     BCC     50004$

000136'                                         LET DT.KBRSP(R0) := #CM.ARG
(4)  000136' 012760  000000G 000022                                     MOV     #CM.ARG,DT.KBRSP
000144'                                         INLINE <BR      1$>
(2)  000144' 000446                                                     BR      1$

000146'                                 ENDIF
(4)  000146'                                                    50004$:

000146'                                 CALL NUMCHK IN <R0,R1> OUT <R1,R4>
(4)  000146' 162705  000004                                             SUB     #2*2,R5
(3)  000152' 010546                                                     MOV     R5,-(SP)
(5)  000154' 010145                                                     MOV     R1,-(R5)
(4)  000156' 010045                                                     MOV     R0,-(R5)
(3)  000160' 004767  000000G                                            JSR     PC,NUMCHK
(3)  000164' 012605                                                     MOV     (SP)+,R5
(4)  000166' 012501                                                     MOV     (R5)+,R1
(4)  000170' 012504                                                     MOV     (R5)+,R4

                                ;+
                                ; WE GOT THE SECOND PART OF THE ARGUMENT,IF INVALID, LEAVE
                                ;-

000172'                                 IF.ERROR THEN
(6)  000172' 103001                                                     BCC     50005$
```

```
                                        INLINE <BR      1$>                                  BR      1$
(2)  000174' 000432
                                    ENDIF                                          50005$:
(4)  000176'
                                ;+
                                ; NUMBER OK, VERIFY NO JUNK ARGUMENTS EXIST
                                ;-
                                    CALL ARGCHK IN <R1> OUT <R1>                           SUB     #1*2,R5
(4)  000176' 162705  000002                                                                MOV     R5,-(SP)
(3)  000202' 010546                                                                        MOV     R1,-(R5)
(4)  000204' 010145                                                                        JSR     PC,ARGCHK
(3)  000206' 004767  000000G                                                               MOV     (SP)+,R5
(3)  000212' 012605                                                                        MOV     (R5)+,R1
(4)  000214' 012501
                                ;+
                                ; IF THERE ARE MORE JUNK ARGUMENTS - LEAVE, BUT STUFF ERROR MSG FIRST
                                ;-
                                    IF.NO.ERROR THEN                                       BCS     50006$
(6)  000216' 103404
                                        LET DT.KBRSP(R0) := #CM.ARG                        MOV     #CM.ARG,DT.KBRSP
(4)  000220' 012760  000000G 000022
                                        INLINE <BR      1$>                                BR      1$
(2)  000226' 000415
                                    ENDIF                                          50006$:
(4)  000230'
                                ;+
                                ; NO JUNK ARGUMENTS - SO REPLACE FILL COUNT/CHAR WITH
                                ; NEW VALUES
                                ;-
                                    LET (R3)+ :B= R4                                       MOVB    R4,(R3)+
(4)  000230' 110423
                                    LET (R3) :B= R2                                        MOVB    R2,(R3)
(4)  000232' 110213
                                ENDIF                                              50002$:
(4)  000234'
                                ;+
                                ; CONVERT THE NUMBER IN DT.FCNT TO ASCII, STUFF MESSAGE
                                ; AND RETURN
                                ;-
                                CALL BOA16 IN <DT.FCNT(R0),#KF.MG2>                        MOV     R5,-(SP)
(3)  000234' 010546                                                                        MOV     #KF.MG2,-(R5)
(5)  000236' 012745  000007'                                                               MOV     DT.FCNT(R0),-(R5
(4)  000242' 016045  000036
```

```
(3)  000246' 004767  000000G                                                    JSR     PC,BOA16
(3)  000252' 012605                                                             MOV     (SP)+,R5
701
702  000254'                        LET DT.KBRSP(R0) := #KF.MG1
(4)  000254' 012760  000000' 000022                                            MOV     #KF.MG1,DT.KBRSP
703  000262'                        INLINE <1$:>
(2)  000262'                                                                    1$:
704  000262'                        CALL RESREG
(3)  000262' 004767  000000G                                                    JSR     PC,RESREG
705
706  000266'                        ENDRTN
(3)  000266'                                                                50000$:
(3)  000266'                                                                50001$:
(2)  000266' 000207                                                             RTS     PC
707          000001                .END
```

KFILL.MAC 28-JUL-78 09:18 SYMBOL TABLE

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16 = ****** G
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KFILL   000020RG
KF.MG1  000000R
KF.MG2  000007R
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
NUMCHK= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
```

KFILL.MAC 28-JUL-78 09:18 SYMBOL TABLE

```
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000110
$NSK2 = 000110
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050007
$TEMP = 000300
$TSK0 = 050002
$TSK1 = 050006
$$ARGC= 000004
$$BYTE= 000000
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000216R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000270R

. ABS.  000000     000
        000270     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KFILL,DSKZ:KFILL=SPMAC/ML,EQUATE,KFILL
 RUN-TIME: 13 3 .3 SECONDS
 RUN-TIME RATIO: 31/16=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KMAP   PROCESS THE 'MAP' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KMAP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'MAP' KEYBOARD COMMAND. IT WILL
                        ;       SEARCH THE MODULE LIST AND ENQUEUE THE MAP MESSAGE FOR
                        ;       EACH MODULE. IF A MODULE NAME IS SPECIFIED, A MAP
                        ;       MESSAGE IS OUTPUTTED FOR THAT MODULE ONLY.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG,CM.BADNAME
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK - CHECK FOR JUNK ARGUMENTS
                        ;       2. ENQTQ - ENQUEUE A MESSAGE
                        ;       3. NAMCHK - VERIFY A MODULE NAME
                        ;       4. SAVREG - SAVE REGISTERS
                        ;       5. RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KMAP IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KMAP

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

          000001                $LSTTAG=1
          000001                $LSTIN=1


                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KMAP            ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  SAVREG          ;SAVE REGS 0-4
                                .GLOBL  RESREG          ;RESTORE REGS 0-4
                                .GLOBL  ENQTQ           ;ENQUEUE A MESSAGE
                                .GLOBL  CM.ARG          ;'ILLEGAL ARGUMENT' ERROR MSG
                                .GLOBL  NAMCHK          ;CHECK A MODULE NAME
                                .GLOBL  CM.BADNAME      ;'INVALID MODULE NAME' ERROR MESSAGE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  KM.HLD: .WORD   0       ;HOLD ADDRESS OF MESSAGE
000002' 041445  042115  000076  KM.CMD: .ASCIZ  /%CMD>/
000010' 041045  054523  000076  KM.RUN: .ASCIZ  /%BSY>/
                                .EVEN
```

```
                                .SBTTL  KMAP ROUTINE

000016'                         ROUTINE KMAP <DTADR,BUFPTR>
000016'                                                                 KMAP:

                                ;+
                                ; INITIALIZE AND SAVE DTABLE ADDRESS AND DECODE BUFFER PTR
                                ;-

000016'                         CALL SAVREG
000016' 004767  000000G                                                 JSR     PC,SAVREG
000022'                         LET R0 := DTADR(R5)
000022' 016500  000000                                                  MOV     DTADR(R5),R0
000026'                         LET R1 := BUFPTR(R5)
000026' 016501  000002                                                  MOV     BUFPTR(R5),R1



                                ;+
                                ; IF IN RUNMODE , STUFF FAKE COMMAND PROMPT SINCE
                                ; THIS COMMAND USES ENQTQ, IT WILL O/P THE NORMAL
                                ; PROMPT BEFORE ANY OF THE MAP MESSAGES WILL BE
                                ; PRINTED - THEN THIS DUMMY PROMPT WILL BE
                                ; OUTPUTTED......
                                ;-
000032'                         IF #RUNMODE SETIN DT.ST0(R0) THEN
000032' 032760  100000  000010                                          BIT     #RUNMODE,DT.ST0(
000040' 001404                                                          BEQ     50002$
000042'                                 LET KM.HLD := #KM.RUN
000042' 012767  000010' 177730                                          MOV     #KM.RUN,KM.HLD
000050'                         ELSE
000050' 000403                                                          BR      50003$
000052'                                                         50002$:
000052'                                 LET KM.HLD := #KM.CMD
000052' 012767  000002' 177720                                          MOV     #KM.CMD,KM.HLD
000060'                         ENDIF
000060'                                                         50003$:

                                ;+
                                ; SEE IF ANY ARGUMENTS ARE INCLUDED IN DECODE BUFFER
                                ;-

000060'                         CALL ARGCHK IN <R1> OUT <R1>
000060' 162705  000002                                                  SUB     #1*2,R5
000064' 010546                                                          MOV     R5,-(SP)
000066' 010145                                                          MOV     R1,-(R5)
000070' 004767  000000G                                                 JSR     PC,ARGCHK
000074' 012605                                                          MOV     (SP)+,R5
000076' 012501                                                          MOV     (R5)+,R1

                                ;+
                                ; IF THERE ARE NO ARGUMENTS DO A FULL MAP (ALL MODULES). GET ADDRESS
                                ; OF MODULE LIST. THEN, WHILE
                                ; THE ENTRY IN THE MODULE LIST IS NOT 0, ENQUEUE THE MAP MESSAGE
                                ; FOR THE MODULES.
```

```
                                ;-

                                IF.ERROR THEN
000100'                                                         BCC     50004$
(6)  000100' 103023
                                        LET R2 := DT.MLST(R0)
000102'                                                         MOV     DT.MLST(R0),R2
(4)  000102' 016002  000032             WHILE (R2) NE #ENDLST DO
000106'                                                 50005$:
(4)  000106'                                                    CMP     (R2),#ENDLST
(6)  000106' 021227  000000                                     BEQ     50006$
(9)  000112' 001415                             CALL ENQTQ IN <R0,#MSGMAP,#0,(R2)+,#0>
000114'                                                         MOV     R5,-(SP)
(3)  000114' 010546                                             MOV     #0,-(R5)
(8)  000116' 012745  000000                                     MOV     (R2)+,-(R5)
(7)  000122' 012245                                             MOV     #0,-(R5)
(6)  000124' 012745  000000                                     MOV     #MSGMAP,-(R5)
(5)  000130' 012745  000021                                     MOV     R0,-(R5)
(4)  000134' 010045                                             JSR     PC,ENQTQ
(3)  000136' 004767  000000G                                    MOV     (SP)+,R5
(3)  000142' 012605
                                        ENDDO
000144'                                                         BR      50005$
(4)  000144' 000760                                     50006$:
(3)  000146'
                                ELSE
000146'                                                         BR      50007$
(4)  000146' 000450                                     50004$:
(3)  000150'

                                ;+
                                ; THERE IS AN ARGUMENT - GO CHECK IT OUT
                                ;-

                                        CALL NAMCHK IN <R0,R1> OUT <R2,R1>
000150'                                                         SUB     #2*2,R5
(4)  000150' 162705  000004                                     MOV     R5,-(SP)
(3)  000154' 010546                                             MOV     R1,-(R5)
(5)  000156' 010145                                             MOV     R0,-(R5)
(4)  000160' 010045                                             JSR     PC,NAMCHK
(3)  000162' 004767  000000G                                    MOV     (SP)+,R5
(3)  000166' 012605                                             MOV     (R5)+,R2
(4)  000170' 012502                                             MOV     (R5)+,R1
(4)  000172' 012501
                                ;+
                                ; IF MODULE NAME IS O.K. - JUST VERIFY THERE IS NO JUNK ARGUMENT
                                ;-
                                        IF.NO.ERROR THEN
000174'                                                         BCS     50010$
(6)  000174' 103432
                                                CALL ARGCHK IN <R1> OUT <R1>
000176'                                                         SUB     #1*2,R5
(4)  000176' 162705  000002                                     MOV     R5,-(SP)
(3)  000202' 010546                                             MOV     R1,-(R5)
(4)  000204' 010145                                             JSR     PC.ARGCHK
(3)  000206' 004767  000000G                                    MOV     (SP)+,R5
(3)  000212' 012605                                             MOV     (R5)+,R1
(4)  000214' 012501
```

```
                                ;+
                                ; IF NO JUNK ARGUMENTS.... ENQUEUE THE MESSAGE FOR THIS MODULE
                                ;-
000216'                                 IF.ERROR THEN
(6) 000216' 103015                                                              BCC     50011$

                                                CALL ENQTQ IN <R0,#MSGMAP,#0,R2,#0>
(3) 000220' 010546                                                              MOV     R5,-(SP)
(8) 000222' 012745  000000                                                      MOV     #0,-(R5)
(7) 000226' 010245                                                              MOV     R2,-(R5)
(6) 000230' 012745  000000                                                      MOV     #0,-(R5)
(5) 000234' 012745  000021                                                      MOV     #MSGMAP,-(R5)
(4) 000240' 010045                                                              MOV     R0,-(R5)
(3) 000242' 004767  000000G                                                     JSR     PC,ENQTQ
(3) 000246' 012605                                                              MOV     (SP)+,R5

000250'                                 ELSE
(4) 000250' 000403                                                              BR      50012$
(3) 000252'                                                             50011$:

                                ;+
                                ; THERE IS A JUNK ARGUMENT IN DECODE BUFFER... STUFF ERROR MSG...
                                ;-
000252'                                         LET DT.KBRSP(R0) := #CM.ARG
(4) 000252' 012760  000000G 000022                                              MOV     #CM.ARG,DT.KBRSP

000260'                                 ENDIF
(4) 000260'                                                             50012$:

000260'                                 ELSE
(4) 000260' 000403                                                              BR      50013$
(3) 000262'                                                             50010$:

                                ;+
                                ; BAD MODULE NAME....STUFF ERROR MSG....
                                ;-

000262'                                 LET DT.KBRSP(R0) := #CM.BADNAME
(4) 000262' 012760  000000G 000022                                              MOV     #CM.BADNAME,DT.K

000270'                                 ENDIF
(4) 000270'                                                             50013$:

000270'                         ENDIF
(4) 000270'                                                             50007$:


                                ;+
                                ; IF NO ERRORS, ALSO ENQUEUE A PROMPT
                                ;-

000270'                         IF DT.KBRSP(R0) EQ #0 THEN
(6) 000270' 005760  000022                                                      TST     DT.KBRSP(R0)
(9) 000274' 001015                                                              BNE     50014$
```

```
  697  000276'                              CALL ENQTQ IN <R0,#MSGSTD,KM.HLD,#0,#0>
  (3)  000276' 010546                                                        MOV     R5,-(SP)
  (8)  000300' 012745  000000                                                MOV     #0,-(R5)
  (7)  000304' 012745  000000                                                MOV     #0,-(R5)
  (6)  000310' 016745  177464                                                MOV     KM.HLD,-(R5)
  (5)  000314' 012745  000000                                                MOV     #MSGSTD,-(R5)
  (4)  000320' 010045                                                        MOV     R0,-(R5)
  (3)  000322' 004767  000000G                                               JSR     PC,ENQTQ
  (3)  000326' 012605                                                        MOV     (SP)+,R5
  698  000330'                              ENDIF
  (4)  000330'                                                        50014$:
  699
  700                                       ;+
  701                                       ; CLEAN UP AND GOODBYE
  702                                       ;-
  703
  704  000330'                              CALL RESREG
  (3)  000330' 004767  000000G                                               JSR     PC,RESREG
  705
  706  000334'                              ENDRTN
  (3)  000334'                                                        50000$:
  (3)  000334'                                                        50001$:
  (2)  000334' 000207                                                        RTS     PC
  707
  708          000001                       .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CM.BAD= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KDE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KMAP    000016RG
KM.CMD  000002R
KM.HLD  000000R
KM.RUN  000010R
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK= ****** G
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
```

```
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$CASE= 000000
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$DST = 000000
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $ISK2 = 000001          $$ELOC= 000402
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $LOCTA= 177777          $$ERFL= 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000001
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $NESTL= 177777          $$LOC = 000274R
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK1 = 000110          $$REG = 177777
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $NSK2 = 000110          $$RETU= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $NSK3 = 000110          $$RTN1= 050000
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $SAVLE= 177777          $$RTN2= 050001
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $SSK0 = 050006          $$SRC = 000000
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $TAGLE= 177777          $$TGSV= 000000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TAGNU= 050015          $$TGS1= 000000
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $TEMP = 000300          $$TGS2= 000000
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $TSK0 = 050014          $$TO  = 000000
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $TSK1 = 050013          $$$TAG= 050000
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $TSK2 = 050012          .     = 000336R
UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$ARGC= 000004
UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$BYTE= 000403

. ABS.  000000     000
        000336     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KMAP,DSKZ:KMAP=SPMAC/ML,EQUATE,KMAP
 RUN-TIME: 13 4 .4 SECONDS
 RUN-TIME RATIO: 37/18=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KMOD   PROCESS THE 'MOD' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KMOD
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'MOD' KEYBOARD COMMAND. IT
                        ;       CAN BE USED WITH OR WITHOUT AN ARGUMENT. IF NO
                        ;       ARGUMENT, THE LAST PREVIOUSLY MODIFIED LOCATION
                        ;       WILL BE OPENED. THERE CAN BE ONE OR TWO ARGUMENTS: A MODULE
                        ;       NAME AND A NUMBER. IF A MODULE NAME, A NUMBER MUST FOLLOW.
                        ;       THE MODULE NAME WILL BE VERIFIED. IF NO MODULE BY THAT NAME
                        ;       AN ERROR MESSAGE IS ISSUED. THE NUMBER WILL BE VERIFIED. IF
                        ;       NOT OCTAL OR NOT EVEN, AN ERROR MESSAGE IS ISSUED. THE
                        ;       OPERATOR MAY ENTER NEW CONTENTS FOR AN OPENED LOCATION.
                        ;       IT MAY BE TERMINATED BY CTRL U, LF OR CR.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.KBRSP - KEYBOARD RESPONSE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG,CM.BADNAME,CM.ODD,CM.RUN,CM.ADR,DX.KFL
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ENQTQ - ENQUEUE A MESSAGE
                        ;       2. NUMCHK - CHECK A NUMBER
                        ;       3. NAMCHK - CHECK A NAME
                        ;       4. ARGCHK - CHECK AN ARGUMENT
                        ;       5. HRDADR - CHECK AN ADDRESS FOR EXISTENCE
                        ;       6. CMDCPY - COPY KEYBOARD BUFFER TO DECODE BUFFER
                        ;       7. MSGDEQ - DEQUEUE A MESSAGE
                        ;       8. SAvREG
                        ;       9. RESREG
                        ;       10. HRDADRCHK
                        ;       11. BOA16
                        ;       12. KBINI
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KMOD IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;       AND   BUFPTR = DECODE BUFFER POINTER
```

```
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KMOD

                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1

                 000001 $LSTTAG=1
                 000001 $LSTIN=1

                        ;***************
                        ;
                        ; REFERENCED BY OTHER MODULES:
                        ;
                        .GLOBL  KMOD            ;MODULE ENTRY POINT
                        ;
                        ;***************
                        ;
                        ; GLOBAL REFERENCES:
                        ;
                        .GLOBL  CM.ARG          ;'ILLEGAL ARGUMENT'
                        .GLOBL  MSGDEQ          ;MESSAGE DEQUEUER
                        .GLOBL  CMDCPY          ;COPY KEYBOARD BUF TO DECODE BUF
                        .GLOBL  CM.ADR          ;'ILLEGAL ADDRESS'
                        .GLOBL  CM.RUN          ;'ILLEGAL COMMAND IN RUN MODE
                        .GLOBL  NAMCHK          ;CHECK AN OPTION MODULE NAME
                        .GLOBL  CM.BADNAME      ;'INVALID MODULE MODULE NAME'
                        .GLOBL  ENQTQ           ;ENQUEUE A MESSAGE
                        .GLOBL  NUMCHK          ;CHECK A NUMBER
                        .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                        .GLOBL  HRDADRCHK       ;CHECK AN ADDRESS ROUTINE
                        .GLOBL  CM.ODD          ;'NUMBER MUST BE EVEN
                        .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONVERSION
                        .GLOBL  SAVREG          ;SAVE REGISTERS
                        .GLOBL  KBINI           ;KEYBOARD INIT ROUTINE
                        .GLOBL  RESREG          ;RESTORE REGISTERS
                        .GLOBL  DX.KFL          ;CARRIAGE RETURN FLAG
                        ;
                        ;***************
                        ;
                        ; LOCAL STORAGE:
                        ;
000000' 000000          KM.LOC: .WORD   0       ;HOLD LOCATION TO BE MODIFIED
000002' 000006          KM.MG1: .BLKB   6       ;HOLD ASCII LOCATION
000010' 020057                  .ASCII  '/'     ;SEPARATOR
000012' 000006          KM.MG2: .BLKB   6       ;HOLD ASCII CONTENTS OF LOCATION
000020' 020040                  .ASCII  /  /    ;SPACES
000022'    000                  .BYTE   0       ;TERMINATOR (NO CR DESIRED)
000023'    000          KM.ABT: .BYTE   0       ;ABORT FLAG
000024'    000          KM.FLG: .BYTE   0       ;KBINT FLAG
                        ;
                 000026' .EVEN
                        ;***************
```

```
                        KMOD ROUTINE

                        .SBTTL  KMOD ROUTINE

                        ROUTINE KMOD <DTADR,BUFPTR>                                KMOD:

                        ;+
                        ;SAVE REGISTERS AND SAVE DTABLE ADDRESS + BUFFER POINTER
                        ;-

                        CALL SAVREG                                                JSR     PC,SAVREG

                        LET R0 := DTADR(R5)                                        MOV     DTADR(R5),R0

                        LET R1 := BUFPTR(R5)                                       MOV     BUFPTR(R5),R1

                        ;+
                        ; TYPE OUT THE CR AT END OF MOD COMMAND
                        ;-

                        CALL MSGDEQ IN <R0>                                        MOV     R5,-(SP)
                                                                                   MOV     R0,-(R5)
                                                                                   JSR     PC,MSGDEQ
                                                                                   MOV     (SP)+,R5

                        ;+
                        ; INIT ABORT FLAG
                        ;-

                        LET KM.ABT :B= #0                                          CLRB    KM.ABT


                        ;+
                        ; IF IN RUNMODE, LOAD MESSAGE AND LEAVE
                        ;-

                        IF #RUNMODE SETIN DT.ST0(R0) THEN                          BIT     #RUNMODE,DT.ST0(
                                                                                   BEQ     50002$

                                LET DT.KBRSP(R0) := #CM.RUN                        MOV     #CM.RUN,DT.KBRSP

                                INLINE <JMP     1$>                                JMP     1$

                        ENDIF                                                    50002$:

                        ;+
                        ; INIT R2 TO 0. WILL BE MODULE HEADER ADDRESS LATER IF SPECIFIED IN
                        ; ARGUMENT
                        ;-
                        LET R2 := #0                                               CLR     R2
```

```
                                ;+
                                ; GET THE FIRST ARGUMENT
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>

000104'                                                          SUB     #1*2,R5
(4)  000104' 162705  000002                                      MOV     R5,-(SP)
(3)  000110' 010546                                              MOV     R1,-(R5)
(4)  000112' 010145                                              JSR     PC,ARGCHK
(3)  000114' 004767  000000G                                     MOV     (SP)+,R5
(3)  000120' 012605                                              MOV     (R5)+,R1
(4)  000122' 012501

                                ;+
                                ; IF IT'S NOT A CR, SEE IF ALPHABETIC CHAR
                                ;-
                                IF.NO.ERROR THEN                                 BCS     50003$

000124'
(6)  000124' 103532

                                        IFB (R1) LT #'0 ORB (R1) GT #'9 THEN

000126'                                                          CMPB    (R1),#'0
(6)  000126' 121127  000060                                      BLT     50004$
(8)  000132' 002403                                              CMPB    (R1),#'9
(6)  000134' 121127  000071                                      BLE     50005$
(9)  000140' 003436                                      50004$:
(6)  000142'

                                ;+
                                ; IT'S ALPHABETIC - GO VERIFY MODULE NAME....
                                ;-

                                        CALL NAMCHK IN <R0,R1> OUT <R2,R1>

000142'                                                          SUB     #2*2,R5
(4)  000142' 162705  000004                                      MOV     R5,-(SP)
(3)  000146' 010546                                              MOV     R1,-(R5)
(5)  000150' 010145                                              MOV     R0,-(R5)
(4)  000152' 010045                                              JSR     PC,NAMCHK
(3)  000154' 004767  000000G                                     MOV     (SP)+,R5
(3)  000160' 012605                                              MOV     (R5)+,R2
(4)  000162' 012502                                              MOV     (R5)+,R1
(4)  000164' 012501

                                ;+
                                ; IF MODULE NAME BAD - STUFF ERROR MSG AND RETURN
                                ;-
                                        IF.ERROR THEN                            BCC     50006$

000166'
(6)  000166' 103005

                                                LET DT.KBRSP(R0) := #CM.BADNAME

000170'                                                          MOV     #CM.BADNAME,DT.K
(4)  000170' 012760  000000G 000022

                                                INLINE <JMP     1$>

000176'                                                          JMP     1$
(2)  000176' 000167  000714

                                        ENDIF                            50006$:

000202'
(4)  000202'
```

```
                                ;+
                                ; NAME O.K. ... CHECK NEXT ARGUMENT
                                ;-

000202'                                 CALL ARGCHK IN <R1> OUT <R1>
(4) 000202' 162705  000002                                              SUB     #1*2,R5
(3) 000206' 010546                                                      MOV     R5,-(SP)
(4) 000210' 010145                                                      MOV     R1,-(R5)
(3) 000212' 004767  000000G                                             JSR     PC,ARGCHK
(3) 000216' 012605                                                      MOV     (SP)+,R5
(4) 000220' 012501                                                      MOV     (R5)+,R1

                                ;+
                                ; IF THE ARGUMENT IS A CR.... WHEN A MODULE NAME IS SPECIFIED, A NUMBER
                                ; MUST FOLLOW...BUST COMMAND!!!... STUFF ERROR MSG
                                ;-
000222'                                 IF.ERROR THEN
(6) 000222' 103005                                                      BCC     50007$

000224'                                         LET DT.KBRSP(R0) := #CM.ARG
(4) 000224' 012760  000000G 000022                                      MOV     #CM.ARG,DT.KBRSP
000232'                                         INLINE <JMP     1$>
(2) 000232' 000167  000660                                              JMP     1$

000236'                                 ENDIF
(4) 000236'                                                     50007$:

000236'                         ENDIF
(4) 000236'                                                     50005$:

                                ;+
                                ; THERE WAS ANOTHER ARGUMENT... GO CHECK IT.. SHOULD BE A NUMBER...
                                ;-

000236'                         CALL NUMCHK IN <R0,R1> OUT <R1,R3>
(4) 000236' 162705  000004                                              SUB     #2*2,R5
(3) 000242' 010546                                                      MOV     R5,-(SP)
(5) 000244' 010145                                                      MOV     R1,-(R5)
(4) 000246' 010045                                                      MOV     R0,-(R5)
(3) 000250' 004767  000000G                                             JSR     PC,NUMCHK
(3) 000254' 012605                                                      MOV     (SP)+,R5
(4) 000256' 012501                                                      MOV     (R5)+,R1
(4) 000260' 012503                                                      MOV     (R5)+,R3

                                ;+
                                ; IF NUMBER WAS TOO LARGE... OR NON OCTAL... BUST COMMAND...
                                ;-
000262'                         IF.ERROR THEN
(6) 000262' 103002                                                      BCC     50010$

000264'                                 INLINE <JMP     1$>
(2) 000264' 000167  000626                                              JMP     1$
```

```
                                        ENDIF
                                50010$:

                                ;+
                                ; NOW MAKE SURE IT'S AN EVEN NUMBER
                                ;-

                                        LET R3 := R3 ROTATE -1
                                                                        ROR     R3

                                        IFCOND CS THEN
                                                                        BCC     50011$

                                ;+
                                ; UH OH... AN ODD ADDRESS... BUST COMMAND.... STUFF ERROR AND LEAVE..
                                ;-

                                                LET DT.KBRSP(R0) := #CM.ODD
                                                                        MOV     #CM.ODD,DT.KBRSP
                                                INLINE <JMP     1$>
                                                                        JMP     1$

                                        ENDIF
                                50011$:

                                ;+
                                ; RESTORE NUMBER
                                ;-

                                                LET R3 := R3 ROTATE 1
                                                                        ROL     R3


                                ;+
                                ; NOW... JUST CHECK FOR ANY JUNK ARGUMENTS....
                                ;-

                                        CALL ARGCHK IN <R1> OUT <R1>
                                                                        SUB     #1*2,R5
                                                                        MOV     R5,-(SP)
                                                                        MOV     R1,-(R5)
                                                                        JSR     PC,ARGCHK
                                                                        MOV     (SP)+,R5
                                                                        MOV     (R5)+,R1

                                ;+
                                ; IF JUNK ARGUMENTS FOUND ! SHOULD HAVE BEEN CR ! BUST COMMAND....
                                ; STUFF ERROR MSG & RETURN....
                                ;-
                                        IF.NO.ERROR THEN
                                                                        BCS     50012$


                                                LET DT.KBRSP(R0) := #CM.ARG
                                                                        MOV     #CM.ARG,DT.KBRSP
                                                INLINE <JMP     1$>
```

```
(2)  000340' 000167  000552                                                    JMP     1$
777
778  000344'                            ENDIF                                50012$:
(4)  000344'
779
780                                     ;+
781                                     ; FORM FINAL ADDRESS - ADD MODULE HEADER ADDRESS(IF ANY) TO NUMBER
782                                     ; IN R3 AND STORE IN R3
783                                     ;-
784
785  000344'                            LET R3 := R3 + R2                    ADD     R2,R3
(6)  000344' 060203
786
787
788                                     ;+
789                                     ; SEE IF ADDRESS IS > 32K.... NEVER ALLOWED....
790                                     ;-
791
792  000346'                            IFCOND CS THEN                       BCC     50013$
(6)  000346' 103005
793
794                                     ;+
795                                     ; NUMBER > 32K... SORRY... NOT ALLOWED....
796
797  000350'                                    LET DT.KBRSP(R0) := #CM.ADR   MOV     #CM.ADR,DT.KBRSP
(4)  000350' 012760  000000G 000022
798  000356'                                    INLINE <JMP     1$>           JMP     1$
(2)  000356' 000167  000534
799
800  000362'                            ENDIF                                50013$:
(4)  000362'
801
802
803                                     ;+
804                                     ; BUT DOES IT REALLY EXIST....
805                                     ;-
806
807  000362'                            CALL HRDADR IN <R3>                  MOV     R5,-(SP)
(3)  000362' 010546                                                         MOV     R3,-(R5)
(4)  000364' 010345                                                         JSR     PC,HRDADR
(3)  000366' 004767  000000G                                                MOV     (SP)+,R5
(3)  000372' 012605
808
809                                     ;+
810                                     ; IF NO SUCH MEMORY ADDRESS....STUFF ERROR.... AND SO LONG....
811                                     ;-
812  000374'                            IF.ERROR THEN                        BCC     50014$
(6)  000374' 103005
813
814
815  000376'                                    LET DT.KBRSP(R0) := #CM.ADR   MOV     #CM.ADR,DT.KBRSP
(4)  000376' 012760  000000G 000022
816  000404'                                    INLINE <JMP     1$>           JMP     1$
(2)  000404' 000167  000506
817
818  000410'                            ENDIF
```

```
(4)  000410'                                                                    50014$:
819
820
821
822  000410'                        ELSE                                                BR      50015$
(4)  000410' 000402
(3)  000412'                                                                    50003$:
823
824                                 ;+
825                                 ; IT'S A CR SO MOVE KM.LOC TO R3
826                                 ;-
827
828  000412'                                LET R3 := KM.LOC                            MOV     KM.LOC,R3
(4)  000412' 016703  177362
829
830  000416'                        ENDIF
(4)  000416'                                                                    50015$:
831
832                                 ;  ********* LOOP WHILE NOT LONE CR ON INPUT *************
833
834
835
836
837
838                                 ;+
839                                 ; CLEAR KM.ABT AND R4 (BOTH FLAGS)
840                                 ;-
841  000416'                        INLINE <5$:>                                        5$:
(2)  000416'
842  000416'                                LET KM.ABT :B= #0                           CLRB    KM.ABT
(4)  000416' 105067  177401
843  000422'                                LET R4 := #0                                CLR     R4
(4)  000422' 005004
844
845                                 ;+
846                                 ; SEE IF ADDRESS EXISTS
847                                 ;-
848
849  000424'                                CALL HRDADR IN <R3>                         MOV     R5,-(SP)
(3)  000424' 010546                                                                     MOV     R3,-(R5)
(4)  000426' 010345                                                                     JSR     PC,HRDADR
(3)  000430' 004767  000000G                                                           MOV     (SP)+,R5
(3)  000434' 012605
850
851                                 ;+
852                                 ; IF THIS NEW ADDRESS IS A BUMMER... STUFF ERROR MSG
853                                 ;-
854  000436'                                IF.ERROR THEN                               BCC     50016$
(6)  000436' 103006
855
856  000440'                                       LET DT.KBRSP(R0) := #CM.ADR           MOV     #CM.ADR,DT.KBRSP
(4)  000440' 012760  000000G 000022
857  000446'                                       INLINE <JMP     1$>                   JMP     1$
(2)  000446' 000167  000444
858
859  000452'                                ELSE
```

```
(4)  000452' 000402                                                          BR      50017$
(3)  000454'                                                          50016$:
860
861
862                                 ;+
863                                 ; SAVE R3 IN KM.LOC
864                                 ;-
865
866  000454'                                    LET KM.LOC := R3
(4)  000454' 010367  177320                                                  MOV     R3,KM.LOC
867
868  000460'                                    ENDIF
(4)  000460'                                                          50017$:
869
870                                 ;+
871                                 ; CONVERT LOCAL STORAGE TO ASCII AND OUTPUT
872                                 ;-
873
874  000460'                                    CALL BOA16 IN <KM.LOC,#KM.MG1>
(3)  000460' 010546                                                          MOV     R5,-(SP)
(5)  000462' 012745  000002'                                                 MOV     #KM.MG1,-(R5)
(4)  000466' 016745  177306                                                  MOV     KM.LOC,-(R5)
(3)  000472' 004767  000000G                                                 JSR     PC,BOA16
(3)  000476' 012605                                                          MOV     (SP)+,R5
875  000500'                                    CALL BOA16 IN <@KM.LOC,#KM.MG2>
(3)  000500' 010546                                                          MOV     R5,-(SP)
(5)  000502' 012745  000012'                                                 MOV     #KM.MG2,-(R5)
(4)  000506' 017745  177266                                                  MOV     @KM.LOC,-(R5)
(3)  000512' 004767  000000G                                                 JSR     PC,BOA16
(3)  000516' 012605                                                          MOV     (SP)+,R5
876
877
878                                 ;+
879                                 ; ENQUEUE IT
880                                 ;-
881  000520'                                    CALL ENQTQ IN <R0,#MSGSTD,#KM.MG1,#0,#0>
(3)  000520' 010546                                                          MOV     R5,-(SP)
(8)  000522' 012745  000000                                                  MOV     #0,-(R5)
(7)  000526' 012745  000000                                                  MOV     #0,-(R5)
(6)  000532' 012745  000002'                                                 MOV     #KM.MG1,-(R5)
(5)  000536' 012745  000000                                                  MOV     #MSGSTD,-(R5)
(4)  000542' 010045                                                          MOV     R0,-(R5)
(3)  000544' 004767  000000G                                                 JSR     PC,ENQTQ
(3)  000550' 012605                                                          MOV     (SP)+,R5
882                                 ;+
883                                 ; AND PRINT IT
884                                 ;-
885
886  000552'                                    CALL MSGDEQ IN <R0>
(3)  000552' 010546                                                          MOV     R5,-(SP)
(4)  000554' 010045                                                          MOV     R0,-(R5)
(3)  000556' 004767  000000G                                                 JSR     PC,MSGDEQ
(3)  000562' 012605                                                          MOV     (SP)+,R5
887
888                                 ;+
889                                 ; NOW GET SOME NEW INPUT FROM KEYBOARD, WHEN A CR
```

```
                                ; IS STRUCK, INTERRUPT THE REPEAT-LOOP
                                ;-
000564'                                 LET DX.KFL := #0
(4)  000564' 005067  000000G                                                    CLR     DX.KFL
                        CALL KBINI IN <R0>
000570'                                                                         MOV     R5,-(SP)
(3)  000570' 010546                                                             MOV     R0,-(R5)
(4)  000572' 010045                                                             JSR     PC,KBINI
(3)  000574' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000600' 012605
                                        REPEAT
000602'                                                                 50020$:
(3)  000602'
                                                CALL MSGDEQ IN <R0>
000602'                                                                         MOV     R5,-(SP)
(3)  000602' 010546                                                             MOV     R0,-(R5)
(4)  000604' 010045                                                             JSR     PC,MSGDEQ
(3)  000606' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000612' 012605
                                        UNTILB DX.KFL NE #0
000614'                                                                         TSTB    DX.KFL
(3)  000614' 105767  000000G                                                    BEQ     50020$
(6)  000620' 001770
                                INLINE <2$:>
000622'                                                                 2$:
(2)  000622'
                                ;+
                                ; INIT KM.FLG
                                ;-
000622'                                 LET KM.FLG :B= #0
(4)  000622' 105067  177176                                                     CLRB    KM.FLG
                                ;+
                                ; IF FIRST CHAR WAS A CR... JUST LEAVE....
                                ;-
000626'                                 LET R1 := DT.KBUF(R0)
(4)  000626' 016001  000030                                                     MOV     DT.KBUF(R0),R1
000632'                                 IFB (R1) EQ #CR THEN
(6)  000632' 121127  000015                                                     CMPB    (R1),#CR
(9)  000636' 001002                                                             BNE     50021$
000640'                                         INLINE <JMP     1$>
(2)  000640' 000167  000252                                                     JMP     1$
000644'                                 ENDIF
(4)  000644'                                                            50021$:
                                ;+
                                ; GET ORIGINAL DECODE BUFFER POINTER... COPY KEYBOARD BUFFER INTO
                                ; DECODE BUFFER.....
```

```
                                ;-
                                        LET R1 := BUFPTR(R5)
000644'                                                                 MOV     BUFPTR(R5),R1
(4) 000644' 016501  000002
                                        CALL CMDCPY IN <R0,R1> OUT <R4>
000650'                                                                 SUB     #1*2,R5
(4) 000650' 162705  000002                                              MOV     R5,-(SP)
(3) 000654' 010546                                                      MOV     R1,-(R5)
(5) 000656' 010145                                                      MOV     R0,-(R5)
(4) 000660' 010045                                                      JSR     PC,CMDCPY
(3) 000662' 004767  000000G                                             MOV     (SP)+,R5
(3) 000666' 012605                                                      MOV     (R5)+,R4
(4) 000670' 012504
                                ;+
                                ; GET FIRST ARGUMENT
                                ;-
                                        CALL ARGCHK IN <R1> OUT <R1>
000672'                                                                 SUB     #1*2,R5
(4) 000672' 162705  000002                                              MOV     R5,-(SP)
(3) 000676' 010546                                                      MOV     R1,-(R5)
(4) 000700' 010145                                                      JSR     PC,ARGCHK
(3) 000702' 004767  000000G                                             MOV     (SP)+,R5
(3) 000706' 012605                                                      MOV     (R5)+,R1
(4) 000710' 012501
                                ;+
                                ; IF THE FIRST CHAR IS NOT A TERMINATOR (CR,LF,^U)... GO CHECK NUMBER...
                                ;-
                                        IF R4 EQ #0 ANDB (R1) NE #LF ANDB (R1) NE #CR THEN
000712'                                                                 TST     R4
(6) 000712' 005704                                                      BNE     50022$
(9) 000714' 001054                                                      CMPB    (R1),#LF
(6) 000716' 121127  000012                                              BEQ     50022$
(9) 000722' 001451                                                      CMPB    (R1),#CR
(6) 000724' 121127  000015                                              BEQ     50022$
(9) 000730' 001446
                                                CALL NUMCHK IN <R0,R1> OUT <R1,R2>
000732'                                                                 SUB     #2*2,R5
(4) 000732' 162705  000004                                              MOV     R5,-(SP)
(3) 000736' 010546                                                      MOV     R1,-(R5)
(5) 000740' 010145                                                      MOV     R0,-(R5)
(4) 000742' 010045                                                      JSR     PC,NUMCHK
(3) 000744' 004767  000000G                                             MOV     (SP)+,R5
(3) 000750' 012605                                                      MOV     (R5)+,R1
(4) 000752' 012501                                                      MOV     (R5)+,R2
(4) 000754' 012502
                                                IF.NO.ERROR THEN
000756'                                                                 BCS     50023$
(6) 000756' 103430
                                ;+
                                ; NUMBER O.K. - GET NEXT ARGUMENT
                                ;-
```

```
                                        CALL ARGCHK IN <R1> OUT <R1>
000760'                                                                 SUB     #1*2,R5
(4)  000760' 162705  000002                                             MOV     R5,-(SP)
(3)  000764' 010546                                                     MOV     R1,-(R5)
(4)  000766' 010145                                                     JSR     PC,ARGCHK
(3)  000770' 004767  000000G                                           MOV     (SP)+,R5
(3)  000774' 012605                                                     MOV     (R5)+,R1
(4)  000776' 012501

                    ;+
                    ; IF NOT A TERMINATOR (CR OR LF) STUFF ERROR MSG
                    ;-
                                        IFB (R1) NE #LF ANDB (R1) NE #CR THEN
001000'                                                                 CMPB    (R1),#LF
(6)  001000' 121127  000012                                             BEQ     50024$
(9)  001004' 001412                                                     CMPB    (R1),#CR
(6)  001006' 121127  000015                                             BEQ     50024$
(9)  001012' 001407

                                            LET DT.KBRSP(R0) := #CM.ARG
001014'                                                                 MOV     #CM.ARG,DT.KBRSP
(4)  001014' 012760  000000G 000022
                                            LET KM.ABT :B= #1
001022'                                                                 MOVB    #1,KM.ABT
(4)  001022' 112767  000001  176773

                                        ELSE
001030'                                                                 BR      50025$
(4)  001030' 000402                                             50024$:
(3)  001032'

                    ;+
                    ; IT WAS A TERMINATOR... CHANGE CONTENTS OF LOCATION(IN LOCAL STORAGE)
                    ;-

                                            LET @KM.LOC := R2
001032'                                                                 MOV     R2,@KM.LOC
(4)  001032' 010277  176742

                                        ENDIF
001036'                                                         50025$:
(4)  001036'

                                    ELSE
001036'                                                                 BR      50026$
(4)  001036' 000403                                             50023$:
(3)  001040'

                    ;+
                    ; BAD NUMBER.... SET ABORT FLAG
                    ;-

                                            LET KM.ABT :B= #1
001040'                                                                 MOVB    #1,KM.ABT
(4)  001040' 112767  000001  176755

                                    ENDIF
001046'                                                         50026$:
(4)  001046'

                                ENDIF
001046'                                                         50022$:
(4)  001046'
```

```
                                ;+
                                ; SEE IF TERMINATOR IS LF AND ABORT FLAG = 0 (NO ABORT)
                                ;-

001046'                                 IFB (R1) EQ #LF  ANDB KM.ABT EQ #0 AND R4 EQ #0 THEN
(6)  001046' 121127  000012                                                     CMPB    (R1),#LF
(9)  001052' 001007                                                             BNE     50027$
(6)  001054' 105767  176743                                                     TSTB    KM.ABT
(9)  001060' 001004                                                             BNE     50027$
(6)  001062' 005704                                                             TST     R4
(9)  001064' 001002                                                             BNE     50027$

001066'                                         LET R3 := R3 + #2
(6)  001066' 062703  000002                                                     ADD     #2,R3

001072'                                 ENDIF                           50027$:
(4)  001072'
                                ;+
                                ; UNTIL TERMINATOR = CR AND ABORT FLAG NOT SET (= 1)
                                ;-

001072'                                 INLINE <CMPB (R1),#CR>                  CMPB (R1),#CR
(2)  001072' 121127  000015
001076'                                 INLINE <BEQ 6$>                         BEQ 6$
(2)  001076' 001402
001100'                                 INLINE <JMP 5$>                         JMP 5$
(2)  001100' 000167  177312
001104'                                 INLINE <6$:>                            6$:
(2)  001104'
001104'                                 INLINE <TSTB KM.ABT>                    TSTB KM.ABT
(2)  001104' 105767  176713
001110'                                 INLINE <BEQ 7$>                         BEQ 7$
(2)  001110' 001402
001112'                                 INLINE <JMP 5$>                         JMP 5$
(2)  001112' 000167  177300
001116'                                 INLINE <7$:>                            7$:
(2)  001116'


001116'                         INLINE <1$:>                                    1$:
(2)  001116'
001116'                          CALL RESREG                                    JSR     PC,RESREG
(3)  001116' 004767  000000G

001122'                         ENDRTN                          50000$:
(3)  001122'                                                    50001$:
(3)  001122'                                                            RTS     PC
(2)  001122' 000207

             000001             .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16 = ****** G
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CMDCPY= ****** G
CM.ADR= ****** G
CM.ARG= ****** G
CM.BAD= ****** G
CM.ODD= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.KFL= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KBINI = ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KMOD    000026RG
KM.ABT  000023R
KM.FLG  000024R
KM.LOC  000000R
KM.MG1  000002R
KM.MG2  000012R
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK= ****** G
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
NUMCHK= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
```

```
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000110
$NSK2 = 000110
$NSK3 = 000110
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050030
$TEMP = 000300
$TSK0 = 050027
$TSK1 = 050026
$TSK2 = 050025
$$ARGC= 000004
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 001064R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 001124R

. ABS.  000000     000
        001124     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KMOD,DSKZ:KMOD=SPMAC/ML,EQUATE,KMOD
 RUN-TIME: 20 10 .4 SECONDS
 RUN-TIME RATIO: 58/32=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KMONOF PROCESS MON, MOFF KEYBOARD COMMANDS
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       KMONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
                        ;       1. KMON - TURN THE UNIBUS MAP HARDWARE ON
                        ;       2. KMOFF - TURN THE UNIBUS MAP HARDWARE OFF
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  KMON  PROCESS MON KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KMON
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE "MON" KEYBOARD COMMAND.
                        ;       IT WILL TEST TO SEE IF 22 BIT ADDRESSING IS AVAILABLE, AND THAT THE
                        ;       EXERCISER IS NOT IN RUN MODE. IT WILL
                        ;       ENABLE THE UNIBUS MAP HARDWARE.
                        ;       IF IN RUN MODE OR NO 22 BIT ADDRESSING AVAILABLE, THE APPROPRIATE
                        ;       ERROR MESSAGE IS OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0
                        ;       DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ST0
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG,CM.RUN
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KMON IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  KMOFF PROCESS MOFF KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KMOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL PROCESS THE "MOFF" KEYBOARD COMMAND. IT
                        ;       WILL TEST FOR THE PRESENCE OF 22 BIT ADDRESSING. IF NO 22 BIT ADDRESSING,
                        ;       IT WILL STUFF A NO UNIBUS MAP MESSAGE. IF UNIBUS MAP IS AVAILABLE, A CHECK
                        ;       IS MADE TO SEE IF THE EXERCISER IS IN RUN MODE. IF IN RUN
                        ;       MODE, THE ERROR MESSAGE IS STUFFED. IF NOT IN RUN MODE,
                        ;       THE STATUS WORD WILL BE UPDATED(THE UNIBUS MAP BIT CLEARED)
                        ;       AND THE UNIBUS MAP TURNED OFF.
                        ;       LOCATION.

                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;       COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.CF0
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ST0
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG,CM.RUN
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KMOFF IN <DTADR,CMDBUF>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             CMDBUF = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINTITIONS AND REFERENCES FOR KMON

                                .MCALL  STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTIN=1
                        000001  $LSTTAG=1


                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KMOFF           ;MODULE ENTRY POINT
                                .GLOBL  KMON            ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  ARGCHK          ;CHECK FOR EXISTENCE OF AN ARGUMENT
                                .GLOBL  CM.ARG          ;'ILLEGAL OR INVALID ARG' MESSAGE
                                .GLOBL  CM.RUN          ;"NOT A VALID COMMAND IN RUN MODE" MESSAGE
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 047516  052440  044516  MB.NOMB:        .ASCIZ  /NO UNIBUS MAP%/
000006' 052502  020123  040515
000014' 022520     000
000017'    125  044516  052502  MB.ON:  .ASCIZ  /UNIBUS MAP ON%/
000024' 020123  040515  020120
000032' 047117  000045
000036' 047125  041111  051525  MB.OFF: .ASCIZ  /UNIBUS MAP OFF%/
000044' 046440  050101  047440
000052' 043106  000045
000056'    000                  MB.FLG: .BYTE   0               ;UNIBUS MAP ON/OFF FLAG
                000060'         .EVEN
```

```
                                .SBTTL  KMON    ROUTINE

000060'                         ROUTINE KMON <DTADR,BUFPTR>
(2)  000060'                                                            KMON:

                                ;+
                                ; SET ON FLAG
                                ;-

000060'                         LET MB.FLG :B= #0
(4)  000060' 105067  177772                                                     CLRB    MB.FLG
000064'                         INLINE <BR MBROU>
(2)  000064' 000404                                                             BR MBROU
000066'                         ENDRTN
(3)  000066'                                                            50000$:
(3)  000066'                                                            50001$:
(2)  000066' 000207                                                             RTS     PC

000070'                         ROUTINE KMOFF <DTADR,BUFPTR>
(2)  000070'                                                            KMOFF:

                                ;+
                                ; SET THE OFF FLAG
                                ;-

000070'                         LET MB.FLG :B= #1
(4)  000070' 112767  000001  177760                                             MOVB    #1,MB.FLG

                                ;+
                                ; SAVE REGISTERS AND DATA TABLE ADDRESS
                                ;-

000076'                         INLINE <MBROU:>
(2)  000076'                                                            MBROU:
000076'                         PUSH R0,R1
(2)  000076' 010046                                                             MOV     R0,-(SP)
(3)  000100' 010146                                                             MOV     R1,-(SP)
000102'                         LET R0 := DTADR(R5)
(4)  000102' 016500  000000                                                     MOV     DTADR(R5),R0
000106'                         LET R1 := BUFPTR(R5)
(4)  000106' 016501  000002                                                     MOV     BUFPTR(R5),R1

                                ;+
                                ; IF NOT IN RUN MODE CONTINUE
                                ;-

000112'                         IF #RUNMODE NOTSETIN DT.ST0(R0) THEN
(6)  000112' 032760  100000  000010                                             BIT     #RUNMODE,DT.ST0(
(9)  000120' 001054                                                             BNE     50002$

                                ;+
                                ; SEE IF ANY ARGUMENTS EXIST & IF
                                ; THEY DO , THIS IS A BAD COMMAND
                                ;-

000122'                                 CALL ARGCHK IN <R1> OUT <R1>
```

```
(4)  000122' 162705  000002                                                    SUB     #1*2,R5
(3)  000126' 010546                                                            MOV     R5,-(SP)
(4)  000130' 010145                                                            MOV     R1,-(R5)
(3)  000132' 004767  000000G                                                   JSR     PC,ARGCHK
(3)  000136' 012605                                                            MOV     (SP)+,R5
(4)  000140' 012501                                                            MOV     (R5)+,R1
705
706                                     ;+
707                                     ; THE ARGUMENT IS A CR SO ALL IS O.K.
708                                     ; SEE IF 22 BIT ADDRESSING IS PRESENT AND IF IT IS,
709                                     ; IF MB FLAG IS 0 THEN SET THE KTSTAT BIT IN DT.ST0 OTHERWISE CLEAR IT.
710                                     ;-
711  000142'                                    IF.ERROR THEN                  BCC     50003$
(6)  000142' 103037
712
713
714
715  000144'                                            IF #ADDR22 SETIN DT.CF0(R0) THEN
                                                                               BIT     #ADDR22,DT.CF0(R
(6)  000144' 032760  001000  000014                                            BEQ     50004$
(9)  000152' 001427
716                                     ;+
717                                     ; IF MB.FLG EQUALS 0 THEN TURN MAP ON
718                                     ;-
719
720  000154'                                          IFB MB.FLG EQ #0 THEN
                                                                               TSTB    MB.FLG
(6)  000154' 105767  177676                                                    BNE     50005$
(9)  000160' 001012
721  000162'                                                 LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MAPSTAT
                                                                               BIS     #MAPSTAT,DT.ST0(
(6)  000162' 052760  000200  000010
722  000170'                                                 LET @#SR3 := @#SR3 SET.BY #BIT05
                                                                               BIS     #BIT05,@#SR3
(6)  000170' 052737  000040  172516
723                                     ;+
724                                     ; LOAD UNIBUS MAP ON MESSAGE
725                                     ;-
726
727  000176'                                                 LET DT.KBRSP(R0) := #MB.ON
                                                                               MOV     #MB.ON,DT.KBRSP(
(4)  000176' 012760  000017' 000022
728  000204'                                          ELSE                     BR      50006$
(4)  000204' 000411
(3)  000206'                                                                50005$:
729
730                                     ;+
731                                     ; TURN IT OFF
732                                     ;-
733
734  000206'                                                 LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #MAPSTAT
                                                                               BIC     #MAPSTAT,DT.ST0(
(6)  000206' 042760  000200  000010
735  000214'                                                 LET @#SR3 := @#SR3 CLR.BY #BIT05
                                                                               BIC     #BIT05,@#SR3
(6)  000214' 042737  000040  172516
736                                     ;+
737                                     ; LOAD THE UNIBUS MAP OFF MESSAGE
738                                     ;-
739
740  000222'                                                 LET DT.KBRSP(R0) := #MB.OFF
                                                                               MOV     #MB.OFF,DT.KBRSP
(4)  000222' 012760  000036' 000022
741  000230'                                          ENDIF
```

```
(4)  000230'                                                                    50006$:
742  000230'                                    ELSE
(4)  000230' 000403                                                                     BR      50007$
(3)  000232'                                                                    50004$:
743                                 ;+
744                                 ; LOAD THE NO UNIBUS MAP MESSAGE
745                                 ;-
746
747  000232'                                        LET DT.KBRSP(R0) := #MB.NOMB
(4)  000232' 012760  000000' 000022                                                     MOV     #MB.NOMB,DT.KBRS
748  000240'                                    ENDIF
(4)  000240'                                                                    50007$:
749
750  000240'                                ELSE
(4)  000240' 000403                                                                     BR      50010$
(3)  000242'                                                                    50003$:
751                                 ;+
752                                 ; LOAD THE ILLEGAL ARGUMENT MESSAGE
753                                 ;-
754
755  000242'                                LET DT.KBRSP(R0) := #CM.ARG
(4)  000242' 012760  000000G 000022                                                     MOV     #CM.ARG,DT.KBRSP
756  000250'                                ENDIF
(4)  000250'                                                                    50010$:
757  000250'                            ELSE
(4)  000250' 000403                                                                     BR      50011$
(3)  000252'                                                                    50002$:
758
759                                 ;+
760                                 ; LOAD THE RUN MODE MESSAGE
761                                 ;-
762
763  000252'                                LET DT.KBRSP(R0) := #CM.RUN
(4)  000252' 012760  000000G 000022                                                     MOV     #CM.RUN,DT.KBRSP
764  000260'                            ENDIF
(4)  000260'                                                                    50011$:
765
766                                 ;+
767                                 ; CLEAN UP
768                                 ;-
769
770  000260'                        POP R1,R0
(2)  000260' 012601                                                                     MOV     (SP)+,R1
(3)  000262' 012600                                                                     MOV     (SP)+,R0
771
772  000264'                        ENDRTN
(3)  000264'                                                                    50000$:
(3)  000264'                                                                    50001$:
(2)  000264' 000207                                                                     RTS     PC
773
774          000001                .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CM.ARG= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR  = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KMOFF    000070RG
KMON     000060RG
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MBROU    000076R
MB.FLG   000056R
MB.NOM   000000R
MB.OFF   000036R
MB.ON    000017R
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
```

```
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$BYTE= 000402
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $ISK2 = 000001          $$CASE= 000000
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $ISK3 = 000001          $$DST = 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LOCTA= 177777          $$ELOC= 000402
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $LSTIN= 000001          $$ERFL= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $LSTTA= 000001          $$FLAG= 000001
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NESTL= 177777          $$FROM= 000001
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK0 = 000300          $$LOC = 000160R
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $NSK1 = 000110          $$LOCN= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $NSK2 = 000110          $$REG = 177777
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $NSK3 = 000110          $$RETU= 000000
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $NSK4 = 000110          $$RTN1= 050000
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $SAVLE= 177777          $$RTN2= 050001
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TAGLE= 177777          $$SRC = 000000
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $TAGNU= 050012          $$TGSV= 000000
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $TEMP = 000300          $$TGS1= 000000
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $TSK0 = 050011          $$TGS2= 000000
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $TSK1 = 050010          $$TO  = 000001
UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $TSK2 = 050007          $$TAG= 050000
UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $TSK3 = 050006          .     = 000266R
UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$ARGC= 000004

. ABS.  000000    000
        000266    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KMONOF,DSKZ:KMONOF=SPMAC/ML,EQUATE,KMONOF
 RUN-TIME: 13 4 .4 SECONDS
 RUN-TIME RATIO: 34/18=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KRUN   PROCESS THE 'RUN' COMMAND & 'RUNL' COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KRUN
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'RUN' AND THE 'RUNL' KEYBOARD
                        ;       COMMANDS. IF AN ADDRESS IS INCLUDED, IT WILL BE VERIFIED.
                        ;       ALL OPTION MODULES HEADERS WILL BE INITIALIZED. THE SYS-
                        ;       TEM ERROR COUNT, POWER FAIL COUNT, AND RELOCATION FLAG
                        ;       WILL BE INITIALIZED. THE KT WILL BE MAPPED TO THE NEW
                        ;       LOCATION (IF ANY SPECIFIED). THE WRITE BUFFER LIMITS
                        ;       WILL BE SET UP.
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;       2. DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.MLST
                        ;       2. DT.ST0
                        ;       3. DT.SSIZ
                        ;       4. DT.CF0
                        ;       5. DT.ADDR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.ST0
                        ;       2. DT.KBRSP
                        ;       3. DT.ESIZ
                        ;       4. DT.SYP
                        ;       5. DT.EXS
                        ;       6. DT.PFL
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. BE.MCNT - NUMBER OF MONITOR SYS ERRORS
                        ;       2. PR.RL1 - RELOCATION FLAG
                        ;       3. CM.ADDR
                        ;       4. MD.COD
                        ;       5. CM.RUN
                        ;       6. CM NBAD
                        ;       7. CM.NUM
                        ;       8. CM.ARG
                        ;       9. PA.CNT
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK
                        ;       2. CLRTIM
                        ;       3. SAVREG
                        ;       4. RESREG
                        ;       5. WBFLIM
                        ;       6. WSTBUS
```

```
564                             ;       7. CLKCHK
565                             ;       8. APTSEL
566                             ;       9. PRRLOC
567                             ;       10. MSGDEQ
568                             ;       11. SETVEC
569                             ;       12. UNIMAP
570                             ;
571                             ; FUNCTIONAL SIDE EFFECTS:
572                             ;       NONE
573                             ;
574                             ; CALLING SEQUENCE:
575                             ;       CALL KRUN(L) IN <DTADR,BUFPTR>
576                             ;       WHERE DTADR = DTABLE ADDRESS   ;   BUFPTR = COMMAND DECODE BUFFER P
577                             ;
578                             ; VERSION:
579                             ;       0.0
580                             ;
581                             ;       EDIT            DATE            BY              REASON
582                             ;--
583
584
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KRUN

                                .MCALL STRUCT
                                STRUCT
                                .PRINT  ;SPMAC: VERSION 1.1

                        000001  $LSTTAG=1
                        000001  $LSTIN=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KRUN            ;MODULE ENTRY POINT
                                .GLOBL  KRUNL           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CLKCHK          ;SEE IF A CLOCK IS HERE
                                .GLOBL  APTSEL          ;APT SELECT ROUTINE
                                .GLOBL  MD.COD          ;CODE
                                .GLOBL  CM.ADR          ;'ILLEGAL ADDRESS'
                                .GLOBL  PRRLOC          ;RELOCATE THE CODE
                                .GLOBL  MSGDEQ          ;MESSAGE DEQUE
                                .GLOBL  SETVEC          ;SET UP UNUSED VECTORS
                                .GLOBL  UNIMAP          ;UNIBUS MAPPING
                                .GLOBL  CM.RUN          ;'ILLEGAL COMMAND IN RUN MODE'
                                .GLOBL  CM.NBAD         ;'NOT AN OCTAL #'
                                .GLOBL  CM.NUM          ;'NUMBER TOO LARGE'
                                .GLOBL  CM.ARG          ;'ILLEGAL OR MISSING ARGUMENT'
                                .GLOBL  CLRTIM          ;CLEAR TIME TABLE
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  PR.RL1          ;RELOCATION FLAG
                                .GLOBL  PA.CNT          ;NUMBER OF PARITY ERRORS
                                .GLOBL  BE.MCNT         ;MONITOR SYSTEM ERROR COUNT
                                .GLOBL  WBFLIM          ;ESTABLISH WRITE BUFFER LIMITS
                                .GLOBL  WSTBUS          ;WRITE WORST CASE UNIBUS PATTERN
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  KR.FLG: .WORD   0       ;'1' IF CMD IS RUNL - '0' IF CMD IS RUN
000002' 000000                  KR.MOV: .WORD   0       ;'1' IF MOVING CODE - '0' IF NOT
000004' 052515  052123  041040  KR.SML: .ASCIZ  /MUST BE GREATER THAN OR EQUAL TO 4K (20000)%/
000012' 020105  051107  040505
000020' 042524  020122  044124
000026' 047101  047440  020122
000034' 050505  040525  020114
000042' 047524  032040  020113
000050' 031050  030060  030060
000056' 022451     000
```

```
  634  000061'    101  042104  042522  KR.BIG: .ASCIZ  /ADDRESS O.K. BUT EXERCISER WON'T FIT%/
       000066' 051523  047440  045456
       000074' 020056  052502  020124
       000102' 054105  051105  044503
       000110' 042523  020122  047527
       000116' 023516  020124  044506
       000124' 022524     000
  635  000127'    115  051525  020124  KR.KT:  .ASCIZ  /MUST HAVE KT ON%/
       000134' 040510  042526  045440
       000142' 020124  047117  000045
  636  000150' 047516  046440  042117  KR.MOD: .ASCIZ  /NO MODULES SELECTED%/
       000156' 046125  051505  051440
       000164' 046105  041505  042524
       000172' 022504     000
  637  000175'    115  050101  041040  KR.MAP: .ASCIZ  /MAP BOX MUST BE ON%/
       000202' 054117  046440  051525
       000210' 020124  042502  047440
       000216' 022516     000
  638          000222'                 .EVEN
  639                                  ;
  640                                  ;***************
  641
  642
```

```
                                .SBTTL  KRUN ROUTINE

                                ROUTINE KRUN <DTADR,BUFPTR>                     KRUN:

                                ;+
                                ; INDICATE 'RUN' COMMAND
                                ;-

                                LET KR.FLG := #0                                        CLR     KR.FLG
                                INLINE <BR      RUNSUB>                                 BR      RUNSUB
                                ENDRTN                                          50000$:
                                                                                50001$:
                                                                                        RTS     PC

                                ROUTINE KRUNL <DTADR,BUFPTR>                    KRUNL:

                                ;+
                                ; INDICATE 'RUNL COMMAND
                                ;-

                                LET KR.FLG := #1                                        MOV     #1,KR.FLG
```

```
                                INLINE <RUNSUB:>                                        RUNSUB:

                                ;+
                                ;INIT FLAG,SAVE REGISTERS,DTABLE AND BUF PTR
                                ;-


                                LET KR.MOV := #0                                        CLR     KR.MOV
000240' 005067  177536
                                CALL SAVREG                                             JSR     PC,SAVREG
000244' 004767  000000G
                                LET R0 := DTADR(R5)                                     MOV     DTADR(R5),R0
000250' 016500  000000
                                LET R1 := BUFPTR(R5)                                    MOV     BUFPTR(R5),R1
000254' 016501  000002
                                ;+
                                ; IF IN AUTO OR APTPRES MODE, DON'T CHECK ANY ARGUMENTS - JUST GO TO WORK
                                ;-

                                IF #AUTO SETIN DT.CF0(R0) OR #APTPRES SETIN DT.CF0(R0) THEN
000260' 032760  000010  000014                                                          BIT     #AUTO,DT.CF0(R0)
000266' 001004                                                                          BNE     50002$
000270' 032760  000200  000014                                                          BIT     #APTPRES,DT.CF0(
000276' 001402                                                                          BEQ     50003$
                                                                                50002$:
                                        INLINE <JMP     339$>                           JMP     339$
000300'
                                ENDIF                                           50003$:


                                ;+
                                ; SEE IF IN RUNMODE - IF SO STUFF ERROR AND SCRAM
                                ;-

                                IF #RUNMODE SETIN DT.ST0(R0) THEN
000304' 032760  100000  000010                                                          BIT     #RUNMODE,DT.ST0(
000312' 001405                                                                          BEQ     50004$
                                        LET DT.KBRSP(R0) := #CM.RUN                     MOV     #CM.RUN,DT.KBRSP
000314' 012760  000000G 000022
                                        INLINE <JMP     1$>                             JMP     1$
000322' 000167  001166

                                ENDIF                                           50004$:


                                ;+
                                ; WE ARE NOT IN RUNMODE....ALL IS OK, SAVE DT.ADDR.....GET AN ARGUMENT IF ANY.....
                                ;-


                                ;+
                                ; SAVE DT.ADDR
                                ;-
```

```
                                LET R3 := DT.ADDR(R0)                                   MOV     DT.ADDR(R0),R3

                                        CALL ARGCHK IN <R1> OUT <R1>                    SUB     #1*2,R5
                                                                                        MOV     R5,-(SP)
                                                                                        MOV     R1,-(R5)
                                                                                        JSR     PC,ARGCHK
                                                                                        MOV     (SP)+,R5
                                                                                        MOV     (R5)+,R1
                                ;+
                                ; IF NO ARGS, JUMP AHEAD
                                ;-


                                        IF.ERROR THEN                                   BCC     50005$

                                                INLINE <JMP     339$>                   JMP     339$

                                        ENDIF                                   50005$:
                                ;+
                                ; NOT A CARRIAGE RETURN..SET FLAG.. GET BY ANY LEADING 0'S AND VERIFY NUMBER....
                                ;-


                                        LET KR.MOV := #1                                MOV     #1,KR.MOV

                                        WHILEB (R1) EQ #'0 DO                   50006$:
                                                                                        CMPB    (R1),#'0
                                                                                        BNE     50007$
                                                LET R1 := R1 + #1                       INC     R1

                                        ENDDO                                           BR      50006$
                                                                                50007$:


                                ;+
                                ; INIT C-BIT AND REGISTERS
                                ;-

                                        LET CARRY := 0                                  CLC

                                        LET R4 := #0                                    CLR     R4

                                        LET R3 := #0                                    CLR     R3

                                        LET R2 := #0                                    CLR     R2
```

```
                                ;+
                                ; NUMBER VERIFICATION CONSISTS OF CHECKING UNTIL SOMETHING OTHER
                                ; THAN A '0' THRU '7' IS DETECTED IN THE DECODE BUFFER.....THE OCTAL
                                ; ASCII CHARACTERS WILL BE DE-ASCIIED, SHIFTED INTO R3 AND ANY
                                ; OVERFLOW SHIFTED INTO R4....A COUNT OF THE NUMBER OF CHARACTERS IS
                                ; MAINTAINED......
                                ;-

000410'                                 WHILEB (R1) GE #'0 ANDB (R1) LE #'7 DO
000410'                                                                         50010$:
(6) 000410' 121127  000060                                                      CMPB    (R1),#'0
(9) 000414' 002420                                                              BLT     50011$
(6) 000416' 121127  000067                                                      CMPB    (R1),#'7
(9) 000422' 003015                                                              BGT     50011$

                                ;+
                                ; GET A BYTE
                                ;-
000424'                                         LET R0 :B= (R1)
(4) 000424' 111100                                                              MOVB    (R1),R0
                                ;+
                                ; DE-ASCII IT
                                ;-

000426'                                         LET R0 := R0 - #'0
(6) 000426' 162700  000060                                                      SUB     #'0,R0

                                ;+
                                ; SHIFT OLD STUFF(LO ORDER) TO RIGHT
                                ;-

000432'                                         LET R3 := R3 SHIFT 1
(7) 000432' 006303                                                              ASL     R3
                                ;+
                                ;ROTATE HIGH ORDER
                                ;-
000434'                                         LET R4 := R4 ROTATE  1
(7) 000434' 006104                                                              ROL     R4
000436'                                         LET R3 := R3 SHIFT 1
(7) 000436' 006303                                                              ASL     R3
000440'                                         LET R4 := R4 ROTATE 1
(7) 000440' 006104                                                              ROL     R4
000442'                                         LET R3 := R3 SHIFT 1
(7) 000442' 006303                                                              ASL     R3
000444'                                         LET R4 := R4 ROTATE 1
(7) 000444' 006104                                                              ROL     R4

                                ;+
                                ; ADD THE NEW CHAR
                                ;-
000446'                                         LET R3 := R3 + R0
(6) 000446' 060003                                                              ADD     R0,R3

                                END
```

```
                                ;+
                                ; INCREMENT CHAR COUNTER
                                ;-
000450'                                 LET R2 := R2 + #1
(6)  000450' 005202                                                     INC     R2

                                ;+
                                ; GET NEXT BYTE
                                ;-

000452'                                 LET R1 := R1 + #1
(6)  000452' 005201                                                     INC     R1

000454'                         ENDDO
(4)  000454' 000755                                                     BR      50010$
(3)  000456'                                                    50011$:

                                ;+
                                ; RESTORE DTABLE TO R0
                                ;-

000456'                         LET R0 := DTADR(R5)
(4)  000456' 016500  000000                                             MOV     DTADR(R5),R0

                                ;+
                                ; R1 IS NOW POINTING TO FIRST NON-OCTAL-ASCII-CHAR....IT MUST BE
                                ; CR OR SPACE...IF NOT - STUFF ERROR MSG AND SCRAM....
                                ;-

000462'                         IFB (R1) NE #CR ANDB (R1) NE #SPACE THEN
(6)  000462' 121127  000015                                             CMPB    (R1),#CR
(9)  000466' 001410                                                     BEQ     50012$
(6)  000470' 121127  000040                                             CMPB    (R1),#SPACE
(9)  000474' 001405                                                     BEQ     50012$

000476'                                 LET DT.KBRSP(R0) := #CM.NBAD
(4)  000476' 012760  000000G 000022                                     MOV     #CM.NBAD,DT.KBRS
000504'                                 INLINE <JMP     1$>
(2)  000504' 000167  001004                                             JMP     1$
000510'                         ENDIF
(4)  000510'                                                    50012$:

                                ;+
                                ; IF THE ADDRESS GIVEN WAS MORE THAN 8 CHARS...(NOT COUNTING LEADING 0'S)..STUFF
                                ; ERROR MSG AND RETURN...
                                ;-

000510'                         IF R2 GT #10 THEN
(6)  000510' 020227  000010                                             CMP     R2,#10
(9)  000514' 003405                                                     BLE     50013$
000516'                                 LET DT.KBRSP(R0) := #CM.ADR
(4)  000516' 012760  000000G 000022                                     MOV     #CM.ADR,DT.KBRSP
000524'                                 INLINE <JMP     1$>
(2)  000524' 000167  000764                                             JMP     1$
```

```
822  000530'                                ENDIF
(4)  000530'                                                                50013$:
823
824
825                                 ;+
826                                 ; CONVERT THE NUMBER TO PAR FORMAT...TO DO, SHIFT THE HIGH AND LOW
827                                 ; ADDRESS BITS TO THE RIGHT 6 TIMES.....
828                                 ;-
829
830
831                                 ;+
832                                 ; INIT COUNTER
833                                 ;-
834  000530'                                LET R2 := #0                                       CLR     R2
(4)  000530' 005002
835
836  000532'                                WHILE R2 NE #6 DO                                  50014$:
(4)  000532'                                                                                    CMP     R2,#6
(6)  000532' 020227  000006                                                                     BEQ     50015$
(9)  000536' 001404
837
838  000540'                                        LET R4 := R4 SHIFT -1                       ASR     R4
(7)  000540' 006204
839  000542'                                        LET R3 := R3 ROTATE -1                      ROR     R3
(7)  000542' 006003
840  000544'                                        LET R2 := R2 + #1                           INC     R2
(6)  000544' 005202
841
842  000546'                                ENDDO                                              BR      50014$
(4)  000546' 000771
(3)  000550'                                                                50015$:
843
844
845                                 ;+
846                                 ; R3 IS NOW PAR FORMAT. IF R4, THE HIGH ORDER BITS, IS NOT ZERO,
847                                 ; THE ADDRESS WAS TOO BIG....STUFF ERROR AND SCRAM...
848                                 ;-
849
850  000550'                                IF R4 NE #0 THEN                                   TST     R4
(6)  000550' 005704
(9)  000552' 001405                                                                             BEQ     50016$
851  000554'                                        LET DT.KBRSP(R0) := #CM.ADR
(4)  000554' 012760  000000G 000022                                                             MOV     #CM.ADR,DT.KBRSP
852  000562'                                        INLINE <JMP     1$>
(2)  000562' 000167  000726                                                                     JMP     1$
853
854  000566'                                ENDIF
(4)  000566'                                                                50016$:
855                                 ;+
856                                 ; ALSO BUST COMMAND IF THE LOW ORDER WORD IS LESS THAN 4K
857                                 ;-
858
859
860  000566'                                IF R3 LOS #177 THEN                                CMP     R3,#177
(6)  000566' 020327  000177
(9)  000572' 101005                                                                             BHI     50017$
```

```
                                        LET DT.KBRSP(R0) := #KR.SML
000574'                                                                 MOV     #KR.SML,DT.KBRSP
(4)  000574' 012760  000004' 000022
                                        INLINE <JMP     1$>
000602'                                                                 JMP     1$
(2)  000602' 000167  000706
                                ENDIF
000606'                                                         50017$:
(4)  000606'
                        ;+
                        ; NEXT, SEE IF THE EXERCISER CAN POSSIBLY FIT INTO THE NEW ADDRESS SPACE
                        ; AND SEE IF ADDRESS EXISTS AT ALL.....GET THE EXERCISER SIZE, CONVERT
                        ; TO PAR FORMAT, IF THE NEW ADDRESS DESIRED IS GREATER THAN THE SYSTEM
                        ; SIZE....STUFF ERROR MSG AND RETURN...IF NOT, SUBTRACT THE NEW ADDRESS
                        ; FROM THE SYSTEM SIZE....SEE IF THE EXERCISER WILL FIT....IF NOT
                        ; STUFF ERROR MSG AND RETURN.....
                        ;-
                                IF R3 HI DT.SSIZ(R0) THEN
000606'                                                                 CMP     R3,DT.SSIZ(R0)
(6)  000606' 020360  000046                                             BLOS    50020$
(9)  000612' 101405
                                        LET DT.KBRSP(R0) := #CM.ADR
000614'                                                                 MOV     #CM.ADR,DT.KBRSP
(4)  000614' 012760  000000G 000022
                                        INLINE <JMP     1$>
000622'                                                                 JMP     1$
(2)  000622' 000167  000666
                                ENDIF
000626'                                                         50020$:
(4)  000626'

                                LET R2 := DT.ESIZ(R0)
000626'                                                                 MOV     DT.ESIZ(R0),R2
(4)  000626' 016002  000044
                                LET R4 := #0
000632'                                                                 CLR     R4
(4)  000632' 005004
                                WHILE R4 NE #6 DO
000634'                                                         50021$:
(4)  000634'                                                            CMP     R4,#6
(6)  000634' 020427  000006                                             BEQ     50022$
(9)  000640' 001404
                                        LET CARRY := 0
000642'                                                                 CLC
(4)  000642' 000241
                                        LET R2 := R2 ROTATE -1
000644'                                                                 ROR     R2
(7)  000644' 006002
                                        LET R4 := R4 + #1
000646'                                                                 INC     R4
(6)  000646' 005204
                                ENDDO
000650'                                                                 BR      50021$
(4)  000650' 000771
(3)  000652'                                                    50022$:

                        ;+
                        ; R2 NOW HAS EXERCISER SIZE IN PAR FORMAT....SEE IF ROOM EXISTS FOR IT...
                        ; FIRST SUBTRACT 4K FROM IT (THE UN MOVABLE PORTION) AND SUBTRACT NEW
                        ; ADDR FROM SYS SIZE....
                        ;-

                                LET R2 := R2 - #177
000652'                                                                 SUB     #177,R2
(6)  000652' 162702  000177
                                LET R4 := DT.SSIZ(R0) - R3
000656'
```

```
(4)  000656' 016004  000046                                                  MOV     DT.SSIZ(R0),R4
(6)  000662' 160304                                                          SUB     R3,R4
898
899                                 ;+
900                                 ; IF R4 (THE DIFFERENCE BETWEEN THE TOP OF MEMORY AND RELOCATION
901                                 ; ADDRESS) IS LESS THAN THE EXERCISER SIZE, WE OBVIOUSLY DON'T HAVE ROOM.
902                                 ; STUFF ERROR MSG AND RETURN.
903                                 ;-
904
905  000664'                                IF R4 LE R2 THEN                    CMP     R4,R2
(6)  000664' 020402                                                          BGT     50023$
(9)  000666' 003005
906  000670'                                    LET DT.KBRSP(R0) := #KR.BIG   MOV     #KR.BIG,DT.KBRSP
(4)  000670' 012760  000061' 000022
907  000676'                                    INLINE <JMP     1$>           JMP     1$
(2)  000676' 000167  000612
908  000702'                                ENDIF
(4)  000702'                                                          50023$:
909
910                                 ;+
911                                 ; AT THIS POINT, THE NUMBER HAS BEEN VERIFIED.... FINALLY CHECK FOR
912                                 ; JUNK ARGUMENTS
913                                 ;-
914
915  000702'                                CALL ARGCHK IN <R1> OUT <R1>        SUB     #1*2,R5
(4)  000702' 162705  000002                                                  MOV     R5,-(SP)
(3)  000706' 010546                                                          MOV     R1,-(R5)
(4)  000710' 010145                                                          JSR     PC,ARGCHK
(3)  000712' 004767  000000G                                                 MOV     (SP)+,R5
(3)  000716' 012605                                                          MOV     (R5)+,R1
(4)  000720' 012501
916
917  000722'                                IF.NO.ERROR THEN                    BCS     50024$
(6)  000722' 103405
918  000724'                                    LET DT.KBRSP(R0) := #CM.ARG   MOV     #CM.ARG,DT.KBRSP
(4)  000724' 012760  000000G 000022
919  000732'                                    INLINE <JMP     1$>           JMP     1$
(2)  000732' 000167  000556
920
921  000736'                                ENDIF
(4)  000736'                                                          50024$:
922
923                                 ;+
924                                 ; JUST CHECK TO MAKE SURE KT IS ENABLED....
925                                 ;-
926
927  000736'                                IF #KTSTAT NOTSETIN DT.ST0(R0) THEN BIT     #KTSTAT,DT.ST0(R
(6)  000736' 032760  000020  000010                                          BNE     50025$
(9)  000744' 001005
928  000746'                                    LET DT.KBRSP(R0) := #KR.KT    MOV     #KR.KT,DT.KBRSP(
(4)  000746' 012760  000127' 000022
929  000754'                                    INLINE <JMP     1$>           JMP     1$
(2)  000754' 000167  000534
930  000760'                                ENDIF
(4)  000760'                                                          50025$:
931
```

```
                                ;+
                                ; IF R3 IS BIGGER THAN OR EQUAL TO 6000 AND MAPPING IS NOT ON, BUST
                                ; COMMAND....THAT IS , YOU CANNOT RUN ABOVE 96K (ASSUMES AN EXERCISER SIZE
                                ; OF 28K) BECAUSE YOU MAY OVERLAP SOME OF THE EXERCISER ABOVE
                                ; 124K AND WITH MAP BOX OFF, YOU WILL DIE.....
                                ;-

000760'                                 IF R3 HIS #6000 AND #MAPSTAT NOTSETIN DT.ST0(R0) THEN
(6)  000760' 020327  006000                                                     CMP     R3,#6000
(9)  000764' 103411                                                             BLO     50026$
(6)  000766' 032760  000200  000010                                            BIT     #MAPSTAT,DT.ST0(
(9)  000774' 001005                                                             BNE     50026$
000776'                                         LET DT.KBRSP(R0) := #KR.MAP
(4)  000776' 012760  000175' 000022                                            MOV     #KR.MAP,DT.KBRSP
001004'                                         INLINE <JMP     1$>
(2)  001004' 000167  000504                                                     JMP     1$
001010'                                 ENDIF
(4)  001010'                                                            50026$:

001010'                         339$:
                                ;+
                                ; IF UNDER APT, USE APT'S TABLE TO SELECT THE MODULES
                                ; AND SET UP DEVICE COUNT
                                ;-

001010'                         IF #APTPRES SETIN DT.CF0(R0) THEN
(6)  001010' 032760  000200  000014                                            BIT     #APTPRES,DT.CF0(
(9)  001016' 001405                                                             BEQ     50027$
001020'                                 CALL APTSEL IN <R0>
(3)  001020' 010546                                                             MOV     R5,-(SP)
(4)  001022' 010045                                                             MOV     R0,-(R5)
(3)  001024' 004767  000000G                                                    JSR     PC,APTSEL
(3)  001030' 012605                                                             MOV     (SP)+,R5
001032'                         ENDIF
(4)  001032'                                                            50027$:

001032'                         LET R1 := DT.MLST(R0)    ;GET ADDRESS OF MODULE LIST
(4)  001032' 016001  000032                                                     MOV     DT.MLST(R0),R1

001036'                         LET R4 := #0
(4)  001036' 005004                                                             CLR     R4
                                ;+
                                ; BEFORE HONORING COMMAND, MAKE SURE AT LEAST ONE MODULE IS SELECTED
                                ;-

001040'                         WHILE (R1) NE #ENDLST DO
(4)  001040'                                                            50030$:
(6)  001040' 021127  000000                                                     CMP     (R1),#ENDLST
(9)  001044' 001410                                                             BEQ     50031$

                                ;+
                                ; GET A MODULE ADDRESS
                                ;-
001046'                                 LET R2 := (R1)+
```

```
                                        IF #BIT14 SETIN STAT(R2) THEN
                                                                                        MOV     (R1)+,R2
                                                                                        BIT     #BIT14,STAT(R2)
                                                                                        BEQ     50032$
                                                LET R4 := #1
                                                                                        MOV     #1,R4
                                        ENDIF
                                                                                50032$:
                                ENDDO
                                                                                        BR      50030$
                                                                                50031$:

                                ;+
                                ; IF R4 IS ZERO...NO MODULES SELECTED.....STUFF ERROR AND GOODBYE...
                                ;-

                                IF R4 EQ #0 THEN
                                                                                        TST     R4
                                                                                        BNE     50033$
                                        LET DT.KBRSP(R0) := #KR.MOD
                                                                                        MOV     #KR.MOD,DT.KBRSP
                                        INLINE <JMP     1$>
                                                                                        JMP     1$
                                ENDIF
                                                                                50033$:


                                ;+
                                ; !!!!! GO TO WORK !!!!!!    GET THE MODULE HEADERS CLEANED UP !
                                ;-

                                ;+
                                ; BUT FIRST PRINT THE CR - IT TAKES SO LONG TO WRITE WSTBUS
                                ; PATTERN THAT THE CR WOULD NOT BE ECHOED FOR A LONG TIME
                                ; AND IT MAY CAUSE WORRY - HENCE THE CALL TO MSGDEQ WILL PRINT
                                ; IT RIGHT AWAY....
                                ;-


                                        REPEAT
                                                                                50034$:
                                                CALL MSGDEQ IN <R0>
                                                                                        MOV     R5,-(SP)
                                                                                        MOV     R0,-(R5)
                                                                                        JSR     PC,MSGDEQ
                                                                                        MOV     (SP)+,R5
                                        UNTIL MD.COD EQ #MSGNUL
                                                                                        CMP     MD.COD,#MSGNUL
                                                                                        BNE     50034$


                                ;+
```

```
1006                                    ;IF THE CLOCK IS PRESENT, CLEAR CLOCK TABLE....SEE IF IT IS...
1007                                    ;-
1008
1009  001126'                                   CALL CLKCHK IN <R0>                                   MOV     R5,-(SP)
 (3)  001126' 010546                                                                                  MOV     R0,-(R5)
 (4)  001130' 010045                                                                                  JSR     PC,CLKCHK
 (3)  001132' 004767  000000G                                                                        MOV     (SP)+,R5
 (3)  001136' 012605
1010
1011  001140'                                   IF #CLKPRES SETIN DT.CF0(R0) THEN                     BIT     #CLKPRES,DT.CF0(
 (6)  001140' 032760  000001  000014                                                                 BEQ     50035$
 (9)  001146' 001405
1012
1013  001150'                                         CALL CLRTIM IN <R0>                             MOV     R5,-(SP)
 (3)  001150' 010546                                                                                  MOV     R0,-(R5)
 (4)  001152' 010045                                                                                  JSR     PC,CLRTIM
 (3)  001154' 004767  000000G                                                                        MOV     (SP)+,R5
 (3)  001160' 012605
1014
1015  001162'                                   ENDIF                                         50035$:
 (4)  001162'
1016
1017
1018                                    ;+
1019                                    ; GET MODULE LIST
1020                                    ;-
1021  001162'                           LET R1 := DT.MLST(R0)                                         MOV     DT.MLST(R0),R1
 (4)  001162' 016001  000032
1022
1023  001166'                           WHILE (R1) NE #ENDLST DO                              50036$:
 (4)  001166'                                                                                         CMP     (R1),#ENDLST
 (6)  001166' 021127  000000                                                                          BEQ     50037$
 (9)  001172' 001450
1024
1025
1026
1027                                    ;+
1028                                    ; CLEAR OUT HEADER STUFF AND COPY CONFIG WORD TO RES1 IN HEADER
1029                                    ;-
1030  001174'                                 LET R2 := (R1)+                                         MOV     (R1)+,R2
 (4)  001174' 012102
1031
1032
1033  001176'                                 LET PASCNT(R2) := #0                                    CLR     PASCNT(R2)
 (4)  001176' 005062  000034
1034  001202'                                 LET ICOUNT(R2) := #0                                    CLR     ICOUNT(R2)
 (4)  001202' 005062  000040
1035  001206'                                 LET SOFCNT(R2) := #0                                    CLR     SOFCNT(R2)
 (4)  001206' 005062  000042
1036  001212'                                 LET HRDCNT(R2) := #0                                    CLR     HRDCNT(R2)
 (4)  001212' 005062  000044
1037  001216'                                 LET SOFPAS(R2) := #0                                    CLR     SOFPAS(R2)
 (4)  001216' 005062  000046
1038  001222'                                 LET HRDPAS(R2) := #0                                    CLR     HRDPAS(R2)
 (4)  001222' 005062  000050
1039  001226'                                 LET SYSCNT(R2) := #0
```

```
                                                                                        CLR     SYSCNT(R2)
 (4)  001226' 005062  000052
1040  001232'                        LET XFLAG(R2) :B= #0                              CLRB    XFLAG(R2)
 (4)  001232' 105062  000005
1041  001236'                        LET RES1(R2) := DT.CF0(R0)                        MOV     DT.CF0(R0),RES1(
 (4)  001236' 016062  000014  000056
1042
1043                            ;+
1044                            ; CLEAR ACTIVE AND DROPPED BIT IN STAT WORD OF MODULE
1045                            ;-
1046
1047  001244'                        LET STAT(R2) := STAT(R2) CLR.BY #BIT13!ACTBIT     BIC     #BIT13!ACTBIT,ST
 (6)  001244' 042762  024000  000026
1048
1049
1050
1051
1052
1053
1054
1055                            ;+
1056                            ; IF AN IOMODR(P)(X) CLEAR CDATA ERROR AND WORD COUNTS
1057                            ;-
1058
1059  001252'                        IF #IOMODR SETIN STAT(R2) OR #IOMODP SETIN STAT(R2) OR #IOMODX SETIN STAT(R2) TH
 (6)  001252' 032762  112000  000026                                                   BIT     #IOMODR,STAT(R2)
 (8)  001260' 001010                                                                   BNE     50040$
 (6)  001262' 032762  102000  000026                                                   BIT     #IOMODP,STAT(R2)
 (8)  001270' 001004                                                                   BNE     50040$
 (6)  001272' 032762  110000  000026                                                   BIT     #IOMODX,STAT(R2)
 (9)  001300' 001404                                                                   BEQ     50041$
 (6)  001302'                                                                  50040$:
1060
1061                            ;+
1062                            ; THIS MODULE NEEDS X - TRA WORK - CLEAR ERROR COUNTS
1063                            ;-
1064
1065  001302'                             LET CDERCT(R2) := #0                         CLR     CDERCT(R2)
 (4)  001302' 005062  000146
1066  001306'                             LET CDWDCT(R2) := #0                         CLR     CDWDCT(R2)
 (4)  001306' 005062  000144
1067  001312'                        ENDIF
 (4)  001312'                                                                  50041$:
1068
1069  001312'                   ENDDO                                                 BR      50036$
 (4)  001312' 000725
 (3)  001314'                                                                  50037$:
1070
1071
1072
1073                            ;+
1074                            ; CLEAR SOME OTHER GOODIES
1075                            ;-
1076
1077
1078                            ;+
1079                            ; CLEAR OUT SYS PASS COUNT
```

```
1080                                    ;-
1081  001314'                                   LET @DT.SYP(R0) := #0
 (4)  001314' 005070  000072                                                CLR     @DT.SYP(R0)
1082
1083
1084                                    ;+
1085                                    ; CLEAR # OF POWER FAILS
1086                                    ;-
1087  001320'                                   LET DT.PFL(R0) := #0
 (4)  001320' 005060  000062                                                CLR     DT.PFL(R0)
1088
1089                                    ;+
1090                                    ; CLEAR # OF MONITOR SYS ERRORS
1091                                    ;-
1092  001324'                                   LET BE.MCNT := #0
 (4)  001324' 005067  000000G                                               CLR     BE.MCNT
1093
1094                                    ;+
1095                                    ; CLR # OF PARITY ERRORS
1096                                    ;-
1097  001330'                                   LET PA.CNT := #0
 (4)  001330' 005067  000000G                                               CLR     PA.CNT
1098
1099                                    ;+
1100                                    ; CLEAR EXERCISER TOTAL SYS ERRORS
1101                                    ;-
1102  001334'                                   LET DT.EXS(R0) := #0
 (4)  001334' 005060  000060                                                CLR     DT.EXS(R0)
1103
1104                                    ;+
1105                                    ; INIT RELOCATION FLAG
1106                                    ;-
1107  001340'                                   LET PR.RL1 := #1
 (4)  001340' 012767  000001  000000G                                       MOV     #1,PR.RL1
1108
1109
1110
1111                                    ;+
1112                                    ; STUFF UNUSED VECTORS
1113                                    ;-
1114
1115  001346'                                   CALL SETVEC
 (3)  001346' 004767  000000G                                               JSR     PC,SETVEC
1116
1117                                    ;+
1118                                    ; WRITE WORST CASE UNIBUS PATTERN
1119                                    ;-
1120
1121  001352'                                   CALL WSTBUS IN <R0>
 (3)  001352' 010546                                                        MOV     R5,-(SP)
 (4)  001354' 010045                                                        MOV     R0,-(R5)
 (3)  001356' 004767  000000G                                               JSR     PC,WSTBUS
 (3)  001362' 012605                                                        MOV     (SP)+,R5
1122
1123
1124                                    ;+
```

```
                                ; MOVE THE CODE IF NECESSARY
                                ;-
                                        IF KR.MOV EQ #1 THEN
001364'                                                                 CMP     KR.MOV,#1
(6)  001364' 026727  176412  000001                                     BNE     50042$
(9)  001372' 001010
                                                CALL PRRLOC IN <R0,R3>
001374'                                                                 MOV     R5,-(SP)
(3)  001374' 010546                                                     MOV     R3,-(R5)
(5)  001376' 010345                                                     MOV     R0,-(R5)
(4)  001400' 010045                                                     JSR     PC,PRRLOC
(3)  001402' 004767  000000G                                           MOV     (SP)+,R5
(3)  001406' 012605
                                                LET DT.ADDR(R0) := R3
001410'                                                                 MOV     R3,DT.ADDR(R0)
(4)  001410' 010360  000042
                                        ENDIF
001414'                                                         50042$:
(4)  001414'
                                ;+
                                ; IF THIS IS LOCK MODE OR KT OFF, CLEAR THE RELMODE BIT
                                ;-
                                        IF KR.FLG EQ #1 OR #KTSTAT NOTSETIN DT.ST0(R0) THEN
001414'                                                                 CMP     KR.FLG,#1
(6)  001414' 026727  176360  000001                                     BEQ     50043$
(8)  001422' 001404                                                     BIT     #KTSTAT,DT.ST0(R
(6)  001424' 032760  000020  000010                                     BNE     50044$
(9)  001432' 001004                                             50043$:
(6)  001434'
                                                LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RELMODE
001434'                                                                 BIC     #RELMODE,DT.ST0(
(6)  001434' 042760  020000  000010
                                        ELSE
001442'                                                                 BR      50045$
(4)  001442' 000403                                             50044$:
(3)  001444'
                                ;+
                                ; NOT LOCK MODE
                                ;-
                                                LET DT.ST0(R0) := DT.ST0(R0) SET.BY #RELMODE
001444'                                                                 BIS     #RELMODE,DT.ST0(
(6)  001444' 052760  020000  000010
                                        ENDIF
001452'                                                         50045$:
(4)  001452'
                                ;+
                                ;SETUP THE WRITE BUFFER LIMITS
                                ;-
                                        CALL WBFLIM IN <R0>
001452'                                                                 MOV     R5,-(SP)
(3)  001452' 010546
```

```
 (4)  001454' 010045                                                                    MOV     R0,-(R5)
 (3)  001456' 004767  000000G                                                          JSR     PC,WBFLIM
 (3)  001462' 012605                                                                   MOV     (SP)+,R5
1161
1162                                ;+
1163                                ; IF 22 BIT ADDRESSING... MAP THE UNIBUS MAP REGS.
1164                                ;-
1165
1166  001464'                               IF #ADDR22 SETIN DT.CF0(R0) THEN           BIT     #ADDR22,DT.CF0(R
 (6)  001464' 032760  001000  000014                                                  BEQ     50046$
 (9)  001472' 001405
1167
1168  001474'                                     CALL UNIMAP IN <R0>                 MOV     R5,-(SP)
 (3)  001474' 010546                                                                   MOV     R0,-(R5)
 (4)  001476' 010045                                                                   JSR     PC,UNIMAP
 (3)  001500' 004767  000000G                                                          MOV     (SP)+,R5
 (3)  001504' 012605
1169
1170  001506'                               ENDIF                                  50046$:
 (4)  001506'
1171
1172
1173                                ;+
1174                                ; SET RUNMODE BIT.... AND DONE
1175                                ;-
1176
1177  001506'                               LET DT.ST0(R0) := DT.ST0(R0) SET.BY #RUNMODE  BIS     #RUNMODE,DT.ST0(
 (6)  001506' 052760  100000  000010
1178
1179
1180  001514'                           INLINE <1$:>                                 1$:
 (2)  001514'
1181  001514'                           CALL RESREG                                  JSR     PC,RESREG
 (3)  001514' 004767  000000G
1182
1183  001520'                           ENDRTN                                 50000$:
 (3)  001520'                                                                 50001$:
 (3)  001520'                                                                         RTS     PC
 (2)  001520' 000207
1184
1185          000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
APTSEL= ****** G
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BE.MCN= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKCHK= ****** G
CLKPRE= 000001
CLRTIM= ****** G
CM.ADR= ****** G
CM.ARG= ****** G
CM.NBA= ****** G
CM.NUM= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KRUN    000222RG
KRUNL   000232RG
KR.BIG  000061R
KR.FLG  000000R
KR.KT   000127R
KR.MAP  000175R
KR.MOD  000150R
KR.MOV  000002R
KR.SML  000004R
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.COD= ****** G
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PA.CNT= ****** G
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRRLOC= ****** G
PR.RL1= ****** G
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
RUNSUB  000240R
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
```

```
SENDLS= 177777          TMPIO = 000002          WBUFRQ= 000140          $F$NO = 000403          $TSK0 = 050046
SETVEC= ****** G        TQOVF = 000002          WBUFSZ= 000142          $F$OR = 000320          $TSK1 = 050037
SOFCNT= 000042          UIPAR0= 177640          WDFR  = 000116          $F$RTI= 000350          $TSK2 = 050041
SOFPAS= 000046          UIPAR1= 177642          WDTO  = 000114          $F$RTN= 000300          $$ARGC= 000004
SPACE = 000040          UIPAR2= 177644          WSTBUS= ****** G        $F$SEL= 000140          $$BYTE= 000403
SPOINT= 000032          UIPAR3= 177646          WTINRE= 000352          $F$THE= 000330          $$CASE= 000000
SPVALU= 002200          UIPAR4= 177650          WTWHMI= 000222          $F$TRU= 000404          $$DST = 000000
SR0   = 177572          UIPAR5= 177652          XFLAG = 000005          $F$UNT= 000130          $$ELOC= 000402
SR1   = 177574          UIPAR6= 177654          XOFF  = 000023          $F$WHI= 000120          $$ERFL= 000000
SR2   = 177576          UIPAR7= 177656          XON   = 000021          $F$YES= 000402          $$FLAG= 000001
SR3   = 172516          UIPDR0= 177600          $BGNLE= 177777          $IFLEV= 177777          $$FROM= 000000
STAT  = 000026          UIPDR1= 177602          $ERFLG= 000400          $ISK0 = 000001          $$LOC = 001472R
STATBI= 064757          UIPDR2= 177604          $F$AND= 000310          $LOCTA= 177777          $$LOCN= 000000
STAT1 = 000027          UIPDR3= 177606          $F$BAD= 000401          $LSTIN= 000001          $$REG = 177777
SUSPND= 000001          UIPDR4= 177610          $F$BLA= 000170          $LSTTA= 000001          $$RETU= 000000
SVR0  = 000062          UIPDR5= 177612          $F$CAS= 000150          $NESTL= 177777          $$RTN1= 050000
SVR1  = 000064          UIPDR6= 177614          $F$DEC= 000220          $NSK0 = 000300          $$RTN2= 050001
SVR2  = 000066          UIPDR7= 177616          $F$DO = 000340          $NSK1 = 000110          $$SRC = 000000
SVR3  = 000070          UNIMAP= ****** G        $F$FAL= 000405          $NSK2 = 000110          $$TGSV= 000000
SVR4  = 000072          WASADR= 000104          $F$GOO= 000400          $SAVLE= 177777          $$TGS1= 000000
SVR5  = 000074          WBFLIM= ****** G        $F$IF = 000110          $SSK0 = 050037          $$TGS2= 000000
SVR6  = 000076          WBSTAT= 000040          $F$INC= 000210          $TAGLE= 177777          $$TO  = 000000
SYSCNT= 000052          WBUFEA= 000136          $F$LOO= 000200          $TAGNU= 050047          $$$TAG= 050000
SYSERR= 000100          WBUFPA= 000134          $F$NAM= 000160          $TEMP = 000300          .     = 001522R

. ABS.  000000    000
        001522    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KRUN,DSKZ:KRUN=SPMAC/ML,EQUATE,KRUN
 RUN-TIME: 29 19 .4 SECONDS
 RUN-TIME RATIO: 82/49=1.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KRUNS   PROCESS THE 'RUN' COMMAND FOR SMALL MONITORS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KRUNS
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS FOR NON-KT MONITORS ONLY.
                        ;       THIS ROUTINE PROCESSES THE 'RUN' KEYBOARD
                        ;       COMMAND. IF AN ARGUMENT IS INCLUDED,AN ERROR MSG. IS ISSUED.
                        ;       ALL OPTION MODULES HEADERS WILL BE INITIALIZED. THE SYS-
                        ;       TEM ERROR COUNT AND  POWER FAIL COUNT
                        ;       WILL BE INITIALIZED. THE WRITE BUFFER LIMITS
                        ;       WILL BE SET UP.
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;       2. DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.MLST
                        ;       2. DT.STO
                        ;       3. DT.CFO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.STO
                        ;       2. DT.KBRSP
                        ;       3. DT.ESIZ
                        ;       4. DT.PFL
                        ;       5. DT.EXS
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. BE.MCNT
                        ;       2. MD.COD
                        ;       3. PA.CNT
                        ;       4. CM.RUN
                        ;       5. CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK
                        ;       2. CLRTIM
                        ;       3. SAVREG
                        ;       4. RESREG
                        ;       5. WBFLIM
                        ;       6. WSTBUS
                        ;       7. CLKCHK
                        ;       8. APTSEL
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
```

```
                        ;       CALL KRUN IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = DTABLE ADDRESS
                        ;          BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KRUN

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTTAG=1
        000001                  $LSTIN=1
                                ;***************
                                ;
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KRUN            ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CLKCHK          ;SEE IF A CLOCK IS HERE
                                .GLOBL  APTSEL          ;APT SELECT ROUTINE
                                .GLOBL  MD.COD          ;CODE
                                .GLOBL  MSGDEQ          ;MESSAGE DEQUE
                                .GLOBL  CM.RUN          ;'ILLEGAL COMMAND IN RUN MODE'
                                .GLOBL  CM.ARG          ;'ILLEGAL OR MISSING ARGUMENT'
                                .GLOBL  CLRTIM          ;CLEAR TIME TABLE
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  PA.CNT          ;NUMBER OF PARITY ERRORS
                                .GLOBL  BE.MCNT         ;MONITOR SYSTEM ERROR COUNT
                                .GLOBL  WBFLIM          ;ESTABLISH WRITE BUFFER LIMITS
                                .GLOBL  WSTBUS          ;WRITE WORST CASE UNIBUS PATTERN
                                ;
                                ;****************
                                ;
                                ; GLOBAL EQUATES:
                                ;
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 047045  020117  047515  KR.MOD: .ASCIZ  /%NO MODULES SELECTED%/
000006' 052504  042514  020123
000014' 042523  042514  052103
000022' 042105  000045
                                .EVEN
                                ;
                                ;****************
```

```
                                .SBTTL  KRUN ROUTINE

                                ROUTINE KRUN <DTADR,BUFPTR>
000026'                                                                 KRUN:
(2)  000026'


                                ;+
                                ; SAVE REGISTERS AND SAVE DTABLE ADDR AND BUFFER POINTER
                                ;-

                                CALL SAVREG
000026'                                                                 JSR     PC,SAVREG
(3)  000026' 004767  000000G
                                LET R0 := DTADR(R5)
000032'                                                                 MOV     DTADR(R5),R0
(4)  000032' 016500  000000
                                LET R1 := BUFPTR(R5)
000036'                                                                 MOV     BUFPTR(R5),R1
(4)  000036' 016501  000002
                                ;+
                                ; IF IN AUTO OR APTPRES MODE, DON'T CHECK ANY ARGUMENTS - JUST GO TO WORK
                                ;-

                                IF #AUTO SETIN DT.CF0(R0) OR #APTPRES SETIN DT.CF0(R0) THEN
000042'                                                                 BIT     #AUTO,DT.CF0(R0)
(6)  000042' 032760  000010  000014
(8)  000050' 001004                                                     BNE     50002$
(6)  000052' 032760  000200  000014                                    BIT     #APTPRES,DT.CF0(
(9)  000060' 001401                                                     BEQ     50003$
(6)  000062'                                                    50002$:
                                        INLINE <BR      339$>
000062'                                                                 BR      339$
(2)  000062' 000431
                                ENDIF
000064'                                                         50003$:
(4)  000064'


                                ;+
                                ; SEE IF IN RUNMODE - IF SO STUFF ERROR AND SCRAM
                                ;-

                                IF #RUNMODE SETIN DT.ST0(R0) THEN
000064'                                                                 BIT     #RUNMODE,DT.ST0(
(6)  000064' 032760  100000  000010
(9)  000072' 001405                                                     BEQ     50004$
                                        LET DT.KBRSP(R0) := #CM.RUN
000074'                                                                 MOV     #CM.RUN,DT.KBRSP
(4)  000074' 012760  000000G 000022
                                      ; INLINE <JMP     1$>
000102'                                                                 JMP     1$
(2)  000102' 000167  000424

                                ENDIF
000106'                                                         50004$:
(4)  000106'


                                ;+
                                ; WE ARE NOT IN RUNMODE....ALL IS OK....GET AN ARGUMENT IF ANY.....
                                ;-
```

```
                                ;+
                                ; IF NO ARGS, JUMP AHEAD
                                ;-


000106'                         IF.ERROR THEN
(6)  000126' 103002                                                     BCC     50005$

                                        INLINE <BR      339$>
000130'                                                                 BR      339$
(2)  000130' 000406

                                ELSE
000132'                                                                 BR      50006$
(4)  000132' 000405
(3)  000134'                                                    50005$:


                                ;+
                                ; THERE ARE ARGUMENTS.....TOO BAD - NOT ALLOWED IN THIS MONITOR....
                                ; STUFF ERROR MSG AND SEE YA.....NEXT TIME TYPE 'RUN' ONLY
                                ;-
000134'                                 LET DT.KBRSP(R0) := #CM.ARG
(4)  000134' 012760  000000G 000022                                     MOV     #CM.ARG,DT.KBRSP
000142'                                 INLINE <JMP     1$>
(2)  000142' 000167  000364                                             JMP     1$
000146'                         ENDIF
(4)  000146'                                                    50006$:

000146'                         INLINE <339$:>
(2)  000146'                                                            339$:

                                ;+
                                ; IF UNDER APT, USE APT'S TABLE TO SELECT THE MODULES
                                ; AND SET UP DEVICE COUNT
                                ;-

000146'                         IF #APTPRES SETIN DT.CF0(R0) THEN
(6)  000146' 032760  000200  000014                                    BIT     #APTPRES,DT.CF0(
(9)  000154' 001405                                                     BEQ     50007$
000156'                                 CALL APTSEL IN <R0>
(3)  000156' 010546                                                     MOV     R5,-(SP)
(4)  000160' 010045                                                     MOV     R0,-(R5)
(3)  000162' 004767  000000G                                           JSR     PC,APTSEL
(3)  000166' 012605                                                     MOV     (SP)+,R5
000170'                         ENDIF
(4)  000170'                                                    50007$:


                                ;+
                                ; GET ADDR OF MODULE LIST
                                ;-
```

```
                                LET R1 := DT.MLST(R0)                                   MOV     DT.MLST(R0),R1
(4)  000170' 016001  000032
                                ;+
                                ; FLAG
                                ;-
                                LET R4 := #0                                            CLR     R4
(4)  000174' 005004
                                ;+
                                ; WHILE NOT AT END OF MODULE LIST...SEE IF A MODULE IS SELECTED...
                                ; IF SO, INCREMENT A COUNT
                                ;-
                                WHILE (R1) NE #0 DO                             50010$:
(4)  000176'                                                                            TST     (R1)
(6)  000176' 005711                                                                     BEQ     50011$
(9)  000200' 001407
                                        LET R2 := (R1)+                                 MOV     (R1)+,R2
(4)  000202' 012102
                                        IF #BIT14 SETIN STAT(R2) THEN                   BIT     #BIT14,STAT(R2)
(6)  000204' 032762  040000  000026                                                    BEQ     50012$
(9)  000212' 001401
                                                LET R4 := R4 + #1                       INC     R4
(6)  000214' 005204
                                        ENDIF                                   50012$:
(4)  000216'
                                ENDDO                                                   BR      50010$
(4)  000216' 000767                                                             50011$:
(3)  000220'
                                ;+
                                ; IF R4 IS ZERO...NO MODULES SELECTED.....STUFF ERROR AND GOODBYE...
                                ;-
                                IF R4 EQ #0 THEN                                        TST     R4
(6)  000220' 005704                                                                     BNE     50013$
(9)  000222' 001005
                                        LET DT.KBRSP(R0) := #KR.MOD                     MOV     #KR.MOD,DT.KBRSP
(4)  000224' 012760  000000' 000022
                                        INLINE <JMP     1$>                             JMP     1$
(2)  000232' 000167  000274
                                ENDIF                                           50013$:
(4)  000236'
                                ;+
                                ; !!!!! GO TO WORK !!!!!!    GET THE MODULE HEADERS CLEANED UP !
                                ; BUT FIRST PRINT THE CR - IT TAKES SO LONG TO WRITE WSTBUS
                                ; PATTERN THAT THE CR WOULD NOT BE ECHOED FOR A LONG TIME
                                ; AND IT MAY CAUSE WORRY - HENCE THE CALL TO MSGDEQ WILL PRINT
```

```
                                ; IT RIGHT AWAY....
                                ;-



                                REPEAT
                                                                        50014$:
                                        CALL MSGDEQ IN <R0>
                                                                                MOV     R5,-(SP)
                                                                                MOV     R0,-(R5)
                                                                                JSR     PC,MSGDEQ
                                                                                MOV     (SP)+,R5
                                UNTIL MD.COD EQ #MSGNUL
                                                                                CMP     MD.COD,#MSGNUL
                                                                                BNE     50014$



                                ;+
                                ; IF THIS IS THE SCRAWNY MONITOR... DON'T EVEN CONSIDER THE CLOCK
                                ; STUFF
                                ;-

                                IF #NCPUOP NOTSETIN DT.CF0(R0) THEN
                                                                                BIT     #NCPUOP,DT.CF0(R
                                                                                BNE     50015$
                                ;+
                                ;IF THE CLOCK IS PRESENT, CLEAR CLOCK TABLE....SEE IF IT IS...
                                ;-

                                        CALL CLKCHK IN <R0>
                                                                                MOV     R5,-(SP)
                                                                                MOV     R0,-(R5)
                                                                                JSR     PC,CLKCHK
                                                                                MOV     (SP)+,R5

                                        IF #CLKPRES SETIN DT.CF0(R0) THEN
                                                                                BIT     #CLKPRES,DT.CF0(
                                                                                BEQ     50016$

                                                CALL CLRTIM IN <R0>
                                                                                MOV     R5,-(SP)
                                                                                MOV     R0,-(R5)
                                                                                JSR     PC,CLRTIM
                                                                                MOV     (SP)+,R5

                                        ENDIF
                                                                        50016$:
                                ENDIF
                                                                        50015$:



                                ;+
                                ; GET MODULE LIST
                                ;-
                                LET R1 := DT.MLST(R0)
                                                                                MOV     DT.MLST(R0),R1
```

```
                                ;+
                                ; WHILE NOT AT END OF MODULE LIST, CLR OUT MODULE HEADER LOCATIONS
                                ; AND COPY CONFIG WORD 0 INTO RES1.
                                ;-

000330'                         WHILE (R1) NE #0 DO                                     50017$:
(4)  000330'                                                                                    TST     (R1)
(6)  000330' 005711                                                                             BEQ     50020$
(9)  000332' 001450
                                        LET R2 := (R1)+                                         MOV     (R1)+,R2
000334'
(4)  000334' 012102
                                        LET PASCNT(R2) := #0                                    CLR     PASCNT(R2)
000336'
(4)  000336' 005062  000034             LET ICOUNT(R2) := #0                                    CLR     ICOUNT(R2)
000342'
(4)  000342' 005062  000040             LET SOFCNT(R2) := #0                                    CLR     SOFCNT(R2)
000346'
(4)  000346' 005062  000042             LET HRDCNT(R2) := #0                                    CLR     HRDCNT(R2)
000352'
(4)  000352' 005062  000044             LET SOFPAS(R2) := #0                                    CLR     SOFPAS(R2)
000356'
(4)  000356' 005062  000046             LET HRDPAS(R2) := #0                                    CLR     HRDPAS(R2)
000362'
(4)  000362' 005062  000050             LET SYSCNT(R2) := #0                                    CLR     SYSCNT(R2)
000366'
(4)  000366' 005062  000052             LET XFLAG(R2) :B= #0                                    CLRB    XFLAG(R2)
000372'
(4)  000372' 105062  000005             LET RES1(R2) := DT.CF0(R0)                              MOV     DT.CF0(R0),RES1(
000376'
(4)  000376' 016062  000014  000056

                                ;+
                                ; CLEAR ACTIVE AND DROPPED BIT IN STAT WORD OF MODULE
                                ;-

                                        LET STAT(R2) := STAT(R2) CLR.BY #BIT13!ACTBIT           BIC     #BIT13!ACTBIT,ST
000404'
(6)  000404' 042762  024000  000026


                                ;+
                                ; IF AN IOMODR(P)(X) CLEAR CDATA ERROR AND WORD COUNTS
                                ;-

                                        IF #IOMODR SETIN STAT(R2) OR #IOMODP SETIN STAT(R2) OR #IOMODX SETIN STAT(R2) TH
000412'                                                                                         BIT     #IOMODR,STAT(R2)
(6)  000412' 032762  112000  000026                                                            BNE     50021$
(8)  000420' 001010
```

```
(6)  000422' 032762  102000  000026                                                    BIT     #IOMODP,STAT(R2)
(8)  000430' 001004                                                                    BNE     50021$
(6)  000432' 032762  110000  000026                                                    BIT     #IOMODX,STAT(R2)
(9)  000440' 001404                                                                    BEQ     50022$
(6)  000442'                                                                  50021$:
813
814
815  000442'                                  LET CDERCT(R2) := #0
(4)  000442' 005062  000146                                                            CLR     CDERCT(R2)
816  000446'                                  LET CDWDCT(R2) := #0
(4)  000446' 005062  000144                                                            CLR     CDWDCT(R2)
817  000452'                              ENDIF
(4)  000452'                                                                  50022$:
818
819  000452'                          ENDDO
(4)  000452' 000726                                                                    BR      50017$
(3)  000454'                                                                  50020$:
820
821
822
823                                   ;+
824                                   ; CLEAR SOME OTHER GOODIES.....THAT IS, SYSTEM PASS COUNT...MONITOR SYS ERRORS
825                                   ; PARITY ERRORS, AND TOTAL SYS ERRORS
826                                   ;-
827
828  000454'                          LET @DT.SYP(R0) := #0
(4)  000454' 005070  000072                                                            CLR     @DT.SYP(R0)
829  000460'                          LET DT.PFL(R0) := #0
(4)  000460' 005060  000062                                                            CLR     DT.PFL(R0)
830  000464'                          LET BE.MCNT := #0
(4)  000464' 005067  000000G                                                           CLR     BE.MCNT
831  000470'                          LET PA.CNT := #0
(4)  000470' 005067  000000G                                                           CLR     PA.CNT
832  000474'                          LET DT.EXS(R0) := #0
(4)  000474' 005060  000060                                                            CLR     DT.EXS(R0)
833
834
835
836                                   ;+
837                                   ; WRITE WORST CASE UNIBUS PATTERN
838                                   ;-
839
840  000500'                          CALL WSTBUS IN <R0>
(3)  000500' 010546                                                                    MOV     R5,-(SP)
(4)  000502' 010045                                                                    MOV     R0,-(R5)
(3)  000504' 004767  000000G                                                           JSR     PC,WSTBUS
(3)  000510' 012605                                                                    MOV     (SP)+,R5
841
842
843
844                                   ;+
845                                   ;SETUP THE WRITE BUFFER LIMITS
846                                   ;-
847
848  000512'                          CALL WBFLIM IN <R0>
(3)  000512' 010546                                                                    MOV     R5,-(SP)
```

```
(4)  000514' 010045                                                          MOV     R0,-(R5)
(3)  000516' 004767  000000G                                                 JSR     PC,WBFLIM
(3)  000522' 012605                                                          MOV     (SP)+,R5
849
850
851                                ;+
852                                ; SET RUNMODE BIT.... AND DONE
853                                ;-
854
855  000524'                       LET DT.ST0(R0) := DT.ST0(R0) SET.BY #RUNMODE
(6)  000524' 052760  100000  000010                                          BIS     #RUNMODE,DT.ST0(
856
857
858  000532'                       INLINE <1$:>
(2)  000532'                                                                 1$:
859  000532'                       CALL RESREG
(3)  000532' 004767  000000G                                                 JSR     PC,RESREG
860
861  000536'                       ENDRTN
(3)  000536'                                                          50000$:
(3)  000536'                                                          50001$:
(2)  000536' 000207                                                          RTS     PC
862
863          000001               .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
APTSEL= ****** G
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BE.MCN= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKCHK= ****** G
CLKPRE= 000001
CLRTIM= ****** G
CM.ARG= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KRUN    000026RG
KR.MOD  000000R
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.COD= ****** G
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PA.CNT= ****** G
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
```

```
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBFLIM= ****** G
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WSTBUS= ****** G
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000120
$NSK2 = 000110
$SAVLE= 177777
$SSK0 = 050020
$TAGLE= 177777
$TAGNU= 050023
$TEMP = 000300
$TSK0 = 050017
$TSK1 = 050020
$TSK2 = 050022
$$ARGC= 000004
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000440R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000540R

. ABS.  000000     000
        000540     001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:KRUNS,DSKZ:KRUNS=SPMAC/ML,EQUATE,KRUNS
RUN-TIME: 18 9 .4 SECONDS
RUN-TIME RATIO: 54/27=1.9
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KSUM   PROCESS THE 'SUM' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KSUM
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'SUM' KEYBOARD COMMAND. IT WILL
                        ;       SEARCH THE MODULE LIST AND ENQUEUE THE SUM MESSAGE FOR
                        ;       EACH MODULE. IF A MODULE NAME IS SPECIFIED, A SUMMARY
                        ;       MESSAGE IS OUTPUTTED FOR THAT
                        ;       MODULE ONLY AND NO HEADER OR TRAILER MESSAGES
                        ;       ARE OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST, DT.STO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG, CM.BADNAME
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK - CHECK FOR JUNK ARGUMENTS
                        ;       2. ENQTQ - ENQUEUE A MESSAGE
                        ;       3. NAMCHK - VERIFY A MODULE NAME
                        ;       4. SAVREG - SAVE REGISTERS
                        ;       5. RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KSUM IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KSUM

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTTAG=1
        000001                  $LSTIN=1


                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  KSUM                    ;MODULE ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  ENQTQ           ;ENQUEUE A MESSAGE
                                .GLOBL  CM.ARG          ;'ILLEGAL ARGUMENT' ERROR MSG
                                .GLOBL  NAMCHK          ;CHECK A MODULE NAME
                                .GLOBL  CM.BADNAME      ;'INVALID MODULE NAME' ERROR MESSAGE
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  KS.HLD:         .WORD   0       ;HOLD ADDRESS OF PROMPT
000002' 041445  042115  000076  KS.CMD: .ASCIZ  /%CMD>/
000010' 041045  054523  000076  KS.RUN: .ASCIZ  /%BSY>/
                                .EVEN
                                ;
```

```
                                .SBTTL  KSUM ROUTINE

                                ROUTINE KSUM <DTADR,BUFPTR>
000016'                                                                 KSUM:
(2)  000016'

                                ;+
                                ; SAVE REGISTERS, DTABLE ADDRESS, AND DECODE BUFFER PTR
                                ;-

                                CALL SAVREG
000016'                                                                         JSR     PC,SAVREG
(3)  000016' 004767  000000G
                                LET R0 := DTADR(R5)
000022'                                                                         MOV     DTADR(R5),R0
(4)  000022' 016500  000000
                                LET R1 := BUFPTR(R5)
000026'                                                                         MOV     BUFPTR(R5),R1
(4)  000026' 016501  000002

                                ;+
                                ; IF IN RUN MODE - STUFF A PROMPT TO BE ENQUEUED LATER
                                ; IN THIS MODULE - IT IS USED BECAUSE THE NORMAL PROMPT
                                ; IS OUTPUTTED BEFORE ANY SUM MESSAGES ARE OUTPUTTED -
                                ; THIS ONE WILL FOLLOW THE SUM MESSAGES....
                                ;-

                                IF #RUNMODE SETIN DT.ST0(R0) THEN
000032'                                                                         BIT     #RUNMODE,DT.ST0(
(6)  000032' 032760  100000  000010                                            BEQ     50002$
(9)  000040' 001404
                                        LET KS.HLD := #KS.RUN
000042'                                                                         MOV     #KS.RUN,KS.HLD
(4)  000042' 012767  000010' 177730
                                ELSE
000050'                                                                         BR      50003$
(4)  000050' 000403
(3)  000052'                                                            50002$:
                                        LET KS.HLD := #KS.CMD
000052'                                                                         MOV     #KS.CMD,KS.HLD
(4)  000052' 012767  000002' 177720
                                ENDIF
000060'                                                                 50003$:
(4)  000060'

                                ;+
                                ; SEE IF ANY ARGUMENTS ARE INCLUDED IN DECODE BUFFER
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>
000060'                                                                         SUB     #1*2,R5
(4)  000060' 162705  000002                                                     MOV     R5,-(SP)
(3)  000064' 010546                                                             MOV     R1,-(R5)
(4)  000066' 010145                                                             JSR     PC,ARGCHK
(3)  000070' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000074' 012605                                                             MOV     (R5)+,R1
(4)  000076' 012501

                                ;+
                                ; IF THERE ARE NO ARGUMENTS DO A FULL SUMMARY (ALL MODULES). GET ADDRESS
                                ; OF MODULE LIST, ENQUEUE THE RUN SUMMARY HEADER MESSAGE, THEN, WHILE
                                ; THE ENTRY IN THE MODULE LIST IS NOT 0, ENQUEUE THE SUMMARY MESSAGE
                                ; FOR THE MODULES. WHEN MODULE LIST ENTRY = 0, ENQUEUE THE TRAILING MESSAGE..
                                ;-
```

```
641
642  000100'                        IF.ERROR THEN                                  BCC     50004$
(6)  000100' 103026
643
644  000102'                                LET R1 := #MSGSMH                      MOV     #MSGSMH,R1
(4)  000102' 012701  000014
645  000106'                                LET R2 := #0                           CLR     R2
(4)  000106' 005002
646  000110'                                LET R3 := #0                           CLR     R3
(4)  000110' 005003
647  000112'                                CALL NQ                                JSR     PC,NQ
(3)  000112' 004767  000176
648
649  000116'                                LET R1 := #MSGSMB                      MOV     #MSGSMB,R1
(4)  000116' 012701  000015
650  000122'                                LET R4 := DT.MLST(R0)                  MOV     DT.MLST(R0),R4
(4)  000122' 016004  000032
651  000126'                                WHILE (R4) NE #0 DO                    50005$: TST     (R4)
(4)  000126'                                                                      BEQ     50006$
(6)  000126' 005714
(9)  000130' 001404
652  000132'                                        LET R3 := (R4)+                MOV     (R4)+,R3
(4)  000132' 012403
653  000134'                                        CALL NQ                        JSR     PC,NQ
(3)  000134' 004767  000154
654  000140'                                ENDDO                                  BR      50005$
(4)  000140' 000772
(3)  000142'                                                                      50006$:
655
656  000142'                                LET R1 := #MSGSMS                      MOV     #MSGSMS,R1
(4)  000142' 012701  000016
657  000146'                                LET R2 := #0                           CLR     R2
(4)  000146' 005002
658  000150'                                CALL NQ                                JSR     PC,NQ
(3)  000150' 004767  000140
659
660  000154'                        ELSE                                           BR      50007$
(4)  000154' 000442                                                               50004$:
(3)  000156'
661
662                                 ;+
663                                 ; THERE IS AN ARGUMENT - GO CHECK IT OUT
664                                 ;-
665
666  000156'                                CALL NAMCHK IN <R0,R1> OUT <R2,R1>     SUB     #2*2,R5
(4)  000156' 162705  000004                                                       MOV     R5,-(SP)
(3)  000162' 010546                                                               MOV     R1,-(R5)
(5)  000164' 010145                                                               MOV     R0,-(R5)
(4)  000166' 010045                                                               JSR     PC,NAMCHK
(3)  000170' 004767  000000G                                                      MOV     (SP)+,R5
(3)  000174' 012605                                                               MOV     (R5)+,R2
(4)  000176' 012502                                                               MOV     (R5)+,R1
(4)  000200' 012501
667  000202'                                IF.NO.ERROR THEN                       BCS     50010$
(6)  000202' 103424
668
```

```
                                ;+
                                ; MODULE NAME IS O.K. - NOW JUST VERIFY THERE IS NO JUNK ARGUMENT
                                ;-

000204'                                 CALL ARGCHK IN <R1> OUT <R1>
(4) 000204' 162705  000002                                              SUB     #1*2,R5
(3) 000210' 010546                                                      MOV     R5,-(SP)
(4) 000212' 010145                                                      MOV     R1,-(R5)
(3) 000214' 004767  000000G                                             JSR     PC,ARGCHK
(3) 000220' 012605                                                      MOV     (SP)+,R5
(4) 000222' 012501                                                      MOV     (R5)+,R1
000224'                                 IF.ERROR THEN
(6) 000224' 103007                                                      BCC     50011$

                                ;+
                                ; NO JUNK ARGUMENTS.... ENQUEUE THE MESSAGE FOR THIS MODULE
                                ;-

000226'                                     LET R1 := #MSGSMB
(4) 000226' 012701  000015                                              MOV     #MSGSMB,R1
000232'                                     LET R3 := R2
(4) 000232' 010203                                                      MOV     R2,R3
000234'                                     LET R2 := #0
(4) 000234' 005002                                                      CLR     R2
000236'                                     CALL NQ
(3) 000236' 004767  000052                                              JSR     PC,NQ

000242'                                 ELSE
(4) 000242' 000403                                                      BR      50012$
(3) 000244'                                                         50011$:

                                ;+
                                ; THERE IS A JUNK ARGUMENT IN DECODE BUFFER... STUFF ERROR MSG...
                                ;-
000244'                                     LET DT.KBRSP(R0) := #CM.ARG
(4) 000244' 012760  000000G 000022                                      MOV     #CM.ARG,DT.KBRSP

000252'                                 ENDIF
(4) 000252'                                                         50012$:

000252'                             ELSE
(4) 000252' 000403                                                      BR      50013$
(3) 000254'                                                         50010$:

                                ;+
                                ; BAD MODULE NAME....STUFF ERROR MSG....
                                ;-

000254'                                 LET DT.KBRSP(R0) := #CM.BADNAME
(4) 000254' 012760  000000G 000022                                      MOV     #CM.BADNAME,DT.K

000262'                                 ENDIF
(4) 000262'                                                         50013$:

000262'                         ENDIF
(4) 000262'                                                         50007$:
```

```
                                ;+
                                ; IF NO ERRORS - ENQUEUE A PROMPT
                                ;-

000262'                         IF  DT.KBRSP(R0) EQ #0 THEN
(6)  000262' 005760  000022                                                 TST     DT.KBRSP(R0)
(9)  000266' 001007                                                         BNE     50014$
000270'                                 LET R1 := #MSGSTD
(4)  000270' 012701  000000                                                 MOV     #MSGSTD,R1
000274'                                 LET R2 := KS.HLD
(4)  000274' 016702  177500                                                 MOV     KS.HLD,R2
000300'                                 LET R3 := #0
(4)  000300' 005003                                                         CLR     R3
000302'                                 CALL NQ
(3)  000302' 004767  000006                                                 JSR     PC,NQ
000306'                         ENDIF
(4)  000306'                                                        50014$:

                                ;+
                                ; CLEAN UP AND GOODBYE
                                ;-

000306'                         CALL RESREG
(3)  000306' 004767  000000G                                               JSR     PC,RESREG

000312'                         ENDRTN
(3)  000312'                                                        50000$:
(3)  000312'                                                        50001$:
(2)  000312' 000207                                                         RTS     PC

                                ;+
                                ; THIS IS THE ACTUAL CALL TO THE TYPE-QUEUE ENQUEUEING ROUTINE.
                                ;-

000314'                         ROUTINE NQ
(2)  000314'                                                        NQ:

000314'                         CALL ENQTQ IN <R0,R1,R2,R3,#0>
(3)  000314' 010546                                                         MOV     R5,-(SP)
(8)  000316' 012745  000000                                                 MOV     #0,-(R5)
(7)  000322' 010345                                                         MOV     R3,-(R5)
(6)  000324' 010245                                                         MOV     R2,-(R5)
(5)  000326' 010145                                                         MOV     R1,-(R5)
(4)  000330' 010045                                                         MOV     R0,-(R5)
(3)  000332' 004767  000000G                                               JSR     PC,ENQTQ
(3)  000336' 012605                                                         MOV     (SP)+,R5

000340'                         ENDRTN
(3)  000340'                                                        50000$:
(3)  000340'                                                        50001$:
(2)  000340' 000207                                                         RTS     PC
          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CM.BAD= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KSUM    000016RG
KS.CMD  000002R
KS.HLD  000000R
KS.RUN  000010R
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK= ****** G
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NQ      000314R
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
```

```
SVR1  = 000064          UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$BYTE= 000403
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$CASE= 000000
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$DST = 000000
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $ISK2 = 000001          $$ELOC= 000402
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $LOCTA= 177777          $$ERFL= 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LSTIN= 000001          $$FLAG= 000001
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $LSTTA= 000001          $$FROM= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $NESTL= 177777          $$LOC = 000266R
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK0 = 000300          $$LOCN= 000000
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK1 = 000110          $$REG = 177777
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $NSK2 = 000110          $$RETU= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $NSK3 = 000110          $$RTN1= 050000
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $SAVLE= 177777          $$RTN2= 050001
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $SSK0 = 050006          $$SRC = 000000
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $TAGLE= 177777          $$TGSV= 000000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TAGNU= 050002          $$TGS1= 000000
UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $TEMP = 000300          $$TGS2= 000000
UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $TSK0 = 050014          $$TO  = 000005
UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $TSK1 = 050013          $$$TAG= 050000
UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $TSK2 = 050012          .     = 000342R
UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$ARGC= 000000

. ABS.  000000     000
        000342     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KSUM,DSKZ:KSUM=SPMAC/ML,EQUATE,KSUM
 RUN-TIME: 15 5 .4 SECONDS
 RUN-TIME RATIO: 56/21=2.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KSWR   PROCESS THE 'SWR' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KSWR
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'SWR' KEYBOARD COMMAND.
                        ;       IF NO ARGUMENT IS FOUND IN THE DECODE BUFFER, THE CONTENTS
                        ;       OF THE SOFTWARE SWITCH REGISTER ARE OUTPUTTED. THE NUMBER
                        ;       IS VERIFIED TO BE OCTAL AND 16 BITS OR LESS IN LENGTH. IF
                        ;       THE NUMBER IS VALID, IT WILL REPLACE THE CONTENTS OF THE
                        ;       SOFTWARE SWITCH REGISTER AND THE NEW CONTENTS WILL BE OUTPUTTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DT.KBRSP
                        ;       2. DT.SWR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ARGCHK - VERIFY AN ARGUMENT
                        ;       2. BOA16 - BINARY TO OCTAL ASCII CONVERSION
                        ;       3. SAVREG - SAVE REGISTERS
                        ;       4. RESREG - RESTORE REGISTERS
                        ;       5. NUMCHK - CHECK FOR VALID NUMBER
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KSWR IN <DTADR,BUFPTR>
                        ;       WHERE:  DTADR = ADDRESS OF DTABLE
                        ;               BUFPTR = CMD DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KSWR

                                .MCALL  STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KSWR            ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONVERSION
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  NUMCHK          ;CHECK AN OCTAL ASCII NUMBER
                                .GLOBL  CM.ARG          ;"INVALID OR MISSING ARGUMENT" MESSAGE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 053523  027522    040   KS.MG1: .ASCII  'SWR/ '         ;SWR MESSAGE
000005' 000006                  KS.MG2: .BLKB   6               ;HOLD ASCII DT.SWR CONTENTS
000013'    045     000                  .ASCIZ  /%/
        000016'                 .EVEN
                                ;****************
```

```
                                .SBTTL  KSWR ROUTINE


                                ROUTINE KSWR <DTADR,BUFPTR>
000016'                                                                 KSWR:
                                ;+
                                ; SAVE REGISTERS, DATA TABLE ADDRESS, AND DECODE BUFFER POINTER
                                ;-

                                CALL SAVREG
000016' 004767  000000G                                                 JSR     PC,SAVREG
                                LET R0 := DTADR(R5)
000022' 016500  000000                                                  MOV     DTADR(R5),R0
                                LET R1 := BUFPTR(R5)
000026' 016501  000002                                                  MOV     BUFPTR(R5),R1
                                ;+
                                ; GET TO FIRST ARGUMENT
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>
000032' 162705  000002                                                  SUB     #1*2,R5
000036' 010546                                                          MOV     R5,-(SP)
000040' 010145                                                          MOV     R1,-(R5)
000042' 004767  000000G                                                 JSR     PC,ARGCHK
000046' 012605                                                          MOV     (SP)+,R5
000050' 012501                                                          MOV     (R5)+,R1


                                ;+
                                ; IF IT'S NOT A CARRIAGE RETURN, VERIFY THAT IT IS NUMBER
                                ;-

                                IF.NO.ERROR THEN
000052' 103435                                                          BCS     50002$
                                        CALL NUMCHK IN  <R0,R1> OUT <R1,R2>
000054' 162705  000004                                                  SUB     #2*2,R5
000060' 010546                                                          MOV     R5,-(SP)
000062' 010145                                                          MOV     R1,-(R5)
000064' 010045                                                          MOV     R0,-(R5)
000066' 004767  000000G                                                 JSR     PC,NUMCHK
000072' 012605                                                          MOV     (SP)+,R5
000074' 012501                                                          MOV     (R5)+,R1
000076' 012502                                                          MOV     (R5)+,R2


                                ;+
                                ; IF THERE IS AN ERROR IN THE NUMBER, LEAVE
                                ;-

                                        IF.ERROR THEN
000100' 103002                                                          BCC     50003$
                                                INLINE <BR      1$>
000102' 000434                                                          BR      1$

                                        ELSE
000104'
```

```
(4)  000104' 000420                                                                  BR      50004$
(3)  000106'                                                                 50003$:
633
634                                 ;+
635                                 ; NUMBER OK, VERIFY NO JUNK ARGUMENTS EXIST
636                                 ;-
637
638  000106'                                CALL ARGCHK IN <R1> OUT <R1>
(4)  000106' 162705  000002                                                          SUB     #1*2,R5
(3)  000112' 010546                                                                  MOV     R5,-(SP)
(4)  000114' 010145                                                                  MOV     R1,-(R5)
(3)  000116' 004767  000000G                                                         JSR     PC,ARGCHK
(3)  000122' 012605                                                                  MOV     (SP)+,R5
(4)  000124' 012501                                                                  MOV     (R5)+,R1
639
640
641                                 ;+
642                                 ; IF THERE ARE MORE JUNK ARGUMENTS - LEAVE, BUT STUFF ERROR MSG FIRST
643                                 ;-
644
645  000126'                                IF.NO.ERROR THEN
(6)  000126' 103405                                                                  BCS     50005$
646  000130'                                       LET DT.KBRSP(R0) := #CM.ARG
(4)  000130' 012760  000000G 000022                                                  MOV     #CM.ARG,DT.KBRSP
647  000136'                                       INLINE <BR      1$>
(2)  000136' 000416                                                                  BR      1$
648
649  000140'                                ELSE
(4)  000140' 000402                                                                  BR      50006$
(3)  000142'                                                                 50005$:
650
651                                 ;+
652                                 ; NO JUNK ARGUMENTS - SO REPLACE SOFTWARE SWITCH REG WITH
653                                 ; NEW VALUE
654                                 ;-
655
656  000142'                                       LET DT.SWR(R0) := R2
(4)  000142' 010260  000056                                                          MOV     R2,DT.SWR(R0)
657
658  000146'                                ENDIF
(4)  000146'                                                                 50006$:
659  000146'                                ENDIF
(4)  000146'                                                                 50004$:
660
661  000146'                            ENDIF
(4)  000146'                                                                 50002$:
662
663                                 ;+
664                                 ; CONVERT THE NUMBER IN DT.SWR TO ASCII, STUFF MESSAGE
665                                 ; AND RETURN
666                                 ;-
667  000146'                            CALL BOA16 IN <DT.SWR(R0),#KS.MG2>
(3)  000146' 010546                                                                  MOV     R5,-(SP)
(5)  000150' 012745  000005'                                                         MOV     #KS.MG2,-(R5)
(4)  000154' 016045  000056                                                          MOV     DT.SWR(R0),-(R5)
(3)  000160' 004767  000000G                                                         JSR     PC,BOA16
```

```
(3)  000164' 012605                                                    MOV     (SP)+,R5
668
669  000166'                          LET DT.KBRSP(R0) := #KS.MG1
(4)  000166' 012760  000000' 000022                                    MOV     #KS.MG1,DT.KBRSP
670
671                                   ;+
672                                   ; RESTORE REGISTERS AND RETURN.
673                                   ;-
674
675  000174'                          INLINE <1$:>                     1$:
(2)  000174'
676  000174'                          CALL RESREG
(3)  000174' 004767  000000G                                           JSR     PC,RESREG
677
678  000200'                          ENDRTN
(3)  000200'                                                   50000$:
(3)  000200'                                                   50001$:
(2)  000200' 000207                                                    RTS     PC
679          000001                   .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16 = ****** G
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KSWR    000016RG
KS.MG1  000000R
KS.MG2  000005R
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
NUMCHK= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
```

```
SVR4  = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$BYTE= 000000
SVR5  = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$CASE= 000000
SVR6  = 000076          UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$DST = 000000
SYSCNT= 000052          WASADR= 000104          $F$DO = 000340          $ISK2 = 000001          $$ELOC= 000402
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LOCTA= 177777          $$ERFL= 000000
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000001
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NESTL= 177777          $$LOC = 000126R
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK1 = 000110          $$REG = 177777
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $NSK2 = 000110          $$RETU= 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $NSK3 = 000110          $$RTN1= 050000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $SAVLE= 177777          $$RTN2= 050001
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGLE= 177777          $$SRC = 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TAGNU= 050007          $$TGSV= 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TEMP = 000300          $$TGS1= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK0 = 050002          $$TGS2= 000000
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $TSK1 = 050004          $$TO  = 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK2 = 050006          $$$TAG= 050000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$ARGC= 000004          .     = 000202R


. ABS.  000000    000
        000202    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KSWR,DSKZ:KSWR=SPMAC/ML,EQUATE,KSWR
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 41/15=2.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  KTONOF PROCESS KTON, KTOFF KEYBOARD COMMANDS
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       KTONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
                        ;       1. KKTON - TURN THE MEMORY MANAGEMENT HARDWARE ON
                        ;       2. KKTOFF - TURN THE MEMORY MANAGEMENT OFF
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  KKTON  PROCESS KTON KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KKTON
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE "KTON" KEYBOARD COMMAND.
                        ;       IT WILL TEST TO SEE IF KT IS AVAILABLE, AND THAT THE
                        ;       EXERCISER IS NOT IN RUN MODE. IT WILL SET THE KT STATUS
                        ;       BIT IN STATUS WORD 0 AND ENABLE THE KT HARDWARE.
                        ;       IF IN RUN MODE OR NO KT AVAILABLE, THE APPROPRIATE
                        ;       ERROR MESSAGE IS STUFFED INTO THE DATA TABLE'S ADDRESS
                        ;       OF KB CMD RESPONSE.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.CF0
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ST0
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;       CM.RUN
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK  - CHECK FOR EXISTENCE OF AN ARGUMENT
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KKTON IN <DTADR,BUFPTR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  KKTOFF PROCESS KTOFF KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KKTOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE WILL PROCESS THE KTOFF KEYBOARD COMMAND. IT
                        ;       WILL TEST FOR THE PRESENCE OF KT. IF NO KT AVAILABLE, IT
                        ;       WILL STUFF A NO KT MESSAGE. IF KT IS AVAILABLE, A CHECK
                        ;       IS MADE TO SEE IF THE EXERCISER IS IN RUN MODE. IF IN RUN
                        ;       MODE, THE ERROR MESSAGE IS STUFFED. IF NOT IN RUN MODE,
                        ;       THE STATUS WORD WILL BE UPDATED(THE KT BIT CLEARED)
                        ;       AND THE KT TURNED OFF. NOTE: THE MESSAGES ARE STUFFED
                        ;       INTO THE DATA TABLE'S ADDRESS OF KB CMD RESPONSE
                        ;       LOCATION.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;       COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0
                        ;       DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.ST0
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KKTOFF IN <DTADR,CMDBUF>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             CMDBUF = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINTITIONS AND REFERENCES FOR KKTON

                                .MCALL  STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1


                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KKTOFF          ;MODULE ENTRY POINT
                                .GLOBL  KKTON           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  ARGCHK          ;CHECK FOR EXISTENCE OF AN ARGUMENT
                                .GLOBL  CM.ARG          ;'ILLEGAL OR INVALID ARG' MESSAGE
                                .GLOBL  CM.RUN          ;"NOT A VALID COMMAND IN RUN MODE" MESSAGE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 047516  045440  022524  KT.NOKT:        .ASCIZ  /NO KT%/
000006'    000
000007'    113  020124  047117  KT.ON:  .ASCIZ  /KT ON%/
000014' 000045
000016' 052113  047440  043106  KT.OFF: .ASCIZ  /KT OFF%/
000024' 000045
000026'    000                  KT.FLG: .BYTE   0               ;KT ON/OFF FLAG
                000030'         .EVEN
```

```
                                .SBTTL  KKTON   ROUTINE

                                ROUTINE KKTON <DTADR,BUFPTR>
                                                                KKTON:
                                ;+
                                ; SET UP THE ON FLAG
                                ;-
                                LET KT.FLG :B= #0
                                                                        CLRB    KT.FLG
                                INLINE <BR KTROU>
                                                                        BR KTROU
                                ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC

                                ROUTINE KKTOFF <DTADR,BUFPTR>
                                                                KKTOFF:
                                ;+
                                ; SET UP THE OFF FLAG
                                ;-
                                LET KT.FLG :B= #1
                                                                        MOVB    #1,KT.FLG
                                ;+
                                ; SAVE REGISTERS AND DATA TABLE ADDRESS
                                ;-

                                INLINE <KTROU:>
                                                                KTROU:
                                PUSH R0,R1
                                                                        MOV     R0,-(SP)
                                                                        MOV     R1,-(SP)
                                LET R0 := DTADR(R5)
                                                                        MOV     DTADR(R5),R0
                                LET R1 := BUFPTR(R5)
                                                                        MOV     BUFPTR(R5),R1
                                ;+
                                ; IF NOT IN RUNMODE THEN CONTINUE
                                ;-

                                IF #RUNMODE NOTSETIN DT.ST0(R0) THEN
                                                                        BIT     #RUNMODE,DT.ST0(
                                                                        BNE     50002$
                                ;+
                                ; SEE IF ANY ARGUMENTS EXIST & IF
                                ; THEY DO , THIS IS A BAD COMMAND
                                ;-

                                        CALL ARGCHK IN <R1> OUT <R1>
                                                                        SUB     #1*2,R5
                                                                        MOV     R5,-(SP)
```

```
(4)  000100' 010145                                                    MOV     R1,-(R5)
(3)  000102' 004767  000000G                                           JSR     PC,ARGCHK
(3)  000106' 012605                                                    MOV     (SP)+,R5
(4)  000110' 012501                                                    MOV     (R5)+,R1
706
707                                 ;+
708                                 ; THE ARGUMENT IS A CR SO ALL IS O.K.
709                                 ; SEE IF KT IS PRESENT AND IF IT IS,
710                                 ; IF KT FLAG IS 0 THEN SET THE KTSTAT BIT IN DT.ST0 OTHERWISE CLEAR IT
711                                 ;-
712
713  000112'                                IF.ERROR THEN                  BCC     50003$
(6)  000112' 103054
714  000114'                                       IF #KTPRES SETIN DT.CF0(R0) THEN
(6)  000114' 032760  000400  000014                                        BIT     #KTPRES,DT.CF0(R
(9)  000122' 001444                                                        BEQ     50004$
715  000124'                                              IFB KT.FLG EQ #0 THEN
(6)  000124' 105767  177676                                                TSTB    KT.FLG
(9)  000130' 001021                                                        BNE     50005$
716  000132'                                                    LET DT.ST0(R0) := DT.ST0(R0) SET.BY #KTSTAT
(6)  000132' 052760  000020  000010                                        BIS     #KTSTAT,DT.ST0(R
717  000140'                                                    LET @#SR0 := #1
(4)  000140' 012737  000001  177572                                        MOV     #1,@#SR0
718  000146'                                                    IF #ADDR22 SETIN DT.CF0(R0) THEN
(6)  000146' 032760  001000  000014                                        BIT     #ADDR22,DT.CF0(R
(9)  000154' 001403                                                        BEQ     50006$
719  000156'                                                           LET @#SR3 := @#SR3 SET.BY #BIT04
(6)  000156' 052737  000020  172516                                        BIS     #BIT04,@#SR3
720  000164'                                                    ENDIF
(4)  000164'                                                                50006$:
721                                 ;+
722                                 ; LOAD THE KTON MESSAGE IN DT.KBRSP
723                                 ;-
724
725  000164'                                                    LET DT.KBRSP(R0) := #KT.ON
(4)  000164' 012760  000007' 000022                                        MOV     #KT.ON,DT.KBRSP(
726  000172'                                              ELSE
(4)  000172' 000417                                                        BR      50007$
(3)  000174'                                                                50005$:
727  000174'                                                    LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #KTSTAT
(6)  000174' 042760  000020  000010                                        BIC     #KTSTAT,DT.ST0(R
728  000202'                                                    LET @#SR0 := #0
(4)  000202' 005037  177572                                                CLR     @#SR0
729  000206'                                                    IF #ADDR22 SETIN DT.CF0(R0) THEN
(6)  000206' 032760  001000  000014                                        BIT     #ADDR22,DT.CF0(R
(9)  000214' 001403                                                        BEQ     50010$
730  000216'                                                           LET @#SR3 := @#SR3 CLR.BY #BIT04
(6)  000216' 042737  000020  172516                                        BIC     #BIT04,@#SR3
731  000224'                                                    ENDIF
(4)  000224'                                                                50010$:
732                                 ;+
733                                 ; LOAD KTOFF MESSAGE INTO DT.KBRSP
734                                 ;-
735
736  000224'                                                    LET DT.KBRSP(R0) := #KT.OFF
(4)  000224' 012760  000016' 000022                                        MOV     #KT.OFF,DT.KBRSP
```

```
737  000232'                                                    ENDIF
(4)  000232'                                                                50007$:
738  000232'                                                ELSE
(4)  000232' 000403                                                             BR      50011$
(3)  000234'                                                                50004$:
739
740                                 ;+
741                                 ; LOAD THE NO KT MESSAGE INTO DT.KBRSP
742                                 ;-
743
744  000234'                                                    LET DT.KBRSP(R0) := #KT.NOKT
(4)  000234' 012760  000000' 000022                                             MOV     #KT.NOKT,DT.KBRS
745  000242'                                                ENDIF
(4)  000242'                                                                50011$:
746
747  000242'                                            ELSE
(4)  000242' 000403                                                             BR      50012$
(3)  000244'                                                                50003$:
748
749                                 ;+
750                                 ; LOAD THE ILLEGAL ARGUMENT INTO DT.KBRSP
751                                 ;-
752
753  000244'                                            LET DT.KBRSP(R0) := #CM.ARG
(4)  000244' 012760  000000G 000022                                             MOV     #CM.ARG,DT.KBRSP
754  000252'                                            ENDIF
(4)  000252'                                                                50012$:
755  000252'                                ELSE
(4)  000252' 000403                                                             BR      50013$
(3)  000254'                                                                50002$:
756
757                                 ;+
758                                 ; LOAD THE RUN MODE MESSAGE
759                                 ;-
760
761  000254'                                            LET DT.KBRSP(R0) := #CM.RUN
(4)  000254' 012760  000000G 000022                                             MOV     #CM.RUN,DT.KBRSP
762  000262'                                ENDIF
(4)  000262'                                                                50013$:
763
764                                 ;+
765                                 ; CLEAN UP
766                                 ;-
767
768  000262'                                POP R1,R0
(2)  000262' 012601                                                             MOV     (SP)+,R1
(3)  000264' 012600                                                             MOV     (SP)+,R0
769
770  000266'                                ENDRTN
(3)  000266'                                                                50000$:
(3)  000266'                                                                50001$:
(2)  000266' 000207                                                             RTS     PC
771
772
773          000001                        .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CM.ARG= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR  = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KKTOFF  000040RG
KKTON   000030RG
KTERRO= 000040
KTPRES= 000400
KTROU   000046R
KTSTAT= 000020
KTXTND= 040000
KT.FLG  000026R
KT.NOK  000000R
KT.OFF  000016R
KT.ON   000007R
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLD= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
```

```
SVR3   = 000070          UIPDR6= 177614          $F$DO  = 000340          $ISK4  = 000001          $$CASE= 000000
SVR4   = 000072          UIPDR7= 177616          $F$FAL= 000405          $LOCTA= 177777          $$DST  = 000000
SVR5   = 000074          WASADR= 000104          $F$GOO= 000400          $LSTIN= 000001          $$ELOC= 000402
SVR6   = 000076          WBSTAT= 000040          $F$IF  = 000110          $LSTTA= 000001          $$ERFL= 000000
SYSCNT= 000052           WBUFEA= 000136          $F$INC= 000210          $NESTL= 177777          $$FLAG= 000001
SYSERR= 000100           WBUFPA= 000134          $F$LOO= 000200          $NSK0  = 000300          $$FROM= 000001
TMPIO  = 000002          WBUFRQ= 000140          $F$NAM= 000160          $NSK1  = 000110          $$LOC  = 000214R
TQOVF  = 000002          WBUFSZ= 000142          $F$NO  = 000403          $NSK2  = 000110          $$LOCN= 000000
UIPAR0= 177640           WDFR   = 000116         $F$OR  = 000320          $NSK3  = 000110          $$REG  = 177777
UIPAR1= 177642           WDTO   = 000114         $F$RTI= 000350          $NSK4  = 000110          $$RETU= 000000
UIPAR2= 177644           WTINRE= 000352          $F$RTN= 000300          $NSK5  = 000110          $$RTN1= 050000
UIPAR3= 177646           WTWHMI= 000222          $F$SEL= 000140          $SAVLE= 177777          $$RTN2= 050001
UIPAR4= 177650           XFLAG  = 000005         $F$THE= 000330          $TAGLE= 177777          $$SRC  = 000000
UIPAR5= 177652           XOFF   = 000023         $F$TRU= 000404          $TAGNU= 050014          $$TGSV= 000000
UIPAR6= 177654           XON    = 000021         $F$UNT= 000130          $TEMP  = 000300          $$TGS1= 000000
UIPAR7= 177656           $BGNLE= 177777          $F$WHI= 000120          $TSK0  = 050013          $$TGS2= 000000
UIPDR0= 177600           $ERFLG= 000400          $F$YES= 000402          $TSK1  = 050012          $$TO   = 000001
UIPDR1= 177602           $F$AND= 000310          $IFLEV= 177777          $TSK2  = 050011          $$TAG= 050000
UIPDR2= 177604           $F$BAD= 000401          $ISK0  = 000001          $TSK3  = 050007          .      = 000270R
UIPDR3= 177606           $F$BLA= 000170          $ISK1  = 000001          $TSK4  = 050010
UIPDR4= 177610           $F$CAS= 000150          $ISK2  = 000001          $$ARGC= 000004
UIPDR5= 177612           $F$DEC= 000220          $ISK3  = 000001          $$BYTE= 000403

. ABS.  000000     000
        000270     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KTONOF,DSKZ:KTONOF=SPMAC/ML,EQUATE,KTONOF
 RUN-TIME: 14 5 .4 SECONDS
 RUN-TIME RATIO: 54/20=2.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  LPONOF  PROCESS THE 'LPON' AND 'LPOFF' KEYBOARD COMMANDS
                        .IDENT  /V0.0/


                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       LPONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE CONSISTS OF TWO MODULES:
                        ;       1. KLPON - PROCESS THE KEYBOARD COMMAND 'LPON'
                        ;       2. KLPOFF - PROCESS THE KEYBOARD COMMAND 'LPOFF'
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR LPONOF

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
          000001                $LSTIN=1
          000001                $LSTTAG = 1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KLPON           ;'LPON' MODULE ENTRY POINT
                                .GLOBL  KLPOFF          ;'LPOFF' MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CM.ARG          ;ILLEGAL ARGUMENT MESSAGE
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  LPINT           ;LP INTERRUPT ROUTINE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 050114  047440  022516  LP.MGN: .ASCIZ  /LP ON%/
000006'    000
000007'    114  020120  043117  LP.MGF: .ASCIZ  /LP OFF%/
000014' 022506     000
000017'    000                  LP.FLG: .BYTE   0               ;ON/OFF FLAG BYTE
                                .EVEN
```

```
                        .SBTTL  KLPON   PROCESS THE 'LPON' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KLPON
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'LPON' KEYBOARD ROUTINE.
                        ;       IT WILL FIRST VERIFY THAT NO ARGUMENT APPEARS
                        ;       IN THE COMMAND DECODE BUFFER. IF AN ARGUMENT IS IN THE
                        ;       BUFFER, AN ERROR MESSAGE IS ISSUED. OTHERWISE,
                        ;       THE LP ON/OFF STATUS BIT IN THE STATUS
                        ;       INDICATOR IS SET.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DTABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;       DT.ST0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       LPINT ROUTINE, LOC. 200 AND 202
                        ;       CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK - CHECK FOR UNWANTED ARGUMENTS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KLPON IN <DTADR,BUFPTR>
                        ;       WHERE:   DTADR = ADDRESS OF DTABLE
                        ;                BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  KLPOFF  PROCESS THE 'ROTOF' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KLPOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'LPOFF' KEYBOARD ROUTINE.
                        ;       IT WILL FIRST VERIFY THAT NO ARGUMENT APPEARS
                        ;       IN THE COMMAND DECODE BUFFER. IF AN ARGUMENT IS IN THE
                        ;       BUFFER, AN ERROR MESSAGE IS ISSUED. OTHERWISE,
                        ;       THE LP ON/OFF STATUS BIT IN THE STATUS
                        ;       INDICATOR IS CLEARED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DTABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;       3. ADDRESS OF COMMAND RESPONSE BUFFER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;       DT.STO
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK - CHECK FOR UNWANTED ARGUMENTS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KLPOFF IN <DTADR,BUFPTR>
                        ;       WHERE:   DTADR = ADDRESS OF DTABLE
                        ;                BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  KLPON ROUTINE


                                ROUTINE KLPON <DTADR,BUFPTR>                    KLPON:
                                ;+
                                ; SET ON FLAG
                                ;-

                                LET LP.FLG :B= #0                                       CLRB    LP.FLG
                                INLINE <BR LPROU>                                       BR LPROU
                                ENDRTN                                          50000$:
                                                                                50001$:
                                                                                        RTS     PC


                                ROUTINE KLPOFF <DTADR,BUFPTR>                   KLPOFF:
                                ;+
                                ; SET OFF FLAG
                                ;-

                                LET LP.FLG :B= #1                                       MOVB    #1,LP.FLG
                                ;+
                                ; INITIALIZE AND SAVE DTABLE ADDRESS AND DECODE BUFFER POINTER
                                ;-

                                INLINE <LPROU:>                                 LPROU:
                                PUSH R0,R1                                              MOV     R0,-(SP)
                                                                                        MOV     R1,-(SP)
                                LET R0 := DTADR(R5)                                     MOV     DTADR(R5),R0
                                LET R1 := BUFPTR(R5)                                    MOV     BUFPTR(R5),R1


                                ;+
                                ; MAKE SURE NO ADDITIONAL ARGUMENTS ARE IN DECODE BUFFER
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>                            SUB     #1*2,R5
                                                                                        MOV     R5,-(SP)
                                                                                        MOV     R1,-(R5)
                                                                                        JSR     PC,ARGCHK
                                                                                        MOV     (SP)+,R5
                                                                                        MOV     (R5)+,R1
```

```
                                ;+
                                ; THERE IS NO <CR> YET SO IT'S A BAD COMMAND...STUFF ERROR
                                ; MESSAGE AND BEAT IT....
                                ;-

                                IF.NO.ERROR THEN                                        BCS     50002$
                                        LET DT.KBRSP(R0) := #CM.ARG                     MOV     #CM.ARG,DT.KBRSP

                                ELSE                                                    BR      50003$
                                                                                50002$:

                                ;+
                                ; NOW DETERMINE IF LPON OR LPOFF COMMAND
                                ;-

                                        IFB LP.FLG EQ #0 THEN                           TSTB    LP.FLG
                                                                                        BNE     50004$
                                ;+
                                ; SET THE LPSTAT BIT
                                ; IN DT.STO...AND LOAD UP INTERRUP ROUTINE
                                ;-

                                                LET DT.STO(R0) := DT.STO(R0) SET.BY #LPSTAT
                                                                                        BIS     #LPSTAT,DT.STO(R
                                                LET @#200 := #LPINT                     MOV     #LPINT,@#200
                                                LET @#202 := #PRI1                      MOV     #PRI1,@#202
                                                LET DT.KBRSP(R0) := #LP.MGN             MOV     #LP.MGN,DT.KBRSP

                                        ELSE                                            BR      50005$
                                                                                50004$:

                                ;+
                                ; CLEAR THE LPSTAT BIT IN DT.STO.......
                                ;-

                                                LET DT.STO(R0) := DT.STO(R0) CLR.BY #LPSTAT
                                                                                        BIC     #LPSTAT,DT.STO(R
                                                LET DT.KBRSP(R0) := #LP.MGF             MOV     #LP.MGF,DT.KBRSP
                                        ENDIF
                                                                                50005$:
                                ENDIF                                           50003$:

                                ;+
                                ; CLEAN UP AND LEAVE
```

```
                                ;-
                                POP R1,R0
                                                        MOV     (SP)+,R1
                                                        MOV     (SP)+,R0
                                ENDRTN
                                                    50000$:
                                                    50001$:
                                                        RTS     PC
                                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KLPOFF  000030RG
KLPON   000020RG
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPINT = ****** G
LPROU   000036R
LPSTAT= 000001
LP.FLG  000017R
LP.MGF  000007R
LP.MGN  000000R
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
```

```
SVR4  = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$ELOC= 000402
SVR5  = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$ERFL= 000000
SVR6  = 000076          UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$FLAG= 000001
SYSCNT= 000052          WASADR= 000104          $F$DO = 000340          $LOCTA= 177777          $$FROM= 000001
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LSTIN= 000001          $$LOC = 000110R
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $NESTL= 177777          $$REG = 177777
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK0 = 000300          $$RETU= 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK1 = 000110          $$RTN1= 050000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK2 = 000110          $$RTN2= 050001
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $SAVLE= 177777          $$SRC = 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TAGNU= 050006          $$TGS1= 000000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TEMP = 000300          $$TGS2= 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TSK0 = 050003          $$TO  = 000001
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TSK1 = 050005          $$$TAG= 050000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $$ARGC= 000004          .     = 000166R
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $$BYTE= 000402
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $$CASE= 000000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$DST = 000000

. ABS.  000000    000
        000166    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:LPONOF,DSKZ:LPONOF=SPMAC/ML,EQUATE,LPONOF
 RUN-TIME: 12 3 .4 SECONDS
 RUN-TIME RATIO: 43/16=2.6
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                                .TITLE  LSTCHK CHECK MODULES TO DETERMINE IF ALL HAVE MADE END OF PASS
                                .IDENT /V0.0/

                                ;++
                                ; MODULE PACKAGE NAME:
                                ;       LSTCHK
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
                                ;               1. CHKEOP - CHECK FOR END-OF-PASS
                                ;               2. CHKACT - CHECK ALL MODULES INACTIVE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR LSTCHK

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTTAG=1
        000001                  $LSTIN=1
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  CHKEOP          ;MODULE ENTRY POINT
                                .GLOBL  CHKACT          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
```

```
                                .SBTTL  CHKEOP CHECK THAT ALL MODULES HAVE DONE END OF PASS
                                .IDENT  /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       CHKEOP
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       THIS MODULE IS CALLED TO CHECK END OF PASS CONDITION
                                ;       OF ALL MODULES. IF ALL MODULES HAVE MADE END OF PASS
                                ;       AT THIS RELOCATION, A RETURN NO ERROR OCCURS. IF
                                ;       ONLY BACKGROUND MODULES ARE CONFIGURED, ONLY
                                ;       ONE MUST COMPLETE END OF PASS FOR THE RETURN NO ERROR
                                ;       CONDITION
                                ;
                                ; INPUTS:
                                ;       DTABLE
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.MLST
                                ;       DT.BLST
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE ROUTINES CALLED:
                                ;       SAVREG
                                ;       RESREG
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL CHKEOP IN <DTADR>
                                ;       WHERE DTADR IS DATA TABLE ADDRESS
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            DATE            BY              REASON
                                ;--
```

```
                                       .SBTTL  CHKEOP ROUTINE

                                       ROUTINE CHKEOP <DTADR>
                                                                                    CHKEOP:


                                       ;+
                                       ; SAVE REGISTERS AND GET DTABLE ADDR AND MODULE LIST
                                       ;-
                                       CALL SAVREG
                                                                                            JSR     PC,SAVREG
                                       LET R0 := DTADR(R5)
                                                                                            MOV     DTADR(R5),R0
                                       LET R1 := DT.MLST(R0)
                                                                                            MOV     DT.MLST(R0),R1

                                       ;+
                                       ; INIT COUNTERS
                                       ;-

                                       LET R3 := #0
                                                                                            CLR     R3
                                       LET R4 := #0
                                                                                            CLR     R4

                                       ;+
                                       ; WHILE NOT AT END OF LIST AND NOT AT END OF IOMOD LIST,
                                       ; GET A MODULE HEADER ADDRESS. IF THE MODULE IS SELECTED AND NOT
                                       ; DROPPED, INCREMENT COUNT IN R4. IF, IN ADDITION, IT IS
                                       ; ACTIVE, INCREMENT R3 COUNT. IF ACTIVE, CHECK EOP BIT - IF SET
                                       ; CONTINUE TO NEXT MODULE. IF NOT SET - RETURN ERROR - CAN'T RELOCATE.
                                       ;-

                                       WHILE (R1) NE #ENDLST AND R1 NE DT.BLST(R0) DO
                                                                                    50002$:
                                                                                            CMP     (R1),#ENDLST
                                                                                            BEQ     50003$
                                                                                            CMP     R1,DT.BLST(R0)
                                                                                            BEQ     50003$
                                              LET R2 := (R1)+
                                                                                            MOV     (R1)+,R2
                                              IF #BIT14 SETIN STAT(R2) AND #BIT13 NOTSETIN STAT(R2) THEN
                                                                                            BIT     #BIT14,STAT(R2)
                                                                                            BEQ     50004$
                                                                                            BIT     #BIT13,STAT(R2)
                                                                                            BNE     50004$
                                                    LET R4 := R4 + #1
                                                                                            INC     R4
                                                    IF #ACTBIT SETIN STAT(R2) THEN
                                                                                            BIT     #ACTBIT,STAT(R2)
                                                                                            BEQ     50005$
                                                           LET R3 := R3 + #1
                                                                                            INC     R3
                                                           IFB #EOPBIT NOTSETIN XFLAG(R2) THEN
                                                                                            BITB    #EOPBIT,XFLAG(R2
                                                                                            BNE     50006$
```

```
  637  000102'                                                  INLINE <BR      1$>
  (2)  000102' 000443                                                                      BR      1$
  638  000104'                                                ENDIF
  (4)  000104'                                                                          50006$:
  639  000104'                                            ENDIF
  (4)  000104'                                                                          50005$:
  640  000104'                                        ENDIF
  (4)  000104'                                                                          50004$:
  641  000104'                                    ENDDO
  (4)  000104' 000745                                                                      BR      50002$
  (3)  000106'                                                                          50003$:
  642
  643
  644                                        ;+
  645                                        ; IF R3 IS ZERO....WE HAVE NO ACTIVE IOMODS - SEE IF WE HAVE ANY BKMODS
  646                                        ; THAT HAVE DONE EOP. IF ONE HAS, IT'S O.K. TO RELOCATE SO
  647                                        ; RETURN NO ERROR. IF # IOMODS ACTIVE  IS NOT ZERO AND WE GOT THIS
  648                                        ; FAR, WE CAN DEFINITELY RELOCATE (THE ELSE SIDE) .... RETURN
  649                                        ; NO ERROR.....
  650                                        ;-
  651
  652  000106'                               IF R3 EQ #0 THEN
  (6)  000106' 005703                                                                      TST     R3
  (9)  000110' 001026                                                                      BNE     50007$
  653  000112'                                        WHILE (R1) NE #ENDLST DO
  (4)  000112'                                                                          50010$:
  (6)  000112' 021127  000000                                                              CMP     (R1),#ENDLST
  (9)  000116' 001422                                                                      BEQ     50011$
  654  000120'                                            LET R2 := (R1)+
  (4)  000120' 012102                                                                      MOV     (R1)+,R2
  655  000122'                                            IF #BIT14 SETIN STAT(R2) AND #BIT13 NOTSETIN STAT(R2) THEN
  (6)  000122' 032762  040000  000026                                                      BIT     #BIT14,STAT(R2)
  (9)  000130' 001414                                                                      BEQ     50012$
  (6)  000132' 032762  020000  000026                                                      BIT     #BIT13,STAT(R2)
  (9)  000140' 001010                                                                      BNE     50012$
  656  000142'                                                LET R3 := R3 + #1
  (6)  000142' 005203                                                                      INC     R3
  657  000144'                                                IFB #EOPBIT SETIN XFLAG(R2) THEN
  (6)  000144' 132762  000001  000005                                                      BITB    #EOPBIT,XFLAG(R2
  (9)  000152' 001403                                                                      BEQ     50013$
  658  000154'                                                    CALL RESREG
  (3)  000154' 004767  000000G                                                             JSR     PC,RESREG
  659  000160'                                                    RETURN NO.ERROR
  (4)  000160' 000420                                                                      BR      50000$
  660  000162'                                                ENDIF
  (4)  000162'                                                                          50013$:
  661  000162'                                            ENDIF
  (4)  000162'                                                                          50012$:
  662  000162'                                        ENDDO
  (4)  000162' 000753                                                                      BR      50010$
  (3)  000164'                                                                          50011$:
  663  000164'                               ELSE
  (4)  000164' 000403                                                                      BR      50014$
  (3)  000166'                                                                          50007$:
  664
  665                                        ;+
```

```
                                ; THERE WAS AT LEAST ONE IOMOD ACTIVE....SINCE WE GOT THIS
                                ; FAR, THE EOP BIT MUST HAVE BEEN SET IN ALL- SO RELOCATE.....
                                ;-

000166'                                 CALL RESREG                                             JSR     PC,RESREG
(3)  000166' 004767  000000G
000172'                                 RETURN NO.ERROR                                         BR      50000$
(4)  000172' 000413
                                ENDIF                                                   50014$:
(4)  000174'


                                ;+
                                ; IF WE EVER GET HERE.... !!!!!! ..... THE SITUATION COULD
                                ; BE THIS:    THE NUMBER OF ACTIVE IOMODS IS ZERO.
                                ; SO WE TRIED TO USE BACKGROUND EOP BITS AS RELOCATION CRITERIA AND
                                ; COULDN'T - IF THERE ARE NO BACKGROUND MODULES SELECTED AND NOT DROPPED,
                                ; (OR MAYBE NONE AT ALL (R3 WOULD BE ZERO))THERE ARE IOMODS THAT
                                ; ARE NOT CURRENTLY ACTIVE BUT COULD RUN SOMEPLACE (R4 NE #0)
                                ; THEN WE MAY AS WELL RELOCATE OVER AND OVER AGAIN IN HOPES OF FINDING
                                ; A PLACE WHERE THEY CAN RUN.....................
                                ;-

000174'                         IF R3 EQ #0 AND R4 NE #0 THEN                                   TST     R3
(6)  000174' 005703                                                                             BNE     50015$
(9)  000176' 001005                                                                             TST     R4
(6)  000200' 005704                                                                             BEQ     50015$
(9)  000202' 001403
000204'                                 CALL RESREG                                             JSR     PC,RESREG
(3)  000204' 004767  000000G
000210'                                 RETURN NO.ERROR                                         BR      50000$
(4)  000210' 000404
                                ENDIF                                                   50015$:
(4)  000212'


                                ;+
                                ; IF WE GET HERE - WE SIMPLY ARE WAITING FOR A BACKGROUND TO EOP
                                ;-

000212'                         INLINE <1$:>                                                    1$:
(2)  000212'
000212'                         CALL RESREG                                             JSR     PC,RESREG
(3)  000212' 004767  000000G
000216'                         RETURN ERROR                                            SEC
(2)  000216' 000261                                                                     BR      50001$
(4)  000220' 000401


000222'                         ENDRTN                                                  50000$:
(3)  000222'                                                                            CLC
(2)  000222' 000241                                                                     50001$:
(3)  000224'                                                                            RTS     PC
(2)  000224' 000207
```

```
                        .SBTTL  CHKACT CHECK FOR ACTIVE MODULES
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CHKACT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE SIMPLY CHECKS ALL OPTION MODULES TO
                        ;       SEE IF THEY ARE ACTIVE
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DTABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. SAVREG
                        ;       2. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CHKACT IN <DT>
                        ;               WHERE DT = DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                       .SBTTL  CHKACT ROUTINE

                                       ROUTINE CHKACT <DTADR>
000226'                                                                                  CHKACT:
(2)  000226'


                                       ;+
                                       ; SAVE REGISTERS AND GET DTABLE ADDR. AND ADDR. OF MODULE LIST
                                       ;-

                                       CALL SAVREG
000226'                                                                                          JSR     PC,SAVREG
(3)  000226' 004767  000000G
                                       LET R0 := DTADR(R5)
000232'                                                                                          MOV     DTADR(R5),R0
(4)  000232' 016500  000000
                                       LET R1 := DT.MLST(R0)
000236'                                                                                          MOV     DT.MLST(R0),R1
(4)  000236' 016001  000032

                                       ;+
                                       ; SEARCH LIST OF MODULES UNTIL END OR UNTIL AN ACTIVE
                                       ; BIT IS SET IN STAT WORD
                                       ;-

                                       WHILE (R1) NE #ENDLST DO
000242'                                                                                  50002$:
(4)  000242'                                                                                     CMP     (R1),#ENDLST
(6)  000242' 021127  000000                                                                      BEQ     50003$
(9)  000246' 001412
                                               LET R2 := (R1)+
000250'                                                                                          MOV     (R1)+,R2
(4)  000250' 012102
                                               IF #ACTBIT SETIN STAT(R2) THEN
000252'                                                                                          BIT     #ACTBIT,STAT(R2)
(6)  000252' 032762  004000  000026                                                             BEQ     50004$
(9)  000260' 001404
                                                       CALL RESREG
000262'                                                                                          JSR     PC,RESREG
(3)  000262' 004767  000000G
                                                       RETURN ERROR
000266'                                                                                          SEC
(2)  000266' 000261                                                                              BR      50001$
(4)  000270' 000405
                                               ENDIF
000272'                                                                                  50004$:
(4)  000272'
                                       ENDDO
000272'                                                                                          BR      50002$
(4)  000272' 000763
(3)  000274'                                                                             50003$:

                                       ;+
                                       ; WE WENT ALL THE WAY THRU .... GUESS NONE ACTIVE
                                       ;-

                                       CALL RESREG
000274'                                                                                          JSR     PC,RESREG
(3)  000274' 004767  000000G
                                       RETURN NO.ERROR
000300'                                                                                          BR      50000$
(4)  000300' 000400

                                       ENDRTN
000302'                                                                                  50000$:
(3)  000302'                                                                                     CLC
(2)  000302' 000241                                                                      50001$:
(3)  000304'                                                                                     RTS     PC
(2)  000304' 000207
```

```
781
782          000001                   .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CHKACT  000226RG
CHKEOP  000000RG
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
```

```
TQOVF = 000002          WBUFPA= 000134          $F$IF  = 000110          $LSTTA= 000001          $$DST  = 000000
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NESTL= 177777          $$ELOC= 000403
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK0 = 000300          $$ERFL= 000000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK1 = 000120          $$FLAG= 000001
UIPAR3= 177646          WDTO  = 000114          $F$NO  = 000403          $NSK2 = 000110          $$FROM= 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR  = 000320          $NSK3 = 000110          $$LOC  = 000260R
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $NSK4 = 000110          $$LOCN= 000000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $SAVLE= 177777          $$REG  = 177777
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $SSK0 = 050003          $$RETU= 000001
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TAGLE= 177777          $$RTN1= 050000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TAGNU= 050005          $$RTN2= 050001
UIPDR2= 177604          $ERFLG= 000000          $F$UNT= 000130          $TEMP  = 000300          $$SRC  = 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK0  = 050002          $$TGSV= 000000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $TSK1  = 050003          $$TGS1= 000000
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $TSK2  = 050004          $$TGS2= 000000
UIPDR6= 177614          $F$CAS= 000150          $ISK0  = 000001          $TSK3  = 050012          $$TO   = 000000
UIPDR7= 177616          $F$DEC= 000220          $ISK1  = 000001          $TSK4  = 050013          $$$TAG= 050000
WASADR= 000104          $F$DO  = 000340          $ISK2  = 000001          $$ARGC= 000002          .      = 000306R
WBSTAT= 000040          $F$FAL= 000405          $LOCTA= 177777          $$BYTE= 000403
WBUFEA= 000136          $F$GOO= 000400          $LSTIN= 000001          $$CASE= 000000

. ABS.  000000     000
        000306     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:LSTCHK,DSKZ:LSTCHK=SPMAC/ML,EQUATE,LSTCHK
 RUN-TIME: 17 7 .4 SECONDS
 RUN-TIME RATIO: 67/25=2.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  MSGDEQ - DE-QUEUE NEXT MESSAGE
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE NAME:
                        ;       MSGDEQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE CONTAINS THE TWO ROUTINES MSGDEQ AND MSGDRET.
                        ;       THE MSGDEQ ROUTINE DE-QUEUES THE NEXT MESSAGE TO BE PROCESSED,
                        ;       WHICH CONSISTS OF PERFORMING THE FOLLOWING:
                        ;               1. DEQUEUING NEXT MESSAGE
                        ;               2. ENABLING/INHIBITTING PRINTING OF MESSAGE BASED
                        ;                  ON THE SOFTWARE SWITCH REGISTER SETTING
                        ;               3. HOOKING MESSAGE TO APPROPRIATE OUTPUT DRIVER
                        ;
                        ;       THE MSGDRET ROUTINE PERFORMS THE NECESSARY CLEAN-UP ASSOCIATED
                        ;       WITH THE PROCESSING OF THE LAST MESSAGE. THIS CONSISTS OF:
                        ;               1. ACTIVATING REQUESTOR MODULE, I.E., THE MODULE THAT
                        ;                  JUST HAD ITS MESSAGE PROCESSED
                        ;               2. RESETTING THE CONTROL-O FLAG
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.SWR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. CTRLOF               ;CONTROL-O FLAG
                        ;       2. DX.KFL               ;CARRIAGE RETURN FLAG
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       1. ENQCQ                MAKE ENTRY IN CONTROL QUEUE
                        ;       2. KBINI                INITIALIZE KEYBOARD HANDLER ROUTINE
                        ;       3. MSGDGET              GET NEXT MESSAGE MODULE
                        ;       4. MSGDHOOK             HOOK MESSAGE TO OUTPUT DRIVER
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL MSGDEQ IN <DTABLE>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
(1) 000000'                             $LSTIN=1
        000001                          $LSTTAG=1
        000001                          $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  MSGDEQ                  ;MODULE'S ENTRY POINT
                                .GLOBL  MD.BSY                  ;BUSY FLAG
                                .GLOBL  MD.COD                  ;MESSAGE TYPE INDICATOR
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQCQ                   ;CONTROL QUEUE EN-QUEING MODULE
                                .GLOBL  BA.STAT
                                .GLOBL  KBINI                   ;KB I/O INITIATOR MODULE
                                .GLOBL  DX.KFL                  ;CARRIAGE RETURN FLAG
                                .GLOBL  MSGDGET                 ;MESSAGE DE-QUEING MODULE
                                .GLOBL  MSGDHOCK                ;OUTPUT MESSAGE TO DRIVER MODULE
                                .GLOBL  CTRLOF                  ;CONTROL-O FLAG
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
                                ;
000000' 000000                  MD.COD: .WORD   0                ;MESSAGE TYPE CODE
000002' 000000                  MD.MSG: .WORD   0                ;ADDRESS OF MESSAGE
000004' 000000                  MD.HDR: .WORD   0                ;ADDRESS OF MODULE'S HEADER
000006' 000000                  MD.RET: .WORD   0                ;ADDRESS AT WHICH MODULE IS TO BE RESUMED
000010' 000000                  MD.DTBL: .WORD  0                ;ADDRESS OF DATA TABLE
000012' 000000                  MD.BSY: .WORD   0                ;MESSAGE-QUEUE BUSY INDICATOR
        000005                          .REPT   ^D<5>            ;SAVE 5 WORDS FOR R5 STACK
                                        .WORD   0
                                        .ENDR
000026'                         MD.R5S:                          ;LOCAL R5 STACK
000026' 000050                  MD.KBECH: .BLKW ^D40             ;KEYBOARD ECHO BUFFER
000146'    001      000         MD.NUL: .BYTE   1,0              ;NULL CHAR MESSAGE
                                ;
                                ;***************
                                .EVEN
                                ;
```

```
                                .SBTTL  PROCESS NEXT MESSAGE

000150'                         ROUTINE MSGDEQ <DTABLE>
(2)  000150'                                                                    MSGDEQ:

                                ;+
                                ; MAKE SURE THE MESSAGE QUEUE MECHANISM IS NOT BUSY. IF
                                ; IT IS, SIMPLY RETURN TO THE CALLER
                                ;-

000150'                         IF MD.BSY EQ #1 THEN
(6)  000150' 026727  177636  000001                                             CMP     MD.BSY,#1
(9)  000156' 001001                                                             BNE     50002$
000160'                             RETURN
(4)  000160' 000561                                                             BR      50000$
000162'                         ENDIF
(4)  000162'                                                                    50002$:

                                ;+
                                ; SAVE REGISTERS, THEN GET THE DATA TABLE ADDRESS AND SAVE IT IN
                                ; LOCAL STORAGE.
                                ;-

000162'                         PUSH    R0,R1
(2)  000162' 010046                                                             MOV     R0,-(SP)
(3)  000164' 010146                                                             MOV     R1,-(SP)
000166'                         LET R0 := DTABLE(R5)
(4)  000166' 016500  000000                                                     MOV     DTABLE(R5),R0
000172'                         LET MD.DTBL := R0
(4)  000172' 010067  177612                                                     MOV     R0,MD.DTBL

                                ;+
                                ; GET THE NEXT MESSAGE TO BE PRINTED
                                ;-

000176'                         CALL MSGDGET IN <R0,#MD.KBECH,#MD.NULL> OUT <MD.COD,MD.MSG,MD.HDR,MD.RET>
(4)  000176' 162705  000010                                                     SUB     #4*2,R5
(3)  000202' 010546                                                             MOV     R5,-(SP)
(6)  000204' 012745  000146'                                                    MOV     #MD.NULL,-(R5)
(5)  000210' 012745  000026'                                                    MOV     #MD.KBECH,-(R5)
(4)  000214' 010045                                                             MOV     R0,-(R5)
(3)  000216' 004767  000000G                                                    JSR     PC,MSGDGET
(3)  000222' 012605                                                             MOV     (SP)+,R5
(4)  000224' 012567  177550                                                     MOV     (R5)+,MD.COD
(4)  000230' 012567  177546                                                     MOV     (R5)+,MD.MSG
(4)  000234' 012567  177544                                                     MOV     (R5)+,MD.HDR
(4)  000240' 012567  177542                                                     MOV     (R5)+,MD.RET

                                ;+
                                ; IF THIS IS THE NULL CHARACTER MESSAGE AND IF OUTPUTTING THE
                                ; NULL IS ENABLED, THEN HOOK UP THE MESSAGE.  OTHERWISE,
                                ; JUST RETURN.  IN EITHER CASE, DON'T BOTHER MAKING ANY OF THE OTHER TESTS.
                                ;-

000244'                         IF MD.COD EQ #MSGNUL THEN
(6)  000244' 026727  177530  177775                                             CMP     MD.COD,#MSGNUL
```

```
(9)  000252' 001007                                                                    BNE     50003$
653  000254'                        IF #ENBNUL SETIN DT.SWR(R0) THEN
(6)  000254' 032760  000001  000056                                                   BIT     #ENBNUL,DT.SWR(R
(9)  000262' 001402                                                                    BEQ     50004$
654  000264'                                INLINE <BR      1$>
(2)  000264' 000470                                                                    BR      1$
655  000266'                        ELSE
(4)  000266' 000401                                                                    BR      50005$
(3)  000270'                                                                        50004$:
656  000270'                                INLINE <BR      2$>
(2)  000270' 000513                                                                    BR      2$
657  000272'                        ENDIF
(4)  000272'                                                                        50005$:
658  000272'                    ENDIF
(4)  000272'                                                                        50003$:
659
660
661                             ;+
662                             ; IF THIS IS AN ERROR MESSAGE OR A MESSAGE COMING DIRECTLY
663                             ; FROM A MODULE, AND IF THE PRINTING OF THESE TYPES OF MESSAGES
664                             ; IS INHIBITTED, DON'T PRINT THE MESSAGE.  INSTEAD, JUST ENTER THE MODULE IN
665                             ; THE CONTROL QUEUE AND RETURN.
666                             ;-
667
668  000272'                    IF MD.COD EQ #MSGSTD OR MD.COD GE #MSGSKEL AND MD.COD LT #MSGSYS THEN
(6)  000272' 026727  177502  000000                                                   CMP     MD.COD,#MSGSTD
(8)  000300' 001404                                                                    BEQ     50006$
(6)  000302' 026727  177472  000003                                                   CMP     MD.COD,#MSGSKEL
(9)  000310' 002422                                                                    BLT     50007$
(6)  000312'                                                                        50006$:
(6)  000312' 026727  177462  000012                                                   CMP     MD.COD,#MSGSYS
(9)  000320' 002016                                                                    BGE     50007$
669  000322'                        IF #INHEPRT SETIN DT.SWR(R0) THEN
(6)  000322' 032760  020000  000056                                                   BIT     #INHEPRT,DT.SWR(
(9)  000330' 001412                                                                    BEQ     50010$
670  000332'                                CALL ENQCQ IN <R0,MD.HDR,MD.RET>
(3)  000332' 010546                                                                    MOV     R5,-(SP)
(6)  000334' 016745  177446                                                           MOV     MD.RET,-(R5)
(5)  000340' 016745  177440                                                           MOV     MD.HDR,-(R5)
(4)  000344' 010045                                                                    MOV     R0,-(R5)
(3)  000346' 004767  000000G                                                          JSR     PC,ENQCQ
(3)  000352' 012605                                                                    MOV     (SP)+,R5
671  000354'                                INLINE <BR      2$>
(2)  000354' 000461                                                                    BR      2$
672  000356'                        ENDIF
(4)  000356'                                                                        50010$:
673  000356'                    ENDIF
(4)  000356'                                                                        50007$:
674
675
676                             ;+
677                             ; IF THIS IS AND END-OF-PASS MESSAGE, AND IF THE PRINTING OF THIS TYPE OF
678                             ; MESSAGE IS NOT ENABLED, DON'T PRINT THE MESSAGE.  INSTEAD, IF THIS
679                             ; MESSAGE CAME FROM A BKMOD, CLEAR THE DEFERRED-RETURN BIT IN THE
680                             ; BACKGROUND STATUS INDICATOR; ELSE ENQUEUE THE
681                             ; MODULE IN THE CONTROL QUEUE. THEN JUST RETURN.
```

```
  682                                   ;-
  683
  684  000356'                          IF MD.COD EQ #MSGEOP AND #ENBEOP NOTSETIN DT.SWR(R0) THEN
  (6)  000356' 026727  177416  000013                                                        CMP     MD.COD,#MSGEOP
  (9)  000364' 001030                                                                        BNE     50011$
  (6)  000366' 032760  010000  000056                                                        BIT     #ENBEOP,DT.SWR(R
  (9)  000374' 001024                                                                        BNE     50011$
  685  000376'                                  LET R1 := MD.HDR
  (4)  000376' 016701  177402                                                                MOV     MD.HDR,R1
  686  000402'                                  IF #BKMOD SETIN STAT(R1) THEN
  (6)  000402' 032761  000020  000026                                                        BIT     #BKMOD,STAT(R1)
  (9)  000410' 001404                                                                        BEQ     50012$
  687  000412'                                          LET BA.STAT := BA.STAT CLR.BY #BKDEF
  (6)  000412' 042767  000002  000000G                                                       BIC     #BKDEF,BA.STAT
  688  000420'                                  ELSE
  (4)  000420' 000411                                                                        BR      50013$
  (3)  000422'                                                                       50012$:
  689  000422'                                          CALL ENQCQ IN <R0,MD.HDR,MD.RET>
  (3)  000422' 010546                                                                        MOV     R5,-(SP)
  (6)  000424' 016745  177356                                                                MOV     MD.RET,-(R5)
  (5)  000430' 016745  177350                                                                MOV     MD.HDR,-(R5)
  (4)  000434' 010045                                                                        MOV     R0,-(R5)
  (3)  000436' 004767  000000G                                                               JSR     PC,ENQCQ
  (3)  000442' 012605                                                                        MOV     (SP)+,R5
  690  000444'                                  ENDIF
  (4)  000444'                                                                       50013$:
  691  000444'                                  INLINE <BR      2$>
  (2)  000444' 000425                                                                        BR      2$
  692  000446'                          ENDIF
  (4)  000446'                                                                       50011$:
  693
  694
  695                                   ;+
  696                                   ; SET THE MESSAGE-QUEUE BUSY FLAG.
  697                                   ;-
  698
  699  000446'                          INLINE <1$:>
  (2)  000446'                                                                       1$:
  700  000446'                          LET MD.BSY := #1
  (4)  000446' 012767  000001  177336                                                        MOV     #1,MD.BSY
  701
  702
  703                                   ;+
  704                                   ; IF NO TTY , DON'T BOTHER CALLING MSGDHOOK - JUST CALL MSGRET WHICH
  705                                   ; WILL RTS BACK TO HERE
  706                                   ;-
  707
  708  000454'                          IF #NOAPTY NOTSETIN DT.CF0(R0) THEN
  (6)  000454' 032760  000002  000014                                                        BIT     #NOAPTY,DT.CF0(R
  (9)  000462' 001014                                                                        BNE     50014$
  709  000464'                                  CALL MSGDHOOK IN <R0,MD.COD,MD.MSG,#MSGDRET>
  (3)  000464' 010546                                                                        MOV     R5,-(SP)
  (7)  000466' 012745  000526'                                                               MOV     #MSGDRET,-(R5)
  (6)  000472' 016745  177304                                                                MOV     MD.MSG,-(R5)
  (5)  000476' 016745  177276                                                                MOV     MD.COD,-(R5)
  (4)  000502' 010045                                                                        MOV     R0,-(R5)
```

```
(3)  000504' 004767  000000G                                                    JSR     PC,MSGDHOOK
(3)  000510' 012605                                                             MOV     (SP)+,R5
710  000512'                        ELSE
(4)  000512' 000402                                                             BR      50015$
(3)  000514'                                                            50014$:
711  000514'                                CALL MSGDRET
(3)  000514' 004767  000006                                                     JSR     PC,MSGDRET
712  000520'                        ENDIF
(4)  000520'                                                            50015$:
713
714                                 ;+
715                                 ; RETURN TO CALLER
716                                 ;-
717
718  000520'                        INLINE <2$:>
(2)  000520'                                                                    2$:
719  000520'                        POP     R1,R0
(2)  000520' 012601                                                             MOV     (SP)+,R1
(3)  000522' 012600                                                             MOV     (SP)+,R0
720  000524'                        ENDRTN
(3)  000524'                                                            50000$:
(3)  000524'                                                            50001$:
(2)  000524' 000207                                                             RTS     PC
```

```
                                .SBTTL  PERFORM CLEAN-UP FOR THE LAST MESSAGE


                                ;+
                                ; SET THE MESSAGE QUEUE AS BEING NOT BUSY, AND RESET THE
                                ; CONTROL-O FLAG
                                ;-

000526'                         MSGDRET:  LET MD.BSY := #0
(4)  000526' 005067  177260                                                     CLR     MD.BSY
                                        LET CTRLOF := #0
000532'                                                                         CLR     CTRLOF
(4)  000532' 005067  000000G

                                ;+
                                ; SWITCH TO A LOCAL R5 STACK
                                ;-

                                        PUSH R5,R0,R1
000536'                                                                         MOV     R5,-(SP)
(2)  000536' 010546                                                             MOV     R0,-(SP)
(3)  000540' 010046                                                             MOV     R1,-(SP)
(4)  000542' 010146
                                        LET R1 := MD.DTBL
000544'                                                                         MOV     MD.DTBL,R1
(4)  000544' 016701  177240
                                        LET R5 := #MD.R5S
000550'                                                                         MOV     #MD.R5S,R5
(4)  000550' 012705  000026'

                                ;+
                                ; DETERMINE IF THE LAST MESSAGE WAS ISSUED BY AN OPTION MODULE
                                ; OR MONITOR (I.E. MD.HDR IS NOT 0).
                                ; AND IF IT WAS, RESUME ITS EXECUTION BY MAKING AN ENTRY FOR IT
                                ; IN THE CONTROL QUEUE
                                ;-

                                        IF MD.HDR NE #0 THEN
000554'                                                                         TST     MD.HDR
(6)  000554' 005767  177224                                                     BEQ     50016$
(9)  000560' 001422
                                                IF MD.COD EQ #MSGSTD OR MD.COD GE #MSGSKEL THEN
000562'                                                                         CMP     MD.COD,#MSGSTD
(6)  000562' 026727  177212  000000                                             BEQ     50017$
(8)  000570' 001404                                                             CMP     MD.COD,#MSGSKEL
(6)  000572' 026727  177202  000003                                             BLT     50020$
(9)  000600' 002412
(6)  000602'                                                                50017$:
                                                    CALL ENQCQ IN <MD.DTBL,MD.HDR,MD.RET>
000602'                                                                         MOV     R5,-(SP)
(3)  000602' 010546                                                             MOV     MD.RET,-(R5)
(6)  000604' 016745  177176                                                     MOV     MD.HDR,-(R5)
(5)  000610' 016745  177170                                                     MOV     MD.DTBL,-(R5)
(4)  000614' 016745  177170                                                     JSR     PC,ENQCQ
(3)  000620' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000624' 012605
                                                ENDIF
000626'                                                                     50020$:
(4)  000626'
                                        ENDIF
000626'                                                                     50016$:
(4)  000626'


                                ;+
```

```
                                ; IF THE MESSAGE WAS ONE OF THE KEYBOARD PROMPTS, THEN UNLOCK THE
                                ; KEYBOARD AND ALLOW THE NEXT COMMAND TO BE ENTERED
                                ;-

000626'                                 IF MD.COD EQ #MSGPRM THEN
(6)  000626' 026727  177146  177776                                     CMP     MD.COD,#MSGPRM
(9)  000634' 001006                                                     BNE     50021$
000636'                                         CALL KBINI IN <MD.DTBL>
(3)  000636' 010546                                                     MOV     R5,-(SP)
(4)  000640' 016745  177144                                             MOV     MD.DTBL,-(R5)
(3)  000644' 004767  000000G                                            JSR     PC,KBINI
(3)  000650' 012605                                                     MOV     (SP)+,R5
000652'                                 ENDIF
(4)  000652'                                                    50021$:


```

```
                                .SBTTL  RESTORE THE INTERRUPTED MODULE'S R5 BEFORE RETURNING


                                ;+
                                ; NOW RETURN TO THE INTERRUPTED ROUTINE
                                ;-


                                ;+
                                ; DO AN RTS IF NO TTY - AN RTI IF A TTY PRESENT
                                ;-
                                        IF #NOAPTY NOTSETIN DT.CF0(R1) THEN
000652' 032761  000002  000014                                          BIT     #NOAPTY,DT.CF0(R
(6) 000652' 032761  000002  000014                                      BIT     #NOAPTY,DT.CF0(R
(9) 000660' 001005                                                      BNE     50022$
                                                POP R1,R0,R5
000662'                                                                 MOV     (SP)+,R1
(2) 000662' 012601                                                      MOV     (SP)+,R0
(3) 000664' 012600                                                      MOV     (SP)+,R5
(4) 000666' 012605
                                                INLINE  <RTI>
000670'                                                                 RTI
(2) 000670' 000002
                                        ELSE
000672'                                                                 BR      50023$
(4) 000672' 000404                                              50022$:
(3) 000674'
                                                POP R1,R0,R5
000674'                                                                 MOV     (SP)+,R1
(2) 000674' 012601                                                      MOV     (SP)+,R0
(3) 000676' 012600                                                      MOV     (SP)+,R5
(4) 000700' 012605
                                                INLINE <RTS     PC>
000702'                                                                 RTS     PC
(2) 000702' 000207
                                        ENDIF
000704'                                                         50023$:
(4) 000704'

                000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BA.STA= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLOF= ****** G
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.KFL= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQCQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KBINI = ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.BSY  000012RG
MD.COD  000000RG
MD.DTB  000010R
MD.HDR  000004R
MD.KBE  000026R
MD.MSG  000002R
MD.NUL  000146R
MD.RET  000006R
MD.R5S  000026R
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ  000150RG
MSGDER= 000005
MSGDGE= ****** G
MSGDHO= ****** G
MSGDRE  000526R
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRIO  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
```

```
STAT  = 000026          UIPAR6= 177654          XOFF  = 000023          $F$THE= 000330          $$ARGC= 000002
STATBI= 064757          UIPAR7= 177656          XON   = 000021          $F$TRU= 000404          $$BYTE= 000403
STAT1 = 000027          UIPDR0= 177600          $BGNLE= 177777          $F$UNT= 000130          $$CASE= 000000
SUSPND= 000001          UIPDR1= 177602          $ERFLG= 000400          $F$WHI= 000120          $$DST = 000000
SVR0  = 000062          UIPDR2= 177604          $F$AND= 000310          $F$YES= 000402          $$ELOC= 000402
SVR1  = 000064          UIPDR3= 177606          $F$BAD= 000401          $IFLEV= 177777          $$ERFL= 000000
SVR2  = 000066          UIPDR4= 177610          $F$BLA= 000170          $ISK0 = 000001          $$FLAG= 000001
SVR3  = 000070          UIPDR5= 177612          $F$CAS= 000150          $ISK1 = 000001          $$FROM= 000000
SVR4  = 000072          UIPDR6= 177614          $F$DEC= 000220          $LOCTA= 177777          $$LOC = 000660R
SVR5  = 000074          UIPDR7= 177616          $F$DO = 000340          $LSTIN= 000001          $$LOCN= 000000
SVR6  = 000076          WASADR= 000104          $F$FAL= 000405          $LSTTA= 000001          $$REG = 177777
SYSCNT= 000052          WBSTAT= 000040          $F$GOO= 000400          $NESTL= 177777          $$RETU= 000000
SYSERR= 000100          WBUFEA= 000136          $F$IF = 000110          $NSK0 = 000110          $$RTN1= 050000
TMPIO = 000002          WBUFPA= 000134          $F$INC= 000210          $NSK1 = 000110          $$RTN2= 050001
TQOVF = 000002          WBUFRQ= 000140          $F$LOO= 000200          $NSK2 = 000110          $$SRC = 000000
UIPAR0= 177640          WBUFSZ= 000142          $F$NAM= 000160          $SAVLE= 177777          $$TGSV= 000000
UIPAR1= 177642          WDFR  = 000116          $F$NO = 000403          $TAGLE= 177777          $$TGS1= 000000
UIPAR2= 177644          WDTO  = 000114          $F$OR = 000320          $TAGNU= 050024          $$TGS2= 000000
UIPAR3= 177646          WTINRE= 000352          $F$RTI= 000350          $TEMP = 050023          $$TO  = 000001
UIPAR4= 177650          WTWHMI= 000222          $F$RTN= 000300          $TSK0 = 050023          $$$TAG= 050000
UIPAR5= 177652          XFLAG = 000005          $F$SEL= 000140          $TSK1 = 050020          .     = 000704R

. ABS.  000000     000
        000704     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:MSGDEQ,DSKZ:MSGDEQ=SPMAC/ML,EQUATE,MSGDEQ
 RUN-TIME: 18 8 .4 SECONDS
 RUN-TIME RATIO: 85/26=3.2
 CORE USED:  14K  (27 PAGES)
```

```
508                             .TITLE  MSGDGET - DE-QUEUE THE NEXT MESSAGE TO BE PRINTED
509                             .IDENT  /V0.0/
510
511                             ;++
512                             ; MODULE NAME:
513                             ;       MSGDGET
514                             ;
515                             ; FUNCTIONAL DESCRIPTION:
516                             ;       THE DE-QUEUE-NEXT-MESSAGE MODULE IS CALLED WHENEVER IT IS NECESSARY
517                             ;       TO FETCH THE ADDRESS OF THE NEXT MESSAGE TO BE PRINTED.
518                             ;       MESSAGES ARE  DE-QUEUED ACCORDING TO THE FOLLOWING PRIORITY:
519                             ;               1. OPERATOR INPUT STRING
520                             ;               2. KEYBOARD COMMAND RESPONSE
521                             ;               3. KEYBOARD COMMAND PROMPT
522                             ;               4. TYPE QUEUE ENTRY
523                             ;               5. DEFAULT MESSAGE (ONE-CHAR NULL MESSAGE)
524                             ;
525                             ; INPUTS:
526                             ;       1. ADDRESS OF DATA TABLE
527                             ;       2. ADDRESS OF KEYBOARD'S ECHO BUFFER
528                             ;       3. ADDRESS OF DEFAULT MESSAGE (ONE-CHAR NULL MESSAGE)
529                             ;
530                             ; IMPLICIT INPUTS:
531                             ;       1. DT.KBRSP
532                             ;       2. DT.KBUF
533                             ;       3. DT.KBECH
534                             ;       4. DT.KBPRM
535                             ;
536                             ; OUTPUTS:
537                             ;       1. FOUR-WORD MESSAGE PACKET:
538                             ;               WORD 1:   TYPE CODE
539                             ;                    2:   ADDRESS OF MESSAGE
540                             ;                    3:   OPTION MODULE'S HEADER ADDRESS
541                             ;                    4:   OPTION MODULE'S RETURN ADDRESS
542                             ;
543                             ; IMPLICIT OUTPUTS:
544                             ;       1. UPDATED DT.KBECH
545                             ;
546                             ; PATHOLOGICAL CONNECTIONS:
547                             ;       1. KEYBOARD INPUT POINTER (KB.INP)
548                             ;
549                             ; SUBORDINATE MODULES CALLED:
550                             ;       1. DEQTQ                        ;DE-QUEUE NEXT TYPE QUEUE ENTRY
551                             ;       2. FILLMSG                      ;FILL-IN SKELETAL MESSAGE
552                             ;       3. SAVREG                       ;SAVE REGISTERS
553                             ;       4. RESREG                       ;RESTORE REGISTERS
554                             ;
555                             ; CALLING SEQUENCE:
556                             ;       CALL MSGDGET IN <DTABLE,KBECH,DFTMSG> OUT <MSGTYPE,MSGADDR,HDRADDR,RETADDR>
557                             ;               WHERE   DTABLE  - DTABLE ADDRESS
558                             ;                       KBECH   - ECHO BUFFER POINTER
559                             ;                       DFTMSG  - DEFAULT MESSAGE
560                             ;                       MSGTYPE - MESSAGE TYPE
561                             ;                       MSGADDR - MESSAGE ADDRESS
562                             ;                       HDRADDR - HEADER ADDRESS
563                             ;                       RETADDR - RETURN ADDRESS
```

```
                                ;
                                ; VERSION:      0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;--
```

```
                        .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                .MCALL  STRUCT
000000'                         STRUCT
(i) 000000'             .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN = 1
        000001                  $LSTTAG = 1


                        ;
                        ;****************
                        ;
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  MSGDGET                 ;MODULE'S ENTRY POINT
                        ;
                        ;****************
                        ;
                        ; GLOBAL REFERENCES
                        ;
                        .GLOBL  DEQTQ                   ;TYPE QUEUE DE-QUEUING MODULE
                        .GLOBL  FILLMSG                 ;FILL IN SKELETAL MESSAGE MODULE
                        .GLOBL  KB.INP                  ;KEYBOARD'S INPUT BUFFER POINTER
                        .GLOBL  SAVREG                  ;SAVE REGISTER ROUTINE
                        .GLOBL  RESREG                  ;RESTORE REGISTER ROUTINE
                        ;
                        ;****************
                        ;
```

```
                        .SBTTL  DE-QUEUE NEXT MESSAGE

                                ROUTINE MSGDGET <DTABLE,KBECH,DFTMSG,MSGTYPE,MSGADDR,HDRADDR,RETADDR>
                                                                                MSGDGET:

                        ;+
                        ; SAVE REGISTERS AND GET DTABLE ADDRESS AND KEYBOARD ECHO POINTER
                        ;-

                                CALL SAVREG
                                                                                JSR     PC,SAVREG
                                LET R0 := DTABLE(R5)
                                                                                MOV     DTABLE(R5),R0
                                LET R1 := KBECH(R5)
                                                                                MOV     KBECH(R5),R1

                        ;+
                        ; SEE IF ANY OPERATOR INPUT IS AWAITING ECHOING, AND IF SO, ZERO THE KEYBOARD
                        ; ECHO BUFFER, COPY THE CHARACTERS TO BE ECHOED INTO IT AND USE THIS AS THE
                        ; FIRST MESSAGE
                        ;-

                                IF DT.KBECH(R0) LO KB.INP THEN
                                                                                CMP     DT.KBECH(R0),KB.
                                                                                BHIS    50002$
                                    LET R2 := #^D<40>
                                                                                MOV     #^D<40>,R2
                                    WHILE R2 GT #0 DO
                                                                        50003$:
                                                                                TST     R2
                                                                                BLE     50004$
                                        LET (R1)+ := #0
                                                                                CLR     (R1)+
                                        LET R2 := R2 - #1
                                                                                DEC     R2
                                    ENDDO
                                                                                BR      50003$
                                                                        50004$:

                        ;+
                        ; LOAD UP ECHO BUFFER FROM INPUT BUFFER
                        ;-

                                    LET R1 := KBECH(R5)
                                                                                MOV     KBECH(R5),R1
                                    WHILE DT.KBECH(R0) LT KB.INP DO
                                                                        50005$:
                                                                                CMP     DT.KBECH(R0),KB.
                                                                                BGE     50006$
                                        LET (R1)+ :B= @DT.KBECH(R0)
                                                                                MOVB    @DT.KBECH(R0),(R
                                        LET DT.KBECH(R0) := DT.KBECH(R0) + #1
                                                                                INC     DT.KBECH(R0)
                                    ENDDO
                                                                                BR      50005$
                                                                        50006$:
```

```
                        ;+
                        ; SET UP OUTPUT ARGUMENTS FOR ECHO TYPE MESSAGE TYPE
                        ;-

000070'                         LET R1 := #MSGECH                       MOV     #MSGECH,R1
(4) 000070' 012701  177777
000074'                         LET R2 := KBECH(R5)                     MOV     KBECH(R5),R2
(4) 000074' 016502  000002
000100'                         LET R3 := #0                            CLR     R3
(4) 000100' 005003
000102'                         LET R4 := #0                            CLR     R4
(4) 000102' 005004
000104'                     ELSE                                        BR      50007$
(4) 000104' 000466
(3) 000106'                                                         50002$:

                        ;+
                        ; SEE IF THERE IS A KEYBOARD COMMAND RESPONSE WAITING TO BE OUTPUT,
                        ; IF SO, TAKE IT NEXT, AND THEN CLEAR KEYBOARD RESPONSE INDICATOR
                        ;-

000106'                         IF DT.KBRSP(R0) NE #0 THEN              TST     DT.KBRSP(R0)
(6) 000106' 005760  000022                                              BEQ     50010$
(9) 000112' 001411
000114'                             LET R1 := #MSGRES                   MOV     #MSGRES,R1
(4) 000114' 012701  000001
000120'                             LET R2 := DT.KBRSP(R0)              MOV     DT.KBRSP(R0),R2
(4) 000120' 016002  000022
000124'                             LET R3 := #0                        CLR     R3
(4) 000124' 005003
000126'                             LET R4 := #0                        CLR     R4
(4) 000126' 005004
000130'                             LET DT.KBRSP(R0) := #0              CLR     DT.KBRSP(R0)
(4) 000130' 005060  000022
000134'                         ELSE                                    BR      50011$
(4) 000134' 000452
(3) 000136'                                                         50010$:

                        ;+
                        ; SEE IF THERE IS A KEYBOARD PROMPT AWAITING OUTPUT, IF SO, TAKE
                        ; IT, AND CLEAR THE KEYBOARD PROMPT INDICATOR
                        ;-

000136'                         IF DT.KBPRM(R0) NE #0 THEN              TST     DT.KBPRM(R0)
(6) 000136' 005760  000026                                              BEQ     50012$
(9) 000142' 001411
000144'                             LET R1 := #MSGPRM                   MOV     #MSGPRM,R1
(4) 000144' 012701  177776
000150'                             LET R2 := DT.KBPRM(R0)              MOV     DT.KBPRM(R0),R2
(4) 000150' 016002  000026
000154'                             LET R3 := #0                        CLR     R3
(4) 000154' 005003
000156'                             LET R4 := #0                        CLR     R4
(4) 000156' 005004
000160'                             LET DT.KBPRM(R0) := #0
```

```
(4)  000160' 005060  000026                                                  CLR     DT.KBPRM(R0)
671  000164'                                    ELSE
(4)  000164' 000436                                                          BR      50013$
(3)  000166'                                                              50012$:
672
673                                 ;+
674                                 ; SEE IF THERE IS ANYTHING IN THE TYPE QUEUE WAITING TO BE PRINTED,
675                                 ; IF SO, CHECK TO SEE IF ITS A SKELETAL MESSAGE THAT HAS TO BE
676                                 ; FILLED-IN
677                                 ;-
678
679  000166'                            CALL DEQTQ OUT <R1,R2,R3,R4>
(4)  000166' 162705  000010                                                  SUB     #4*2,R5
(3)  000172' 004767  000000G                                                JSR     PC,DEQTQ
(4)  000176' 012501                                                          MOV     (R5)+,R1
(4)  000200' 012502                                                          MOV     (R5)+,R2
(4)  000202' 012503                                                          MOV     (R5)+,R3
(4)  000204' 012504                                                          MOV     (R5)+,R4
680  000206'                        IF.NO.ERROR THEN
(6)  000206' 103417                                                          BCS     50014$
681  000210'                            IF  R1 GE #MSGSKEL THEN
(6)  000210' 020127  000003                                                  CMP     R1,#MSGSKEL
(9)  000214' 002413                                                          BLT     50015$
682  000216'                                CALL FILLMSG IN <R0,R1,R2,R3> OUT <R2>
(4)  000216' 162705  000002                                                  SUB     #1*2,R5
(3)  000222' 010546                                                          MOV     R5,-(SP)
(7)  000224' 010345                                                          MOV     R3,-(R5)
(6)  000226' 010245                                                          MOV     R2,-(R5)
(5)  000230' 010145                                                          MOV     R1,-(R5)
(4)  000232' 010045                                                          MOV     R0,-(R5)
(3)  000234' 004767  000000G                                                JSR     PC,FILLMSG
(3)  000240' 012605                                                          MOV     (SP)+,R5
(4)  000242' 012502                                                          MOV     (R5)+,R2
683  000244'                            ENDIF
(4)  000244'                                                              50015$:
684  000244'                        ELSE
(4)  000244' 000406                                                          BR      50016$
(3)  000246'                                                              50014$:
685
686                                 ;+
687                                 ; THE TYPE QUEUE MUST BE EMPTY, SO WE'LL RETURN THE DEFAULT MESSAGE
688                                 ; WHICH IS THE ONE-CHAR NULL MESSAGE
689                                 ;-
690
691  000246'                            LET R1 := #MSGNUL
(4)  000246' 012701  177775                                                  MOV     #MSGNUL,R1
692  000252'                            LET R2 := DFTMSG(R5)
(4)  000252' 016502  000004                                                  MOV     DFTMSG(R5),R2
693  000256'                            LET R3 := #0
(4)  000256' 005003                                                          CLR     R3
694  000260'                            LET R4 := #0
(4)  000260' 005004                                                          CLR     R4
695  000262'                        ENDIF
(4)  000262'                                                              50016$:
696  000262'                    ENDIF
(4)  000262'                                                              50013$:
```

```
697  000262'                                  ENDIF
(4)  000262'                                                                  50011$:
698  000262'                              ENDIF
(4)  000262'                                                                  50007$:
699
700                                  ;+
701                                  ; LOAD UP OUTPUT ARGUMENTS
702                                  ;-
703
704  000262'                              LET MSGTYPE(R5) := R1
(4)  000262' 010165  000006                                                           MOV     R1,MSGTYPE(R5)
705  000266'                              LET MSGADDR(R5) := R2
(4)  000266' 010265  000010                                                           MOV     R2,MSGADDR(R5)
706  000272'                              LET HDRADDR(R5) := R3
(4)  000272' 010365  000012                                                           MOV     R3,HDRADDR(R5)
707  000276'                              LET RETADDR(R5) := R4
(4)  000276' 010465  000014                                                           MOV     R4,RETADDR(R5)
708
709                                  ;+
710                                  ; RESTORE REGISTERS AND RETURN TO CALLER
711                                  ;-
712
713  000302'                              CALL RESREG
(3)  000302' 004767  000000G                                                          JSR     PC,RESREG
714  000306'                              ENDRTN
(3)  000306'                                                                  50000$:
(3)  000306'                                                                  50001$:
(2)  000306' 000207                                                                   RTS     PC
715
716          000001                      .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DEQTQ = ****** G
DFTMSG= 000004
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
FILLMS= ****** G
HDRADD= 000012
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KBECH = 000002
KB.INP= ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGADD= 000010
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDGE  000000RG
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGTYP= 000006
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RETADD= 000014
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
```

```
SVR2  = 000066          UIPDR5= 177612          $F$DEC= 000220          $ISK3 = 000001          $$ARGC= 000016
SVR3  = 000070          UIPDR6= 177614          $F$DO = 000340          $ISK4 = 000001          $$BYTE= 000403
SVR4  = 000072          UIPDR7= 177616          $F$FAL= 000405          $LOCTA= 177777          $$CASE= 000000
SVR5  = 000074          WASADR= 000104          $F$GOO= 000400          $LSTIN= 000001          $$DST = 000000
SVR6  = 000076          WBSTAT= 000040          $F$IF = 000110          $LSTTA= 000001          $$ELOC= 000402
SYSCNT= 000052          WBUFEA= 000136          $F$INC= 000210          $NESTL= 177777          $$ERFL= 000000
SYSERR= 000100          WBUFPA= 000134          $F$LOO= 000200          $NSK0 = 000300          $$FLAG= 000001
TMPIO = 000002          WBUFRQ= 000140          $F$NAM= 000160          $NSK1 = 000110          $$FROM= 000000
TQOVF = 000002          WBUFSZ= 000142          $F$NO = 000403          $NSK2 = 000110          $$LOC = 000214R
UIPAR0= 177640          WDFR  = 000116          $F$OR = 000320          $NSK3 = 000110          $$LOCN= 000000
UIPAR1= 177642          WDTO  = 000114          $F$RTI= 000350          $NSK4 = 000110          $$REG = 177777
UIPAR2= 177644          WTINRE= 000352          $F$RTN= 000300          $NSK5 = 000110          $$RETU= 000000
UIPAR3= 177646          WTWHMI= 000222          $F$SEL= 000140          $SAVLE= 177777          $$RTN1= 050000
UIPAR4= 177650          XFLAG = 000005          $F$THE= 000330          $SSK0 = 050006          $$RTN2= 050001
UIPAR5= 177652          XOFF  = 000023          $F$TRU= 000404          $TAGLE= 177777          $$SRC = 000000
UIPAR6= 177654          XON   = 000021          $F$UNT= 000130          $TAGNU= 050017          $$TGSV= 000000
UIPAR7= 177656          $BGNLE= 177777          $F$WHI= 000120          $TEMP = 000300          $$TGS1= 000000
UIPDR0= 177600          $ERFLG= 000400          $F$YES= 000402          $TSK0 = 050007          $$TGS2= 000000
UIPDR1= 177602          $F$AND= 000310          $IFLEV= 177777          $TSK1 = 050011          $$TO  = 000000
UIPDR2= 177604          $F$BAD= 000401          $ISK0 = 000001          $TSK2 = 050013          $$$TAG= 050000
UIPDR3= 177606          $F$BLA= 000170          $ISK1 = 000001          $TSK3 = 050016          .     = 000310R
UIPDR4= 177610          $F$CAS= 000150          $ISK2 = 000001          $TSK4 = 050015

. ABS.  000000     000
        000310     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:MSGDGE,DSKZ:MSGDGE=SPMAC/ML,EQUATE,MSGDGE
 RUN-TIME: 16 6 .4 SECONDS
 RUN-TIME RATIO: 102/23=4.2
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  MSGDHOOK - HOOK MESSAGE TO OUTPUT DRIVER
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE NAME:
                        ;       MSGDHOOK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE HOOKS THE ADDRESS OF THE NEXT MESSAGE
                        ;       TO BE PRINTED TO THE APPROPRIATE OUTPUT DRIVER
                        ;       AND THEN INITIATES THE PRINTING OF THE MESSAGE
                        ;       BY CALLING THE DRIVER AT ITS I/O INITIATOR ENTRY POINT.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. MESSAGE TYPE CODE
                        ;       3. ADDRESS OF MESSAGE
                        ;       4. RETURN ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.ST0
                        ;       2. DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       TTINI
                        ;       LPINI
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL MSGDHOOK IN <DTABLE,MSGCOD,MSGADDR,RETADR>
                        ;               WHERE DTABLE  = ADDRESS OF DATA TABLE
                        ;                     MSGCOD  = MESSAGE TYPE CODE
                        ;                     MSGADDR = MESSAGE ADDRESS
                        ;                     RETADDR = RETURN ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINTIONS AND REFERENCES


                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
(1) 000000'                             $LSTIN = 1
        000001                          $LSTTAG = 1
        000001                          $LSTTAG = 1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  MSGDHOOK                ;MODULE'S ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  TTINI                   ;TERMINAL DRIVER MODULE
                                .GLOBL  LPINI                   ;LINE PRINTER DRIVER MODULE
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  MS.EXT: .WORD   0               ;EXIT THROUGH HERE IF NO TTY(UNDER APT)
                                                                ;AND ITS A MSGPOP
```

```
                                .SBTTL  MSGDHOOK ROUTINE

                                ROUTINE MSGDHOOK <DTABLE,MSGCODE,MSGADDR,RETADR>
                                                                        MSGDHOOK:


                                ;+
                                ; DO THE NECESSARY SET-UP STUFF
                                ;-

                                CALL SAVREG
                                                                        JSR     PC,SAVREG
                                LET R1 := MSGCODE(R5)
                                                                        MOV     MSGCODE(R5),R1
                                LET R2 := MSGADDR(R5)
                                                                        MOV     MSGADDR(R5),R2
                                LET R3 := RETADR(R5)
                                                                        MOV     RETADR(R5),R3
                                LET MS.EXT := R3
                                                                        MOV     R3,MS.EXT
                                LET R0 := DTABLE(R5)
                                                                        MOV     DTABLE(R5),R0


                                ;+
                                ; IF THERE IS NO TTY (UNDER APT) WE WILL BE EXITTING DIRECTLY (BELOW)
                                ;-

                                IF #NOAPTY NOTSETIN DT.CF0(R0) THEN
                                                                        BIT     #NOAPTY,DT.CF0(R
                                                                        BNE     50002$
                                ;+
                                ; IS THE "LPON" KEYBOARD COMMAND IN EFFECT? IF IT IS,
                                ; THIS MESSAGE IS TO GO TO THE LINE PRINTER
                                ;-

                                        IF #LPSTAT SETIN DT.ST0(R0) THEN
                                                                        BIT     #LPSTAT,DT.ST0(R
                                                                        BEQ     50003$
                                            CALL LPINI IN <R0,R1,R2,R3>
                                                                        MOV     R5,-(SP)
                                                                        MOV     R3,-(R5)
                                                                        MOV     R2,-(R5)
                                                                        MOV     R1,-(R5)
                                                                        MOV     R0,-(R5)
                                                                        JSR     PC,LPINI
                                                                        MOV     (SP)+,R5
                                        ELSE
                                                                        BR      50004$
                                                                    50003$:

                                ;+
                                ; IF NO ONE ELSE WANTS IT, PASS IT TO THE TERMINAL OUTPUT DRIVER
```

```
                        ;-
000074'                         CALL TTINI IN <R0,R1,R2,R3>
000074' 010546                                                  MOV     R5,-(SP)
000076' 010345                                                  MOV     R3,-(R5)
000100' 010245                                                  MOV     R2,-(R5)
000102' 010145                                                  MOV     R1,-(R5)
000104' 010045                                                  MOV     R0,-(R5)
000106' 004767  000000G                                         JSR     PC,TTINI
000112' 012605                                                  MOV     (SP)+,R5
000114'                         ENDIF
000114'                                                 50004$:
000114'                 ELSE
000114' 000406                                                  BR      50005$
000116'                                                 50002$:
                        ;+
                        ; IF WE ARE UNDER APT WITH NO TTY,
                        ; CLEAN UP IN AN UNNATURAL WAY (SIMULATE THE NORMAL RETURN
                        ; THAT IS, RESTORE REGS, DO ONE FINAL POP, WHICH KILLS
                        ; THE RTS PC. RESTORE R5  THEN JUMP THRU MS.EXT)
                        ;-
000116'                         CALL RESREG
000116' 004767  000000G                                         JSR     PC,RESREG

000122'                         INLINE <TST     (SP)+>
000122' 005726                                                  TST     (SP)+
000124'                         INLINE <MOV     (SP)+,R5>
000124' 012605                                                  MOV     (SP)+,R5
000126'                         INLINE <JMP @MS.EXT>
000126' 000177  177646                                          JMP @MS.EXT
000132'                 ENDIF
000132'                                                 50005$:


                        ;+
                        ; CHOW BABY!!!!! !!
                        ;-

000132'                 CALL RESREG
000132' 004767  000000G                                         JSR     PC,RESREG
000136'                 ENDRTN
000136'                                                 50000$:
000136'                                                 50001$:
000136' 000207                                                  RTS     PC


000001                          .END
```

```
ACSR  = 000102   CTRLC = 000003   EOPBIT= 000001   MODHOL= 002000   RANNUM= 000054
ACTBIT= 004000   CTRLO = 000017   ERRTYP= 000106   MODSEL= 001000   RBUFEA= 000130
ADDR22= 001000   CTRLU = 000025   EVNTBE= 000200   MSGADD= 000004   RBUFPA= 000126
ADR   = 000006   DCEVNT= 000011   EVNTHD= 000200   MSGCKD= 000010   RBUFSZ= 000132
APTFER= 000004   DEFRTN= 000400   EVNTKT= 000203   MSGCKS= 000011   RBUFVA= 000124
APTPRE= 000200   DIAGMC= 000000   EVNTPE= 000202   MSGCOD= 000002   RDSERV= 000101
ASB   = 000106   DROPMO= 100000   EVNTRE= 000201   MSGDER= 000005   RDWHMI= 000022
ASSEMB= 000010   DSEVNT= 000014   FATERR= 100000   MSGDHO  000002RG RELERR= 000020
ASTAT = 000104   DTABLE= 000000   HRDCNT= 000044   MSGDRP= 000017   RELMOD= 020000
AUTO  = 000010   DT.ADD= 000042   HRDPAS= 000050   MSGECH= 177777   RELTIM= 010000
AUTOST= 020000   DT.AP = 000100   ICONT = 000036   MSGEOP= 000013   RESREG= ****** G
AWAS  = 000110   DT.APK= 000076   ICOUNT= 000040   MSGHDR= 000004   RES1  = 000056
BIT0  = 000001   DT.BLS= 000034   IDNUM = 000122   MSGHNG= 000022   RES2  = 000060
BIT00 = 000001   DT.CF0= 000014   IE    = 000100   MSGHRD= 000007   RETADR= 000006
BIT01 = 000002   DT.CF1= 000016   INDPAR= 000040   MSGMAP= 000021   RICHAR= 031060
BIT02 = 000004   DT.ERR= 000020   INHDRP= 040000   MSGNUL= 177775   RPTDAT= 002000
BIT03 = 000010   DT.ESI= 000044   INHEPR= 020000   MSGPOP= 000002   RSTRT = 000112
BIT04 = 000020   DT.EVN= 000000   INHREL= 001000   MSGPRM= 177776   RUBOUT= 000177
BIT05 = 000040   DT.EXS= 000060   INHRRE= 000400   MSGRES= 000001   RUNMOD= 100000
BIT06 = 000100   DT.FCH= 000037   INIT  = 000030   MSGSFT= 000006   R5VALU= 001740
BIT07 = 000200   DT.FCN= 000036   INTR  = 000120   MSGSKE= 000003   SAM   = 075464
BIT08 = 000400   DT.HMX= 000104   IOMOD = 100000   MSGSMB= 000015   SAVREG= ****** G
BIT09 = 001000   DT.KBE= 000024   IOMODP= 102000   MSGSMH= 000014   SBADR = 000102
BIT1  = 000002   DT.KBP= 000026   IOMODR= 112000   MSGSMS= 000016   SBKMOD= 000000
BIT10 = 002000   DT.KBR= 000022   IOMODX= 110000   MSGSTD= 000000   SBKSEL= 010000
BIT11 = 004000   DT.KBU= 000030   JACK  = 035060   MSGSYS= 000012   SC.ADR= 000006
BIT12 = 010000   DT.MLS= 000032   KIPAR0= 172340   MSGVEC= 000020   SC.ALC= 000014
BIT13 = 020000   DT.MTI= 000110   KIPAR1= 172342   MS.EXT  000000R  SC.APC= 000016
BIT14 = 040000   DT.OFF= 000070   KIPAR2= 172344   NBKMOD= 001000   SC.CKL= 000002
BIT15 = 100000   DT.PAS= 000074   KIPAR3= 172346   NCPUOP= 000020   SC.CKP= 000004
BIT2  = 000004   DT.PC = 000002   KIPAR4= 172350   NOAPTY= 000002   SC.CLO= 000000
BIT3  = 000010   DT.PFL= 000062   KIPAR5= 172352   NULL  = 000000   SC.HLD= 000010
BIT4  = 000020   DT.PSW= 000004   KIPAR6= 172354   OWEN  = 024020   SC.SCA= 000012
BIT5  = 000040   DT.PTA= 000064   KIPAR7= 172356   PAERR = 000010   SENDLS= 177777
BIT6  = 000100   DT.RCS= 000102   KIPDR0= 172300   PARPRE= 002000   SOFCNT= 000042
BIT7  = 000200   DT.REL= 000040   KIPDR1= 172302   PARSTA= 000100   SOFPAS= 000046
BIT8  = 000400   DT.SCT= 000066   KIPDR2= 172304   PASCNT= 000034   SPACE = 000040
BIT9  = 001000   DT.SMX= 000106   KIPDR3= 172306   PDPLSI= 020000   SPOINT= 000032
BKDEF = 000002   DT.SP = 000006   KIPDR4= 172310   PDP60 = 004000   SPVALU= 002200
BKMOD = 000020   DT.SSI= 000046   KIPDR5= 172312   PDP70 = 010000   SR0   = 177572
BKMODE= 040000   DT.ST0= 000010   KIPDR6= 172314   PRI0  = 000000   SR1   = 177574
BKSLSH= 000134   DT.ST1= 000012   KIPDR7= 172316   PRI1  = 000040   SR2   = 177576
CAPRES= 000004   DT.SWR= 000056   KTERRO= 000040   PRI4  = 000200   SR3   = 172516
CASTAT= 000004   DT.SYP= 000072   KTPRES= 000400   PRI5  = 000240   STAT  = 000026
CDERCT= 000146   DT.WBU= 000050   KTSTAT= 000020   PRI6  = 000300   STATBI= 064757
CDWDCT= 000144   DT.WHL= 000054   KTXTND= 040000   PRI7  = 000340   STAT1 = 000027
CKTIM = 100000   DT.WLL= 000052   LF    = 000012   PR0   = 000000   SUSPND= 000001
CLKPRE= 000001   DVID1 = 000014   LPINI = ****** G  PR4   = 000200   SVR0  = 000062
CONFIG= 000056   ECCMEM= 000100   LPSTAT= 000001   PR5   = 000240   SVR1  = 000064
CQOVF = 000001   ECCSTA= 000010   MAPSTA= 000200   PR6   = 000300   SVR2  = 000066
CR    = 000015   ENBEOP= 010000   MED   = 076600   PR7   = 000340   SVR3  = 000070
CSRA  = 000100   ENBNUL= 000001   MEMPAS= 040000   PS    = 177776   SVR4  = 000072
CSRC  = 000102   ENDLST= 000000   MODEXH= 004000   PSW   = 177776   SVR5  = 000074
```

```
SVR6  = 000076          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$ERFL= 000000
SYSCNT= 000052          UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$FLAG= 000001
SYSERR= 000100          WASADR= 000104          $F$DO = 000340          $LOCTA= 177777          $$FROM= 000000
TMPIO = 000002          WBSTAT= 000040          $F$FAL= 000405          $LSTIN= 000001          $$LOC = 000050R
TQOVF = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
TTINI = ****** G        WBUFPA= 000134          $F$IF = 000110          $NESTL= 177777          $$REG = 177777
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK0 = 000300          $$RETU= 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK1 = 000110          $$RTN1= 050000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK2 = 000110          $$RTN2= 050001
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $SAVLE= 177777          $$SRC = 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TAGNU= 050006          $$TGS1= 000000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TEMP = 000300          $$TGS2= 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TSK0 = 050005          $$TO  = 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TSK1 = 050004          $$$TAG= 050000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $$ARGC= 000010          .     = 000140R
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $$BYTE= 000403
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $$CASE= 000000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$DST = 000000
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$ELOC= 000402

. ABS.  000000     000
        000140     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:MSGDHO,DSKZ:MSGDHO=SPMAC/ML,EQUATE,MSGDHO
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 69/16=4.3
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  NAMCHK VERIFY A MODULE NAME
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       NAMCHK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE IS CALLED TO VERIFY A MODULE NAME. IT
                        ;       RECEIVES THE ADDRESS OF THE FIRST CHARACTER OF THE MODULE
                        ;       NAME AND SEARCHES THE MODULE LIST FOR A FIVE CHARACTER MATCH.
                        ;       IF THE MODULE NAME IS FOUND, THE HEADER ADDRESS OF THE
                        ;       MODULE IS RETURNED. IF IT IS NOT FOUND, A RETURN WITH
                        ;       ERROR OCCURS. THE UPDATED COMMAND DECODE BUFFER IS
                        ;       RETURNED WHETHER OR NOT THERE WAS AN ERROR.
                        ;       A RETURN WITH ERROR OCCURS IF THE MODULE NAME IS
                        ;       MORE THAN 5 CHARACTERS IN LENGTH.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;
                        ; OUTPUTS:
                        ;       1. HEADER ADDRESS OF MODULE
                        ;       2. UPDATED COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL NAMCHK IN <DTADR,MODBUF> OUT <HEAD,NEWBUF>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;             MODBUF = ADDRESS OF 5 BYTE BUFFER CONTAINING MODULE NAME
                        ;             HEAD = HEADER ADDRESS OF VERIFIED MODULE
                        ;             NEWBUF = UPDATED DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR NAMCHK

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  NAMCHK          ;ROUTINE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
```

```
                                .SBTTL  NAMCHK ROUTINE

000000'                         ROUTINE NAMCHK  <DTADR,MODBUF,HEAD,NEWBUF>
(2) 000000'                                                                 NAMCHK:


                                ;+
                                ; SAVE REGISTERS, UPDATE CMD DECODE BUFFER POINTER, AND SEE IF IT'S
                                ; A SPACE OR <CR>.... IF NOT, POP R1 AND RETURN ERROR BECAUSE
                                ; MODULE NAME MAY NOT EXCEED 5 CHARACTERS.....
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                                 JSR     PC,SAVREG
000004'                         LET R1 := MODBUF(R5)
(4) 000004' 016501  000002                                                  MOV     MODBUF(R5),R1
000010'                         LET NEWBUF(R5) := R1 + #5
(4) 000010' 010165  000006                                                  MOV     R1,NEWBUF(R5)
(6) 000014' 062765  000005  000006                                          ADD     #5,NEWBUF(R5)
000022'                         IFB 5(R1) NE #SPACE ANDB 5(R1) NE #CR THEN
(6) 000022' 126127  000005  000040                                          CMPB    5(R1),#SPACE
(9) 000030' 001405                                                          BEQ     50002$
(6) 000032' 126127  000005  000015                                          CMPB    5(R1),#CR
(9) 000040' 001401                                                          BEQ     50002$
000042'                                 INLINE <BR      1$>
(2) 000042' 000433                                                          BR      1$
000044'                         ENDIF
(4) 000044'                                                                 50002$:

                                ;+
                                ; SAVE DTABLE ADR, THEN MODULE LIST
                                ;-

000044'                         LET R0 := DTADR(R5)
(4) 000044' 016500  000000                                                  MOV     DTADR(R5),R0
000050'                         LET R0 := DT.MLST(R0)
(4) 000050' 016000  000032                                                  MOV     DT.MLST(R0),R0

                                ;+
                                ; SAVE FIRST MODULE ADDR IN R2
                                ;-

000054'                         LET R2 := (R0)
(4) 000054' 011002                                                          MOV     (R0),R2

                                ;+
                                ; WHILE MODULE LIST IS NOT EXHAUSTED (=0).....
                                ;-

000056'                         WHILE (R0) NE #ENDLST DO
(4) 000056'                                                                 50003$:
(6) 000056' 021027  000000                                                  CMP     (R0),#ENDLST
(9) 000062' 001423                                                          BEQ     50004$

                                ;+
                                ; INIT COUNT - WHILE CHARACTER IN MODULE HEADER = CHARACTER IN
```

```
                                ; BUFFER......
                                ;-
                                        LET R3 := #5
000064'                                                                 MOV     #5,R3
(4) 000064' 012703  000005
                                        WHILEB (R2)+ EQ (R1)+ DO
000070'                                                         50005$:
(4) 000070'                                                             CMPB    (R2)+,(R1)+
(6) 000070' 122221                                                      BNE     50006$
(9) 000072' 001011
                                ;+
                                ; ADVANCE TO NEXT CHARACTER, DECREMENT COUNT, SEE IF COUNT = 0.
                                ; IF IT IS ZERO, SAVE CURRENT HEADER ADDRESS (R0) AND
                                ; RETURN WITH NO ERROR.
                                ;-
                                                LET R3 := R3 - #1
000074'                                                                 DEC     R3
(6) 000074' 005303
                                                IF R3 EQ #0 THEN
000076'                                                                 TST     R3
(6) 000076' 005703                                                      BNE     50007$
(9) 000100' 001005
000102'                                                 LET HEAD(R5) := (R0)
(4) 000102' 011065  000004                                              MOV     (R0),HEAD(R5)
                                                CALL RESREG
000106'                                                                 JSR     PC,RESREG
(3) 000106' 004767  000000G
                                                RETURN NO.ERROR
000112'                                                                 BR      50000$
(4) 000112' 000413
                                                ENDIF
000114'                                                         50007$:
(4) 000114'
                                        ENDDO
000114'                                                                 BR      50005$
(4) 000114' 000765                                              50006$:
(3) 000116'
                                ;+
                                ; GET TO THE NEXT TABLE ADDRESS
                                ;-
                                        LET R0 := R0 + #2
000116'                                                                 ADD     #2,R0
(6) 000116' 062700  000002
                                ;+
                                ; SAVE NEW MODULE ADDRESS IN R2
                                ;-
                                        LET R2 := (R0)
000122'                                                                 MOV     (R0),R2
(4) 000122' 011002
                                ;+
                                ; RESTORE MODULE BUFFER POINTER
                                ;-
                                        LET R1 := MODBUF(R5)
000124'                                                                 MOV     MODBUF(R5),R1
(4) 000124' 016501  000002
                                ENDDO
000130'                                                                 BR      50003$
(4) 000130' 000752
```

```
(3)  000132'                                                                    50004$:
668
669                             ;+
670                             ; CLEAN UP AND RETURN WITH ERROR BECAUSE MODULE NAME NOT FOUND
671                             ;-
672
673  000132'                    INLINE <1$:>
(2)  000132'                                                                    1$:
674  000132'                    CALL RESREG
(3)  000132' 004767  000000G                                                   JSR     PC,RESREG
675
676  000136'                    RETURN ERROR
(2)  000136' 000261                                                            SEC
(4)  000140' 000401                                                            BR      50001$
677
678  000142'                    ENDRTN
(3)  000142'                                                                    50000$:
(2)  000142' 000241                                                            CLC
(3)  000144'                                                                    50001$:
(2)  000144' 000207                                                            RTS     PC
679
680          000001            .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HEAD  = 000004
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODBUF= 000002
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPDP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK  000000RG
NBKMOD= 001000
NCPUOP= 000020
NEWBUF= 000006
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LSTTA= 000001          $$ELOC= 000403
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $NESTL= 177777          $$ERFL= 000000
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $NSK0 = 000300          $$FLAG= 000001
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK1 = 000120          $$FROM= 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK2 = 000120          $$LOC = 000100R
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK3 = 000110          $$LOCN= 000000
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $SAVLE= 177777          $$REG = 177777
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $SSK0 = 050004          $$RETU= 000001
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TAGLE= 177777          $$RTN1= 050000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGNU= 050010          $$RTN2= 050001
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TEMP = 000300          $$SRC = 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TSK0 = 050003          $$TGSV= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK1 = 050004          $$TGS1= 000000
UIPDR2= 177604          $ERFLG= 000000          $F$UNT= 000130          $TSK2 = 050005          $$TGS2= 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK3 = 050006          $$TO  = 000000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $TSK4 = 050007          $$$TAG= 050000
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$ARGC= 000010          .     = 000146R
UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$BYTE= 000403
UIPDR7= 177616          $F$DEC= 000220          $LOCTA= 177777          $$CASE= 000000
WASADR= 000104          $F$DO = 000340          $LSTIN= 000001          $$DST = 000000

. ABS.  000000     000
        000146     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:NAMCHK,DSKZ:NAMCHK=SPMAC/ML,EQUATE,NAMCHK
 RUN-TIME: 13 3 .4 SECONDS
 RUN-TIME RATIO: 79/18=4.3
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE NEWBA - ESTABLISH NEW BASE ADDRESS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       NEWBA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CALCULATES A NEW RELOCATION BASE ADDRESS BY USING
                        ;       EITHER THE CONSTANT OFFSET OR A RANDOMLY GENERATED
                        ;       OFFSET VALUE
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS, CURRENT ADDRESS OFFSET,
                        ;       FIRST RELOCATION FLAG
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.REL, DT.ESIZ, DT.SSIZ, DT.SWR, DT.STO
                        ;
                        ; OUTPUTS:
                        ;       NEW ADDRESS OFFSET
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       RANDOM                  ;RANDOM NUMBER GENERATOR
                        ;       SAVREG                  ;SAVE REGISTERS
                        ;       RESREG                  ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL NEWBA IN <A,B C> OUT <D>
                        ;               A=ADDRESS OF DATA TABLE
                        ;               B=CURRENT ADDRESS OFFSET
                        ;               C=FIRST RELOCATION FLAG
                        ;               D=NEW ADDRESS OFFSET
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFENITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                    000001      $LSTIN=1
                    000001      $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  NEWBA                   ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  RANDOM                  ;RANDOM NUMBER GENERATOR
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  NB.CNT:         .WORD 0         ;NO. OF RANDOM RELOCATIONS SINCE LAST
                                                                ;   MOVE TO BOTTOM OF MEMORY
000002' 000000                  NB.CYCLNG:      .WORD 0         ;CYCLING FLAG - INDICATES, WHEN SET, RELOCATING
                                                                ;   BY CONSTANT OFFSET (CYCLING)
000004' 000000                  NB.MAX:         .WORD 0         ;MAX. NO. OF RANDOM RELOCATIONS ALLOWED
                                                                ;   BEFORE RELOCATING BACK TO BOTTOM
000006' 000000                  NB.NBR:         .WORD 0         ;STORAGE FOR RANDOM NUMBER
000010' 000000                  NB.STADDR:      .WORD 0         ;STORAGE FOR EXERCISER'S STARTING ADDRESS
                                                                ;   OFFSET VALUE
000012' 000000                  NB.OFLAG:       .WORD 0         ;O-FLAG - INDICATES, WHEN SET, THAT THERE
                                                                ;   HAS BEEN A RELOCATION TO BOTTOM
000014' 000000                  NB.TOP:         .WORD 0         ;OFFSET NEEDED TO MOVE THE EXERCISER
                                                                ;   TO HIGHEST MEMORY
000016' 000000                  NB.BIT:         .WORD 0         ;BIT PATTERN FOR USE WITH RANDOM NUMBER
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL NEWBA ROUTINE

                                ROUTINE NEWBA <TABL,OLDPAR,RL1,NEWPAR>
                                                                        NEWBA:


                                ;+
                                ;SAVE REGISTERS
                                ;-

                                CALL SAVREG
                                                                        JSR     PC,SAVREG

                                ;+
                                ;SET R0 TO START OF DATA TABLE
                                ;-

                                LET R0 := TABL(R5)
                                                                        MOV     TABL(R5),R0

                                ;+
                                ;IF THIS IS THE FIRST RELOCATION, DO THE NECESSARY INITIALIZATION
                                ;-

                                IF RL1(R5) EQ #1 THEN
                                                                        CMP     RL1(R5),#1
                                                                        BNE     50002$

                                ;+
                                ;INITIALIZE FLAGS AND NB.CNT
                                ;-

                                        LET NB.CYCLNG := #1
                                                                        MOV     #1,NB.CYCLNG
                                        LET NB.0FLAG := #0
                                                                        CLR     NB.0FLAG
                                        LET NB.CNT := #0
                                                                        CLR     NB.CNT

                                ;+
                                ;SAVE THE INITIAL ADDRESS OFFSET
                                ;-

                                        LET NB.STADDR := OLDPAR(R5)
                                                                        MOV     OLDPAR(R5),NB.ST

                                ;+
                                ;CALCULATE NB.MAX BY DIVIDING BY 2 THE NUMBER OF K'S OF MEMORY.
                                ;THIS IS DONE BY GETTING THE CONTENTS OF DT.SSIZ AND SHIFTING
                                ;TO THE RIGHT 6 TIMES. (5 TIMES TO GET NO. OF K'S, AND ONCE MORE TO
                                ;DIVIDE BY 2).
                                ;DON'T, HOWEVER, ALLOW NB.MAX TO BE GREATER THAN 30(10).
```

```
650                                     ;-
651
652  000064'                                    LET R1 := DT.SSIZ(R0)
(4)  000064' 016001  000046                                 MOV     DT.SSIZ(R0),R1
653  000070'                                    LET R1 := R1 SHIFT #-6
(7)  000070' 006201                                         ASR     R1
(7)  000072' 006201                                         ASR     R1
(7)  000074' 006201                                         ASR     R1
(7)  000076' 006201                                         ASR     R1
(7)  000100' 006201                                         ASR     R1
(7)  000102' 006201                                         ASR     R1
654  000104'                                    IF R1 GT #^D30 THEN
(6)  000104' 020127  000036                                 CMP     R1,#^D30
(9)  000110' 003402                                         BLE     50003$
655  000112'                                            LET R1 := #^D30
(4)  000112' 012701  000036                                 MOV     #^D30,R1
656  000116'                                    ENDIF
(4)  000116'                                50003$:
657  000116'                                    LET NB.MAX := R1
(4)  000116' 010167  177662                                 MOV     R1,NB.MAX
```

```
                                ;+
                                ;CALCULATE THE ADDRESS OFFSET NEEDED TO MOVE THE EXERCISER TO THE
                                ;TOP OF CORE.  THIS IS DONE USING THE FOLLOWNG ALGORITHM:
                                ; 1. GET THE SIZE OF THE MOVABLE PORTION OF THE EXERCISER
                                ; 2. SHIFT OUT THE SIX LEAST SIGNIFICANT BITS
                                ; 3. ADD ONE TO THE RESULT TO ALLOW ROOM FOR THE BLOCKS JUST
                                ;    SHIFTED OUT
                                ; 4. SUBTRACT THIS FROM THE SYSTEM SIZE.  IF, HOWEVER, THE
                                ;    UNIBUS MAP IS SHUT OFF, (IT MAY NOT
                                ;    EVEN EXIST), THEN DON'T ALLOW RELOCATION ABOVE 124K.
                                ;-
000122'                                 LET R2 := DT.ESIZ(R0) - #20000
(4) 000122' 016002  000044                                              MOV     DT.ESIZ(R0),R2
(6) 000126' 162702  020000                                              SUB     #20000,R2
000132'                                 LET R2 := R2 SHIFT #-6
(7) 000132' 006202                                                      ASR     R2
(7) 000134' 006202                                                      ASR     R2
(7) 000136' 006202                                                      ASR     R2
(7) 000140' 006202                                                      ASR     R2
(7) 000142' 006202                                                      ASR     R2
(7) 000144' 006202                                                      ASR     R2
000146'                                 LET R2 := R2 CLR.BY #176000
(6) 000146' 042702  176000                                              BIC     #176000,R2
000152'                                 LET R2 := R2 + #1
(6) 000152' 005202                                                      INC     R2

000154'                                 IF #MAPSTAT NOTSETIN DT.ST0(R0) AND DT.SSIZ(R0) HIS #7600 THEN
(6) 000154' 032760  000200  000010                                      BIT     #MAPSTAT,DT.ST0(
(9) 000162' 001007                                                      BNE     50004$
(6) 000164' 026027  000046  007600                                      CMP     DT.SSIZ(R0),#760
(9) 000172' 103403                                                      BLO     50004$
000174'                                         LET R1 := #7600
(4) 000174' 012701  007600                                              MOV     #7600,R1
000200'                                 ELSE
(4) 000200' 000402                                                      BR      50005$
(3) 000202'                                                     50004$:
000202'                                         LET R1 := DT.SSIZ(R0)
(4) 000202' 016001  000046                                              MOV     DT.SSIZ(R0),R1
000206'                                 ENDIF
(4) 000206'                                                     50005$:

000206'                                 LET NB.TOP := R1 - R2
(4) 000206' 010167  177602                                              MOV     R1,NB.TOP
(6) 000212' 160267  177576                                              SUB     R2,NB.TOP


000216'                         ENDIF
(4) 000216'                                                     50002$:


                                ;+
                                ;ARE WE RELOCATING BY THE CONSTANT OFFSET (CYCLING) ?   WE ARE IF
```

```
                        ;THE CYCLING FLAG IS SET OR IF BIT 8 OF THE SOFTWARE SWITCH
                        ;REGISTER IS SET.
                        ;-

000216'                 IF NB.CYCLNG EQ #1 OR #BIT08 SETIN DT.SWR(R0) THEN
(6) 000216' 026727 177560 000001                                        CMP     NB.CYCLNG,#1
(8) 000224' 001404                                                      BEQ     50006$
(6) 000226' 032760 000400 000056                                        BIT     #BIT08,DT.SWR(R0
(9) 000234' 001435                                                      BEQ     50007$
(6) 000236'                                                         50006$:


                        ;+
                        ;RELOCATING BY CONSTANT OFFSET (CYCLING)
                        ;-

                        ;+
                        ;ADD THE OFFSET CONSTANT TO THE CURRENT ADDRESS OFFSET TO FIND
                        ;A NEW BASE
                        ;-

000236'                         LET NEWPAR(R5) := OLDPAR(R5) + DT.REL(R0)
(4) 000236' 016565 000002 000006                                        MOV     OLDPAR(R5),NEWPA
(6) 000244' 066065 000040 000006                                        ADD     DT.REL(R0),NEWPA


                        ;+
                        ;SEE IF THIS NEW BASE VALUE IS TOO LARGE FOR THE SYSTEM.
                        ;-

000252'                         IF NEWPAR(R5) HI NB.TOP THEN
(6) 000252' 026567 000006 177534                                        CMP     NEWPAR(R5),NB.TO
(9) 000260' 101406                                                      BLOS    50010$


                        ;+
                        ;THE NEW ADDRESS WILL BE TOO HIGH - WE'VE HIT THE TOP OF CORE, SO
                        ;IT'S TIME TO RELOCATE TO THE BOTTOM.  THE NEW OFFSET VALUE WILL BE 200.
                        ;THIS WILL PUT THE EXERCISER BACK TOGETHER AGAIN AT THE BOTTOM OF
                        ;MEMORY.  SET NB.0FLAG TO INDICATE WE'VE GONE TO THE BOTTOM, AND RETURN.
                        ;-

000262'                             LET NEWPAR(R5) := #200
(4) 000262' 012765 000200 000006                                        MOV     #200,NEWPAR(R5)
000270'                             LET NB.0FLAG := #1
(4) 000270' 012767 000001 177514                                        MOV     #1,NB.0FLAG


000276'                         ENDIF
(4) 000276'                                                         50010$:


                        ;+
                        ;NEW ADDRESS IS OK.  BUT BEFORE WE LEAVE, WE HAVE TO SEE IF WE HAVE
                        ;FINISHED CYCLING.  THIS IS DONE BY CHECKING IF THE NEW BASE IS
                        ;GREATER THAN OR EQUAL TO THE STARTING ADDRESS OFFSET, AND IF NB.0FLAG
```

```
                                ;IS SET
                                ;-

                                        IF NEWPAR(R5) HIS NB.STADDR AND NB.0FLAG EQ #1 THEN
(6)  000276' 026567  000006  177504                                                    CMP     NEWPAR(R5),NB.ST
(9)  000304' 103410                                                                    BLO     50011$
(6)  000306' 026727  177500  000001                                                    CMP     NB.0FLAG,#1
(9)  000314' 001004                                                                    BNE     50011$

                                ;+
                                ;CYCLING IS COMPLETED.  CLEAR THE CYCLING FLAG.
                                ;ALSO CLEAR NB.0FLAG.
                                ;-

                                                LET NB.CYCLNG := #0
(4)  000316' 005067  177460                                                            CLR     NB.CYCLNG
                                                LET NB.0FLAG := #0
(4)  000322' 005067  177464                                                            CLR     NB.0FLAG

                                        ENDIF
(4)  000326'                                                                50011$:

                                ELSE
(4)  000326' 000463                                                                    BR      50012$
(3)  000330'                                                                50007$:

                                ;+
                                ;RELOCATING BY RANDOM OFFSET VALUES
                                ;-

                                ;+
                                ;IF THE LAST RELOCATION WAS BACK TO THE BOTTOM OF MEMORY, THEN
                                ;WE MUST NOW RELOCATE TO THE TOP-MOST PIECE OF MEMORY.
                                ;-

                                        IF NB.0FLAG EQ #1 THEN
(6)  000330' 026727  177456  000001                                                    CMP     NB.0FLAG,#1
(9)  000336' 001006                                                                    BNE     50013$

                                ;+
                                ;GET THE PREVIOUSLY CALCULATED OFFSET NEEDED TO GET US TO THE TOP
                                ;OF MEMORY.  ALSO CLEAR NB.0FLAG.
                                ;-

                                                LET NEWPAR(R5) := NB.TOP
(4)  000340' 016765  177450  000006                                                    MOV     NB.TOP,NEWPAR(R5
                                                LET NB.0FLAG := #0
(4)  000346' 005067  177440                                                            CLR     NB.0FLAG

                                        ELSE
(4)  000352' 000451                                                                    BR      50014$
(3)  000354'                                                                50013$:
```

```
                                ;+
                                ;IF THE NUMBER OF PREVIOUS RANDOM RELOCATIONS EQUALS THE MAXIMUM,
                                ;THEN RELOCATE TO THE BOTTOM OF CORE, SET NB.0FLAG, AND RESET THE
                                ;RELOCATION COUNTER
                                ;-

000354'                                 IF NB.CNT EQ NB.MAX THEN
(6) 000354' 026767 177420 177422                                        CMP     NB.CNT,NB.MAX
(9) 000362' 001011                                                      BNE     50015$
000364'                                         LET NEWPAR(R5) := #200
(4) 000364' 012765 000200 000006                                        MOV     #200,NEWPAR(R5)
000372'                                         LET NB.0FLAG := #1
(4) 000372' 012767 000001 177412                                        MOV     #1,NB.0FLAG
000400'                                         LET NB.CNT := #0
(4) 000400' 005067 177374                                               CLR     NB.CNT

000404'                                 ELSE
(4) 000404' 000434                                                      BR      50016$
(3) 000406'                                                     50015$:

                                ;+
                                ;GET A RANDOM NUMBER AND SEE IF IT CAN BE USED TO CREATE AN ADDRESS
                                ;WHICH IS GREATER THAN THE HIGHEST ADDRESS OF THE LOW 4K OF THE
                                ;EXERCISER AND SMALLER THAN OR EQUAL TO NB.TOP.
                                ;IF IT IS TOO BIG, START CLEARING HIGH ORDER BITS ONE AT A
                                ;TIME UNTIL THE NUMBER IS SMALL ENOUGH.  THEN CHECK TO SEE IF IT IS
                                ;TOO SMALL.  IF IT IS, GET ANOTHER RANDOM NUMBER AND START
                                ;AGAIN.   WHEN A GOOD VALUE IS FOUND, SET UP THE OUTPUT AND INCREMENT
                                ; THE COUNTER OF THE NUMBER OF RELOCATIONS.
                                ;-

000406'                         REPEAT
(3) 000406'                                                     50017$:
000406'                                 LET NB.BIT := #100000
(4) 000406' 012767 100000 177402                                        MOV     #100000,NB.BIT
000414'                                 CALL RANDOM OUT <NB.NBR>
(4) 000414' 162705 000002                                               SUB     #1*2,R5
(3) 000420' 004767 000000G                                              JSR     PC,RANDOM
(4) 000424' 012567 177356                                               MOV     (R5)+,NB.NBR
000430'                                 WHILE NB.NBR HI NB.TOP DO
(4) 000430'                                                     50020$:
(6) 000430' 026767 177352 177356                                        CMP     NB.NBR,NB.TOP
(9) 000436' 101406                                                      BLOS    50021$
000440'                                         LET NB.NBR := NB.NBR CLR.BY NB.BIT
(6) 000440' 046767 177352 177340                                        BIC     NB.BIT,NB.NBR
000446'                                         LET NB.BIT := NB.BIT SHIFT #-1
(7) 000446' 006267 177344                                               ASR     NB.BIT
000452'                                 ENDDO
(4) 000452' 000766                                                      BR      50020$
(3) 000454'                                                     50021$:
000454'                         UNTIL NB.NBR HIS #200
(3) 000454' 026727 177326 000200                                        CMP     NB.NBR,#200
(6) 000462' 103751                                                      BLO     50017$
```

```
  813  000464'                                        LET NEWPAR(R5) := NB.NBR
  (4)  000464' 016765  177316  000006                                          MOV     NB.NBR,NEWPAR(R5
  814  000472'                                        LET NB.CNT := NB.CNT + #1
  (6)  000472' 005267  177302                                                 INC     NB.CNT
  815
  816  000476'                                    ENDIF
  (4)  000476'                                                          50016$:
  817  000476'                                ENDIF
  (4)  000476'                                                          50014$:
  818  000476'                            ENDIF
  (4)  000476'                                                          50012$:
  819
  820
  821                                     ;+
  822                                     ;RESTORE REGISTERS AND RETURN
  823                                     ;+
  824
  825  000476'                            CALL RESREG
  (3)  000476' 004767  000000G                                                JSR     PC,RESREG
  826
  827  000502'                            ENDRTN
  (3)  000502'                                                          50000$:
  (3)  000502'                                                          50001$:
  (2)  000502' 000207                                                         RTS     PC
  828
  829          000001                     .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NB.BIT  000016R
NB.CNT  000000R
NB.CYC  000002R
NB.MAX  000004R
NB.NBR  000006R
NB.STA  000010R
NB.TOP  000014R
NB.OFL  000012R
NCPUOP= 000020
NEWBA   000020RG
NEWPAR= 000006
NOAPTY= 000002
NULL  = 000000
OLDPAR= 000002
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANDOM= ****** G
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RL1   = 000004
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
RSVALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
```

```
SVR0  = 000062        UIPDR3= 177606        $F$CAS= 000150        $LOCTA= 177777        $$CASE= 000000
SVR1  = 000064        UIPDR4= 177610        $F$DEC= 000220        $LSTIN= 000001        $$DST = 000000
SVR2  = 000066        UIPDR5= 177612        $F$DO = 000340        $LSTTA= 000001        $$ELOC= 000402
SVR3  = 000070        UIPDR6= 177614        $F$FAL= 000405        $NESTL= 177777        $$ERFL= 000000
SVR4  = 000072        UIPDR7= 177616        $F$GOO= 000400        $NSK0 = 000300        $$FLAG= 000001
SVR5  = 000074        WASADR= 000104        $F$IF = 000110        $NSK1 = 000110        $$FROM= 000000
SVR6  = 000076        WBSTAT= 000040        $F$INC= 000210        $NSK2 = 000110        $$LOC = 000462R
SYSCNT= 000052        WBUFEA= 000136        $F$LOO= 000200        $NSK3 = 000110        $$LOCN= 000000
SYSERR= 000100        WBUFPA= 000134        $F$NAM= 000160        $NSK4 = 000130        $$REG = 177777
TABL  = 000000        WBUFRQ= 000140        $F$NO = 000403        $NSK5 = 000120        $$RETU= 000000
TMPIO = 000002        WBUFSZ= 000142        $F$OR = 000320        $SAVLE= 177777        $$RTN1= 050000
TQOVF = 000002        WDFR  = 000116        $F$RTI= 000350        $SSK0 = 050021        $$RTN2= 050001
UIPAR0= 177640        WDTO  = 000114        $F$RTN= 000300        $TAGLE= 177777        $$SRC = 000000
UIPAR1= 177642        WTINRE= 000352        $F$SEL= 000140        $TAGNU= 050022        $$TGSV= 000000
UIPAR2= 177644        WTWHMI= 000222        $F$THE= 000330        $TEMP = 000300        $$TGS1= 000000
UIPAR3= 177646        XFLAG = 000005        $F$TRU= 000404        $TSK0 = 050012        $$TGS2= 000000
UIPAR4= 177650        XOFF  = 000023        $F$UNT= 000130        $TSK1 = 050014        $$TO  = 000000
UIPAR5= 177652        XON   = 000021        $F$WHI= 000120        $TSK2 = 050016        $$$TAG= 050000
UIPAR6= 177654        $BGNLE= 177777        $F$YES= 000402        $TSK3 = 050017        .     = 000504R
UIPAR7= 177656        $ERFLG= 000400        $IFLEV= 177777        $TSK4 = 050020
UIPDR0= 177600        $F$AND= 000310        $ISK0 = 000001        $TSK5 = 050021
UIPDR1= 177602        $F$BAD= 000401        $ISK1 = 000001        $$ARGC= 000010
UIPDR2= 177604        $F$BLA= 000170        $ISK2 = 000001        $$BYTE= 000403

. ABS.  000000    000
        000504    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:NEWBA,DSKZ:NEWBA=SPMAC/ML,EQUATE,NEWBA
 RUN-TIME: 18 8 .4 SECONDS
 RUN-TIME RATIO: 84/28=2.9
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  NUMCHK OCTAL ASCII TO BINARY CONVERSION ROUTINE
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       NUMCHK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE CONVERTS A 6 DIGIT OCTAL ASCII NUMBER
                        ;       INTO A 16 BIT BINARY NUMBER. IT WILL VERIFY THAT THE NUMBER
                        ;       IS NOT MORE THAN 16 BITS IN LENGTH(LEADING ZEROES ARE
                        ;       ARE NOT COUNTED AS PART OF THE 16 BITS) AND THAT IT IS AN OCTAL NUMBER.
                        ;       A RETURN WITH ERROR OCCURS IF THE NUMBER IS GREATER THAN 16
                        ;       BITS IN LENGTH OR IT IS NOT OCTAL ( I.E. SOMETHING
                        ;       OTHER THAN THE NUMBERS 0-7 WAS DETECTED IN THE NUMBER).
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       1. UPDATED COMMAND DECODE BUFFER POINTER
                        ;       2. THE BINARY NUMBER
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP - THE COMMAND RESPONSE LOCATION
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.NUM                  ;ERROR MESSAGE
                        ;       CM.NBAD                 ;ERROR MESSAGE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL NUMCHK IN <DTADR,CMDPTR> OUT <NEWPTR,NUM>
                        ;       WHERE:
                        ;       DTADR = ADDRESS OF DATA TABLE
                        ;       CMDPTR = COMMAND DECODE BUFFER POINTER
                        ;       NEWPTR = UPDATED CMD DECODE BUFFER POINTER
                        ;       NUM = CONVERTED 16 BIT BINARY NUMBER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINTIONS AND REFERENCES FOR NUMCHK

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

                        000001  $LSTIN=1
                        000001  $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  NUMCHK                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CM.NUM          ;'NUMBER TOO LARGE' MSG ADDRESS
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  CM.NBAD         ;'NOT AN OCTAL NUMBER' MSG ADDRESS
                                ;
                                ;****************
```

```
                                .SBTTL  NUMCHK ROUTINE

                                ROUTINE NUMCHK <DTADR,CMDBUF,NEWPTR,NUM>
000000'                                                                 NUMCHK:
(2)  000000'

                                ;+
                                ; SAVE REGISTERS, DATA TABLE ADDRESS AND CMD DECODE BUFFER POINTER
                                ;-

                                CALL SAVREG
000000'                                                                         JSR     PC,SAVREG
(3)  000000' 004767  000000G

                                LET R0 := DTADR(R5)
000004'                                                                         MOV     DTADR(R5),R0
(4)  000004' 016500  000000
                                LET R1 := CMDBUF(R5)
000010'                                                                         MOV     CMDBUF(R5),R1
(4)  000010' 016501  000002

                                ;+
                                ; INITIALIZE THE OUTPUT REG
                                ;-

                                LET R2 := #0
000014'                                                                         CLR     R2
(4)  000014' 005002

                                ;+
                                ; WHILE THERE IS AN OCTAL ASCII CHAR IN CMD DECODE BUFFER.....
                                ;-

                                WHILEB (R1) GE #'0 ANDB (R1) LE #'7 DO
000016'                                                                 50002$:
(4)  000016'
(6)  000016' 121127  000060                                                     CMPB    (R1),#'0
(9)  000022' 002425                                                             BLT     50003$
(6)  000024' 121127  000067                                                     CMPB    (R1),#'7
(9)  000030' 003022                                                             BGT     50003$

                                ;+
                                ; INITIALIZE THE COUNTER.
                                ;-

                                LET R4 := #0
000032'                                                                         CLR     R4
(4)  000032' 005004

                                ;+
                                ; GET A CHARACTER, SUBTRACT ASCII ZERO, ROTATE 3 LEFT (TO MAKE ROOM),
                                ; AND CHECK TO SEE IF NUM IS NOW > 16 BITS. IF IT IS, STUFF
                                ; ERROR MSG AND RETURN. IF NOT, MOVE R3 TO R2.
                                ;-

                                        LET R3 :B= (R1)+
000034'                                                                         MOVB    (R1)+,R3
(4)  000034' 112103
                                        LET R3 := R3 - #'0
000036'                                                                         SUB     #'0,R3
(6)  000036' 162703  000060
                                        WHILE R4 NE #3 DO
000042'                                                                 50004$:
(4)  000042'
(6)  000042' 020427  000003                                                     CMP     R4,#3
```

```
(9)  000046' 001411                                                                    BEQ     50005$
634  000050'                                LET CARRY := 0
(4)  000050' 000241                                                                    CLC
635  000052'                                LET R2 := R2 ROTATE 1
(7)  000052' 006102                                                                    ROL     R2
636  000054'                                IFCOND CS THEN
(6)  000054' 103004                                                                    BCC     50006$
637  000056'                                       LET DT.KBRSP(R0) := #CM.NUM
(4)  000056' 012760  000000G 000022                                                    MOV     #CM.NUM,DT.KBRSP
638  000064'                                       INLINE <BR      1$>
(2)  000064' 000431                                                                    BR      1$
639  000066'                                ENDIF
(4)  000066'                                                                   50006$:
640  000066'                                LET R4 := R4 + #1
(6)  000066' 005204                                                                    INC     R4
641  000070'                          ENDDO
(4)  000070' 000764                                                                    BR      50004$
(3)  000072'                                                                   50005$:
642
643
644  000072'                                LET R2 :B= R2 SET.BY R3
(6)  000072' 150302                                                                    BISB    R3,R2
645  000074'                          ENDDO
(4)  000074' 000750                                                                    BR      50002$
(3)  000076'                                                                   50003$:
646
647
648                                   ;+
649                                   ; IF THE NON-OCTAL CHAR JUST DETECTED IS NOT A SPACE AND NOT A CR,
650                                   ; STUFF ERROR MSG AND RETURN. ELSE RETURN WITH NO ERROR...
651                                   ;-
652
653  000076'                          IFB (R1) NE #CR ANDB (R1) NE #SPACE ANDB (R1) NE #LF THEN
(6)  000076' 121127  000015                                                            CMPB    (R1),#CR
(9)  000102' 001413                                                                    BEQ     50007$
(6)  000104' 121127  000040                                                            CMPB    (R1),#SPACE
(9)  000110' 001410                                                                    BEQ     50007$
(6)  000112' 121127  000012                                                            CMPB    (R1),#LF
(9)  000116' 001405                                                                    BEQ     50007$
654  000120'                                LET DT.KBRSP(R0) := #CM.NBAD
(4)  000120' 012760  000000G 000022                                                    MOV     #CM.NBAD,DT.KBRS
655  000126'                                INLINE <BR      1$>
(2)  000126' 000410                                                                    BR      1$
656
657  000130'                          ELSE
(4)  000130' 000407                                                                    BR      50010$
(3)  000132'                                                                   50007$:
658
659                                   ;+
660                                   ; THE NUMBER IS OK (16 BITS OR LESS AND OCTAL)
661                                   ; SO SET UP OUTPUTS AND RETURN NO ERROR.
662                                   ;-
663
664  000132'                                LET NUM(R5) := R2
(4)  000132' 010265  000006                                                            MOV     R2,NUM(R5)
665  000136'                                LET NEWPTR(R5) := R1
```

```
   (4)  000136' 010165  000004                                                        MOV     R1,NEWPTR(R5)
 666    000142'                          CALL RESREG
   (3)  000142' 004767  000000G                                                       JSR     PC,RESREG
 667    000146'                          RETURN NO.ERROR
   (4)  000146' 000404                                                              BR      50000$
 668    000150'                        ENDIF
   (4)  000150'                                                                 50010$:
 669
 670    000150'                        INLINE <1$:>
   (2)  000150'                                                                       1$:
 671    000150'                        CALL RESREG
   (3)  000150' 004767  000000G                                                       JSR     PC,RESREG
 672    000154'                        RETURN ERROR
   (2)  000154' 000261                                                              SEC
   (4)  000156' 000401                                                              BR      50001$
 673
 674    000160'                        ENDRTN
   (3)  000160'                                                                 50000$:
   (2)  000160' 000241                                                              CLC
   (3)  000162'                                                                 50001$:
   (2)  000162' 000207                                                              RTS     PC
 675
 676          000001                 .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CMDBUF= 000002
CM.NBA= ****** G
CM.NUM= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NEWPTR= 000004
NOAPTY= 000002
NULL  = 000000
NUM   = 000006
NUMCHK  000000RG
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
```

```
SVR6  = 000076          UIPDR7= 177616          $F$DEC= 000220          $LOCTA= 177777          $$CASE= 000000
SYSCNT= 000052          WASADR= 000104          $F$DO = 000340          $LSTIN= 000001          $$DST = 000000
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LSTTA= 000001          $$ELOC= 000403
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $NESTL= 177777          $$ERFL= 000000
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $NSK0 = 000300          $$FLAG= 000001
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NSK1 = 000110          $$FROM= 000000
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK2 = 000120          $$LOC = 000116R
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK3 = 000110          $$LOCN= 000000
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $SAVLE= 177777          $$REG = 177777
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $SSK0 = 050003          $$RETU= 000001
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $TAGLE= 177777          $$RTN1= 050000
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGNU= 050011          $$RTN2= 050001
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TEMP = 000300          $$SRC = 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TSK0 = 050010          $$TGSV= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK1 = 050003          $$TGS1= 000000
UIPDR2= 177604          $ERFLG= 000000          $F$UNT= 000130          $TSK2 = 050004          $$TGS2= 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK3 = 050005          $$TO  = 000000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $TSK4 = 050006          $$STAG= 050000
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$ARGC= 000010          .     = 000164R
UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$BYTE= 000402

. ABS.  000000     000
        000164     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:NUMCHK,DSKZ:NUMCHK=SPMAC/ML,EQUATE,NUMCHK
 RUN-TIME: 14 4 .3 SECONDS
 RUN-TIME RATIO: 45/19=2.3
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PBREAK PROCESS BREAK MODULE  (PBREAK)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PBREAK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PROCESSES BREAK TRAP CALLS.  IT
                        ;       ENTERS THE OPTION MODULE MAKING THE CALL INTO THE
                        ;       CONTROL QUEUE.
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ENQCQ
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PBREAK IN<DTA>
                        ;
                        ;                       DATA
                        ;               DTA =    TABLE
                        ;                         ADDRESS
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;
                        ;--
```

```
                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
                 000001         $LSTIN=1
                 000001         $LSTTAG=1
                                ;*********************
                                ;REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL PBREAK
                                ;
                                ;*********************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQCQ
                                .GLOBL  SAVREG
                                .GLOBL  RESREG

                                ;*********************
```

```
  573  000000'                        ROUTINE PBREAK <DTA>
  (2)  000000'                                                          PBREAK:
  574
  575                                 ;+
  576                                 ; SAVE REGISTERS NEEDED
  577                                 ;-
  578  000000'                        CALL SAVREG
  (3)  000000' 004767  000000G                                                  JSR     PC,SAVREG
  579
  580                                 ;+
  581                                 ; GET DATA TABLE ADDRESS ARGUMENT PASSED
  582                                 ; GET BEGIN ADDRESS AND DEVELOP RETURN ADDRESS
  583                                 ;-
  584
  585  000004'                        LET R0 := DTA(R5)
  (4)  000004' 016500  000000                                                   MOV     DTA(R5),R0
  586  000010'                        LET R1 := DT.PC(R0)
  (4)  000010' 016001  000002                                                   MOV     DT.PC(R0),R1
  587  000014'                        LET R2 := R1 + #2
  (4)  000014' 010102                                                           MOV     R1,R2
  (6)  000016' 062702  000002                                                   ADD     #2,R2
  588
  589                                 ;+
  590                                 ; MAKE ENTRY IN CONTROL QUEUE
  591                                 ;-
  592
  593  000022'                        CALL ENQCQ IN <R0,(R1),R2>
  (3)  000022' 010546                                                           MOV     R5,-(SP)
  (6)  000024' 010245                                                           MOV     R2,-(R5)
  (5)  000026' 011145                                                           MOV     (R1),-(R5)
  (4)  000030' 010045                                                           MOV     R0,-(R5)
  (3)  000032' 004767  000000G                                                  JSR     PC,ENQCQ
  (3)  000036' 012605                                                           MOV     (SP)+,R5
  594
  595                                 ;+
  596                                 ; RESTORE REGISTERS
  597                                 ;-
  598
  599  000040'                        CALL RESREG
  (3)  000040' 004767  000000G                                                  JSR     PC,RESREG
  600
  601  000044'                        ENDRTN
  (3)  000044'                                                          50000$:
  (3)  000044'                                                          50001$:
  (2)  000044' 000207                                                           RTS     PC
  602
  603          000001                 .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTA   = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQCQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PBREAK  000000RG
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
```

```
TQOVF = 000002          WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
UIPAR0= 177640          WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSKO = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000046R

. ABS.  000000     000
        000046     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PBREAK,DSKZ:PBREAK=SPMAC/ML,EQUATE,PBREAK
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 33/13=2.5
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PEND - PROCESS END TRAP CALL
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE NAME
                        ;       PEND
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'ENDMOD' TRAP CALL (END$).
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. DRPMOD                       ;ROUTINE TO DROP MODULE FROM EXECUTION
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PEND IN <DTABLE>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES
                                
                                
                                
                                
                                        .MCALL  STRUCT
000000'                                 STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN = 1
        000001                          $LSTTAG = 1
                                
                                
                                
                                
                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PEND            ;MODULE'S ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  DRPMOD
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL  PROCESS END TRAP CALL


000000'                                 ROUTINE PEND <DTABLE>
(2)  000000'                                                                    PEND:


                                ;+
                                ; INIT, GET DTABLE ADDR. AND HEADER
                                ;-

000000'                         PUSH R0,R1
(2)  000000' 010046                                                                     MOV     R0,-(SP)
(3)  000002' 010146                                                                     MOV     R1,-(SP)
000004'                         LET R0 := DTABLE(R5)
(4)  000004' 016500  000000                                                             MOV     DTABLE(R5),R0
000010'                         LET R1 := @DT.PC(R0)
(4)  000010' 017001  000002                                                             MOV     @DT.PC(R0),R1

                                ;+
                                ; GO DROP THE OPTION MODULE FROM EXECUTION,
                                ;-

000014'                         CALL DRPMOD IN <R0,R1>
(3)  000014' 010546                                                                     MOV     R5,-(SP)
(5)  000016' 010145                                                                     MOV     R1,-(R5)
(4)  000020' 010045                                                                     MOV     R0,-(R5)
(3)  000022' 004767  000000G                                                            JSR     PC,DRPMOD
(3)  000026' 012605                                                                     MOV     (SP)+,R5

000030'                         POP R1,R0
(2)  000030' 012601                                                                     MOV     (SP)+,R1
(3)  000032' 012600                                                                     MOV     (SP)+,R0
000034'                         ENDRTN
(3)  000034'                                                                    50000$:
(3)  000034'                                                                    50001$:
(2)  000034' 000207                                                                     RTS     PC


        000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PEND    000000RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLD= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
```

```
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000002
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000036R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000

. ABS.  000000     000
        000036     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PEND,DSKZ:PEND=SPMAC/ML,EQUATE,PEND
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 53/13=4.0
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  PENDIT - PROCESS ENDIT TRAP CALL
                        .IDENT  /V0.0/



                        ;++
                        ; MODULE NAME:
                        ;       PENDIT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'ENDIT' TRAP CALL, WHICH CONSISTS
                        ;       OF HANDLING END OF PASS CALLS AND END OF ITERATION CALLS. ALSO, AS
                        ;       PART OF SERVICING THE END OF PASS CALLS, BACKGROUND MODULES ARE
                        ;       TAKEN OFF THE AIR.

                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC
                        ;       2. DT.ST0
                        ;       3. MODULE'S HEADER ADDRESS
                        ;       4. DT.ST1
                        ;       5. DT.CF0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. UPDATED RETURN ADDRESS (DT.PC)
                        ;       2. UPDATED DT.ST0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       BA.STAT                 ;BACKGROUND QUEUE STATUS WORD
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       1. ENQTQ                ;EN-QUEUE TYPE QUEUE ENTRY
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;       4. ENQCQ                ;ENQEUE CONTROL QUEUE ENTRY
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PENDIT IN <DTABLE>
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
000000'                                 $LSTIN = 1
                000001                  $LSTTAG = 1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PENDIT                  ;MODULE'S ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                .GLOBL  ENQTQ                   ;EN-QUEUE TYPE QUEUE ROUTINE
                                .GLOBL  ENQCQ                   ;ENQUEUE INTO THE CONTROL QUEUE
                                .GLOBL  BA.STAT                 ;BACKGROUND QUEUE STATUS WORD
```

```
                                .SBTTL  PROCESS ENDIT ROUTINE


000000'                                 ROUTINE PENDIT <DTABLE>
000000'                                                                 PENDIT:

                                ;+
                                ; DO THE NECESSARY SET-UP STUFF, WHICH CONSISTS OF SAVING THE GPR'S
                                ; AND MARKING THE MODULE FOR AN IMMEDIATE RETURN AFTER
                                ; THE PROCESSING OF THIS TRAP CALL
                                ;-

000000'                                 CALL SAVREG
000000' 004767  000000G                                                 JSR     PC,SAVREG
000004'                                 LET R4 := #0
000004' 005004                                                          CLR     R4
000006'                                 LET R0 := DTABLE(R5)
000006' 016500  000000                                                  MOV     DTABLE(R5),R0
000012'                                 LET DT.STO(R0) := DT.STO(R0) CLR.BY #DEFRTN
000012' 042760  000400  000010                                          BIC     #DEFRTN,DT.STO(R

                                ;+
                                ; GET THE MODULE'S HEADER, THEN GET ITS STATUS WORD, WITH ALL BUT
                                ; THE STATUS BITS MASKED OUT.
                                ;-

000020'                                 LET R1 := @DT.PC(R0)
000020' 017001  000002                                                  MOV     @DT.PC(R0),R1
000024'                                 LET R2 := STAT(R1)
000024' 016102  000026                                                  MOV     STAT(R1),R2
000030'                                 LET R2 := R2 CLR.BY #STATBITS
000030' 042702  064757                                                  BIC     #STATBITS,R2

                                ;+
                                ; IF THE MODULE IS SBK OR NBK, INDICATE THE A DEFERRED RETURN MUST TAKE
                                ; PLACE.
                                ;-

000034'                                 IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
000034' 020227  000000                                                  CMP     R2,#SBKMOD
000040' 001403                                                          BEQ     50002$
000042' 020227  001000                                                  CMP     R2,#NBKMOD
000046' 001003                                                          BNE     50003$
000050'                                                         50002$:
000050'                                         LET DT.STO(R0) := DT.STO(R0) SET.BY #DEFRTN
000050' 052760  000400  000010                                          BIS     #DEFRTN,DT.STO(R
000056'                                 ENDIF
000056'                                                         50003$:

                                ;+
                                ; UPDATE THE RETURN PC.
                                ;-

000056'                                 LET DT.PC(R0) := DT.PC(R0) + #2
000056' 062760  000002  000002                                          ADD     #2,DT.PC(R0)
```

```
                        ;+
                        ; INCREMENT THE MODULE'S ITERATION VARIABLE, AND DETERMINE IF AN
                        ; END OF PASS IS IN ORDER (I.E., IF ITERATIONS EQUAL MAX. OR
                        ; INITIAL PASS OF A BKMOD)
                        ;-

000064'                         LET ICOUNT(R1) := ICOUNT(R1) + #1
(6)  000064' 005261  000040                                                 INC     ICOUNT(R1)
                        IF ICOUNT(R1) HIS ICONT(R1) OR #TMPIO SETIN DT.ST0(R0) THEN
000070'                                                                     CMP     ICOUNT(R1),ICONT
(6)  000070' 026161  000040  000036                                         BHIS    50004$
(8)  000076' 103004                                                         BIT     #TMPIO,DT.ST0(R0
(6)  000100' 032760  000002  000010                                         BEQ     50005$
(9)  000106' 001501                                                 50004$:
(6)  000110'

                        ;+
                        ; ITS END-OF-PASS TIME, SO, MARK THE MODULE AS NOT REQUIRING AN IMMEDIATE
                        ; RETURN FROM THIS CALL, RESET THE ERRORS PER PASS COUNTS,
                        ; BUMP THE END OF PASS COUNT AND RESET THE ITERATION COUNT.
                        ;-

000110'                         LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
(6)  000110' 052760  100000  000012                                         BIS     #CKTIM,DT.ST1(R0
000116'                         LET DT.ST0(R0) := DT.ST0(R0) SET.BY #DEFRTN
(6)  000116' 052760  000400  000010                                         BIS     #DEFRTN,DT.ST0(R
000124'                         LET SOFPAS(R1) := #0
(4)  000124' 005061  000046                                                 CLR     SOFPAS(R1)
000130'                         LET HRDPAS(R1) := #0
(4)  000130' 005061  000050                                                 CLR     HRDPAS(R1)
000134'                         LET ICOUNT(R1) := #0
(4)  000134' 005061  000040                                                 CLR     ICOUNT(R1)
000140'                         LET PASCNT(R1) := PASCNT(R1) + #1
(6)  000140' 005261  000034                                                 INC     PASCNT(R1)


                        ;+
                        ; NOTE THAT AN END OF PASS MESSAGE IS IN ORDER
                        ;-

000144'                         LET R4 := R4 + #1
(6)  000144' 005204                                                         INC     R4


                        ;+
                        ; IF THE MODULE TYPE IS BK, BSK, OR NBK, DO THE FOLLOWING:
                        ;       1. CLEAR THE RETURN ADDRESS FOR THE MESSAGE ENQUEUE CALL.
                        ;       2. CLEAR THE ACTIVE BIT TO TAKE THE MODULE OFF THE AIR.
                        ;       3. IF NBK, SET THE "DROP" BIT.
                        ;       4. IF SBK OR NBK, UNLOCK THE MODULE QUEUE LIST.
                        ; OTHERWISE, LOAD THE MODULE'S RETURN ADDRESS FOR THE MESSAGE ENQUEUE CALL.
                        ;-

000146'                         IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD OR R2 EQ #BKMOD THEN
(6)  000146' 020227  000000                                                 CMP     R2,#SBKMOD
(8)  000152' 001406                                                         BEQ     50006$
(6)  000154' 020227  001000                                                 CMP     R2,#NBKMOD
(8)  000160' 001403                                                         BEQ     50006$
```

```
(6)  000162' 020227  000020                                                    CMP     R2,#BKMOD
(9)  000166' 001025                                                            BNE     50007$
(6)  000170'                                                                50006$:
671  000170'                                LET R3 := #0
(4)  000170' 005003                                                            CLR     R3
672  000172'                                LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
(6)  000172' 042761  004000  000026                                           BIC     #ACTBIT,STAT(R1)
673  000200'                                IF R2 EQ #NBKMOD THEN
(6)  000200' 020227  001000                                                    CMP     R2,#NBKMOD
(9)  000204' 001003                                                            BNE     50010$
674  000206'                                        LET STAT(R1) := STAT(R1) SET.BY #BIT13
(6)  000206' 052761  020000  000026                                           BIS     #BIT13,STAT(R1)
675  000214'                                ENDIF
(4)  000214'                                                                50010$:
676  000214'                                IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
(6)  000214' 020227  000000                                                    CMP     R2,#SBKMOD
(8)  000220' 001403                                                            BEQ     50011$
(6)  000222' 020227  001000                                                    CMP     R2,#NBKMOD
(9)  000226' 001004                                                            BNE     50012$
(6)  000230'                                                                50011$:
677  000230'                                        LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #MODHOLD
(6)  000230' 042760  002000  000010                                           BIC     #MODHOLD,DT.ST0(
678  000236'                                        LET R3 := #0
(4)  000236' 005003                                                            CLR     R3
679  000240'                                ENDIF
(4)  000240'                                                                50012$:
680  000240'                            ELSE
(4)  000240' 000402                                                            BR      50013$
(3)  000242'                                                                50007$:
681  000242'                                LET R3 := RSTRT(R1)
(4)  000242' 016103  000112                                                    MOV     RSTRT(R1),R3
682  000246'                            ENDIF
(4)  000246'                                                                50013$:
683
684                                     ;+
685                                     ; SET THE MODULE'S EOP BIT, UNLESS THIS IS A BKMOD IN DISGUISE AS A IOMOD
686                                     ; (TMPIO = 1), IN WHICH CASE  WE MUST CHANGE
687                                     ; THE BKMOD'S STAT BITS BACK FROM IOMOD TO BKMOD AND
688                                     ; CLEAR TMPIO.
689                                     ;-
690
691  000246'                            IF #TMPIO NOTSETIN DT.ST0(R0) THEN
(6)  000246' 032760  000002  000010                                           BIT     #TMPIO,DT.ST0(R0
(9)  000254' 001004                                                            BNE     50014$
692  000256'                                LET XFLAG(R1) :B= XFLAG(R1) SET.BY #EOPBIT
(6)  000256' 152761  000001  000005                                           BISB    #EOPBIT,XFLAG(R1
693  000264'                            ELSE
(4)  000264' 000412                                                            BR      50015$
(3)  000266'                                                                50014$:
694  000266'                                LET R3 := #0
(4)  000266' 005003                                                            CLR     R3
695  000270'                                LET STAT(R1) := STAT(R1) CLR.BY #IOMOD
(6)  000270' 042761  100000  000026                                           BIC     #IOMOD,STAT(R1)
696  000276'                                LET STAT(R1) := STAT(R1) SET.BY #BKMOD
(6)  000276' 052761  000020  000026                                           BIS     #BKMOD,STAT(R1)
697  000304'                                LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #TMPIO!MODHOLD
```

```
   (6)  000304' 042760  002002  000010                                                    BIC     #TMPIO!MODHOLD,D
   698  000312'                                ENDIF
   (4)  000312'                                                                  50015$:
   699  000312'                            ENDIF
   (4)  000312'                                                                  50005$:
   700
   701                                 ;+
   702                                 ; DETERMINE IF THE EXERCISER IS RUNNING IN RELOCATABLE MODE,
   703                                 ; AUTOMATIC MODE, OR IF SBKSEL IS SET, AND IF SO, DO THE REQUIRED STUFF
   704                                 ;-
   705
   706  000312'                            IF #RELMODE SETIN DT.ST0(R0) OR #AUTO SETIN DT.CF0(R0) OR #SBKSEL SETIN DT.ST1(R
   (6)  000312' 032760  020000  000010                                                    BIT     #RELMODE,DT.ST0(
   (8)  000320' 001010                                                                    BNE     50016$
   (6)  000322' 032760  000010  000014                                                    BIT     #AUTO,DT.CF0(R0)
   (8)  000330' 001004                                                                    BNE     50016$
   (6)  000332' 032760  010000  000012                                                    BIT     #SBKSEL,DT.ST1(R
   (9)  000340' 001432                                                                    BEQ     50017$
   (6)  000342'                                                                  50016$:
   707
   708
   709                                 ;+
   710                                 ; IS IT TIME TO RELOCATE?  OR, IF NOT IN RELOCATION MODE, IS IT TIME TO RETURN TO
   711                                 ; THE XXDP MONITOR?   IF SO, MARK THE MODULE AS NOT REQUIRING
   712                                 ; AN IMMEDIATE RETURN FROM THIS CALL, DE-ACTIVATE THE MODULE, AND
   713                                 ; SET THE MODULE'S RETURN ADDRESS TO 0.
   714                                 ;-
   715
   716  000342'                            IF #RELTIME SETIN DT.ST0(R0) OR #MEMPAS SETIN DT.ST1(R0) THEN
   (6)  000342' 032760  010000  000010                                                    BIT     #RELTIME,DT.ST0(
   (8)  000350' 001004                                                                    BNE     50020$
   (6)  000352' 032760  040000  000012                                                    BIT     #MEMPAS,DT.ST1(R
   (9)  000360' 001422                                                                    BEQ     50021$
   (6)  000362'                                                                  50020$:
   717  000362'                                     LET DT.ST0(R0) := DT.ST0(R0) SET.BY #DEFRTN
   (6)  000362' 052760  000400  000010                                                    BIS     #DEFRTN,DT.ST0(R
   718  000370'                                     LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
   (6)  000370' 042761  004000  000026                                                    BIC     #ACTBIT,STAT(R1)
   719
   720                                 ;+
   721                                 ; IF ITS A BACKGROUND MODULE, RESET THE BKMODE BIT IN THE MONITOR'S STATUS
   722                                 ; INDICATOR WORD 0
   723                                 ;-
   724
   725  000376'                                     IF #BKMOD SETIN STAT(R1) AND #BKMODE SETIN DT.ST0(R0) THEN
   (6)  000376' 032761  000020  000026                                                    BIT     #BKMOD,STAT(R1)
   (9)  000404' 001407                                                                    BEQ     50022$
   (6)  000406' 032760  040000  000010                                                    BIT     #BKMODE,DT.ST0(R
   (9)  000414' 001403                                                                    BEQ     50022$
   726  000416'                                            LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #BKMODE
   (6)  000416' 042760  040000  000010                                                    BIC     #BKMODE,DT.ST0(R
   727  000424'                                     ENDIF
   (4)  000424'                                                                  50022$:
   728
   729
   730  000424'                                LET R3 := #0
```

```
(4)  000424' 005003                                                     CLR     R3
731  000426'                        ;                   ENDIF
(4)  000426'                                                    50021$:
732  000426'                        ;               ENDIF
(4)  000426'                                                    50017$:
733
734                                 ;+
735                                 ; IF IT IS AN END-OF-PASS, ENQUEUE A MESSAGE.
736                                 ; OTHERWISE, IF NBK OR SBK, GET THE RETURN ADDRESS AND MAKE A
737                                 ; CONTROL QUEUE ENTRY.
738                                 ;-
739
740  000426'                        ;               IF R4 NE #0 THEN
(6)  000426' 005704                                                     TST     R4
(9)  000430' 001414                                                     BEQ     50023$
741  000432'                        ;                   CALL ENQTQ IN <R0,#MSGEOP,#0,R1,R3>
(3)  000432' 010546                                                     MOV     R5,-(SP)
(8)  000434' 010345                                                     MOV     R3,-(R5)
(7)  000436' 010145                                                     MOV     R1,-(R5)
(6)  000440' 012745  000000                                             MOV     #0,-(R5)
(5)  000444' 012745  000013                                             MOV     #MSGEOP,-(R5)
(4)  000450' 010045                                                     MOV     R0,-(R5)
(3)  000452' 004767  000000G                                            JSR     PC,ENQTQ
(3)  000456' 012605                                                     MOV     (SP)+,R5
742  000460'                        ;               ELSE
(4)  000460' 000417                                                     BR      50024$
(3)  000462'                                                    50023$:
743  000462'                        ;                   IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
(6)  000462' 020227  000000                                             CMP     R2,#SBKMOD
(8)  000466' 001403                                                     BEQ     50025$
(6)  000470' 020227  001000                                             CMP     R2,#NBKMOD
(9)  000474' 001011                                                     BNE     50026$
(6)  000476'                                                    50025$:
744  000476'                        ;                       LET R2 := DT.PC(R0)
(4)  000476' 016002  000002                                             MOV     DT.PC(R0),R2
745  000502'                        ;                       CALL ENQCQ IN <R0,R1,R2>
(3)  000502' 010546                                                     MOV     R5,-(SP)
(6)  000504' 010245                                                     MOV     R2,-(R5)
(5)  000506' 010145                                                     MOV     R1,-(R5)
(4)  000510' 010045                                                     MOV     R0,-(R5)
(3)  000512' 004767  000000G                                            JSR     PC,ENQCQ
(3)  000516' 012605                                                     MOV     (SP)+,R5
746  000520'                        ;                   ENDIF
(4)  000520'                                                    50026$:
747  000520'                        ;               ENDIF
(4)  000520'                                                    50024$:
748
749                                 ;+
750                                 ; RETURN TO CALLER
751                                 ;-
752
753  000520'                        ;               CALL RESREG
(3)  000520' 004767  000000G                                            JSR     PC,RESREG
754  000524'                        ;               ENDRTN
(3)  000524'                                                    50000$:
(3)  000524'                                                    50001$:
```

```
   (2)  000524' 000207                                                  RTS     PC
   755
   756
   757          000001                          .END            ;THAT'S ALL FOLKS!!!
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BA.STA= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQCQ = ****** G
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PENDIT  000000RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LOCTA= 177777          $SERFL= 000000
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000001
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NESTL= 177777          $$LOC = 000474R
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK1 = 000110          $$REG = 177777
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $NSK2 = 000110          $$RETU= 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $NSK3 = 000110          $$RTN1= 050000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $SAVLE= 177777          $$RTN2= 050001
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGLE= 177777          $$SRC = 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TAGNU= 050027          $$TGSV= 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TEMP = 000300          $$TGS1= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK0 = 050024          $$TGS2= 000000
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $TSK1 = 050026          $$TO  = 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK2 = 050022          $$$TAG= 050000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$ARGC= 000002          .     = 000526R
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$BYTE= 000403
UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$CASE= 000000
UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$DST = 000000
WASADR= 000104          $F$DO = 000340          $ISK2 = 000001          $$ELOC= 000402

. ABS.  000000      000
        000526      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PENDIT,DSKZ:PENDIT=SPMAC/ML,EQUATE,PENDIT
 RUN-TIME: 20 11 .3 SECONDS
 RUN-TIME RATIO: 92/32=2.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PDATER (PROCESS DATER$ TRAP)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PDATER
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS CALLED AS A RESULT OF A DATER$ TRAP
                        ;       EXECUTED BY AN OPTION MODULE.  IT ENQUEUES AN ERROR MESSAGE.
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC
                        ;       2. DT.CFO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. ENQTQ  ;ENQUE THE ERROR MESSAGE
                        ;       2. SAVREG
                        ;       3. RESREG
                        ;       4. APTSER
                        ;       5. DRPMOD
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PDATER IN <DT>
                        ;               WHERE DT = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL PDATER (COMMON DEFINITIONS AND REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                 000001         $LSTIN=1
                 000001         $LSTTAG=1

                                ;*****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PDATER                          ;MODULE ENTRY POINT
                                ;
                                ;*****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQTQ           ;ENQUES ERROR MSG
                                .GLOBL  SAVREG          ;
                                .GLOBL  RESREG          ;
                                .GLOBL  APTSER          ;APT ROUTINE
                                .GLOBL  DRPMOD          ;DROP MODULE
                                ;
                                ;****************
```

```
                                .SBTTL PDATER (CODE)

                                ROUTINE PDATER <AA>
000000'                                                                 PDATER:
(2)  000000'

                                ;+
                                ; THIS ROUTINE ENQUES THE DATA ERROR MESSAGE IN THE TYPE QUEUE.
                                ;
                                ;               R0 = DTABLE ADDRESS
                                ;               R1 = PC+2 OF CALL
                                ;               R2 = PC OF CALL AND IS PASSED AS ADDR OF MSG
                                ;               R3 = RETURN ADDRESS
                                ;          MSGDER = DATA ERROR TYPE MSG CODE
                                ;-

000000'                         CALL SAVREG                             JSR     PC,SAVREG
(3)  000000' 004767  000000G

000004'                         LET R0 := AA(R5)                        MOV     AA(R5),R0
(4)  000004' 016500  000000
000010'                         LET R1 := DT.PC(R0)                     MOV     DT.PC(R0),R1
(4)  000010' 016001  000002
000014'                         LET R4 := @DT.PC(R0)                    MOV     @DT.PC(R0),R4
(4)  000014' 017004  000002

                                ;+
                                ; INCREMENT SOFCNT, AND SOFPAS AND IF UNDER APT, CALL APT SOFT
                                ; ERROR ROUTINE
                                ;-


000020'                         LET SOFCNT(R4) := SOFCNT(R4) + #1       INC     SOFCNT(R4)
(6)  000020' 005264  000042
000024'                         LET SOFPAS(R4) := SOFPAS(R4) + #1       INC     SOFPAS(R4)
(6)  000024' 005264  000046
000030'                         IF #APTPRES SETIN DT.CF0(R0) THEN       BIT     #APTPRES,DT.CF0(
(6)  000030' 032760  000200  000014                                    BEQ     50002$
(9)  000036' 001406
000040'                                 CALL APTSER IN <R0>             MOV     R5,-(SP)
(3)  000040' 010546                                                     MOV     R0,-(R5)
(4)  000042' 010045                                                     JSR     PC,APTSER
(3)  000044' 004767  000000G                                           MOV     (SP)+,R5
(3)  000050' 012605
000052'                         ELSE                                    BR      50003$
(4)  000052' 000431
(3)  000054'                                                    50002$:

                                ;+
                                ; THE ENVIRONMENT IS NOT APT.
                                ; IF THE SWITCH REGISTER SAYS DROP THE MODULE BECAUSE OF THE
                                ; ERROR, DO IT.
                                ;-

000054'                                 IF #BIT15 SETIN DT.SWR(R0) THEN BIT     #BIT15,DT.SWR(R0
(6)  000054' 032760  100000  000056                                    BEQ     50004$
(9)  000062' 001407
000064'                                         CALL DRPMOD IN <R0,R4>
```

```
(3)  000064' 010546                                                                    MOV     R5,-(SP)
(5)  000066' 010445                                                                    MOV     R4,-(R5)
(4)  000070' 010045                                                                    MOV     R0,-(R5)
(3)  000072' 004767  000000G                                                           JSR     PC,DRPMOD
(3)  000076' 012605                                                                    MOV     (SP)+,R5
621  000100'                                ELSE
(4)  000100' 000416                                                                    BR      50005$
(3)  000102'                                                                   50004$:
622
623
624                                     ;+
625                                     ; IF THE ERROR COUNT HAS REACHED THE LIMIT, AND IF THE SWITCH REGISTER
626                                     ; INDICATES THE MODULE SHOULD BE DROPPED, DO IT.
627                                     ;-
628
629  000102'                                        IF SOFCNT(R4) HI DT.SMX(R0) AND #BIT14 NOTSETIN DT.SWR(R0) THEN
(6)  000102' 026460  000042  000106                                                    CMP     SOFCNT(R4),DT.SM
(9)  000110' 101412                                                                    BLOS    50006$
(6)  000112' 032760  040000  000056                                                    BIT     #BIT14,DT.SWR(R0
(9)  000120' 001006                                                                    BNE     50006$
630  000122'                                                CALL DRPMOD IN <R0,R4>
(3)  000122' 010546                                                                    MOV     R5,-(SP)
(5)  000124' 010445                                                                    MOV     R4,-(R5)
(4)  000126' 010045                                                                    MOV     R0,-(R5)
(3)  000130' 004767  000000G                                                           JSR     PC,DRPMOD
(3)  000134' 012605                                                                    MOV     (SP)+,R5
631  000136'                                        ENDIF
(4)  000136'                                                                   50006$:
632  000136'                                ENDIF
(4)  000136'                                                                   50005$:
633  000136'                                ENDIF
(4)  000136'                                                                   50003$:
634
635
636                                     ;+
637                                     ; PUT THE ADDRESS OF THE TRAP CALL IN R2.
638                                     ;-
639
640  000136'                            LET R2 := R1 - #2                               MOV     R1,R2
(4)  000136' 010102                                                                    SUB     #2,R2
(6)  000140' 162702  000002
641
642                                     ;+
643                                     ; PUT THE OPTION MODULE'S RETURN ADDRESS IN R3.
644                                     ;-
645
646  000144'                            LET R3 := R1 + #2                               MOV     R1,R3
(4)  000144' 010103                                                                    ADD     #2,R3
(6)  000146' 062703  000002
647  000152'                            CALL ENQTQ IN <R0,#MSGDER,R2,(R1),R3>
(3)  000152' 010546                                                                    MOV     R5,-(SP)
(8)  000154' 010345                                                                    MOV     R3,-(R5)
(7)  000156' 011145                                                                    MOV     (R1),-(R5)
(6)  000160' 010245                                                                    MOV     R2,-(R5)
(5)  000162' 012745  000005                                                            MOV     #MSGDER,-(R5)
(4)  000166' 010045                                                                    MOV     R0,-(R5)
```

```
                                                                        JSR     PC,ENQTQ
(3)  000170' 004767  000000G                                            MOV     (SP)+,R5
(3)  000174' 012605
648
649  000176'                        CALL RESREG                         JSR     PC,RESREG
(3)  000176' 004767  000000G
650
651  000202'                        ENDRTN                      50000$:
(3)  000202'                                                    50001$:
(3)  000202'                                                            RTS     PC
(2)  000202' 000207
652
653          000001                .END
```

```
AA     = 000000
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
APTSER= ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQDVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDATER  000000RG
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLD= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
```

```
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000110
$NSK2 = 000110
$NSK3 = 000110
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050007
$TEMP = 000300
$TSK0 = 050003
$TSK1 = 050005
$TSK2 = 050006
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000120R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000204R

. ABS.  000000  000
        000204  001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:PDATER,DSKZ:PDATER=SPMAC/ML,EQUATE,PDATER
RUN-TIME: 13 3 .4 SECONDS
RUN-TIME RATIO: 64/17=3.6
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PGETPA (GET PHYSICAL ADDRESS MODULE)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PGETPA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE SERVICES GETPA TRAP CALLS.
                        ;       IT RETURNS A 16-BIT OR 18-BIT VIRTUAL
                        ;       ADDRESS FOR A 16-BIT VIRTUAL ADDRESS GIVEN
                        ;       (DEPENDING ON IF KT IS TURNED ON).
                        ;       A POINTER TO A 3-WORD AREA IS RECEIVED. THE FIRST
                        ;       WORD CONTAINS THE 16-BIT VIRTUAL ADDRESS, THE NEXT TWO
                        ;       WORDS WILL CONTAIN, ON RETURN, THE LOW 16-BITS OF THE ADDRESS AND
                        ;       THE EXTENDED ADDRESS BITS.
                        ;
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DT.PC                ;DT.PC+2 POINTS TO TABLE CONTAINING VA,PA,EA
                        ;                                       WHERE VA = VIRTUAL ADDRESS
                        ;                                             PA = PHYSICAL ADDRESS
                        ;                                             EA = EXTENDED ADDRESS
                        ;
                        ;       2. DT.STO
                        ;       3. DT.OFFSET
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. PHYSICAL ADDRESS WORD
                        ;       2. EXTENDED ADDRESS WORD
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PGETPA IN <DT>
                        ;           WHERE DT = DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL PGETPA (COMMON DEFINITIONS & REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1

                                $LSTIN=1
                                $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PGETPA                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                          ;SAVE REGISTERS
                                .GLOBL  RESREG                          ;RESTORE REGISTERS

                                ;****************
```

```
                                .SBTTL PGETPA (CODE)

                                ROUTINE PGETPA <DT>
000000'                                                                 PGETPA:
(2)  000000'

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
000000'                                                                         JSR     PC,SAVREG
(3)  000000' 004767  000000G

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := DT(R5)
000004'                                                                         MOV     DT(R5),R0
(4)  000004' 016500  000000

                                ;+
                                ; SET R1 TO THE START OF THE ADDRESS TABLE.
                                ;-

                                LET R2 := DT.PC(R0) + #2
000010'                                                                         MOV     DT.PC(R0),R2
(4)  000010' 016002  000002                                                     ADD     #2,R2
(6)  000014' 062702  000002

                                LET R1 := (R2)
000020'                                                                         MOV     (R2),R1
(4)  000020' 011201

                                ;+
                                ; UPDATE DT.PC TO REFLECT THE OPTION MODULE'S RETURN ADDRESS.
                                ;-

                                LET DT.PC(R0) := DT.PC(R0) + #4
000022'                                                                         ADD     #4,DT.PC(R0)
(6)  000022' 062760  000004  000002

                                ;+
                                ; IF MEMORY MANAGEMENT IS NOT TURNED ON, SIMPLY MOVE THE
                                ; 16-BIT VIRTUAL ADDRESS TO THE OUTPUT POSITION OF THE TABLE, CLEAR THE
                                ; HIGH BIT TABLE POSITION, AND RETURN.
                                ;-

                                IF #KTSTAT NOTSETIN DT.ST0(R0) THEN
000030'                                                                         BIT     #KTSTAT,DT.ST0(R
(6)  000030' 032760  000020  000010                                             BNE     50002$
(9)  000036' 001004
                                        LET (R1) := (R1)+
000040'                                                                         MOV     (R1)+,(R1)
(4)  000040' 012111
                                        LET 2(R1) := #0
000042'                                                                         CLR     2(R1)
(4)  000042' 005061  000002

                                ELSE
000046'
```

```
(4)  000046' 000446                                                          BR      50003$
(3)  000050'                                                          50002$:
632
633
634                             ;+
635                             ; ELSE MEMORY MANAGEMENT IS ON.
636                             ;-
637
638                             ;+
639                             ; DETERMINE THE PAR REGISTER REFERENCED BY THE VIRTUAL ADDRESS
640                             ; BY PLACING THE VA IN R2, DETERMINING THE ADDRESS
641                             ; OF THE PROPER PAR REGISTER, AND PLACING THE ADDRESS OF THIS
642                             ; REGISTER IN R3.
643                             ;-
644
645  000050'                            LET R2 := (R1)                       MOV     (R1),R2
(4)  000050' 011102
646  000052'                            LET R2 := SWAP R2                    SWAB    R2
(6)  000052' 000302
647  000054'                            LET R2 := R2 SHIFT #-4               ASR     R2
(7)  000054' 006202                                                         ASR     R2
(7)  000056' 006202                                                         ASR     R2
(7)  000060' 006202                                                         ASR     R2
(7)  000062' 006202
648  000064'                            LET R2 := R2 CLR.BY #177761          BIC     #177761,R2
(6)  000064' 042702  177761
649  000070'                            LET R3 := #KIPAR0 + R2               MOV     #KIPAR0,R3
(4)  000070' 012703  172340                                                 ADD     R2,R3
(6)  000074' 060203
650
651
652                             ;+
653                             ; NOW GET THE CONTENTS OF THIS PAR REGISTER.
654                             ;-
655
656  000076'                            LET R2 := (R3)                       MOV     (R3),R2
(4)  000076' 011302
657
658
659                             ; WE NOW HAVE THE 22-BIT PHYSICAL ADDRESS IN PAR FORMAT.
660                             ; MAKE THIS A VIRTUAL 18-BIT VALUE (STILL PAR FORMAT) BY SUBTRACTING
661                             ; THE CURRENT OFFSET VALUE.
662                             ;-
663
664  000100'                            LET R2 := R2 - DT.OFFSET(R0)         SUB     DT.OFFSET(R0),R2
(6)  000100' 166002  000070
665
666
667                             ;+
668                             ; NOW CONVERT THIS VALUE TO 18 BITS.
669                             ; FIRST FORM THE LOW-ORDER 16-BITS BY SHIFTING THE HIGH
670                             ; ORDER BITS OUT OF THE PAR CONTENTS AND ADDING THE LOW 12 BITS OF THE
671                             ; ORIGINAL VA.  THEN IF THERE IS OVERFLOW, CARRY IT TO THE HIGH BITS.
672                             ;-
673
674  000104'                            LET R3 := R2 SHIFT #6
```

```
(4)  000104' 010203                                                                    MOV     R2,R3
(7)  000106' 006303                                                                    ASL     R3
(7)  000110' 006303                                                                    ASL     R3
(7)  000112' 006303                                                                    ASL     R3
(7)  000114' 006303                                                                    ASL     R3
(7)  000116' 006303                                                                    ASL     R3
(7)  000120' 006303                                                                    ASL     R3
675  000122'                           LET R4 := (R1) CLR.BY #160000
(4)  000122' 011104                                                                    MOV     (R1),R4
(6)  000124' 042704  160000                                                            BIC     #160000,R4
676  000130'                           LET 2(R1) := R3 + R4
(4)  000130' 010361  000002                                                            MOV     R3,2(R1)
(6)  000134' 060461  000002                                                            ADD     R4,2(R1)
677  000140'                           IFCOND CS THEN
(6)  000140' 103002                                                                    BCC     50004$
678  000142'                                   LET R2 := R2 + #200
(6)  000142' 062702  000200                                                            ADD     #200,R2
679  000146'                           ENDIF
(4)  000146'                                                                    50004$:
680
681
682                                  ;+
683                                  ; FORM THE HIGH-ORDER 2 BITS, SHIFTING THEM INTO BIT POSTIONS
684                                  ; 4 AND 5.
685                                  ;-
686
687  000146'                           LET R2 := SWAP R2                                 SWAB    R2
(6)  000146' 000302
688  000150'                           LET R2 := R2 SHIFT #+2
(7)  000150' 006302                                                                    ASL     R2
(7)  000152' 006302                                                                    ASL     R2
689  000154'                           LET R2 := R2 CLR.BY #177717
(6)  000154' 042702  177717                                                            BIC     #177717,R2
690  000160'                           LET 4(R1) := R2
(4)  000160' 010261  000004                                                            MOV     R2,4(R1)
691
692
693  000164'                           ENDIF                                         50003$:
(4)  000164'
694
695
696                                  ;+
697                                  ; RESTORE REGISTERS AND RETURN.
698                                  ;-
699
700  000164'                           CALL RESREG                                       JSR     PC,RESREG
(3)  000164' 004767  000000G
701
702  000170'                           ENDRTN                                        50000$:
(3)  000170'                                                                    50001$:
(3)  000170'                                                                            RTS     PC
(2)  000170' 000207
703
704          000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT    = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PGETPA  000000RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
```

```
UIPAR0= 177640          WBUFEA= 000136          $F$DO  = 000340          $ISK0  = 000001          $$DST  = 000000
UIPAR1= 177642          WBUFPA= 000134          $F$FAL= 000405          $ISK1  = 000001          $$ELOC= 000402
UIPAR2= 177644          WBUFRQ= 000140          $F$GOO= 000400          $LOCTA= 177777          $$ERFL= 000000
UIPAR3= 177646          WBUFSZ= 000142          $F$IF  = 000110          $LSTIN= 000001          $$FLAG= 000001
UIPAR4= 177650          WDFR   = 000116          $F$INC= 000210          $LSTTA= 000001          $$FROM= 000000
UIPAR5= 177652          WDTO   = 000114          $F$LOO= 000200          $NESTL= 177777          $$LOC  = 000140R
UIPAR6= 177654          WTINRE= 000352          $F$NAM= 000160          $NSK0  = 000300          $$LOCN= 000000
UIPAR7= 177656          WTWHMI= 000222          $F$NO  = 000403          $NSK1  = 000110          $$REG  = 177777
UIPDR0= 177600          XFLAG  = 000005          $F$OR  = 000320          $NSK2  = 000110          $$RETU= 000000
UIPDR1= 177602          XOFF   = 000023          $F$RTI= 000350          $SAVLE= 177777          $$RTN1= 050000
UIPDR2= 177604          XON    = 000021          $F$RTN= 000300          $TAGLE= 177777          $$RTN2= 050001
UIPDR3= 177606          $BGNLE= 177777          $F$SEL= 000140          $TAGNU= 050005          $$SRC  = 000000
UIPDR4= 177610          $ERFLG= 000400          $F$THE= 000330          $TEMP  = 000300          $$TGSV= 000000
UIPDR5= 177612          $F$AND= 000310          $F$TRU= 000404          $TSK0  = 050003          $$TGS1= 000000
UIPDR6= 177614          $F$BAD= 000401          $F$UNT= 000130          $TSK1  = 050004          $$TGS2= 000000
UIPDR7= 177616          $F$BLA= 000170          $F$WHI= 000120          $$ARGC= 000002          $$TO   = 000000
WASADR= 000104          $F$CAS= 000150          $F$YES= 000402          $$BYTE= 000403          $$$TAG= 050000
WBSTAT= 000040          $F$DEC= 000220          $IFLEV= 177777          $$CASE= 000000          .      = 000172R


. ABS.  000000     000
        000172     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PGETPA,DSKZ:PGETPA=SPMAC/ML,EQUATE,PGETPA
 RUN-TIME: 14 4 .4 SECONDS
 RUN-TIME RATIO: 72/19=3.7
 CORE USED:  14K  (27 PAGES)
```

```
                                .TITLE PMAP22 - PROCESS MAP22
                                .IDENT /V0.0/

                                ;++
                                ; MODULE NAME:
                                ;       PMAP22
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;       PROCESSES MAP22 TRAP CALLS
                                ;
                                ; INPUTS:
                                ;       DATA TABLE
                                ;
                                ; IMPLICIT INPUTS:
                                ;       DT.PC
                                ;
                                ; OUTPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       DT.PC
                                ;
                                ; PATHOLOGICAL CONNECTIONS:
                                ;       NONE
                                ;
                                ; SUBORDINATE MODULES CALLED:
                                ;       UNIPA                           ;GET PHYS. ADDR. FROM UNIBUS MAP
                                ;
                                ; FUNCTIONAL SIDE EFFECTS:
                                ;       NONE
                                ;
                                ; CALLING SEQUENCE:
                                ;       CALL PMAP22 IN <A>
                                ;               A=ADDRESS OF DATA TABLE
                                ;
                                ; VERSION:
                                ;       0.0
                                ;
                                ;       EDIT            BY              DATE            REASON
                                ;
                                ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES
                                .MCALL STRUCT
000000'                         STRUCT
(1)  000000'                    .PRINT  ;SPMAC: VERSION 1.1
                 000001         $LSTIN=1
                 000001         $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PMAP22                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  UNIPA                           ;GET PHYS. ADDR. FROM UNIBUS MAP
                                ;
```

```
                                .SBTTL PMAP22 ROUTINE

                                ROUTINE PMAP22 <TABL>                                   PMAP22:
                                ;+
                                ; SAVE REGISTERS
                                ;-

                                PUSH R0,R1                                              MOV     R0,-(SP)
                                                                                        MOV     R1,-(SP)

                                ;+
                                ; SET R0 TO START OF DATA TABLE
                                ;-

                                LET R0 := TABL(R5)                                      MOV     TABL(R5),R0

                                ;+
                                ; GET THE ADDRESS OF THE TABLE REQUIRED BY UNIPA.  THIS TABLE
                                ; CONTAINS THE 18-BIT ADDRESS TO BE CONVERTED ALONG WITH A STORAGE
                                ; AREA FOR THE 22-BIT RESULT.
                                ;-

                                LET R1 := DT.PC(R0) + #2                                MOV     DT.PC(R0),R1
                                                                                        ADD     #2,R1

                                ;+
                                ; CALL UNIPA
                                ;-

                                CALL UNIPA IN <(R1)>                                    MOV     R5,-(SP)
                                                                                        MOV     (R1),-(R5)
                                                                                        JSR     PC,UNIPA
                                                                                        MOV     (SP)+,R5

                                ;+
                                ; UPDATE DT.PC TO REFLECT THE OPTION MODULE'S RE-ENTRY
                                ; ADDRESS.  (THAT IS, JUMP OVER THE INPUT ARGUMENT OF
                                ; THE TRAP CALL.)
                                ;-

                                LET DT.PC(R0) := DT.PC(R0) + #4                         ADD     #4,DT.PC(R0)

                                ;+
```

```
   615                                  ; RESTORE REGISTERS
   616                                  ;-
   617
   618  000040'                         POP R1,R0                                          MOV     (SP)+,R1
   (2)  000040' 012601                                                                     MOV     (SP)+,R0
   (3)  000042' 012600
   619
   620
   621                                  ;+
   622                                  ; GOOD-BYE.
   623                                  ;-
   624
   625  000044'                         ENDRTN                                     50000$:
   (3)  000044'                                                                    50001$:
   (3)  000044'                                                                            RTS     PC
   (2)  000044' 000207
   626
   627          000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PMAP22  000000RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TABL  = 000000
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
```

```
UIPAR2= 177644          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR3= 177646          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR4= 177650          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR5= 177652          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR6= 177654          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR7= 177656          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPDR0= 177600          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR1= 177602          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR2= 177604          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR3= 177606          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR4= 177610          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR5= 177612          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR6= 177614          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000001
UIPDR7= 177616          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UNIPA = ****** G        $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000046R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000

. ABS.  000000      000
        000046      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PMAP22,DSKZ:PMAP22=SPMAC/ML,EQUATE,PMAP22
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 62/13=4.6
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  PONOF   PROCESS THE KEYBOARD COMMANDS 'PON' AND 'POFF'
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       PONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
                        ;               1. KPON - PROCESS THE KEYBOARD COMMAND 'PON'
                        ;               2. KPOFF - PROCESS THE KEYBOARD COMMAND 'POFF'
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          BY              DATE            REASON
                        ;--
```

```
                        .SBTTL KPON/KPOFF   PROCESS THE KEYBOARD COMMANDS: 'PON'/'POFF'
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KPON/KPOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'PON'/'POFF' KEYBOARD COMMANDS.
                        ;       IT WILL FIRST VERIFY THAT NO JUNK ARGUMENTS EXIST.
                        ;       IT WILL THEN VERIFY THAT PARITY IS PRESENT ON THE SYSTEM.
                        ;       ALL PARITY CSR'S IN THE PARITY TABLE WILL HAVE THE APPROPRIATE BITS
                        ;       SET OR CLEARED DEPENDING ON COMMAND ISSUED. THE
                        ;       REGISTERS MAY BE NORMAL PARITY OR ECC(ERROR CORRECTION CIRCUITRY).
                        ;       IF CACHE THEN THE CACHE CONTROL
                        ;       REGISTER IS CLEARED. IF THE CPU IS AN 11/70, BITS 0,1
                        ;       IN THE MEMORY CONTROL REGISTER ARE CLEARED. THE PARITY ENABLED
                        ;       MESSAGE WILL BE STUFFED. IF INDIRECT ECC CSR'S THEN THIS COMMAND IS
                        ;       NOT VALID IN BUSY MODE.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PTA
                        ;       DT.CFO
                        ;       DT.STO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.RUN - COMMAND NOT ALLOWED IN RUN MODE MESSAGE
                        ;       CM.ARG - ILLEGAL ARGUMENT MESSAGE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK - CHECK AN ARGUMENT
                        ;       ICSROO - TURN ON/OFF INDIRECT ECC CSRS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KPON IN <DTADR,CMDBUF>
                        ;       WHERE: DTADR = ADDRESS OF DATA TABLE
                        ;              CMDBUF = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR PONOF

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

                000001          $LSTTAG=1
                000001          $LSTIN=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KPON            ;KPON ENTRY POINT
                                .GLOBL  KPOFF           ;KPOFF ENTRY POINT
                                ;
                                ;****************
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CCNTRL          ;CACHE OR MEMORY CONTROL REG
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  ICSR00          ;INDIRECT ECC CSR ROUTINE
                                .GLOBL  CM.ARG          ;'INVALID OR ILLEGAL ARGUMENT' MSG
                                .GLOBL  CM.RUN          ;'ILLEGAL COMMAND IN BUSY MODE' MSG
                                .GLOBL  KONTRL          ;CONTROL WORD

                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  PO.FLAG:        .WORD 0                 ;PON/POFF FLAG
000002' 047516  050040  051101  PO.NUN: .ASCIZ  /NO PARITY%/
000010' 052111  022531     000
000015'    120  051101  052111  PO.OFF: .ASCII  /PARITY MEMORY OFF%/
000022' 020131  042515  047515
000030' 054522  047440  043106
000036'    045
000037'    000                  PO.MB0: .BYTE   0               ;WILL BE STUFFED WITH A % IF
                                                                ;ECC MEM IS PRESENT
000040' 041505  020103  042515  PO.EOF: .ASCIZ  /ECC MEMORY OFF%/
000046' 047515  054522  047440
000054' 043106  000045
000060' 040520  044522  054524  PO.ON:  .ASCII  /PARITY MEMORY ON%/
000066' 046440  046505  051117
000074' 020131  047117     045
000101'    000                  PO.MB1: .BYTE   0               ;WILL BE STUFFED WITH A % IF
                                                                ;ECC MEM IS PRESENT
000102' 041505  020103  042515  PO.EON: .ASCIZ  /ECC MEMORY ON%/
000110' 047515  054522  047440
000116' 022516     000
                000122'         .EVEN
```

```
                                .SBTTL  KPON/KPOFF ROUTINE

                                ROUTINE KPON <DTADR,BUFPTR>                     KPON:
                                ;+
                                ; SET THE FLAG TO REFLECT PON COMMAND
                                ;-

                                LET PO.FLAG := #0                                       CLR     PO.FLAG
                                INLINE <BR PARITY>                                      BR PARITY
                                ENDRTN                                          50000$:
                                                                                50001$:
                                                                                        RTS     PC

                                ROUTINE KPOFF <DTADR,BUFPTR>                    KPOFF:
                                ;+
                                ; SET THE FLAG TO REFLECT POFF COMMAND
                                ;-

                                LET PO.FLAG := #1                                       MOV     #1,PO.FLAG
                                ;+
                                ; SAVE REGISTERS, DTABLE ADDRESS AND BUFPTR
                                ;-

                                INLINE <PARITY:>                                PARITY:
                                PUSH R0,R1                                              MOV     R0,-(SP)
                                                                                        MOV     R1,-(SP)
                                LET R0 := DTADR(R5)                                     MOV     DTADR(R5),R0
                                LET R1 := BUFPTR(R5)                                    MOV     BUFPTR(R5),R1
                                ;+
                                ; PLACE MESSAGE TERMINATORS IN MESSAGES
                                ;-

                                LET PO.MB0 :B= #0                                       CLRB    PO.MB0
                                LET PO.MB1 :B= #0                                       CLRB    PO.MB1
                                ;+
                                ; INSURE NO JUNK ARGUMENTS EXIST
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>
```

```
(4)  000164' 162705  000002                                                    SUB     #1*2,R5
(3)  000170' 010546                                                            MOV     R5,-(SP)
(4)  000172' 010145                                                            MOV     R1,-(R5)
(3)  000174' 004767  000000G                                                   JSR     PC,ARGCHK
(3)  000200' 012605                                                            MOV     (SP)+,R5
(4)  000202' 012501                                                            MOV     (R5)+,R1
665
666  000204'                        IF.NO.ERROR THEN                           BCS     50002$
(6)  000204' 103404
667
668                                 ;+
669                                 ; IT'S NOT A CR SO STUFF ERROR MSG AND SCRAM...
670                                 ;-
671
672  000206'                                LET DT.KBRSP(R0) := #CM.ARG         MOV     #CM.ARG,DT.KBRSP
(4)  000206' 012760  000000G 000022
673
674  000214'                        ELSE                                       BR      50003$
(4)  000214' 000564                                                  50002$:
(3)  000216'
675
676                                 ;+
677                                 ; IT'S A CR SO SEE IF PARITY OR ECC EXISTS AT ALL....
678                                 ;-
679
680  000216'                                IF #PARPRES!ECCMEM NOTSETIN DT.CF0(R0) THEN
                                                                               BIT     #PARPRES!ECCMEM,
(6)  000216' 032760  002100  000014                                           BNE     50004$
(9)  000224' 001004
681
682                                 ;+
683                                 ; NO PARITY..OR ECC... STUFF ERROR MSG AND LEAVE....
684                                 ;-
685
686  000226'                                     LET DT.KBRSP(R0) := #PO.NUN
                                                                               MOV     #PO.NUN,DT.KBRSP
(4)  000226' 012760  000002' 000022
687
688  000234'                                ELSE                               BR      50005$
(4)  000234' 000554                                                  50004$:
(3)  000236'
689
690                                 ;+
691                                 ; DETERMINE IF INDIRECT PARITY REGISTERS IF NOT THEN GO CHECK TABLE.
692                                 ;-
693
694  000236'                                     IF #INDPAR NOTSETIN DT.CF0(R0) THEN
                                                                               BIT     #INDPAR,DT.CF0(R
(6)  000236' 032760  000040  000014                                           BNE     50006$
(9)  000244' 001016
695
696                                 ;+
697                                 ; THERE IS PARITY ON THE SYSTEM.... GET THE PARITY TABLE ADDRESS
698                                 ; THE REGISTERS MAY
699                                 ; BE PARITY OR ECC... IF 11/70, THE PARITY TABLE COULD BE EMPTY.....
700                                 ;-
701
702  000246'                                          LET R1 := DT.PTA(R0)
                                                                               MOV     DT.PTA(R0),R1
(4)  000246' 016001  000064
```

```
                                                WHILE (R1) NE #0 DO
                                                                        50007$:
                                                                                TST     (R1)
                                                                                BEQ     50010$

                        ;+
                        ; DETERMINE WHETHER PON OR POFF COMMAND AND SET OR CLEAR APPROPRIATE BITS
                        ;-

                                                        IF PO.FLAG EQ #0 THEN
                                                                                TST     PO.FLAG
                                                                                BNE     50011$
                                                                LET @(R1)+ := @(R1)+ SET.BY #3
                                                                                BIS     #3,@(R1)+
                                                        ELSE
                                                                                BR      50012$
                                                                        50011$:
                                                                LET @(R1)+ := @(R1)+ CLR.BY #3
                                                                                BIC     #3,@(R1)+
                                                        ENDIF
                                                                        50012$:
                                                ENDDO
                                                                                BR      50007$
                                                                        50010$:

                                        ELSE
                                                                                BR      50013$
                                                                        50006$:

                        ;+
                        ; THERE ARE INDIRECT ECC CSRS THEN IF IN RUN MODE DO NOT LOOK BUT RETURN
                        ;-

                                                IF #RUNMODE SETIN DT.ST0(R0) THEN
                                                                                BIT     #RUNMODE,DT.ST0(
                                                                                BEQ     50014$
                                                        LET DT.KBRSP(R0) := #CM.RUN
                                                                                MOV     #CM.RUN,DT.KBRSP
                                                        INLINE <BR 100$>
                                                                                BR 100$
                                                ELSE
                                                                                BR      50015$
                                                                        50014$:

                        ;+
                        ; IT IS OKAY TO GO TO INDIRECT CSRS
                        ; SEND FLAG A '0' TO INDICATE PON COMMAND
                        ; OR SEND FLAG A '1' TO INDICATE POFF COMMAND
                        ;-

                                                        CALL ICSR00 IN <R0,PO.FLAG>
                                                                                MOV     R5,-(SP)
                                                                                MOV     PO.FLAG,-(R5)
                                                                                MOV     R0,-(R5)
                                                                                JSR     PC,ICSR00
                                                                                MOV     (SP)+,R5
```

```
                                                        ENDIF
                                                                                50015$:
                                                ENDIF
                                                                                50013$:


                                ;+
                                ; IF CACHE PRESENT .......
                                ;-
                                        IF #CAPRES SETIN DT.CF0(R0) THEN
000342' 032760  000004  000014                                                  BIT     #CAPRES,DT.CF0(R
000350' 001415                                                                  BEQ     50016$

                                ;+
                                ; IF PON COMMAND TURN ON CACHE PARITY OTHERWISE
                                ; MUST BE A PARITY POFF COMMAND
                                ;-

                                                IF PO.FLAG EQ #0 THEN
000352' 005767  177422                                                          TST     PO.FLAG
000356' 001004                                                                  BNE     50017$
                                                        LET KONTRL := KONTRL CLR.BY #203
000360' 042767  000203  000000G                                                 BIC     #203,KONTRL
                                                ELSE
000366' 000403                                                                  BR      50020$
                                                                                50017$:
                                                        LET KONTRL := KONTRL SET.BY #203
000370' 052767  000203  000000G                                                 BIS     #203,KONTRL
                                                ENDIF
                                                                                50020$:
                                                LET @CCNTRL := KONTRL
000376' 016777  000000G 000000G                                                 MOV     KONTRL,@CCNTRL

                                        ENDIF
                                                                                50016$:
                                ;+
                                ; NOW STUFF THE APPROPRIATE BITS IN DT.ST0 AND LOAD THE ASSOCIATE MESSAGES
                                ;-


                                        IF #PARPRES SETIN DT.CF0(R0) THEN
000404' 032760  002000  000014                                                  BIT     #PARPRES,DT.CF0(
000412' 001445                                                                  BEQ     50021$
                                                IF #ECCMEM SETIN DT.CF0(R0) THEN
000414' 032760  000100  000014                                                  BIT     #ECCMEM,DT.CF0(R
000422' 001420                                                                  BEQ     50022$
                                                        IF PO.FLAG EQ #0 THEN
000424' 005767  177350                                                          TST     PO.FLAG
000430' 001007                                                                  BNE     50023$
                                                                LET PO.MB1 :B= #'%
000432' 112767  000045  177441                                                  MOVB    #'%,PO.MB1
                                                                LET DT.ST0(R0) := DT.ST0(R0) SET.BY #ECCSTAT
000440' 052760  000010  000010                                                  BIS     #ECCSTAT,DT.ST0(
                                                        ELSE
000446' 000406                                                                  BR      50024$
                                                                                50023$:
000450'
```

```
767  000450'
(4)  000450' 112767  000045  177361                                    LET PO.MB0 :B= #'%
                                                                                    MOVB    #'%,PO.MB0
768  000456'                                                           LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #ECCSTAT
(6)  000456' 042760  000010  000010                                                 BIC     #ECCSTAT,DT.ST0(
769  000464'                                                                 ENDIF
(4)  000464'                                                                                    50024$:
770  000464'                                                           ENDIF
(4)  000464'                                                                                    50022$:
771  000464'                                                           IF PO.FLAG EQ #0 THEN
(6)  000464' 005767  177310                                                         TST     PO.FLAG
(9)  000470' 001007                                                                 BNE     50025$
772  000472'                                                                  LET DT.KBRSP(R0) := #PO.ON
(4)  000472' 012760  000060' 000022                                                 MOV     #PO.ON,DT.KBRSP(
773  000500'                                                                  LET DT.ST0(R0) := DT.ST0(R0) SET.BY #PARSTAT
(6)  000500' 052760  000100  000010                                                 BIS     #PARSTAT,DT.ST0(
774  000506'                                                           ELSE
(4)  000506' 000406                                                                 BR      50026$
(3)  000510'                                                                                    50025$:
775  000510'                                                                  LET DT.KBRSP(R0) := #PO.OFF
(4)  000510' 012760  000015' 000022                                                 MOV     #PO.OFF,DT.KBRSP
776  000516'                                                                  LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #PARSTAT
(6)  000516' 042760  000100  000010                                                 BIC     #PARSTAT,DT.ST0(
777  000524'                                                           ENDIF
(4)  000524'                                                                                    50026$:
778  000524'                                                                    ELSE
(4)  000524' 000420                                                                 BR      50027$
(3)  000526'                                                                                    50021$:
779  000526'                                                           IF PO.FLAG EQ #0 THEN
(6)  000526' 005767  177246                                                         TST     PO.FLAG
(9)  000532' 001007                                                                 BNE     50030$
780  000534'                                                                  LET DT.KBRSP(R0) := #PO.EON
(4)  000534' 012760  000102' 000022                                                 MOV     #PO.EON,DT.KBRSP
781  000542'                                                                  LET DT.ST0(R0) := DT.ST0(R0) SET.BY #ECCSTAT
(6)  000542' 052760  000010  000010                                                 BIS     #ECCSTAT,DT.ST0(
782  000550'                                                           ELSE
(4)  000550' 000406                                                                 BR      50031$
(3)  000552'                                                                                    50030$:
783  000552'                                                                  LET DT.KBRSP(R0) := #PO.EOF
(4)  000552' 012760  000040' 000022                                                 MOV     #PO.EOF,DT.KBRSP
784  000560'                                                                  LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #ECCSTAT
(6)  000560' 042760  000010  000010                                                 BIC     #ECCSTAT,DT.ST0(
785  000566'                                                           ENDIF
(4)  000566'                                                                                    50031$:
786
787  000566'                                                             ENDIF                  50027$:
(4)  000566'
788  000566'                                                       ENDIF                        50005$:
(4)  000566'
789
790  000566'                                                 ENDIF                              50003$:
(4)  000566'
791
792
793                                                         ;+
794                                                         ; CLEAN UP MESS...
795                                                         ;-
```

```
                                ;
000566'                         INLINE <100$:>
000566'                                                                 100$:
000566'                         POP R1,R0
000566' 012601                                                          MOV     (SP)+,R1
000570' 012600                                                          MOV     (SP)+,R0
                                ;
000572'                         ENDRTN
000572'                                                         50000$:
000572'                                                         50001$:
000572' 000207                                                          RTS     PC
        000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CM.RUN= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
ICSROO= ****** G
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KPOFF   000132RG
KPON    000122RG
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARITY  000140R
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PO.EOF  000040R
PO.EON  000102R
PO.FLA  000000R
PO.MB0  000037R
PO.MB1  000101R
PO.NUN  000002R
PO.OFF  000015R
PO.ON   000060R
PRIO  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
```

```
STAT  = 000026        UIPDR1= 177602        $F$BLA= 000170        $ISK3 = 000001        $$BYTE= 000403
STATBI= 064757        UIPDR2= 177604        $F$CAS= 000150        $ISK4 = 000001        $$CASE= 000000
STAT1 = 000027        UIPDR3= 177606        $F$DEC= 000220        $LOCTA= 177777        $$DST = 000000
SUSPND= 000001        UIPDR4= 177610        $F$DO = 000340        $LSTIN= 000001        $$ELOC= 000402
SVR0  = 000062        UIPDR5= 177612        $F$FAL= 000405        $LSTTA= 000001        $$ERFL= 000000
SVR1  = 000064        UIPDR6= 177614        $F$GOO= 000400        $NESTL= 177777        $$FLAG= 000001
SVR2  = 000066        UIPDR7= 177616        $F$IF = 000110        $NSK0 = 000300        $$FROM= 000000
SVR3  = 000070        WASADR= 000104        $F$INC= 000210        $NSK1 = 000110        $$LOC = 000532R
SVR4  = 000072        WBSTAT= 000040        $F$LOO= 000200        $NSK2 = 000110        $$LOCN= 000000
SVR5  = 000074        WBUFEA= 000136        $F$NAM= 000160        $NSK3 = 000110        $$REG = 177777
SVR6  = 000076        WBUFPA= 000134        $F$NO = 000403        $NSK4 = 000110        $$RETU= 000000
SYSCNT= 000052        WBUFRQ= 000140        $F$OR = 000320        $NSK5 = 000110        $$RTN1= 050000
SYSERR= 000100        WBUFSZ= 000142        $F$RTI= 000350        $SAVLE= 177777        $$RTN2= 050001
TMPIO = 000002        WDFR  = 000116        $F$RTN= 000300        $SSK0 = 050010        $$SRC = 000000
TQOVF = 000002        WDTO  = 000114        $F$SEL= 000140        $TAGLE= 177777        $$TGSV= 000000
UIPAR0= 177640        WTINRE= 000352        $F$THE= 000330        $TAGNU= 050032        $$TGS1= 000000
UIPAR1= 177642        WTWHMI= 000222        $F$TRU= 000404        $TEMP = 000300        $$TGS2= 000000
UIPAR2= 177644        XFLAG = 000005        $F$UNT= 000130        $TSK0 = 050003        $$TO  = 000002
UIPAR3= 177646        XOFF  = 000023        $F$WHI= 000120        $TSK1 = 050005        $$$TAG= 050000
UIPAR4= 177650        XON   = 000021        $F$YES= 000402        $TSK2 = 050027        .     = 000574R
UIPAR5= 177652        $BGNLE= 177777        $IFLEV= 177777        $TSK3 = 050031
UIPAR6= 177654        $ERFLG= 000400        $ISK0 = 000001        $TSK4 = 050024
UIPAR7= 177656        $F$AND= 000310        $ISK1 = 000001        $TSK5 = 050012
UIPDR0= 177600        $F$BAD= 000401        $ISK2 = 000001        $$ARGC= 000004

. ABS.  000000     000
        000574     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PONOF,DSKZ:PONOF=SPMAC/ML,EQUATE,PONOF
 RUN-TIME: 19 9 .4 SECONDS
 RUN-TIME RATIO: 111/29=3.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PRBTOD PROCESS THE 'BTOD' MACRO IN OPTION MODULES
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRBTOD
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'BTOD' TRAP ISSUED BY OPTION
                        ;       MODULES. IT WILL RETRIEVE THE NUMBER TO BE CONVERTED TO
                        ;       DECIMAL ASCII FROM DT.PC+2 AND THE TABLE ADDRESS AT WHICH
                        ;       TO STORE THE DECIMAL ASCII FROM DT.PC +4. IT CALLS THE
                        ;       BINARY TO DECIMAL ASCII CONVERSION ROUTINE. THE RETURN PC
                        ;       IS UPDATED SINCE THIS IS DIRECT SERVICE.
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       BDACNV - BINARY TO DECIMAL ASCII CONVERSION ROUTINE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRBTOD IN <DTADR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR PRBTOD

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

                                $LSTIN=1
                        000001  $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  PRBTOD          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  BDACNV          ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
                                ;
```

```
PRBTOD.MAC    28-JUL-78 09:25          PRBTOD ROUTINE

   578                                 .SBTTL  PRBTOD ROUTINE
   579
   580  000000'                        ROUTINE PRBTOD <DTADR>                          PRBTOD:
   (2)  000000'
   581
   582                                 ;+
   583                                 ;INITIALIZE AND SAVE DATA TABLE ADDRESS
   584                                 ;-
   585
   586
   587  000000'                        PUSH R0,R1                                              MOV     R0,-(SP)
   (2)  000000' 010046                                                                         MOV     R1,-(SP)
   (3)  000002' 010146
   588  000004'                        LET R0 := DTADR(R5)                                     MOV     DTADR(R5),R0
   (4)  000004' 016500  000000
   589
   590                                 ;+
   591                                 ; GET DT.PC AND CALL BDACNV
   592                                 ;-
   593
   594  000010'                        LET R1 := DT.PC(R0)                                     MOV     DT.PC(R0),R1
   (4)  000010' 016001  000002
   595  000014'                        CALL BDACNV IN <@2(R1),4(R1)>                           MOV     R5,-(SP)
   (3)  000014' 010546                                                                         MOV     4(R1),-(R5)
   (5)  000016' 016145  000004                                                                 MOV     @2(R1),-(R5)
   (4)  000022' 017145  000002                                                                 JSR     PC,BDACNV
   (3)  000026' 004767  000000G                                                               MOV     (SP)+,R5
   (3)  000032' 012605
   596
   597                                 ;+
   598                                 ; UPDATE RETURN PC
   599                                 ;-
   600
   601  000034'                        LET DT.PC(R0) := DT.PC(R0) + #6                         ADD     #6,DT.PC(R0)
   (6)  000034' 062760  000006  000002
   602
   603                                 ;+
   604                                 ; CLEAN UP AND GOODBYE
   605                                 ;-
   606
   607  000042'                        POP R1,R0                                               MOV     (SP)+,R1
   (2)  000042' 012601                                                                         MOV     (SP)+,R0
   (3)  000044' 012600
   608
   609  000046'                        ENDRTN                                          50000$:
   (3)  000046'                                                                        50001$:
   (3)  000046'                                                                                RTS     PC
   (2)  000046' 000207
   610
   611          000001                 .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BDACNV= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRBTOD  000000RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
```

```
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000002
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000050R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000

. ABS.  000000      000
        000050      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRBTOD,DSKZ:PRBTOD=SPMAC/ML,EQUATE,PRBTOD
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 33/13=2.5
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE PREVT (PROCESS EVENT MODULE)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PREVT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE DISPATCHES CONTROL TO VARIOUS EVENT
                        ;       HANDLING ROUTINES.  THE ADDRESS OF A DATA TABLE
                        ;       IS PASSED TO THIS MODULE. THE EVENT CODE IS
                        ;       EXTRACTED FROM THE TABLE AND AN OFFSET IS GENERATED.
                        ;       THIS OFFSET IS USED TO FORM ADDRESS IN THE DISPATCH
                        ;       TABLE TO SEND CONTROL TO THE PROPER EVENT HANDLING
                        ;       ROUTINE.
                        ;
                        ;       NOTE:  IF EVENT CODE IS 'ZERO', THE STACK IS POPPED
                        ;              AND CONTROL GOES BACK TO CALLER.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       EVENT CODE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       IT CALLS THE ACTUAL EVENT HANDLING ROUTINE
                        ;       THROUGH A POINTER AND THE FORMAT IS:
                        ;               CALL PTR IN <DTADDR>
                        ;                   WHERE DTADDR = DATA TABLE ADDRESS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ; CALLING SEQUENCE:
                        ;       CALL PREVT IN <DTADR>
                        ;               WHERE DTADR = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       -----   ----------  ----------  ------------------
                        ;
                        ;--
```

```
                            .SBTTL COMMON DEFINITIONS & REFERENCES

                            .MCALL STRUCT
000000'                     STRUCT
(1) 000000'                 .PRINT  ;SPMAC: VERSION 1.1
          000001            $LSTIN=1
          000001            $LSTTAG=1
                            ;
                            ;****************
                            ;
                            ;REFERENCED BY OTHER MODULES:
                            ;
                            .GLOBL PREVT                    ;MODULE ENTRY POINT

                            ;****************
                            ; GLOBAL REFERENCES
                            ;


                            .GLOBL PENDIT                   ;ENDIT HANDLER
                            .GLOBL PEND                     ;END HANDLER
                            .GLOBL PRHARD                   ;HRDER HANDLER
                            .GLOBL PDATER                   ;DATER HANDLER
                            .GLOBL PRMSG                    ;MSG HANDLER
                            .GLOBL PBREAK                   ;BREAK HANDLER
                            .GLOBL PRSOFT                   ;SOFER HANDLER
                            .GLOBL PRMSGN                   ;MSGN HANDLER
                            .GLOBL PRGWB                    ;GWBUFF HANDLER
                            .GLOBL PGETPA                   ;GETPA HANDLER
                            .GLOBL PDATCK                   ;DATACK HANDLER
                            .GLOBL PCDATA                   ;CDATA HANDLER
                            .GLOBL PMAP22                   ;MAP22 HANDLER
                            .GLOBL PRMSGS                   ;MSGS HANDLER
                            .GLOBL PRRAND                   ;RAND TRAP HANDLER
                            .GLOBL PARERR                   ;MEMORY PARITY ERROR HANDLER
                            .GLOBL KTERR                    ;MEMORY MGMT VIOL. HANDLER
                            .GLOBL BUSERR                   ;TIMEOUT,ILLEGAL INSTR. ETC.
                            .GLOBL PROTOA                   ;OCTAL-TO-DECIMAL EVENT
                            .GLOBL PRBTOD                   ;BIN-TO-DECIMAL EVENT

                            ;****************
                            ;
                            ; LOCAL STORAGE:
                            ;
                            ; DISPATCH TABLE FOR SOFT-TRAP EVENTS
                            ;
000000'                     PR.TRPTBL:                      ;ADDRESS OF EVENT HANDLER
000000' 000000                      .WORD   0               ; EXIT IS ZERO - DOES NOT NEED A ROUTINE
000002' 000000G                     PRMSG                   ;
000004' 000000G                     PRMSGS                  ;
000006' 000000G                     PRMSGN                  ;
000010' 000000G                     PDATER                  ;
000012' 000000G                     PRHARD                  ;
000014' 000000G                     PRSOFT                  ;
000016' 000000G                     PBREAK                  ;
000020' 000000G                     PEND                    ;
```

```
                                        PDATCK                  ;
                                        PCDATA                  ;
                                        PENDIT                  ;
                                        PRGWB                   ;
                                        PGETPA                  ;
                                        PMAP22                  ;
                                        PRRAND                  ;
                                        PROTOA                  ;
                                        PRBTOD                  ;

                                ;****************
                                ; DISPATCH TABLE FOR HARD TRAPS & MISC. EVENTS
                                ;
000044'                         PR.NTRPTBL:                     ;ADDRESS OF EVENT HANDLER
000044' 000000G                         BUSERR                  ;
000046' 000000G                         BUSERR                  ;
000050' 000000G                         PARERR                  ;
000052' 000000G                         KTERR                   ;
```

```
                        .SBTTL PREVT (CODE)


                        ROUTINE PREVT <DTBL>
000054'                                                 PREVT:
(2)  000054'

                        ;+
                        ; SAVE REGISTERS,FETCH DTABLE ADDRESS & EVENT CODE.
                        ; IF EVENT CODE IS 0, RETURN.
                        ; ELSE CHECK IF EVENT CODE IS A HARDWARE-TRAP TYPE.
                        ;-

                        PUSH R0,R1
000054'                                                         MOV     R0,-(SP)
(2)  000054' 010046                                             MOV     R1,-(SP)
(3)  000056' 010146

                        LET R0 := DTBL(R5)
000060'                                                         MOV     DTBL(R5),R0
(4)  000060' 016500  000000

                        LET R1 := DT.EVNT(R0)
000064'                                                         MOV     DT.EVNT(R0),R1
(4)  000064' 016001  000000

                        IF R1 NE #0 THEN
000070'                                                         TST     R1
(6)  000070' 005701                                             BEQ     50002$
(9)  000072' 001421

                        ;+
                        ; DEVELOP OFFSET INTO TABLE
                        ;-
                                LET R1 := R1 CLR.BY #177700
000074'                                                         BIC     #177700,R1
(6)  000074' 042701  177700
                                LET R1 := R1 SHIFT 1
000100'                                                         ASL     R1
(7)  000100' 006301

                                IF DT.EVNT(R0) GE #EVNTHD THEN
000102'                                                         CMP     DT.EVNT(R0),#EVN
(6)  000102' 026027  000000  000200                            BLT     50003$
(9)  000110' 002403

                        ;+
                        ; IT IS A HARDWARE-TRAP EVENT.
                        ; DEVELOP ADDRESS.
                        ;-

                                    LET R1 := R1 + #PR.NTRPTBL
000112'                                                         ADD     #PR.NTRPTBL,R1
(6)  000112' 062701  000044'
                                ELSE
000116'                                                         BR      50004$
(4)  000116' 000402                                     50003$:
(3)  000120'

                        ;+
                        ; IT IS A SOFTWARE TRAP EVENT.
                        ; FORM AN ADDRESS
                        ;-

                                    LET R1 := R1 + #PR.TRPTBL
000120'                                                         ADD     #PR.TRPTBL,R1
(6)  000120' 062701  000000'
                                ENDIF
000124'                                                 50004$:
(4)  000124'
                                CALL @(R1) IN <R0>
000124'                                                         MOV     R5,-(SP)
(3)  000124' 010546
```

```
(4)  000126' 010045                                                     MOV     R0,-(R5)
(3)  000130' 004771  000000                                             JSR     PC,@(R1)
(3)  000134' 012605                                                     MOV     (SP)+,R5
677
678  000136'                        ENDIF                           50002$:
(4)  000136'
679                                 ;+
680                                 ; RESTORE REGISTERS AND RETURN
681                                 ;-
682
683  000136'                                POP R1,R0
(2)  000136' 012601                                                     MOV     (SP)+,R1
(3)  000140' 012600                                                     MOV     (SP)+,R0
684  000142'                        ENDRTN                          50000$:
(3)  000142'                                                        50001$:
(3)  000142'                                                            RTS     PC
(2)  000142' 000207
685          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUSERR= ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTBL  = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR = ****** G
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARERR= ****** G
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PBREAK= ****** G
PCDATA= ****** G
PDATCK= ****** G
PDATER= ****** G
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PEND  = ****** G
PENDIT= ****** G
PGETPA= ****** G
PMAP22= ****** G
PRBTOD= ****** G
PREVT   000054RG
PRGWB = ****** G
PRHARD= ****** G
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRMSG = ****** G
PRMSGN= ****** G
PRMSGS= ****** G
PROTOA= ****** G
PRRAND= ****** G
PRSOFT= ****** G
PR.NTR  000044R
PR.TRP  000000R
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
```

```
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000110
$NSK2 = 000110
$SAVLE= 177777
$TAGLE= 177777
$TAGNU= 050005
$TEMP = 000300
$TSK0 = 050002
$TSK1 = 050004
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000110R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000001
$$$TAG= 050000
.     = 000144R

. ABS.  000000    000
        000144    001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:PREVT,DSKZ:PREVT=SPMAC/ML,EQUATE,PREVT
RUN-TIME: 12 2 .4 SECONDS
RUN-TIME RATIO: 39/15=2.5
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PRGWB - PROCESS GWBUF
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRGWB
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       PROCESSES GWBUF CALLS FROM OPTION MODULES.  DETERMINES
                        ;       AND PASSES TO A REQUESTING OPTION MODULE A BUFFER
                        ;       SIZE AND STARTING ADDRESS.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ADDR,,DT.SSIZ,DT.ESIZ,DT.PC,DT.STO,DT.WBUF,
                        ;       DT.WLLMT,DT.WHLMT,DT.OFFSET
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.WBUF
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRGWB IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINTIONS AND REFERENCES
                                .MCALL STRUCT
000000'                         STRUCT
(1)  000000'                    .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  PRGWB                           ;MODULE ENTRY POINT
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                          ;SAVE REGISTERS
                                .GLOBL  RESREG                          ;RESTORE REGISTERS
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  PG.AVAIL:       .WORD 0                 ;AVAILABLE BUFFER SIZE - PAR FORMAT
000002' 000000                  PG.SIZE:        .WORD 0                 ;ACTUAL WRITE BUFFER SIZE - PAR FORMAT
                                ;
                                ;***************
                                ;
```

```
                                .SBTTL PRGWB ROUTINE
                                ROUTINE PRGWB <TABL>
000004'                                                                 PRGWB:
(2)  000004'
                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
000004'                                                                         JSR     PC,SAVREG
(3)  000004' 004767  000000G

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := TABL(R5)
000010'                                                                         MOV     TABL(R5),R0
(4)  000010' 016500  000000

                                ;+
                                ; GET THE OPTIONS MODULE'S HEADER ADDRESS
                                ;-

                                LET R1 := @DT.PC(R0)
000014'                                                                         MOV     @DT.PC(R0),R1
(4)  000014' 017001  000002

                                ;+
                                ; CALCULATE THE LARGEST AVAILABLE BUFFER SIZE.  IF THE BUFFER
                                ; ADDRESS POINTER IS POINTING BELOW THE MOVABLE PORTION OF
                                ; THE EXERCISER, THEN THE AVAILABLE BUFFER SIZE IS EVERYTHING FROM THE
                                ; CURRENT POINTER POSITION UP TO THE BASE ADDRESS OF THE MOVABLE PORTION
                                ; OF THE EXERCISER.  IF THE BUFFER ADDRESS POINTER IS ABOVE THE TOP OF
                                ; THE EXERCISER, THEN THE AVAILABLE BUFFER SIZE IS EVERYTHING FROM
                                ; THE CURRENT POINTER POSITION UP TO THE HIGH BUFFER LIMIT.
                                ;-

                                IF DT.WBUF(R0) LO DT.ADDR(R0) THEN
000020'                                                                         CMP     DT.WBUF(R0),DT.A
(6)  000020' 026060  000050  000042                                            BHIS    50002$
(9)  000026' 103007
                                        LET PG.AVAIL := DT.ADDR(R0) - DT.WBUF(R0)
000030'                                                                         MOV     DT.ADDR(R0),PG.A
(4)  000030' 016067  000042  177742                                            SUB     DT.WBUF(R0),PG.A
(6)  000036' 166067  000050  177734
                                ELSE
000044'                                                                         BR      50003$
(4)  000044' 000406
(3)  000046'                                                            50002$:
                                        LET PG.AVAIL := DT.WHLMT(R0) - DT.WBUF(R0)
000046'                                                                         MOV     DT.WHLMT(R0),PG.
(4)  000046' 016067  000054  177724                                            SUB     DT.WBUF(R0),PG.A
(6)  000054' 166067  000050  177716
                                ENDIF
000062'                                                                 50003$:
(4)  000062'

                                ;+
                                ; GET THE REQUESTED BUFFER SIZE FROM THE OPTION MODULE, CONVERT IT TO
```

```
                        ; PAR FORMAT, DOUBLE IT TO CHANGE IT FROM A WORD COUNT TO A MEMORY AREA SIZE, AND
                        ; COMPARE IT TO THE AVAILABLE BUFFER SIZE.  IF THE REQUESTED SIZE IS LARGER THAN THAT
                        ; AVAILABLE, LET THE ACTUAL BUFFER SIZE EQUAL THE AVAILABLE SIZE.  OTHERWISE, LET
                        ; THE ACTUAL SIZE BE THE FULL REQUESTED SIZE.  THEN PASS THE ACTUAL SIZE TO THE
                        ; OPTION MODULE, AFTER CONVERTING IT TO A PHYSICAL WORD COUNT FROM PAR FORMAT.
                        ;-

000062'                 LET R2 := WBUFRQ(R1) SHIFT #-6
(4) 000062' 016102 000140                                       MOV     WBUFRQ(R1),R2
(7) 000066' 006202                                              ASR     R2
(7) 000070' 006202                                              ASR     R2
(7) 000072' 006202                                              ASR     R2
(7) 000074' 006202                                              ASR     R2
(7) 000076' 006202                                              ASR     R2
(7) 000100' 006202                                              ASR     R2
000102'                 LET R2 := R2 CLR.BY #176000
(6) 000102' 042702 176000                                       BIC     #176000,R2
000106'                 LET R2 := R2 SHIFT #+1
(7) 000106' 006302                                              ASL     R2
000110'                 IF R2 HI PG.AVAIL THEN
(6) 000110' 020267 177664                                       CMP     R2,PG.AVAIL
(9) 000114' 101404                                              BLOS    50004$
000116'                         LET PG.SIZE := PG.AVAIL
(4) 000116' 016767 177656 177656                                MOV     PG.AVAIL,PG.SIZE
000124'                 ELSE
(4) 000124' 000402                                              BR      50005$
(3) 000126'                                                 50004$:
000126'                         LET PG.SIZE := R2
(4) 000126' 010267 177650                                       MOV     R2,PG.SIZE
000132'                 ENDIF
(4) 000132'                                                 50005$:
000132'                 LET WBUFSZ(R1) := PG.SIZE SHIFT #+6
(4) 000132' 016761 177644 000142                                MOV     PG.SIZE,WBUFSZ(R
(7) 000140' 006361 000142                                       ASL     WBUFSZ(R1)
(7) 000144' 006361 000142                                       ASL     WBUFSZ(R1)
(7) 000150' 006361 000142                                       ASL     WBUFSZ(R1)
(7) 000154' 006361 000142                                       ASL     WBUFSZ(R1)
(7) 000160' 006361 000142                                       ASL     WBUFSZ(R1)
(7) 000164' 006361 000142                                       ASL     WBUFSZ(R1)
000170'                 LET WBUFSZ(R1) := WBUFSZ(R1) SHIFT #-1
(7) 000170' 006261 000142                                       ASR     WBUFSZ(R1)
000174'                 LET WBUFSZ(R1) := WBUFSZ(R1) CLR.BY #100000
(6) 000174' 042761 100000 000142                                BIC     #100000,WBUFSZ(R


                        ;+
                        ; CONVERT THE BUFFER ADDRESS POINTER FROM PAR FORMAT TO AN 18-BIT
                        ; VIRTUAL ADDRESS, AND PASS IT TO THE OPTION MODULE.
                        ;-

000202'                 LET R2 := DT.WBUF(R0) - DT.OFFSET(R0)
(4) 000202' 016002 000050                                       MOV     DT.WBUF(R0),R2
(6) 000206' 166002 000070                                       SUB     DT.OFFSET(R0),R2
000212'                 LET R3 := R2
(4) 000212' 010203                                              MOV     R2,R3
000214'                 LET R2 := R2 SHIFT #+6
```

```
(7)  000214' 006302                                                          ASL     R2
(7)  000216' 006302                                                          ASL     R2
(7)  000220' 006302                                                          ASL     R2
(7)  000222' 006302                                                          ASL     R2
(7)  000224' 006302                                                          ASL     R2
(7)  000226' 006302                                                          ASL     R2
657  000230'                        LET R3 := SWAP R3                        SWAB    R3
(6)  000230' 000303
658  000232'                        LET R3 := R3 SHIFT #+2                   ASL     R3
(7)  000232' 006303                                                          ASL     R3
(7)  000234' 006303
659  000236'                        LET R3 := R3 CLR.BY #177717              BIC     #177717,R3
(6)  000236' 042703  177717
660  000242'                        LET WBUFPA(R1) := R2                     MOV     R2,WBUFPA(R1)
(4)  000242' 010261  000134
661  000246'                        LET WBUFEA(R1) := R3                     MOV     R3,WBUFEA(R1)
(4)  000246' 010361  000136
662
663
664                                 ;+
665                                 ; IF WRITE BUFFER ROTATION IS ENABLED, FIND A NEW VALUE FOR THE WRITE
666                                 ; BUFFER ADDRESS POINTER.  OTHERWISE, JUST RETURN.
667                                 ;-
668
669  000252'                        IF #WBSTAT SETIN DT.STO(R0) THEN         BIT     #WBSTAT,DT.STO(R
(6)  000252' 032760  000040  000010                                         BEQ     50006$
(9)  000260' 001470
670
671
672                                 ;+
673                                 ; LET THE NEW BUFFER ADDPESS POINTER EQUAL THE CURRENT POINTER PLUS
674                                 ; THE SIZE OF THE BUFFER WE JUST PASSED TO THE OPTION MODULE.
675                                 ; ADD ONE TO THIS VALUE TO GET THE NEXT 100 BYTE BLOCK, TO
676                                 ; ASSURE THAT THERE IS NO OVERLAPPING OF WRITE BUFFERS.
677                                 ;-
678
679  000262'                                LET DT.WBUF(R0) := DT.WBUF(R0) + PG.SIZE + #1
(6)  000262' 066760  177514  000050                                          ADD     PG.SIZE,DT.WBUF(
(7)  000270' 005260  000050                                                  INC     DT.WBUF(R0)
680
681
682                                 ;+
683                                 ; IF THIS NEW BUFFER STARTING ADDRESS IS GREATER THAN OR EQUAL TO THE
684                                 ; HIGH BUFFER LIMIT MINUS 1/2 K (THE 1/2 K IS TO ASSURE THAT WE HAVE A WRITE
685                                 ; BUFFER AREA OF AT LEAST THAT SIZE), THEN MOVE THE BUFFER ADDRESS POINTER TO THE
686                                 ; LOW BUFFER LIMIT.
687                                 ;-
688
689  000274'                                LET R3 := DT.WHLMT(R0) - #20
(4)  000274' 016003  000054                                                  MOV     DT.WHLMT(R0),R3
(6)  000300' 162703  000020                                                  SUB     #20,R3
690  000304'                                IF DT.WBUF(R0) HIS R3 THEN
(6)  000304' 026003  000050                                                  CMP     DT.WBUF(R0),R3
(9)  000310' 103403                                                          BLO     50007$
691  000312'                                       LET DT.WBUF(R0) := DT.WLLMT(R0)
(4)  000312' 016060  000052  000050                                          MOV     DT.WLLMT(R0),DT.
```

```
692  000320'                          ENDIF
(4)  000320'                                                                50007$:
693
694
695                                 ;+
696                                 ; NOW CHECK TO SEE IF THE POINTER HAS BEEN PLACED WITHIN THE CURRENT
697                                 ; POSITION OF THE EXERCISER.  TO DO THIS, WE FIRST CALCULATE THE CURRENT POSITION
698                                 ; OF THE TOP OF THE EXERCISER (WITHIN 100 BYTES, BECAUSE WE ARE WORKING WITH
699                                 ; PAR-FORMATTED WORDS).  THEN SEE IF THE CONTENTS OF THE POINTER IS LESS THAN
700                                 ; THIS VALUE BUT GREATER THAN THE BASE ADDRESS OF THE MOVABLE PORTION OF THE
701                                 ; EXERCISER MINUS 1/2 K.  IF IT IS NOT, THEN WE HAVE A GOOD POINTER VALUE.
702                                 ; BUT IF IT IS, WE MUST THEN CHECK TO SEE IF THE TOP OF THE
703                                 ; EXERCISER IS WITHIN 1/2 K OF THE HIGH BUFFER LIMIT.  IF YES,
704                                 ; AND IF THE MOVABLE PORTION OF THE EXERCISER IS NOT IN LOWEST MEMORY, MOVE THE
705                                 ; BUFFER ADDRESS POINTER DOWN TO THE LOW BUFFER LIMIT.
706                                 ; OTHERWISE, MOVE THE POINTER TO THE TOP OF THE EXERCISER.
707                                 ;-
708
709
710  000320'                            LET R2 := DT.ESIZ(R0) SHIFT #-6
(4)  000320' 016002  000044                                                 MOV     DT.ESIZ(R0),R2
(7)  000324' 006202                                                         ASR     R2
(7)  000326' 006202                                                         ASR     R2
(7)  000330' 006202                                                         ASR     R2
(7)  000332' 006202                                                         ASR     R2
(7)  000334' 006202                                                         ASR     R2
(7)  000336' 006202                                                         ASR     R2
711  000340'                            LET R2 := R2 CLR.BY #176000
(6)  000340' 042702  176000                                                 BIC     #176000,R2
712  000344'                            LET R2 := R2 + #1
(6)  000344' 005202                                                         INC     R2
713  000346'                            LET R2 := R2 + DT.ADDR(R0) - #200
(6)  000346' 066002  000042                                                 ADD     DT.ADDR(R0),R2
(7)  000352' 162702  000200                                                 SUB     #200,R2
714
715  000356'                            LET R3 := DT.ADDR(R0) - #20
(4)  000356' 016003  000042                                                 MOV     DT.ADDR(R0),R3
(6)  000362' 162703  000020                                                 SUB     #20,R3
716
717  000366'                            IF DT.WBUF(R0) LOS R2 AND DT.WBUF(R0) HIS R3 THEN
(6)  000366' 026002  000050                                                 CMP     DT.WBUF(R0),R2
(9)  000372' 101023                                                         BHI     50010$
(6)  000374' 026003  000050                                                 CMP     DT.WBUF(R0),R3
(9)  000400' 103420                                                         BLO     50010$
718  000402'                                LET R4 := DT.WHLMT(R0) - #20
(4)  000402' 016004  000054                                                 MOV     DT.WHLMT(R0),R4
(6)  000406' 162704  000020                                                 SUB     #20,R4
719  000412'                                IF R2 HIS R4 AND DT.ADDR(R0) HI #200 THEN
(6)  000412' 020204                                                         CMP     R2,R4
(9)  000414' 103410                                                         BLO     50011$
(6)  000416' 026027  000042  000200                                        CMP     DT.ADDR(R0),#200
(9)  000424' 101404                                                         BLOS    50011$
720  000426'                                    LET DT.WBUF(R0) := DT.WLLMT(R0)
(4)  000426' 016060  000052  000050                                        MOV     DT.WLLMT(R0),DT.
721  000434'                                ELSE
(4)  000434' 000402                                                         BR      50012$
```

```
(3)  000436'                                                                    50011$:
722  000436'                                    LET DT.WBUF(R0) := R2                    MOV     R2,DT.WBUF(R0)
(4)  000436' 010260  000050
723  000442'                                ENDIF                                50012$:
(4)  000442'
724  000442'                            ENDIF                                    50010$:
(4)  000442'
725
726  000442'                        ENDIF                                        50006$:
(4)  000442'
727
728
729                                 ;+
730                                 ; UPDATE DT.PC TO REFLECT THE ADDRESS TO RETURN TO (I.E., THE ADDRESS
731                                 ; AFTER THE GWBUF TRAP CALL.
732                                 ;-
733
734  000442'                        LET DT.PC(R0) := DT.PC(R0) + #2                  ADD     #2,DT.PC(R0)
(6)  000442' 062760  000002  000002
735
736
737                                 ;+
738                                 ; RESTORE REGISTERS AND RETURN
739                                 ;-
740
741  000450'                        CALL RESREG                                      JSR     PC,RESREG
(3)  000450' 004767  000000G
742
743
744  000454'                        ENDRTN                                       50000$:
(3)  000454'                                                                    50001$:
(3)  000454'                                                                        RTS     PC
(2)  000454' 000207
745          000001                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PG.AVA  000000R
PG.SIZ  000002R
PRGWB   000004RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TABL  = 000000
```

```
TMPIO = 000002          WBSTAT= 000040          $F$DO = 000340          $ISK1 = 000001          $$CASE= 000000
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $ISK2 = 000001          $$DST = 000000
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LOCTA= 177777          $$ELOC= 000402
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $LSTIN= 000001          $$ERFL= 000000
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $LSTTA= 000001          $$FLAG= 000001
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NESTL= 177777          $$FROM= 000000
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK0 = 000300          $$LOC = 000424R
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $NSK1 = 000110          $$LOCN= 000000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $NSK2 = 000110          $$REG = 177777
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $NSK3 = 000110          $$RETU= 000000
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $SAVLE= 177777          $$RTN1= 050000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TAGLE= 177777          $$RTN2= 050001
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TAGNU= 050013          $$SRC = 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TEMP = 000300          $$TGSV= 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $TSK0 = 050006          $$TGS1= 000000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $TSK1 = 050010          $$TGS2= 000000
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $TSK2 = 050012          $$TO  = 000000
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$ARGC= 000002          $$$TAG= 050000
WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$BYTE= 000403          .     = 000456R


. ABS.  000000    000
        000456    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRGWB,DSKZ:PRGWB=SPMAC/ML,EQUATE,PRGWB
 RUN-TIME: 18 8 .4 SECONDS
 RUN-TIME RATIO: 64/27=2.3
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  PRHARD (PROCESS HARD ERRORS)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRHARD
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS CALLED AS A RESULT OF A 'HRDERR' TRAP EXECUTED
                        ;       BY AN OPTION MODULE.
                        ;
                        ;       FIRST THE FOLLOWING IS DONE
                        ;
                        ;           1. THE MODULE'S HARD ERROR COUNT IS INCREMENTED.
                        ;           2. ERROR MESSAGE IS QUEUED UP.
                        ;
                        ;       IF THE ENVIRONMENT IS APT, THEN AN APT ERROR HANDLING ROUTINE
                        ;       IS CALLED.
                        ;
                        ;       IF NOT UNDER APT ENVIRONMENT, THEN
                        ;
                        ;           1. IF BIT15 OF THE SOFTWARE SWITCH REGISTER IS SET, THE
                        ;              MODULE IS DROPPED WITH 'MODULE DROPPED' MESSAGE PRINTED,
                        ;              EVEN ON THE FIRST ERROR.
                        ;           2. IF THE ERROR COUNT IS EQUAL TO OR GREATER THAN THE
                        ;              LIMIT, THEN BIT14 OF THE SOFTWARE SWITCH REGISTER IS
                        ;              CHECKED. IF BIT14 IS SET, THE MODULE IS NOT DROPPED.
                        ;
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0,DT.CF0,DT.HMX
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. DRPMOD               ;DROP A MODULE ROUTINE
                        ;       2. ENQTQ                ;ENQUE AN ERROR MSG ROUTINE
                        ;       3. APTHER               ;APT HARD ERROR ROUTINE
                        ;       4. SAVREG
                        ;       5. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       UNDER APT ENVIRONMENT IT CAN SHUT DOWN THE EXCERCISER.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRHARD IN <DT>
```

```
                        ;       WHERE DT = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL PRHARD (COMMON DEFINITIONS & REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN=1
            000001              $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL PRHARD                   ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL APTHER                   ;HARD ERROR HANDLER UNDER APT
                                .GLOBL ENQTQ                    ;ENQUE A MESSAGE INTO THE TYPE QUEUE
                                .GLOBL DRPMOD                   ;'DROP THE MODULE' ROUTINE
                                .GLOBL SAVREG
                                .GLOBL RESREG
```

```
                                .SBTTL PRHARD (CODE)

000000'                         ROUTINE PRHARD <DTA>
(2) 000000'                                                                 PRHARD:

                                ;+
                                ; SAVE REGS, GET DTABLE ADDRESS AND MODULE
                                ; HEADER ADDRESS.
                                ;-

000000'                         CALL SAVREG
(3) 000000' 004767  000000G                                                 JSR     PC,SAVREG
000004'                         LET R0 := DTA(R5)
(4) 000004' 016500  000000                                                  MOV     DTA(R5),R0
000010'                         LET R1 := DT.PC(R0)
(4) 000010' 016001  000002                                                  MOV     DT.PC(R0),R1
000014'                         LET R2 := (R1)
(4) 000014' 011102                                                          MOV     (R1),R2

                                ;+
                                ; POINT TO TRAP PC
                                ;-
000016'                         LET R3 := R1 - #2
(4) 000016' 010103                                                          MOV     R1,R3
(6) 000020' 162703  000002                                                  SUB     #2,R3

                                ;+
                                ; POINT R1 TO RETURN PC
                                ;-

000024'                         LET R1 := R1 + #4
(6) 000024' 062701  000004                                                  ADD     #4,R1

                                ;+
                                ; INCREMENT MODULE'S ERROR COUNT, AND ENQUE ERROR MESSAGE.
                                ;-

000030'                         LET HRDCNT(R2) := HRDCNT(R2) + #1
(6) 000030' 005262  000044                                                  INC     HRDCNT(R2)
000034'                         LET HRDPAS(R2) := HRDPAS(R2) + #1
(6) 000034' 005262  000050                                                  INC     HRDPAS(R2)
000040'                         CALL ENQTQ IN <R0,#MSGHRD,R3,R2,R1>
(3) 000040' 010546                                                          MOV     R5,-(SP)
(8) 000042' 010145                                                          MOV     R1,-(R5)
(7) 000044' 010245                                                          MOV     R2,-(R5)
(6) 000046' 010345                                                          MOV     R3,-(R5)
(5) 000050' 012745  000007                                                  MOV     #MSGHRD,-(R5)
(4) 000054' 010045                                                          MOV     R0,-(R5)
(3) 000056' 004767  000000G                                                 JSR     PC,ENQTQ
(3) 000062' 012605                                                          MOV     (SP)+,R5

                                ;+
                                ; IF APT ENVIRONMENT, CALL APT ERROR HANDLING ROUTINE
                                ;-

000064'                         IF #APTPRE SETIN DT.CF0(R0) THEN
```

```
(6)  000064' 032760  000200  000014                                        BIT     #APTPRE,DT.CF0(R
(9)  000072' 001406                                                        BEQ     50002$
634  000074'                        CALL APTHERR IN <R0>
(3)  000074' 010546                                                        MOV     R5,-(SP)
(4)  000076' 010045                                                        MOV     R0,-(R5)
(3)  000100' 004767  000000G                                              JSR     PC,APTHERR
(3)  000104' 012605                                                        MOV     (SP)+,R5
635  000106'                    ELSE
(4)  000106' 000431                                                        BR      50003$
(3)  000110'                                                          50002$:
636
637                             ;+
638                             ; THE ENVIRONMENT IS NOT APT.
639                             ; CHECK SWITCH REGISTER IF THE MODULE IS TO BE DROPPED
640                             ; DUE TO THIS ERROR.
641                             ;-
642
643  000110'                        IF #BIT15 SETIN DT.SWR(R0) THEN
(6)  000110' 032760  100000  000056                                        BIT     #BIT15,DT.SWR(R0
(9)  000116' 001407                                                        BEQ     50004$
644  000120'                            CALL DRPMOD IN <R0,R2>
(3)  000120' 010546                                                        MOV     R5,-(SP)
(5)  000122' 010245                                                        MOV     R2,-(R5)
(4)  000124' 010045                                                        MOV     R0,-(R5)
(3)  000126' 004767  000000G                                              JSR     PC,DRPMOD
(3)  000132' 012605                                                        MOV     (SP)+,R5
645  000134'                        ELSE
(4)  000134' 000416                                                        BR      50005$
(3)  000136'                                                          50004$:
646
647                             ;+
648                             ; IF TOO MANY HARD ERRORS AND BIT 14 NOT SET IN SWR, DROP MODULE
649                             ;-
650  000136'                            IF HRDCNT(R2) HIS DT.HMX(R0) AND #BIT14 NOTSETIN DT.SWR(R0) THEN
(6)  000136' 026260  000044  000104                                        CMP     HRDCNT(R2),DT.HM
(9)  000144' 103412                                                        BLO     50006$
(6)  000146' 032760  040000  000056                                        BIT     #BIT14,DT.SWR(R0
(9)  000154' 001006                                                        BNE     50006$
651  000156'                                CALL DRPMOD IN <R0,R2>
(3)  000156' 010546                                                        MOV     R5,-(SP)
(5)  000160' 010245                                                        MOV     R2,-(R5)
(4)  000162' 010045                                                        MOV     R0,-(R5)
(3)  000164' 004767  000000G                                              JSR     PC,DRPMOD
(3)  000170' 012605                                                        MOV     (SP)+,R5
652  000172'                            ENDIF
(4)  000172'                                                          50006$:
653  000172'                        ENDIF
(4)  000172'                                                          50005$:
654  000172'                    ENDIF
(4)  000172'                                                          50003$:
655
656
657                             ;+
658                             ; RESTORE REGS AND RETURN
659                             ;-
660
```

```
   661  000172'                          CALL RESREG                    JSR     PC,RESREG
   (3)  000172' 004767  000000G
   662
   663  000176'                          ENDRTN                      50000$:
   (3)  000176'                                                     50001$:
   (3)  000176'                                                         RTS     PC
   (2)  000176' 000207
   664
   665          000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTHER= ****** G
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DTA    = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRHARD  000000RG
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
```

```
SYSERR= 000100    WBSTAT= 000040    $F$FAL= 000405    $LOCTA= 177777    $$ERFL= 000000
TMPIO = 000002    WBUFEA= 000136    $F$GOO= 000400    $LSTIN= 000001    $$FLAG= 000001
TQOVF = 000002    WBUFPA= 000134    $F$IF = 000110    $LSTTA= 000001    $$FROM= 000000
UIPAR0= 177640    WBUFRQ= 000140    $F$INC= 000210    $NESTL= 177777    $$LOC = 000154R
UIPAR1= 177642    WBUFSZ= 000142    $F$LOO= 000200    $NSK0 = 000300    $$LOCN= 000000
UIPAR2= 177644    WDFR  = 000116    $F$NAM= 000160    $NSK1 = 000110    $$REG = 177777
UIPAR3= 177646    WDTO  = 000114    $F$NO = 000403    $NSK2 = 000110    $$RETU= 000000
UIPAR4= 177650    WTINRE= 000352    $F$OR = 000320    $NSK3 = 000110    $$RTN1= 050000
UIPAR5= 177652    WTWHMI= 000222    $F$RTI= 000350    $SAVLE= 177777    $$RTN2= 050001
UIPAR6= 177654    XFLAG = 000005    $F$RTN= 000300    $TAGLE= 177777    $$SRC = 000000
UIPAR7= 177656    XOFF  = 000023    $F$SEL= 000140    $TAGNU= 050007    $$TGSV= 000000
UIPDR0= 177600    XON   = 000021    $F$THE= 000330    $TEMP = 000300    $$TGS1= 000000
UIPDR1= 177602    $BGNLE= 177777    $F$TRU= 000404    $TSK0 = 050003    $$TGS2= 000000
UIPDR2= 177604    $ERFLG= 000400    $F$UNT= 000130    $TSK1 = 050005    $$TO  = 000000
UIPDR3= 177606    $F$AND= 000310    $F$WHI= 000120    $TSK2 = 050006    $$$TAG= 050000
UIPDR4= 177610    $F$BAD= 000401    $F$YES= 000402    $$ARGC= 000002    .     = 000200R
UIPDR5= 177612    $F$BLA= 000170    $IFLEV= 177777    $$BYTE= 000403
UIPDR6= 177614    $F$CAS= 000150    $ISK0 = 000001    $$CASE= 000000
UIPDR7= 177616    $F$DEC= 000220    $ISK1 = 000001    $$DST = 000000
WASADR= 000104    $F$DO = 000340    $ISK2 = 000001    $$ELOC= 000402

. ABS.  000000     000
        000200     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRHARD,DSKZ:PRHARD=SPMAC/ML,EQUATE,PRHARD
 RUN-TIME: 13 3 .4 SECONDS
 RUN-TIME RATIO: 42/17=2.3
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PRMSG - PROCESS MSG
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRMSG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       PROCESSES MSG CALLS FROM OPTION MODULES BY LOADING
                        ;       THE TYPE-QUEUE WITH THE SPECIFIED MESSAGE
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       ENQTQ - TYPE-QUEUE ENQUEUER
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRMSG IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON REFERENCES AND DEFINITIONS

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PRMSG                   ;MODULE ENTRY POINT
                                ;
                                ;***************
                                ;
                                ;GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQTQ                   ;TYPE-QUEUE ENQUEUER
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;***************
                                ;
```

```
                                .SBTTL PRMSG ROUTINE

                                ROUTINE PRMSG <TABL>
000000'                                                         PRMSG:
(2)  000000'

                                ;+
                                ;SAVE REGISTERS
                                ;-

                                CALL SAVREG
000000'                                                         JSR     PC,SAVREG
(3)  000000' 004767  000000G

                                ;+
                                ;SET R0 TO START OF DTABLE
                                ;-

                                LET R0 := TABL(R5)
000004'                                                         MOV     TABL(R5),R0
(4)  000004' 016500  000000

                                ;+
                                ;GET PC+2 OF THE MSG TRAP CALL FROM THE DATA TABLE
                                ;-

                                LET R1 := DT.PC(R0)
000010'                                                         MOV     DT.PC(R0),R1
(4)  000010' 016001  000002

                                ;+
                                ;GET THE OPTION MODULE HEADER ADDRESS AND MESSAGE ADDRESS.
                                ;-

                                LET R2 := (R1)+
000014'                                                         MOV     (R1)+,R2
(4)  000014' 012102
                                LET R3 := (R1)+
000016'                                                         MOV     (R1)+,R3
(4)  000016' 012103

                                ;+
                                ;ENQUEUE THE MESSAGE, USING THE STANDARD MESSAGE CODE
                                ;-

                                CALL ENQTQ IN <R0,#MSGSTD,R3,R2,R1>
000020'                                                         MOV     R5,-(SP)
(3)  000020' 010546                                             MOV     R1,-(R5)
(8)  000022' 010145                                             MOV     R2,-(R5)
(7)  000024' 010245                                             MOV     R3,-(R5)
(6)  000026' 010345                                             MOV     #MSGSTD,-(R5)
(5)  000030' 012745  000000                                     MOV     R0,-(R5)
(4)  000034' 010045                                             JSR     PC,ENQTQ
(3)  000036' 004767  000000G                                    MOV     (SP)+,R5
(3)  000042' 012605

                                ;+
                                ;RESTORE REGISTERS
```

```
  621                                   ;-
  622
  623  000044'                          CALL RESREG
  (3)  000044' 004767  000000G                                    JSR     PC,RESREG
  624
  625
  626                                   ;+
  627                                   ;RETURN
  628                                   ;-
  629
  630  000050'                          ENDRTN
  (3)  000050'                                                 50000$:
  (3)  000050'                                                 50001$:
  (2)  000050' 000207                                             RTS     PC
  631          000001                   .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PRMSG    000000RG
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TABL   = 000000
TMPIO  = 000002
```

```
TQOVF = 000002          WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
UIPAR0= 177640          WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000052R

. ABS.  000000    000
        000052    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRMSG,DSKZ:PRMSG=SPMAC/ML,EQUATE,PRMSG
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 26/13=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PRMSGN - PROCESS MSGN
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRMSGN
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       PROCESSES MSGN CALLS FROM OPTION MODULES BY LOADING
                        ;       THE TYPE-QUEUE WITH, FIRST, AN ENTRY FOR THE HEADER
                        ;       MESSAGE AND, THEN, AN ENTRY FOR EACH ADDRESS IN THE
                        ;       MESSAGE ADDRESS TABLE
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       PRMSGS - PROCESS MSGS ROUTINE
                        ;       ENQTQ - TYPE-QUEUE ENQUEUER
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRMSGN IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES


                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PRMSGN                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQTQ                   ;TYPE-QUEUE ENQUEUER
                                .GLOBL  PRMSGS                  ;PROCESS MSGS ROUTINE
                                ;
                                ;****************
                                ;
```

```
                        .SBTTL PRMSGN ROUTINE

                        ROUTINE PRMSGN <TABL>
000000'                                                 PRMSGN:
(2)  000000'

                        ;+
                        ;SAVE REGISTERS
                        ;-

                        PUSH R0,R1
000000'                                                         MOV     R0,-(SP)
(2)  000000' 010046                                             MOV     R1,-(SP)
(3)  000002' 010146

                        ;+
                        ;SET R0 TO START OF DATA TABLE
                        ;-

                        LET R0 := TABL(R5)
000004'                                                         MOV     TABL(R5),R0
(4)  000004' 016500  000000

                        ;+
                        ;ENQUEUE THE SKELETAL HEADER MESSAGE, SETTING THE MESSAGE ADDRESS
                        ;PARAMETER TO THE TRAP ADDRESS IN THE OPTION MODULE AND
                        ;THE RETURN ADDRESS PARAMETER TO #0.
                        ;(THE TRAP ADDRESS IS NEEDED FOR THE HEADER MESSAGE.)
                        ;-

                        LET R1 := DT.PC(R0) - #2
000010'                                                         MOV     DT.PC(R0),R1
(4)  000010' 016001  000002                                     SUB     #2,R1
(6)  000014' 162701  000002
                        CALL ENQTQ IN <R0,#MSGHDR,R1,2(R1),#0>
000020'                                                         MOV     R5,-(SP)
(3)  000020' 010546                                             MOV     #0,-(R5)
(8)  000022' 012745  000000                                     MOV     2(R1),-(R5)
(7)  000026' 016145  000002                                     MOV     R1,-(R5)
(6)  000032' 010145                                             MOV     #MSGHDR,-(R5)
(5)  000034' 012745  000004                                     MOV     R0,-(R5)
(4)  000040' 010045                                             JSR     PC,ENQTQ
(3)  000042' 004767  000000G                                    MOV     (SP)+,R5
(3)  000046' 012605

                        ;+
                        ;ENQUEUE THE MESSAGES LISTED IN THE MESSAGE ADDRESS TABLE
                        ;BY CALLING "PROCESS MSGS".
                        ;-

                        CALL PRMSGS IN <R0>
000050'                                                         MOV     R5,-(SP)
(3)  000050' 010546                                             MOV     R0,-(R5)
(4)  000052' 010045                                             JSR     PC,PRMSGS
(3)  000054' 004767  000000G                                    MOV     (SP)+,R5
(3)  000060' 012605

                        ;+
```

```
                                ;RESTORE REGISTERS
                                ;-

000062'                         POP R1,R0
(2)  000062' 012601                                         MOV     (SP)+,R1
(3)  000064' 012600                                         MOV     (SP)+,R0


                                ;+
                                ;RETURN
                                ;-

000066'                         ENDRTN
(3)  000066'                                            50000$:
(3)  000066'                                            50001$:
(2)  000066' 000207                                             RTS     PC

         000001                 .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NDAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PRMSGN  000000RG
PRMSGS= ****** G
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TABL   = 000000
TMPIO  = 000002
TQOVF  = 000002
```

```
UIPAR0= 177640          WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000001
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000070R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000

. ABS.  000000     000
        000070     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRMSGN,DSKZ:PRMSGN=SPMAC/ML,EQUATE,PRMSGN
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 26/13=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PRMSGS - PROCESS MSGS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRMSGS
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       PROCESSES MSGS CALLS FROM OPTION MODULES BY LOADING
                        ;       THE TYPE-QUEUE WITH AN ENTRY FOR EACH ADDRESS
                        ;       IN THE MESSAGE ADDRESS TABLE
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       ENQTQ - TYPE-QUEUE ENQUEUER
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRMSGS IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTIN=1
                        000001  $LSTTAG=1


                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  PRMSGS                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  ENQTQ                   ;TYPE-QUEUE ENQUUEUER
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL PRMSGS ROUTINE

                                ROUTINE PRMSGS <TABL>
(2)  000000'                                                        PRMSGS:

                                ;+
                                ;SAVE REGISTERS
                                ;-

                                CALL SAVREG
(3)  000000' 004767  000000G                                            JSR     PC,SAVREG


                                ;+
                                ;SET R0 TO START OF DATA TABLE
                                ;-

                                LET R0 := TABL(R5)
(4)  000004' 016500  000000                                             MOV     TABL(R5),R0


                                ;+
                                ;GET PC+2 OF THE MSGS TRAP CALL
                                ;-

                                LET R1 := DT.PC(R0)
(4)  000010' 016001  000002                                             MOV     DT.PC(R0),R1


                                ;+
                                ;GET THE OPTIONS MODULE'S HEADER ADDRESS, THE MESSAGE ADDRESS TABLE,
                                ;AND THE RETURN ADDRESS.  (THE MESSAGE ADDRESS TABLE ADDRESS
                                ;IS PLACED IN R2).
                                ;-

                                LET R3 := (R1)+
(4)  000014' 012103                                                     MOV     (R1)+,R3
                                LET R2 := (R1)+
(4)  000016' 012102                                                     MOV     (R1)+,R2


                                ;+
                                ;BUILD AND ENQUEUE A TYPE-QUEUE ENTRY FOR EACH ADDRESS IN THE MESSAGE
                                ;ADDRESS TABLE.  THE RETURN ADDRESS FOR EACH ENTRY EXCEPT THE LAST
                                ;ONE WILL BE 0.  THE LAST ENTRY WILL CONTAIN THE ACTUAL RETURN ADDRESS
                                ;FOR THE OPTION MODULES.
                                ;-

                                WHILE (R2) NE #-1 DO
(4)  000020'                                                        50002$:
(6)  000020' 021227  177777                                             CMP     (R2),#-1
(9)  000024' 001422                                                     BEQ     50003$
                                        IF 2(R2) EQ #-1 THEN
(6)  000026' 026227  000002  177777                                     CMP     2(R2),#-1
(9)  000034' 001002                                                     BNE     50004$
                                            LET R4 := R1
```

```
(4)  000036' 010104                          MOV     R1,R4
626  000040'                         ELSE
(4)  000040' 000401                          BR      50005$
(3)  000042'                                 50004$:
627  000042'                                 CLR     R4
(4)  000042' 005004
628  000044'                         ENDIF
(4)  000044'                                 50005$:
629  000044'                         CALL ENQTQ IN <R0,#MSGSTD,(R2)+,R3,R4>
(3)  000044' 010546                          MOV     R5,-(SP)
(8)  000046' 010445                          MOV     R4,-(R5)
(7)  000050' 010345                          MOV     R3,-(R5)
(6)  000052' 012245                          MOV     (R2)+,-(R5)
(5)  000054' 012745  000000                  MOV     #MSGSTD,-(R5)
(4)  000060' 010045                          MOV     R0,-(R5)
(3)  000062' 004767  000000G                 JSR     PC,ENQTQ
(3)  000066' 012605                          MOV     (SP)+,R5
630  000070'                         ENDDO
(4)  000070' 000753                          BR      50002$
(3)  000072'                                 50003$:
631
632
633
634                                  ;+
635                                  ;RESTORE REGISTERS
636                                  ;-
637
638  000072'                         CALL RESREG
(3)  000072' 004767  000000G                 JSR     PC,RESREG
639
640
641                                  ;+
642                                  ;RETURN
643                                  ;-
644
645  000076'                         ENDRTN
(3)  000076'                                 50000$:
(3)  000076'                                 50001$:
(2)  000076' 000207                          RTS     PC
646
647          000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ  = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PRMSGS  000000RG
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TABL   = 000000
TMPIO  = 000002
```

```
TQOVF = 000002          WBUFEA= 000136          $F$FAL= 000405          $LSTIN= 000001          $$ERFL= 000000
UIPAR0= 177640          WBUFPA= 000134          $F$GOO= 000400          $LSTTA= 000001          $$FLAG= 000001
UIPAR1= 177642          WBUFRQ= 000140          $F$IF = 000110          $NESTL= 177777          $$FROM= 000000
UIPAR2= 177644          WBUFSZ= 000142          $F$INC= 000210          $NSK0 = 000300          $$LOC = 000034R
UIPAR3= 177646          WDFR  = 000116          $F$LOO= 000200          $NSK1 = 000120          $$LOCN= 000000
UIPAR4= 177650          WDTO  = 000114          $F$NAM= 000160          $NSK2 = 000110          $$REG = 177777
UIPAR5= 177652          WTINRE= 000352          $F$NO = 000403          $SAVLE= 177777          $$RETU= 000000
UIPAR6= 177654          WTWHMI= 000222          $F$OR = 000320          $SSK0 = 050003          $$RTN1= 050000
UIPAR7= 177656          XFLAG = 000005          $F$RTI= 000350          $TAGLE= 177777          $$RTN2= 050001
UIPDR0= 177600          XOFF  = 000023          $F$RTN= 000300          $TAGNU= 050006          $$SRC = 000000
UIPDR1= 177602          XON   = 000021          $F$SEL= 000140          $TEMP = 000300          $$TGSV= 000000
UIPDR2= 177604          $BGNLE= 177777          $F$THE= 000330          $TSK0 = 050002          $$TGS1= 000000
UIPDR3= 177606          $ERFLG= 000400          $F$TRU= 000404          $TSK1 = 050003          $$TGS2= 000000
UIPDR4= 177610          $F$AND= 000310          $F$UNT= 000130          $TSK2 = 050005          $$TO  = 000000
UIPDR5= 177612          $F$BAD= 000401          $F$WHI= 000120          $$ARGC= 000002          $$$TAG= 050000
UIPDR6= 177614          $F$BLA= 000170          $F$YES= 000402          $$BYTE= 000403          .     = 000100R
UIPDR7= 177616          $F$CAS= 000150          $IFLEV= 177777          $$CASE= 000000
WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$DST = 000000
WBSTAT= 000040          $F$DO = 000340          $LOCTA= 177777          $$ELOC= 000402

. ABS.  000000      000
        000100      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRMSGS,DSKZ:PRMSGS=SPMAC/ML,EQUATE,PRMSGS
 RUN-TIME: 12 2 .3 SECONDS
 RUN-TIME RATIO: 29/14=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  PROTOA PROCESS THE 'OTOA' MACRO IN OPTION MODULES
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PROTOA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'OTOA' TRAP ISSUED BY OPTION
                        ;       MODULES. IT WILL RETRIEVE THE NUMBER TO BE CONVERTED TO
                        ;       OCTAL ASCII FROM DT.PC+2 AND THE TABLE ADDRESS AT WHICH
                        ;       TO STORE THE OCTAL ASCII FROM DT.PC +4. IT CALLS THE
                        ;       BINARY TO OCTAL ASCII CONVERSION ROUTINE. THE RETURN PC
                        ;       IS UPDATED SINCE THIS IS DIRECT SERVICE.
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       BOA16 - BINARY TO OCTAL ASCII CONVERSION ROUTINE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PROTOA IN <DTADR>
                        ;       WHERE DTADR = ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR PROTOA

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

                        000001  $LSTIN=1
                        000001  $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  PROTOA          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  BOA16           ;BINARY TO OCTAL ASCII CONVERSION ROUTINE

```

```
                                .SBTTL  PROTOA ROUTINE

000000'                         ROUTINE PROTOA <DTADR>
000000'                                                                 PROTOA:

                                ;+
                                ;INITIALIZE AND SAVE DATA TABLE ADDRESS
                                ;-


000000'                         PUSH R0,R1
000000' 010046                                                                  MOV     R0,-(SP)
000002' 010146                                                                  MOV     R1,-(SP)
000004'                         LET R0 := DTADR(R5)
000004' 016500  000000                                                          MOV     DTADR(R5),R0

                                ;+
                                ; GET DT.PC AND CALL BOA16
                                ;-

000010'                         LET R1 := DT.PC(R0)
000010' 016001  000002                                                          MOV     DT.PC(R0),R1
000014'                         CALL BOA16 IN <@2(R1),4(R1)>
000014' 010546                                                                  MOV     R5,-(SP)
000016' 016145  000004                                                          MOV     4(R1),-(R5)
000022' 017145  000002                                                          MOV     @2(R1),-(R5)
000026' 004767  000000G                                                         JSR     PC,BOA16
000032' 012605                                                                  MOV     (SP)+,R5

                                ;+
                                ; UPDATE RETURN PC
                                ;-

000034'                         LET DT.PC(R0) := DT.PC(R0) + #6
000034' 062760  000006  000002                                                  ADD     #6,DT.PC(R0)

                                ;+
                                ; CLEAN UP AND GOODBYE
                                ;-

000042'                         POP R1,R0
000042' 012601                                                                  MOV     (SP)+,R1
000044' 012600                                                                  MOV     (SP)+,R0

000046'                         ENDRTN
000046'                                                                 50000$:
000046'                                                                 50001$:
000046' 000207                                                                  RTS     PC

        000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BOA16  = ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR  = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PROTOA  000000RG
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
```

```
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSKO = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000002
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000050R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000

. ABS.  000000     000
        000050     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PROTOA,DSKZ:PROTOA=SPMAC/ML,EQUATE,PROTOA
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 26/12=2.0
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE PRRAND (PROCESS 'RAND' TRAP CALL)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRRAND
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE SERVICES A 'RAND' TRAP CALL
                        ;       AND RETURNS A RANDOM NUMBER TO A LOCATION IN THE
                        ;       MODULE'S HEADER.
                        ;
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1.DT.PC
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       RANDOM NUMBER
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       RANDOM
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRRAND IN <AA>
                        ;
                        ;           WHERE AA = DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       ------  ----------  ----------  ----------------
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTIN=1
                        000001  $LSTTAG=1


                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL PRRAND           ;MODULE'S ENTRY POINT
                                ;
                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL RANDOM           ;ROUTINE 'RANDOM'
```

```
                                .SBTTL PRRAND (CODE)

000000'                         ROUTINE PRRAND <AA>
(2) 000000'                                                             PRRAND:

                                ;+
                                ; SAVE REGISTERS
                                ;-

000000'                         PUSH R0,R1
(2) 000000' 010046                                                              MOV     R0,-(SP)
(3) 000002' 010146                                                              MOV     R1,-(SP)

                                ;+
                                ; GET DTABLE ADDRESS ,GET HEADER ADDRESS AND GET RANDOM NUMBER ADDRESS
                                ; IN HEADER
                                ;-

000004'                         LET R0 := AA(R5)
(4) 000004' 016500  000000                                                      MOV     AA(R5),R0
000010'                         LET R1 := DT.PC(R0)
(4) 000010' 016001  000002                                                      MOV     DT.PC(R0),R1
000014'                         LET DT.PC(R0) := R1 + #2
(4) 000014' 010160  000002                                                      MOV     R1,DT.PC(R0)
(6) 000020' 062760  000002  000002                                              ADD     #2,DT.PC(R0)
000026'                         LET R1 := (R1)
(4) 000026' 011101                                                              MOV     (R1),R1

                                ;+
                                ; GET RANDOM NUMBER
                                ;-

000030'                         CALL RANDOM OUT <RANNUM(R1)>
(4) 000030' 162705  000002                                                      SUB     #1*2,R5
(3) 000034' 004767  000000G                                                     JSR     PC,RANDOM
(4) 000040' 012561  000054                                                      MOV     (R5)+,RANNUM(R1)

                                ;+
                                ; RESTORE REGISTERS AND RETURN
                                ;-

000044'                         POP R1,R0
(2) 000044' 012601                                                              MOV     (SP)+,R1
(3) 000046' 012600                                                              MOV     (SP)+,R0
000050'                         ENDRTN
(3) 000050'                                                             50000$:
(3) 000050'                                                             50001$:
(2) 000050' 000207                                                              RTS     PC

            000001              .END
```

```
AA     = 000000
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PRRAND  000000RG
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANDOM= ****** G
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
```

```
UIPAR1= 177642          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR2= 177644          WBUFPA= 000134          $F$DO = 000340          $IFLEV= 177777          $$FROM= 000001
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR  = 000116          $F$IF = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG = 000005          $F$NO = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF  = 000023          $F$OR = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TC  = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000052R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000

. ABS.  000000      000
        000052      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRRAND,DSKZ:PRRAND=SPMAC/ML,EQUATE,PRRAND
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 27/12=2.1
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  PRSOFT (PROCESS SOFT ERRORS)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRSOFT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE IS CALLED AS A RESULT OF A 'SOFERR' TRAP EXECUTED
                        ;       BY AN OPTION MODULE.
                        ;
                        ;       FIRST THE FOLLOWING IS DONE:
                        ;
                        ;           1. THE MODULE'S SOFT ERROR COUNT IS INCREMENTED.
                        ;           2. ERROR MESSAGE IS QUEUED UP.
                        ;
                        ;       IF THE ENVIRONMENT IS APT, THEN AN APT ERROR HANDLING ROUTINE
                        ;       IS CALLED AND ON RETURN, THE CONTROL GOES BACK TO THE CALLER.
                        ;
                        ;       IF NOT UNDER APT ENVIRONMENT, THEN
                        ;
                        ;           1. IF BIT15 OF THE SOFTWARE SWITCH REGISTER IS SET, THE
                        ;              MODULE IS DROPPED WITH 'MODULE DROPPED' MESSAGE PRINTED,
                        ;              EVEN ON THE FIRST ERROR.
                        ;           2. IF THE ERROR COUNT IS EQUAL TO OR GREATER THAN THE
                        ;              LIMIT, THEN BIT14 OF THE SOFTWARE SWITCH REGISTER IS
                        ;              CHECKED. IF BIT14 IS SET, THE MODULE IS NOT DROPPED.
                        ;           3. CONTROL RETURNS TO THE CALLER.
                        ;
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. DATA TABLE OFFSETS
                        ;       2. MODULE HEADER OFFSETS
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       1. DATA TABLE
                        ;       2. MODULE HEADER
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. DRPMOD               ;DROP A MODULE ROUTINE
                        ;       2. ENQTQ                ;ENQUE AN ERROR MSG ROUTINE
                        ;       3. APTSER               ;APT SOFT ERROR ROUTINE
                        ;       4. SAVREG
                        ;       5. RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       UNDER APT ENVIRONMENT IT CAN SHUT DOWN THE EXCERCISER.
```

```
                            ;
                            ; CALLING SEQUENCE:
                            ;       CALL PRSOFT IN <DT>
                            ;       WHERE DT = DATA TABLE ADDRESS
                            ;
                            ; VERSION:
                            ;       0.0
                            ;
                            ;       EDIT            DATE            BY              REASON
                            ;--
```

```
                                .SBTTL PRSOFT (COMMON DEFINITIONS & REFERENCES)

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;*****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL PRSOFT                   ;MODULE ENTRY POINT

                                ;*****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL ENQTQ                    ;ENQUE A MESSAGE INTO THE TYPE QUEUE
                                .GLOBL APTSER                   ;APT SOFT ERROR HANDLER
                                .GLOBL DRPMOD                   ;'DROP THE MODULE' ROUTINE
                                .GLOBL SAVREG
                                .GLOBL RESREG

                                ;*****************
```

```
                                .SBTTL PRSOFT (CODE)

                                ROUTINE PRSOFT <DTA>
000000'                                                                 PRSOFT:
(2)  000000'

                                ;+
                                ; SAVE REGS, GET DTABLE ADDRESS AND MODULE
                                ; HEADER ADDRESS.
                                ;-

                                CALL SAVREG
000000'                                                                 JSR     PC,SAVREG
(3)  000000' 004767  000000G
                                LET R0 := DTA(R5)
000004'                                                                 MOV     DTA(R5),R0
(4)  000004' 016500  000000
                                LET R1 := DT.PC(R0)
000010'                                                                 MOV     DT.PC(R0),R1
(4)  000010' 016001  000002
                                LET R2 := (R1)
000014'                                                                 MOV     (R1),R2
(4)  000014' 011102

                                ;+
                                ; POINT R3 TO TRAP ADDRESS
                                ;-

                                LET R3 := R1 - #2
000016'                                                                 MOV     R1,R3
(4)  000016' 010103                                                     SUB     #2,R3
(6)  000020' 162703  000002

                                ;+
                                ; POINT R1 TO RETURN ADDRESS
                                ;-

                                LET R1 := R1 + #4
000024'                                                                 ADD     #4,R1
(6)  000024' 062701  000004

                                ;+
                                ; INCREMENT MODULE'S SOFT ERROR COUNT AND ENQUE THE ERROR
                                ; MESSAGE.
                                ;-
                                LET SOFCNT(R2) := SOFCNT(R2) + #1
000030'                                                                 INC     SOFCNT(R2)
(6)  000030' 005262  000042
                                LET SOFPAS(R2) := SOFPAS(R2) + #1
000034'                                                                 INC     SOFPAS(R2)
(6)  000034' 005262  000046
                                CALL ENQTQ IN <R0,#MSGSFT,R3,R2,R1>
000040'                                                                 MOV     R5,-(SP)
(3)  000040' 010546                                                     MOV     R1,-(R5)
(8)  000042' 010145                                                     MOV     R2,-(R5)
(7)  000044' 010245                                                     MOV     R3,-(R5)
(6)  000046' 010345                                                     MOV     #MSGSFT,-(R5)
(5)  000050' 012745  000006                                             MOV     R0,-(R5)
(4)  000054' 010045                                                     JSR     PC,ENQTQ
(3)  000056' 004767  000000G                                           MOV     (SP)+,R5
(3)  000062' 012605

                                ;+
                                ; IF APT ENVIRONMENT, CALL APT ERROR HANDLING ROUTINE
                                ;-
```

```
                        IF #APTPRE SETIN DT.CF0(R0) THEN
(6)  000064' 032760  000200  000014                                    BIT     #APTPRE,DT.CF0(R
(9)  000072' 001406                                                     BEQ     50002$
                                CALL APTSER IN <R0>
(3)  000074' 010546                                                     MOV     R5,-(SP)
(4)  000076' 010045                                                     MOV     R0,-(R5)
(3)  000100' 004767  000000G                                           JSR     PC,APTSER
(3)  000104' 012605                                                     MOV     (SP)+,R5
                        ELSE
(4)  000106' 000431                                                     BR      50003$
(3)  000110'                                                    50002$:

                        ;+
                        ; THE ENVIRONMENT IS NOT APT.
                        ; CHECK SWITCH REGISTER IF THE MODULE IS TO BE DROPPED
                        ; DUE TO THIS ERROR.
                        ;-

                                IF #BIT15 SETIN DT.SWR(R0) THEN
(6)  000110' 032760  100000  000056                                    BIT     #BIT15,DT.SWR(R0
(9)  000116' 001407                                                     BEQ     50004$
                                    CALL DRPMOD IN <R0,R2>
(3)  000120' 010546                                                     MOV     R5,-(SP)
(5)  000122' 010245                                                     MOV     R2,-(R5)
(4)  000124' 010045                                                     MOV     R0,-(R5)
(3)  000126' 004767  000000G                                           JSR     PC,DRPMOD
(3)  000132' 012605                                                     MOV     (SP)+,R5
                                ELSE
(4)  000134' 000416                                                     BR      50005$
(3)  000136'                                                    50004$:

                        ;+
                        ; IF THE COUNT IS EQUAL TO OR GREATER THAN THE LIMIT,
                        ; CHECK SWITCH REGISTER IF MODULE IS TO BE DROPPED.
                        ;-

                                    IF SOFCNT(R2) HI DT.SMX(R0) AND #BIT14 NOTSETIN DT.SWR(R0)  THEN
(6)  000136' 026260  000042  000106                                    CMP     SOFCNT(R2),DT.SM
(9)  000144' 101412                                                     BLOS    50006$
(6)  000146' 032760  040000  000056                                    BIT     #BIT14,DT.SWR(R0
(9)  000154' 001006                                                     BNE     50006$
                                        CALL DRPMOD IN <R0,R2>
(3)  000156' 010546                                                     MOV     R5,-(SP)
(5)  000160' 010245                                                     MOV     R2,-(R5)
(4)  000162' 010045                                                     MOV     R0,-(R5)
(3)  000164' 004767  000000G                                           JSR     PC,DRPMOD
(3)  000170' 012605                                                     MOV     (SP)+,R5
                                    ENDIF
(4)  000172'                                                    50006$:
                                ENDIF
(4)  000172'                                                    50005$:

                        ENDIF
(4)  000172'                                                    50003$:

                        ;+
```

```
                                    ; RESTORE REGS AND RETURN
                                    ;-

000172'                                 CALL RESREG
(3)  000172' 004767  000000G                                        JSR     PC,RESREG

000176'                             ENDRTN
(3)  000176'                                                    50000$:
(3)  000176'                                                    50001$:
(2)  000176' 000207                                                 RTS     PC

          000001                    .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
APTSER= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DRPMOD= ****** G
DSEVNT= 000014
DTA   = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRSOFT  000000RG
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
```

```
SYSERR= 000100          WBSTAT= 000040          $F$FAL= 000405          $LOCTA= 177777          $$ERFL= 000000
TMPIO = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000001
TQOVF = 000002          WBUFPA= 000134          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR0= 177640          WBUFRQ= 000140          $F$INC= 000210          $NESTL= 177777          $$LOC = 000154R
UIPAR1= 177642          WBUFSZ= 000142          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR2= 177644          WDFR  = 000116          $F$NAM= 000160          $NSK1 = 000110          $$REG = 177777
UIPAR3= 177646          WDTO  = 000114          $F$NO = 000403          $NSK2 = 000110          $$RETU= 000000
UIPAR4= 177650          WTINRE= 000352          $F$OR = 000320          $NSK3 = 000110          $$RTN1= 050000
UIPAR5= 177652          WTWHMI= 000222          $F$RTI= 000350          $SAVLE= 177777          $$RTN2= 050001
UIPAR6= 177654          XFLAG = 000005          $F$RTN= 000300          $TAGLE= 177777          $$SRC = 000000
UIPAR7= 177656          XOFF  = 000023          $F$SEL= 000140          $TAGNU= 050007          $$TGSV= 000000
UIPDR0= 177600          XON   = 000021          $F$THE= 000330          $TEMP = 000300          $$TGS1= 000000
UIPDR1= 177602          $BGNLE= 177777          $F$TRU= 000404          $TSK0 = 050003          $$TGS2= 000000
UIPDR2= 177604          $ERFLG= 000400          $F$UNT= 000130          $TSK1 = 050005          $$TO  = 000000
UIPDR3= 177606          $F$AND= 000310          $F$WHI= 000120          $TSK2 = 050006          $$$TAG= 050000
UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$ARGC= 000002          .     = 000200R
UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$BYTE= 000403
UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$CASE= 000000
UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$DST = 000000
WASADR= 000104          $F$DO = 000340          $ISK2 = 000001          $$ELOC= 000402

. ABS.  000000     000
        000200     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:PRSOFT,DSKZ:PRSOFT=SPMAC/ML,EQUATE,PRSOFT
 RUN-TIME: 13 3 .3 SECONDS
 RUN-TIME RATIO: 32/17=1.8
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE RANDOM (PSEUDO-RANDOM NUMBER GENERATOR)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       RANDOM
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE GENERATES AND RETURNS A 16-BIT RANDOM NUMBER
                        ;       TO THE CALLER.
                        ;
                        ;       THE ALGORITHM USED CONSISTS OF ROTATING TWO BASE WORDS
                        ;       ONE POSITION TO THE RIGHT AND THEN PERFORMING AN
                        ;       EXCLUSIVE-OR OPERATION ON CERTAIN BITS IN WORD-1.  TO
                        ;       INSURE THOROUGH RANDOMNESS OF THE NUMBER, THIS OPERATION
                        ;       IS REPEATED 16 TIMES BEFORE RETURNING THE RANDOM NUMBER
                        ;       FROM THE SECOND WORD.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       A RANDOM NUMBER
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL RANDOM OUT <XXX>
                        ;
                        ;           WHERE XXX = RANDOM NUMBER RETURNED.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       ------  ----------  ----------  ----------------
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
          000001                $LSTIN=1
          000001                $LSTTAG=1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL RANDOM           ;MODULE'S ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 022333                  WORK:   22333           ;WORD-1
000002' 123456                  RAND:   123456          ;WORD-2
                                                        ;THE CHOICE OF VALUES IS ARBITRARY

```

```
                                .SBTTL RANDOM (CODE)

                                ROUTINE RANDOM <RSLT>

RANDOM:

                                ;+
                                ; SAVE REGISTERS
                                ; GET THE TWO BASE WORDS INTO R0 AND R1.
                                ;-

                                CALL SAVREG

                                        JSR     PC,SAVREG

                                LET R0 := WORK

                                        MOV     WORK,R0

                                LET R1 := RAND

                                        MOV     RAND,R1


                                ;+
                                ; SET UP LOOP FOR 16 ITERATIONS, ROTATE THE TWO
                                ; WORDS ONE POSITION TO THE RIGHT, AND THEN DO
                                ; THE EXCLUSIVE-OR OPERATION ON BIT12 AND BIT10
                                ; OF WORD-1, BASED ON "CARRY" CONDITION.
                                ;-

                                DECR R2 FROM #15. TO #0 BY #1

                                        MOV     #15.,R2
                                        BR      50002$
50003$:
                                        DEC     R2
50002$:
                                        TST     R2
                                        BLT     50004$

                                    LET R0 := R0 ROTATE -#1

                                        ROR     R0

                                    LET R1 := R1 ROTATE -#1

                                        ROR     R1

                                    IF.NO.CARRY THEN

                                        BCS     50005$

                                        LET R0 := R0 CLR.BY #BIT12

                                        BIC     #BIT12,R0

                                        IF #BIT10 SETIN R0 THEN

                                        BIT     #BIT10,R0
                                        BEQ     50006$

                                            LET R0 := R0 SET.BY #BIT12

                                        BIS     #BIT12,R0

                                        ENDIF

50006$:

                                    ELSE

                                        BR      50007$
50005$:
                                        LET R0 := R0 SET.BY #BIT12

                                        BIS     #BIT12,R0

                                        IF #BIT10 SETIN R0 THEN

                                        BIT     #BIT10,R0
                                        BEQ     50010$

                                            LET R0 := R0 CLR.BY #BIT12
```

```
(6)  000074' 042700  010000                                                    BIC     #BIT12,R0
618  000100'                          ENDIF
(4)  000100'                                                            50010$:
619  000100'                      ENDIF
(4)  000100'                                                            50007$:
620  000100'                      ENDDEC                                       BR      50003$
(4)  000100' 000752                                                     50004$:
(3)  000102'
621
622                               ;+
623                               ; NOW THAT YOU HAVE DONE THE 16 ITERATIONS AND
624                               ; GOTTEN THE RESULT, STORE THE TWO WORDS BACK
625                               ; INTO THE BASE WORDS. DON'T FORGET TO RETURN
626                               ; THE RESULT.
627                               ;-
628
629  000102'                      LET WORK := R0                               MOV     R0,WORK
(4)  000102' 010067  177672
630  000106'                      LET RAND := R1                               MOV     R1,RAND
(4)  000106' 010167  177670
631  000112'                      LET RSLT(R5) := R1                           MOV     R1,RSLT(R5)
(4)  000112' 010165  000000
632
633                               ;+
634                               ; RESTORE REGISTERS AND RETURN - HURRAH.
635                               ;-
636
637  000116'                      CALL RESREG                                  JSR     PC,RESREG
(3)  000116' 004767  000000G
638
639  000122'                      ENDRTN                               50000$:
(3)  000122'                                                        50001$:
(3)  000122'                                                                 RTS     PC
(2)  000122' 000207
640
641          000001               .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRIO  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RAND    000002R
RANDOM  000004RG
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSLT  = 000000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
```

```
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WORK    000000R
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000220
$NSK2 = 000110
$NSK3 = 000110
$SAVLE= 177777
$SSK0 = 050003
$TAGLE= 177777
$TAGNU= 050011
$TEMP = 000300
$TSK0 = 050004
$TSK1 = 050003
$TSK2 = 050007
$TSK3 = 050010
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000072R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000124R

. ABS.  000000    000
        000124    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RANDOM,DSKZ:RANDOM=SPMAC/ML,EQUATE,RANDOM
 RUN-TIME: 13 3 .4 SECONDS
 RUN-TIME RATIO: 31/17=1.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE RELCTL - RELOCATION CONTROL
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       RELCTL
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CONTROLS THE SEQUENCE OF EVENTS THAT MUST OCCUR BEFORE,
                        ;       DURING, AND AFTER THE RELOCATION PROCESS.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CURRENT BASE ADDRESS, CONFIGURATION WORD 0,
                        ;       STATUS INDICATOR WORD 0, STATUS INDICATOR WORD 1
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       MESSAGE CODE IN MSGDEQ (MD.COD)
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       CLKOFF                  ;TURN OFF SYSTEM CLOCK
                        ;       CLKON                   ;TURN ON SYSTEM CLOCK
                        ;       PRRLOC                  ;PROCESS RELOCATION
                        ;       WBFLIM                  ;DETERMINE WRITE BUFFER LIMITS
                        ;       UNIMAP                  ;LOAD UNIBUS MAP
                        ;       WSTBUS                  ;WRITE WORST-CASE UNIBUS PATTERNS
                        ;       MSGDEQ                  ;MESSAGE DEQUEUER
                        ;       DX.INI                  ;INITIALIZE QUEUES AND KEYBOARD
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL RELCTL IN <A>
                        ;               A=DATA TABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES

                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1
            000001      $LSTIN=1
            000001      $LSTTAG=1

                        ;
                        ;****************
                        ;
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  RELCTL                  ;MODULE ENTRY POINT
                        ;
                        ;****************
                        ;
                        ; GLOBAL REFERENCES
                        ;
                        .GLOBL  CLKOFF
                        .GLOBL  CMDPRC                  ;PROCESS A KEYBOARD COMMAND
                        .GLOBL  DX.KFL                  ;KEYBOARD FLAG
                        .GLOBL  CLKON
                        .GLOBL  PRRLOC
                        .GLOBL  WBFLIM
                        .GLOBL  UNIMAP
                        .GLOBL  WSTBUS
                        .GLOBL  DX.INI
                        .GLOBL  MSGDEQ                  ;MESSAGE DEQUEUER
                        .GLOBL  MD.COD                  ;MESSAGE CODE IN MESDEQ
                        ;
                        ;****************
                        ;
                        ; GLOBAL EQUATES
                        ;
                        .GLOBL  DT.ST0                  ;STATUS INDICATOR WORD 0
                        .GLOBL  DT.ST1                  ;STATUS INDICATOR WORD 1
                        .GLOBL  DT.CF0                  ;CONFIGURATION WORD 0
                        .GLOBL  DT.ADDR                 ;CURRENT BASE ADDRESS
                        .GLOBL  RELTIME                 ;RELOCATION TIME INDICATOR
                        .GLOBL  MSGNUL                  ;NULL MESSAGE TYPE CODE
                        .GLOBL  ADDR22                  ;22-BIT ADDRESSING
                        .GLOBL  RCSR                    ;KB INPUT CSR
                        .GLOBL  IE                      ;INTERRUPT ENABLE
                        .GLOBL  CLKPRES                 ;SYSTEM CLOCK PRESENT
                        .GLOBL  MEMPAS                  ;LAST SYSTEM EOP WAS IN LOWEST MEMORY
                        .GLOBL  AUTO                    ;AUTOMATIC MODE
                        ;
                        ;******************
                        ;
```

```
                                .SBTTL RELCTL ROUTINE

                                ROUTINE RELCTL <TABL>
000000'                         RELCTL:
(2)  000000'

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                PUSH R0
000000'                                 MOV     R0,-(SP)
(2)  000000' 010046

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := TABL(R5)
000002'                                 MOV     TABL(R5),R0
(4)  000002' 016500  000000

                                ;+
                                ; SHUT OFF THE SYSTEM CLOCK IF IT EXISTS.
                                ;-

                                IF #CLKPRES SETIN DT.CF0(R0) THEN
000006'                                 BIT     #CLKPRES,DT.CF0(
(6)  000006' 032760  000001  000014     BEQ     50002$
(9)  000014' 001405
                                        CALL CLKOFF IN <R0>
000016'                                 MOV     R5,-(SP)
(3)  000016' 010546                     MOV     R0,-(R5)
(4)  000020' 010045                     JSR     PC,CLKOFF
(3)  000022' 004767  000000G            MOV     (SP)+,R5
(3)  000026' 012605
                                ENDIF
000030'                         50002$:
(4)  000030'

                                ;+
                                ; SERVICE ANY FINAL KEYBOARD COMMANDS
                                ;-
                                IF DX.KFL EQ #1 THEN
000030'                                 CMP     DX.KFL,#1
(6)  000030' 026727  000000G 000001     BNE     50003$
(9)  000036' 001007
                                        CALL CMDPRC IN  <R0>
000040'                                 MOV     R5,-(SP)
(3)  000040' 010546                     MOV     R0,-(R5)
(4)  000042' 010045                     JSR     PC,CMDPRC
(3)  000044' 004767  000000G            MOV     (SP)+,R5
(3)  000050' 012605
                                        LET DX.KFL := #0
000052'                                 CLR     DX.KFL
(4)  000052' 005067  000000G
                                ENDIF
000056'                         50003$:
(4)  000056'

                                ;+
                                ; WAIT FOR THE TYPE-QUEUE TO EMPTY
```

```
                                ;-
                                REPEAT
                                        CALL MSGDEQ IN <R0>
                                UNTIL MD.COD EQ #MSGNUL

                                ;+
                                ; RELOCATE
                                ;-

                                CALL PRRLOC IN <R0,#0>

                                ;+
                                ; TURN THE SYSTEM CLOCK BACK ON IF IT EXISTS.
                                ;-

                                IF #CLKPRES SETIN DT.CF0(R0) THEN
                                        CALL CLKON IN <R0>
                                ENDIF

                                ;+
                                ; SET THE WRITE BUFFER LIMITS AND INITIALIZE THE WRITE BUFFER POINTER
                                ;-

                                CALL WBFLIM IN <R0>

                                ;+
                                ; IF THE UNIBUS MAP EXISTS, LOAD THE MAP REGISTERS.
                                ;-
```

```
000056'                 50004$:
                                MOV     R5,-(SP)
                                MOV     R0,-(R5)
                                JSR     PC,MSGDEQ
000056' 010546                  MOV     (SP)+,R5
000060' 010045
000062' 004767  000000G
000066' 012605
000070'                         CMP     MD.COD,#MSGNUL
000070' 026727  000000G 177775  BNE     50004$
000076' 001367

000100'                         MOV     R5,-(SP)
000100' 010546                  MOV     #0,-(R5)
000102' 012745  000000          MOV     R0,-(R5)
000106' 010045                  JSR     PC,PRRLOC
000110' 004767  000000G         MOV     (SP)+,R5
000114' 012605

000116'                         BIT     #CLKPRES,DT.CF0(
000116' 032760  000001  000014  BEQ     50005$
000124' 001405
000126'                         MOV     R5,-(SP)
000126' 010546                  MOV     R0,-(R5)
000130' 010045                  JSR     PC,CLKON
000132' 004767  000000G         MOV     (SP)+,R5
000136' 012605
000140'                 50005$:
000140'

000140'                         MOV     R5,-(SP)
000140' 010546                  MOV     R0,-(R5)
000142' 010045                  JSR     PC,WBFLIM
000144' 004767  000000G         MOV     (SP)+,R5
000150' 012605
```

```
                                IF #ADDR22 SETIN DT.CF0(R0) THEN
(6)  000152' 032760  001000  000014                                        BIT     #ADDR22,DT.CF0(R
(9)  000160' 001405                                                        BEQ     50006$
                                        CALL UNIMAP IN <R0>
(3)  000162' 010546                                                        MOV     R5,-(SP)
(4)  000164' 010045                                                        MOV     R0,-(R5)
(3)  000166' 004767  000000G                                               JSR     PC,UNIMAP
(3)  000172' 012605                                                        MOV     (SP)+,R5
                                ENDIF
(4)  000174'                                                            50006$:

                                ;+
                                ; IF THIS RELOCATION WAS TO THE LOWEST PART OF MEMORY, WRITE THE
                                ; WORST-CASE UNIBUS PATTERNS INTO ALL OF FREE MEMORY,
                                ;-

                                IF DT.ADDR(R0) EQ #200 THEN
(6)  000174' 026027  000042  000200                                        CMP     DT.ADDR(R0),#200
(9)  000202' 001014                                                        BNE     50007$
                                        CALL WSTBUS IN <R0>
(3)  000204' 010546                                                        MOV     R5,-(SP)
(4)  000206' 010045                                                        MOV     R0,-(R5)
(3)  000210' 004767  000000G                                               JSR     PC,WSTBUS
(3)  000214' 012605                                                        MOV     (SP)+,R5
                                        IF #AUTO SETIN DT.CF0(R0) THEN
(6)  000216' 032760  000010  000014                                        BIT     #AUTO,DT.CF0(R0)
(9)  000224' 001403                                                        BEQ     50010$
                                                LET DT.ST1(R0) := DT.ST1(R0) SET.BY #MEMPAS
(6)  000226' 052760  040000  000012                                        BIS     #MEMPAS,DT.ST1(R
                                        ENDIF
(4)  000234'                                                            50010$:
                                ENDIF
(4)  000234'                                                            50007$:

                                ;+
                                ; INITIALIZE THE QUEUES AND KEYBOARD.
                                ;-

                                CALL DX.INI
(3)  000234' 004767  000000G                                               JSR     PC,DX.INI

                                ;+
                                ; CLEAR THE RELOCATION TIME INDICATOR IN STATUS INDICATOR WORD 0
                                ;-

                                LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RELTIME
(6)  000240' 042760  010000  000010                                        BIC     #RELTIME,DT.ST0(

                                ;+
                                ; RESTORE REGISTERS
```

```
   716                                  ;-
   717
   718  000246'                         POP R0
   (2)  000246' 012600                                          MOV     (SP)+,R0
   719
   720
   721  000250'                         ENDRTN
   (3)  000250'                                         50000$:
   (3)  000250'                                         50001$:
   (2)  000250' 000207                                          RTS     PC
   722
   723          000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000 G
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010 G
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKOFF= ****** G
CLKON = ****** G
CLKPRE= 000001 G
CMDPRC= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042 G
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014 G
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010 G
DT.ST1= 000012 G
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.INI= ****** G
DX.KFL= ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100 G
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.COD= ****** G
MED   = 076600
MEMPAS= 040000 G
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDEQ= ****** G
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775 G
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRRLOC= ****** G
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RCSR  = ****** G
RDSERV= 000101
RDWHMI= 000022
RELCTL  000000RG
RELERR= 000020
RELMOD= 020000
RELTIM= 010000 G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
```

```
SVR5   = 000074          UIPDR5= 177612          $ERFLG= 000400          $F$UNT= 000130          $$BYTE= 000403
SVR6   = 000076          UIPDR6= 177614          $F$AND= 000310          $F$WHI= 000120          $$CASE= 000000
SYSCNT= 000052           UIPDR7= 177616          $F$BAD= 000401          $F$YES= 000402          $$DST = 000000
SYSERR= 000100           UNIMAP= ****** G        $F$BLA= 000170          $IFLEV= 177777          $$ELOC= 000402
TABL   = 000000          WASADR= 000104          $F$CAS= 000150          $ISK0 = 000001          $$ERFL= 000000
TMPIO  = 000002          WBFLIM= ****** G        $F$DEC= 000220          $ISK1 = 000001          $$FLAG= 000001
TQOVF  = 000002          WBSTAT= 000040          $F$DO  = 000340          $LOCTA= 177777          $$FROM= 000000
UIPAR0= 177640           WBUFEA= 000136          $F$FAL= 000405          $LSTIN= 000001          $$LOC = 000224R
UIPAR1= 177642           WBUFPA= 000134          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
UIPAR2= 177644           WBUFRQ= 000140          $F$IF  = 000110          $NESTL= 177777          $$REG = 177777
UIPAR3= 177646           WBUFSZ= 000142          $F$INC= 000210          $NSK0 = 000300          $$RETU= 000000
UIPAR4= 177650           WDFR   = 000116         $F$LOO= 000200          $NSK1 = 000110          $$RTN1= 050000
UIPAR5= 177652           WDTO   = 000114         $F$NAM= 000160          $NSK2 = 000110          $$RTN2= 050001
UIPAR6= 177654           WSTBUS= ****** G        $F$NO  = 000403          $SAVLE= 177777          $$SRC = 000000
UIPAR7= 177656           WTINRE= 000352          $F$OR  = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPDR0= 177600           WTWHMI= 000222          $F$RTI= 000350          $TAGNU= 050011          $$TGS1= 000000
UIPDR1= 177602           XFLAG  = 000005         $F$RTN= 000300          $TEMP  = 000300          $$TGS2= 000000
UIPDR2= 177604           XOFF   = 000023         $F$SEL= 000140          $TSK0  = 050007          $$TO   = 000000
UIPDR3= 177606           XON    = 000021         $F$THE= 000330          $TSK1  = 050010          $$$TAG= 050000
UIPDR4= 177610           $BGNLE= 177777          $F$TRU= 000404          $$ARGC= 000002          .      = 000252R

. ABS.  000000     000
        000252     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RELCTL,DSKZ:RELCTL=SPMAC/ML,EQUATE,RELCTL
 RUN-TIME: 13 4 .3 SECONDS
 RUN-TIME RATIO: 32/18=1.7
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  RONOF  PROCESS THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS
                        .IDENT  /V0.0/


                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       RONOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE CONSISTS OF TWO MODULES:
                        ;       1. KROTON - PROCESS THE KEYBOARD COMMAND 'ROTON'
                        ;       2. KROTOF - PROCESS THE KEYBOARD COMMAND 'ROTOFF'
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR RONOF

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG = 1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KROTON          ;'ROTON' MODULE ENTRY POINT
                                .GLOBL  KROTOF          ;'ROTOFF' MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  CM.ARG          ;ILLEGAL ARGUMENT MESSAGE
                                .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
                                .GLOBL  SAVREG
                                .GLOBL  RESREG
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 051127  052111  020105  RO.MGN: .ASCIZ  /WRITE BUFFER ROTATION ON%/
000006' 052502  043106  051105
000014' 051040  052117  052101
000022' 047511  020116  047117
000030' 000045
000032' 051127  052111  020105  RO.MGF: .ASCIZ  /WRITE BUFFER ROTATION OFF%/
000040' 052502  043106  051105
000046' 051040  052117  052101
000054' 047511  020116  043117
000062' 022506     000
000065'    000                  RO.FLG: .BYTE   0       ;1 IF ROTOF, 0 IF ROTON
                                ;
                                .EVEN
```

```
                        .SBTTL  KROTON & KROTOF - PROCESS THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KROTON - KROTOF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS
                        ;       IT WILL FIRST VERIFY THAT NO ARGUMENT APPEARS
                        ;       IN THE COMMAND DECODE BUFFER. IF AN ARGUMENT IS IN THE
                        ;       BUFFER, AN ERROR MESSAGE IS ISSUED. OTHERWISE,
                        ;       THE WRITE BUFFER ROTATION BIT IN THE STATUS
                        ;       INDICATOR IS SET IF 'ROTON' AND CLEARED IF 'ROTOFF'.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DTABLE
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;       DT.STO
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       ARGCHK - CHECK FOR UNWANTED ARGUMENTS
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KROTON IN <DTADR,BUFPTR>
                        ;       CALL KROTOF IN <DTADR,BUFPTR>
                        ;       WHERE:   DTADR = ADDRESS OF DTABLE
                        ;                BUFPTR = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  KROTON - KROTOF ROUTINE


                                INLINE <KROTOF:>
                                                                        KROTOF:

                                ;+
                                ; INDICATE IT'S 'ROTOFF'
                                ;-

                                LET RO.FLG :B= RO.FLG + #1
                                                                        INCB    RO.FLG
                                INLINE <BR      GUNCH>
                                                                        BR      GUNCH


                                ROUTINE KROTON <DTADR,BUFPTR>
                                                                    KROTON:


                                ;+
                                ; INDICATE 'ROTON' COMMAND
                                ;-

                                LET RO.FLG :B= #0
                                                                        CLRB    RO.FLG

                                INLINE <GUNCH:>
                                                                        GUNCH:
                                ;+
                                ; SAVE REGISTERS AND SAVE DTABLE ADDRESS AND DECODE BUFFER POINTER
                                ;-

                                CALL SAVREG
                                                                        JSR     PC,SAVREG
                                LET R0 := DTADR(R5)
                                                                        MOV     DTADR(R5),R0
                                LET R1 := BUFPTR(R5)
                                                                        MOV     BUFPTR(R5),R1


                                ;+
                                ; MAKE SURE NO ADDITIONAL ARGUMENTS ARE IN DECODE BUFFER
                                ;-

                                CALL ARGCHK IN <R1> OUT <R1>
                                                                        SUB     #1*2,R5
                                                                        MOV     R5,-(SP)
                                                                        MOV     R1,-(R5)
                                                                        JSR     PC,ARGCHK
                                                                        MOV     (SP)+,R5
                                                                        MOV     (R5)+,R1

                                IF.NO.ERROR THEN
                                                                        BCS     50002$
```

```
                                ;+
                                ; THERE IS NO <CR> YET SO IT'S A BAD COMMAND...STUFF ERROR
                                ; MESSAGE AND BEAT IT....
                                ;-

000136'                                 LET DT.KBRSP(R0) := #CM.ARG
(4) 000136' 012760  000000G 000022                                          MOV     #CM.ARG,DT.KBRSP

000144'                         ELSE
(4) 000144' 000420                                                          BR      50003$
(3) 000146'                                                         50002$:

                                ;+
                                ; IT'S A <CR> SO GO PROCESS.... SET THE WBSTAT BIT
                                ; IN DT.STO IF 'ROTON' AND CLEAR IT IF 'ROTOF'
                                ;-

000146'                                 IFB RO.FLG EQ #0 THEN
(6) 000146' 105767  177713                                                  TSTB    RO.FLG
(9) 000152' 001007                                                          BNE     50004$
000154'                                         LET DT.STO(R0) := DT.STO(R0) SET.BY #WBSTAT
(6) 000154' 052760  000040  000010                                          BIS     #WBSTAT,DT.STO(R
000162'                                         LET DT.KBRSP(R0) := #RO.MGN
(4) 000162' 012760  000000' 000022                                          MOV     #RO.MGN,DT.KBRSP

000170'                                 ELSE
(4) 000170' 000406                                                          BR      50005$
(3) 000172'                                                         50004$:

000172'                                         LET DT.STO(R0) := DT.STO(R0) CLR.BY #WBSTAT
(6) 000172' 042760  000040  000010                                          BIC     #WBSTAT,DT.STO(R
000200'                                         LET DT.KBRSP(R0) := #RO.MGF
(4) 000200' 012760  000032' 000022                                          MOV     #RO.MGF,DT.KBRSP
000206'                                 ENDIF
(4) 000206'                                                         50005$:
000206'                         ENDIF
(4) 000206'                                                         50003$:


                                ;+
                                ; CLEAN UP AND LEAVE
                                ;-

000206'                         CALL RESREG
(3) 000206' 004767  000000G                                                 JSR     PC,RESREG

000212'                         ENDRTN
(3) 000212'                                                         50000$:
(3) 000212'                                                         50001$:
(2) 000212' 000207                                                          RTS     PC
            000001              .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
BUFPTR= 000002
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CM.ARG= ****** G
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GUNCH   000100R
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KROTOF  000066RG
KROTON  000074RG
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RO.FLG  000065R
RO.MGF  000032R
RO.MGN  000000R
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
```

```
SVR3   = 000070          UIPDR4= 177610          $F$BAD= 000401          $F$YES= 000402          $$DST  = 000000
SVR4   = 000072          UIPDR5= 177612          $F$BLA= 000170          $IFLEV= 177777          $$ELOC= 000402
SVR5   = 000074          UIPDR6= 177614          $F$CAS= 000150          $ISK0 = 000001          $$ERFL= 000000
SVR6   = 000076          UIPDR7= 177616          $F$DEC= 000220          $ISK1 = 000001          $$FLAG= 000001
SYSCNT= 000052           WASADR= 000104          $F$DO  = 000340          $LOCTA= 177777          $$FROM= 000000
SYSERR= 000100           WBSTAT= 000040          $F$FAL= 000405          $LSTIN= 000001          $$LOC  = 000152R
TMPIO  = 000002          WBUFEA= 000136          $F$GOO= 000400          $LSTTA= 000001          $$LOCN= 000000
TQOVF  = 000002          WBUFPA= 000134          $F$IF  = 000110          $NESTL= 177777          $$REG  = 177777
UIPAR0= 177640           WBUFRQ= 000140          $F$INC= 000210          $NSK0  = 000300          $$RETU= 000000
UIPAR1= 177642           WBUFSZ= 000142          $F$LOO= 000200          $NSK1  = 000110          $$RTN1= 050000
UIPAR2= 177644           WDFR   = 000116          $F$NAM= 000160          $NSK2  = 000110          $$RTN2= 050001
UIPAR3= 177646           WDTO   = 000114          $F$NO  = 000403          $SAVLE= 177777          $$SRC  = 000000
UIPAR4= 177650           WTINRE= 000352          $F$OR  = 000320          $TAGLE= 177777          $$TGSV= 000000
UIPAR5= 177652           WTWHMI= 000222          $F$RTI= 000350          $TAGNU= 050006          $$TGS1= 000000
UIPAR6= 177654           XFLAG  = 000005          $F$RTN= 000300          $TEMP  = 000300          $$TGS2= 000000
UIPAR7= 177656           XOFF   = 000023          $F$SEL= 000140          $TSK0  = 050003          $$TO   = 000000
UIPDR0= 177600           XON    = 000021          $F$THE= 000330          $TSK1  = 050005          $$$TAG= 050000
UIPDR1= 177602           $BGNLE= 177777          $F$TRU= 000404          $$ARGC= 000004          .      = 000214R
UIPDR2= 177604           $ERFLG= 000400          $F$UNT= 000130          $$BYTE= 000402
UIPDR3= 177606           $F$AND= 000310          $F$WHI= 000120          $$CASE= 000000

. ABS.  000000     000
        000214     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:RONOF,DSKZ:RONOF=SPMAC/ML,EQUATE,RONOF
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 57/16=3.5
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE SAVRES BINDER FOR SAVREG AND RESREG
                        .IDENT /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       SAVRES
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       BINDER FOR THE FOLLOWING MODULES:
                        ;               SAVREG - SAVE REGISTERS
                        ;               RESREG - RESTORE REGISTERS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL  SAVREG SAVE REGISTER R0 THRU R4
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SAVREG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SAVE GENERAL PURPOSE REGISTERS R0 THRU R4
                        ;       ON THE R6 STACK AND RETURNS TO CALLER
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SAVREG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
                                STRUCT
                                .PRINT  ;SPMAC: VERSION 1.1
                                $LSTIN = 1
                                $LSTTAG = 1

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  SAVREG          ;MODULE ENTRY POINT
```

```
  567
  568
  569
  570
  571  000000'
  (1)  000000'
  572           000001
  573           000001
  574
  575
  576
  577
  578
  579
```

```
                        .SBTTL SAVE REGISTER'S R0 THRU R4 ROUTINE


                        ROUTINE SAVREG
000000'                                                         SAVREG:
(2)  000000'

                        ;+
                        ; FIRST ALLOCATE 10 BYTES ON THE STACK POINTER BY SETTING STACK
                        ; POINTER TO SP + 12, THEN LOAD RETURN ADDRESS INTO NEW STACK POINTER
                        ; NOW SAVE REGISTERS
                        ;-

000000'                 LET SP := SP - #12                              SUB     #12,SP
(6)  000000' 162706  000012
000004'                 LET (SP) := +12(SP)                             MOV     +12(SP),(SP)
(4)  000004' 016616  000012
000010'                 LET 2(SP) := R4                                 MOV     R4,2(SP)
(4)  000010' 010466  000002
000014'                 LET 4(SP) := R3                                 MOV     R3,4(SP)
(4)  000014' 010366  000004
000020'                 LET 6(SP) := R2                                 MOV     R2,6(SP)
(4)  000020' 010266  000006
000024'                 LET 10(SP) := R1                                MOV     R1,10(SP)
(4)  000024' 010166  000010
000030'                 LET 12(SP) := R0                                MOV     R0,12(SP)
(4)  000030' 010066  000012


                        ;+
                        ; RETURN TO CALLER
                        ;-


                        ENDRTN
000034'                                                         50000$:
(3)  000034'                                                    50001$:
(3)  000034'                                                            RTS     PC
(2)  000034' 000207
```

```
                        .SBTTL  RESREG RESTORE GENERAL REGISTERS R0 THRU R4
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       RESREG
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL RESTORE GENERAL REGISTERS R0 THRU R4 FROM THE
                        ;       R6 STACK AND RETURN TO CALLER.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL RESREG
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES


                        ;***************
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  RESREG                  ;MODULE ENTRY POINT
```

```
                        .SBTTL RESTORE GENERAL REGISTERS R0 THRU R4 ROUTINE

                        ROUTINE RESREG
                                                                RESREG:

                        ;+
                        ; RESTORE REGISTERS R0 THRU R4 FROM R6 STACK, THEN CLEAN UP STACK FOR
                        ; RETURN TO CALLER
                        ;-

                        LET R4 := 2(SP)                                 MOV     2(SP),R4
                        LET R3 := 4(SP)                                 MOV     4(SP),R3
                        LET R2 := 6(SP)                                 MOV     6(SP),R2
                        LET R1 := 10(SP)                                MOV     10(SP),R1
                        LET R0 := 12(SP)                                MOV     12(SP),R0
                        LET 12(SP) := (SP)                              MOV     (SP),12(SP)
                        LET SP := SP + #12                              ADD     #12,SP


                        ;+
                        ; RETURN TO CALLER
                        ;-

                        ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC

                        .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CFO= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.STO= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG  000036RG
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
RSVALU= 001740
SAM   = 075464
SAVREG  000000RG
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
```

```
UIPAR2= 177644          WBUFPA= 000134          $F$DO  = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR5= 177652          WDFR   = 000116         $F$IF  = 000110         $LSTTA= 000001          $$REG = 177777
UIPAR6= 177654          WDTO   = 000114         $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $NSKO  = 000300         $$RTN1= 050000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR1= 177602          XFLAG  = 000005         $F$NO  = 000403         $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XOFF   = 000023         $F$OR  = 000320         $TAGNU= 050002          $$TGSV= 000000
UIPDR3= 177606          XON    = 000021         $F$RTI= 000350          $TEMP  = 000300         $$TGS1= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000000          $$TGS2= 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO   = 000000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$DST  = 000000         .      = 000074R
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000

. ABS.  000000    000
        000074    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SAVRES,DSKZ:SAVRES=SPMAC/ML,EQUATE,SAVRES
 RUN-TIME: 12 2 .4 SECONDS
 RUN-TIME RATIO: 69/15=4.5
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE  SELDES  PROCESS THE KEYBOARD COMMANDS 'SEL' AND 'DES'
                        .IDENT /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       SELDES
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS PACKAGE CONTAINS TWO ROUTINES:
                        ;       1. KSEL - PROCESS THE KEYBOARD COMMAND 'SEL'
                        ;       2. KDES - PROCESS THE KEYBOARD COMMAND 'DES'
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;
                        ;--
```

```
                                .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR SELDES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1

        000001                  $LSTIN=1
        000001                  $LSTTAG=1
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL  KSEL            ;SELECT A MODULE
                                .GLOBL  KDES            ;DESELCT A MODULE
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES:
                                ;
                                .GLOBL  NAMCHK          ;MODULE NAME VERIFICATION ROUTINE
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  RESREG          ;RESTORE REGISTERS
                                .GLOBL  CM.BADNAME      ;ADDRESS OF BAD MODULE NAME MESSAGE
                                .GLOBL  CM.ARG          ;ADDRESS OF INVALID ARGUMENT MESSAGE
                                .GLOBL  ARGCHK          ;ARGCHK ROUTINE
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE:
                                ;
000000' 000000                  SE.FLG: .WORD   0       ;WHEN A 1 - SELECT, WHEN A 0, DESELECT
                                ;
                                ;****************
```

```
                        .SBTTL  KSEL   PROCESS THE 'SEL' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KSEL
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'SEL' KEYBOARD COMMAND.
                        ;       IT SHALL DETERMINE IF A MODULE NAME HAS BEEN INCLUDED IN
                        ;       THE COMMAND STRING. IF THERE IS NO MODULE NAME, ALL MODULES'
                        ;       STATUS WORDS WILL HAVE THE SELECT BIT SET. IF THERE IS A
                        ;       MODULE NAME IN THE COMMAND STRING, THE NAME MUST FIRST BE VERIFIED
                        ;       AND IF VALID, THE STATUS WORD IN THE MODULE WILL BE UPDATED(SELECT
                        ;       BIT SET).
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;       DT.ST1
                        ;       DT.ST0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;       CM.BADNAME
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KSEL IN <DTADR,CMDBUF>
                        ;       WHERE DTADR = ADDRESS OF DTABLE
                        ;                     CMDBUF = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                       .SBTTL  KSEL ROUTINE

                                       ROUTINE KSEL <DTADR,CMDBUF>                                  KSEL:
000002'                                LET SE.FLG := #1                                                     MOV     #1,SE.FLG
(4)  000002' 012767  000001  177770
                                       INLINE <BR      PRSEDE>                                              BR      PRSEDE
000010' 000403
                                       ENDRTN                                                       50000$:
000012'                                                                                             50001$:
000012'                                                                                                     RTS     PC
(2)  000012' 000207
```

```
                        .SBTTL  KDES   PROCESS THE 'DES' KEYBOARD COMMAND
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       KDES
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS ROUTINE PROCESSES THE 'DES' KEYBOARD COMMAND.
                        ;       IT SHALL DETERMINE IF A MODULE NAME HAS BEEN INCLUDED IN
                        ;       THE COMMAND STRING. IF THERE IS NO MODULE NAME, ALL MODULES'
                        ;       STATUS WORDS WILL HAVE THE SELECT BIT CLEARED. IF THERE IS A
                        ;       MODULE NAME IN THE COMMAND STRING, THE NAME MUST FIRST BE VERIFIED
                        ;       AND IF VALID, THE STATUS WORD IN THE MODULE WILL BE UPDATED(SELECT
                        ;       BIT CLEARED).
                        ;
                        ; INPUTS:
                        ;       1. DTABLE ADDRESS
                        ;       2. COMMAND DECODE BUFFER POINTER
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.MLST
                        ;       DT.ST0
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.KBRSP
                        ;       DT.ST1
                        ;       DT.ST0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       CM.ARG
                        ;       CM.BADNAME
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG
                        ;       RESREG
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KDES IN <DTADR,CMDBUF>
                        ;       WHERE DTADR = ADDRESS OF DTABLE
                        ;                     CMDBUF = COMMAND DECODE BUFFER POINTER
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL  KDES ROUTINE

                                ROUTINE KDES <DTADR,CMDBUF>
                                                                                KDES:
                                ;+
                                ; SET THE FLAG TO DESELECT AND CALL THE COMMON ROUTINE
                                ;-

                                LET SE.FLG := #0
                                                                                CLR     SE.FLG
```

```
                                .SBTTL  PRSEDE ROUTINE

                                INLINE <PRSEDE:>                                                        PRSEDE:
                                ;+
                                ; SAVE REGISTERS AND SAVE DTABLE ADDRESS
                                ;-

000020'                         CALL SAVREG                                                             JSR     PC,SAVREG
000020' 004767  000000G
000024'                         LET R0 := DTADR(R5)                                                     MOV     DTADR(R5),R0
000024' 016500  000000

                                ;+
                                ; GET ADDRESS OF MODULE LIST AND SAVE DECODE BUFFER POINTER
                                ;-

000030'                         LET R1 := DT.MLST(R0)                                                   MOV     DT.MLST(R0),R1
000030' 016001  000032
000034'                         LET R2 := CMDBUF(R5)                                                    MOV     CMDBUF(R5),R2
000034' 016502  000002

                                ;+
                                ; ADVANCE DECODE BUFFER PTR TO FIRST NON SPACE CHARACTER.
                                ; IF IT'S A <CR>, SELECT ALL MODULES - IF NOT A <CR>
                                ; GO VERIFY THE MODULE NAME
                                ;-

000040'                         CALL ARGCHK IN <R2> OUT <R2>                                            SUB     #1*2,R5
000040' 162705  000002                                                                                  MOV     R5,-(SP)
000044' 010546                                                                                          MOV     R2,-(R5)
000046' 010245                                                                                          JSR     PC,ARGCHK
000050' 004767  000000G                                                                                 MOV     (SP)+,R5
000054' 012605                                                                                          MOV     (R5)+,R2
000056' 012502

000060'                         IF.ERROR THEN                                                           BCC     50002$
000060' 103047

                                ;+
                                ; ITS A CR SO SELECT - DESELECT THEM ALL
                                ;-

                                ;+
                                ; IF WE ARE SELECTING, SET BIT 14.
                                ; IF WE ARE DESELECTING, CLEAR BIT 14.
                                ;-

000062'                                 IF SE.FLG GT #0 THEN
```

```
(6)  000062' 005767  177712                                                          TST     SE.FLG
(9)  000066' 003431                                                                  BLE     50003$
740  000070'                                 WHILE (R1) NE #0 DO
(4)  000070'                                                                  50004$:
(6)  000070' 005711                                                                  TST     (R1)
(9)  000072' 001417                                                                  BEQ     50005$
741
742                                  ;+
743                                  ; GET A MODULE HEADER ADDRESS FROM MODULE LIST, UPDATE
744                                  ; STATUS WORD, ADVANCE TO GET NEXT MODULE HEADER ADDRESS.
745                                  ;-
746
747  000074'                                     LET R2 := (R1)+
(4)  000074' 012102                                                                  MOV     (R1)+,R2
748
749  000076'                                     LET STAT(R2) := STAT(R2) SET.BY #BIT14
(6)  000076' 052762  040000  000026                                                 BIS     #BIT14,STAT(R2)
750  000104'                                     LET R4 := STAT(R2) CLR.BY #STATBITS
(4)  000104' 016204  000026                                                          MOV     STAT(R2),R4
(6)  000110' 042704  064757                                                          BIC     #STATBITS,R4
751  000114'                                     IF R4 EQ #SBKMOD THEN
(6)  000114' 020427  000000                                                          CMP     R4,#SBKMOD
(9)  000120' 001003                                                                  BNE     50006$
752  000122'                                            LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
(6)  000122' 052760  010000  000012                                                 BIS     #SBKSEL,DT.ST1(R
753  000130'                                     ENDIF
(4)  000130'                                                                  50006$:
754
755  000130'                                 ENDDO
(4)  000130' 000757                                                                  BR      50004$
(3)  000132'                                                                  50005$:
756  000132'                                 IF #RUNMODE SETIN DT.ST0(R0) THEN
(6)  000132' 032760  100000  000010                                                 BIT     #RUNMODE,DT.ST0(
(9)  000140' 001403                                                                  BEQ     50007$
757  000142'                                        LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODSEL
(6)  000142' 052760  001000  000010                                                 BIS     #MODSEL,DT.ST0(R
758  000150'                                 ENDIF
(4)  000150'                                                                  50007$:
759
760  000150'                                ELSE
(4)  000150' 000412                                                                  BR      50010$
(3)  000152'                                                                  50003$:
761
762  000152'                                     LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
(6)  000152' 052760  100000  000012                                                 BIS     #CKTIM,DT.ST1(R0
763  000160'                                 WHILE (R1) NE #0 DO
(4)  000160'                                                                  50011$:
(6)  000160' 005711                                                                  TST     (R1)
(9)  000162' 001405                                                                  BEQ     50012$
764
765                                  ;+
766                                  ; GET A MODULE HEADER ADDRESS FROM MODULE LIST, UPDATE
767                                  ; STATUS WORD, ADVANCE TO GET NEXT MODULE HEADER ADDRESS.
768                                  ;-
769
770  000164'                                     LET R2 := (R1)+
```

```
(4)  000164' 012102                                                                    MOV     (R1)+,R2
771  000166'                                  LET STAT(R2) := STAT(R2) CLR.BY #BIT14!ACTBIT
(6)  000166' 042762  044000  000026                                                   BIC     #BIT14!ACTBIT,ST
772
773  000174'                                  ENDDO                                    BR      50011$
(4)  000174' 000771                                                          50012$:
(3)  000176'
774
775  000176'                             ENDIF                                50010$:
(4)  000176'
776
777
778  000176'                         ELSE                                             BR      50013$
(4)  000176' 000472                                                          50002$:
(3)  000200'
779
780                                  ;+
781                                  ; THERE IS A MODULE NAME, SO GO CHECK IT OUT
782                                  ;-
783
784  000200'                             CALL NAMCHK IN <R0,R2> OUT <R3,R2>
(4)  000200' 162705  000004                                                          SUB     #2*2,R5
(3)  000204' 010546                                                                  MOV     R5,-(SP)
(5)  000206' 010245                                                                  MOV     R2,-(R5)
(4)  000210' 010045                                                                  MOV     R0,-(R5)
(3)  000212' 004767  000000G                                                         JSR     PC,NAMCHK
(3)  000216' 012605                                                                  MOV     (SP)+,R5
(4)  000220' 012503                                                                  MOV     (R5)+,R3
(4)  000222' 012502                                                                  MOV     (R5)+,R2
785
786                                  ;+
787                                  ; SEE IF NAME O.K.  IF IT WAS, THE MODULE HEADER ADDRESS
788                                  ; WILL BE IN R3.
789                                  ;-
790  000224'                             IF.NO.ERROR THEN                                 BCS     50014$
(6)  000224' 103454
791
792
793                                  ; CALL ARGCHK AGAIN TO SEE IF ANY
794                                  ; MORE ARGUMENTS (IF SO, THEY ARE ILLEGAL)
795                                  ;-
796
797
798  000226'                                  CALL ARGCHK IN <R2> OUT <R2>
(4)  000226' 162705  000002                                                          SUB     #1*2,R5
(3)  000232' 010546                                                                  MOV     R5,-(SP)
(4)  000234' 010245                                                                  MOV     R2,-(R5)
(3)  000236' 004767  000000G                                                         JSR     PC,ARGCHK
(3)  000242' 012605                                                                  MOV     (SP)+,R5
(4)  000244' 012502                                                                  MOV     (R5)+,R2
799  000246'                                  IF.ERROR THEN                           BCC     50015$
(6)  000246' 103037
800
801                                  ;+
802                                  ; IT'S A CR, SO ALL IS WELL, SO IF SELECTING - SET BIT 14
803                                  ; AND IF DESELECTING - CLEAR BIT 14
```

```
                                ;-
                                
                                
                                
                                                IF SE.FLG GT #0 THEN
000250'                                                                         TST     SE.FLG
000250' 005767  177524                                                          BLE     50016$
000254' 003425
                                                        LET STAT(R3) := STAT(R3) SET.BY #BIT14
000256'                                                                         BIS     #BIT14,STAT(R3)
                                                        IF #RUNMODE SETIN DT.ST0(R0) THEN
000264'                                                                         BIT     #RUNMODE,DT.ST0(
000264' 032760  100000  000010                                                  BEQ     50017$
000272' 001403
                                                                LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODSEL
000274'                                                                         BIS     #MODSEL,DT.ST0(R
000274' 052760  001000  000010
                                                        ENDIF
000302'                                                                 50017$:
000302'
                                                        LET R4 := STAT(R3) CLR.BY #STATBITS
000302'                                                                         MOV     STAT(R3),R4
000302' 016304  000026                                                          BIC     #STATBITS,R4
000306' 042704  064757
                                                        IF R4 EQ #SBKMOD THEN
000312'                                                                         CMP     R4,#SBKMOD
000312' 020427  000000                                                          BNE     50020$
000316' 001003
                                                                LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
000320'                                                                         BIS     #SBKSEL,DT.ST1(R
000320' 052760  010000  000012
                                                        ENDIF
000326'                                                                 50020$:
000326'
                                                ELSE
000326'                                                                         BR      50021$
000326' 000406                                                          50016$:
000330'

                                ;+
                                ; IT'S A DESELECT.... DO IT
                                ;-

                                                        LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
000330'                                                                         BIS     #CKTIM,DT.ST1(R0
000330' 052760  100000  000012                                                  LET STAT(R3) := STAT(R3) CLR.BY #BIT14!ACTBIT
000336'                                                                         BIC     #BIT14!ACTBIT,ST
000336' 042763  044000  000026
                                                ENDIF
000344'                                                                 50021$:
000344'
                                        ELSE
000344'                                                                         BR      50022$
000344' 000403                                                          50015$:
000346'

                                ;+
                                ; BAD ! THERE ARE ADDITIONAL ARGUMENTS BEYOND THE MODULE NAME
                                ; SO STUFF THE BAD ARG MESSAGE AND GET OUT
                                ;-
                                        LET DT.KBRSP(R0) := #CM.ARG
000346'                                                                         MOV     #CM.ARG,DT.KBRSP
000346' 012760  000000G 000022
                                        ENDIF
000354'                                                                 50022$:
000354'
                                ELSE
000354'                                                                         BR      50023$
000354' 000403                                                          50014$:
000356'
```

```
                                ;+
                                ; THERE WAS AN ERROR IN SYNTAX OF MODULE NAME
                                ; SO STUFF THE INVALID MODULE NAME ERROR MSG
                                ;-

000356'                                         LET DT.KBRSP(R0) := #CM.BADNAME
(4)  000356' 012760  000000G 000022                                                    MOV     #CM.BADNAME,DT.K
                                        ENDIF
000364'                                                                         50023$:
(4)  000364'

                                ENDIF
000364'                                                                         50013$:
(4)  000364'

                                ;+
                                ; CLEAN UP
                                ;-

                                CALL RESREG
000364'                                                                                 JSR     PC,RESREG
(3)  000364' 004767  000000G

                                ENDRTN
000370'                                                                         50000$:
(3)  000370'                                                                    50001$:
(3)  000370'                                                                            RTS     PC
(2)  000370' 000207

          000001                .END
```

SELDES.MAC 10-AUG-78 17:09 SYMBOL TABLE

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ARGCHK= ****** G
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CMDBUF= 000002
CM.ARG= ****** G
CM.BAD= ****** G
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTADR  = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KDES     000014RG
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KSEL     000002RG
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NAMCHK= ****** G
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PRSEDE   000020R
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SE.FLG   000000R
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
```

```
SVR3   = 000070          UIPDR6= 177614          $F$DO  = 000340          $ISK4  = 000001          $$BYTE= 000403
SVR4   = 000072          UIPDR7= 177616          $F$FAL= 000405          $LOCTA= 177777          $$CASE= 000000
SVR5   = 000074          WASADR= 000104          $F$GOO= 000400          $LSTIN= 000001          $$DST = 000000
SVR6   = 000076          WBSTAT= 000040          $F$IF  = 000110          $LSTTA= 000001          $$ELOC= 000402
SYSCNT= 000052          WBUFEA= 000136          $F$INC= 000210          $NESTL= 177777          $$ERFL= 000000
SYSERR= 000100          WBUFPA= 000134          $F$LOO= 000200          $NSK0  = 000300          $$FLAG= 000001
TMPIO  = 000002          WBUFRQ= 000140          $F$NAM= 000160          $NSK1  = 000110          $$FROM= 000000
TQOVF  = 000002          WBUFSZ= 000142          $F$NO  = 000403          $NSK2  = 000110          $$LOC  = 000316R
UIPAR0= 177640          WDFR   = 000116          $F$OR  = 000320          $NSK3  = 000110          $$LOCN= 000000
UIPAR1= 177642          WDTO   = 000114          $F$RTI= 000350          $NSK4  = 000110          $$REG  = 177777
UIPAR2= 177644          WTINRE= 000352          $F$RTN= 000300          $NSK5  = 000110          $$RETU= 000000
UIPAR3= 177646          WTWHMI= 000222          $F$SEL= 000140          $SAVLE= 177777          $$RTN1= 050000
UIPAR4= 177650          XFLAG  = 000005          $F$THE= 000330          $SSK0  = 050012          $$RTN2= 050001
UIPAR5= 177652          XOFF   = 000023          $F$TRU= 000404          $TAGLE= 177777          $$SRC  = 000000
UIPAR6= 177654          XON    = 000021          $F$UNT= 000130          $TAGNU= 050024          $$TGSV= 000000
UIPAR7= 177656          $BGNLE= 177777          $F$WHI= 000120          $TEMP  = 000300          $$TGS1= 000000
UIPDR0= 177600          $ERFLG= 000400          $F$YES= 000402          $TSK0  = 050013          $$TGS2= 000000
UIPDR1= 177602          $F$AND= 000310          $IFLEV= 177777          $TSK1  = 050023          $$TO   = 000000
UIPDR2= 177604          $F$BAD= 000401          $ISK0  = 000001          $TSK2  = 050022          $$$TAG= 050000
UIPDR3= 177606          $F$BLA= 000170          $ISK1  = 000001          $TSK3  = 050021          .      = 000372R
UIPDR4= 177610          $F$CAS= 000150          $ISK2  = 000001          $TSK4  = 050020
UIPDR5= 177612          $F$DEC= 000220          $ISK3  = 000001          $$ARGC= 000004

. ABS.  000000    000
        000372    001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SELDES,DSKZ:SELDES=SPMAC/ML,EQUATE,SELDES
 RUN-TIME: 17 8 .4 SECONDS
 RUN-TIME RATIO: 91/26=3.4
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE STRAP (FIELD SOFTWARE TRAP)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       STRAP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS ROUTINE IS ENTERED BY ANY ONE OF SEVERAL SOFTWARE TRAPS
                        ;       (I.E. MSGN, GWBUFF, ETC.)FROM DECX/11 OPTION MODULES.
                        ;       AN EVENT CODE IS GENERATED AND
                        ;       PUSHED ONTO THE STACK.
                        ;       THERE IS ONE ENTRY POINT FOR A SOFTWARE TRAP
                        ;       THRU LOC. 34.
                        ;
                        ; INPUTS:
                        ;       NONE.
                        ;
                        ; IMPLICIT INPUTS:
                        ;       SOFTWARE TRAP INSTRUCTION (THRU LOCATION 34)
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       EVENT CODE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE.
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       NONE.
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALLED BY A 1044XX TRAP INSTRUCTION
                        ;         XX = THE CODE FOR THE PARTICULAR TRAP INSTRUCTION.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ; EDIT          BY              DATE            REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES
                                ;
                                        .MCALL  STRUCT
                                        STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
000000'                                 $LSTIN = 1
                000001                  $LSTTAG = 1
                000001

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  STINT           ;ENTRY ADDRESS OF FIELD SOFTWARE TRAP ROUTINE
                                .GLOBL  ST.EXT          ;ADDRESS TO EXIT THROUGH
                                ;
                                ;
                                ;****************
                                ;
                                ;LOCAL STORAGE:
                                ;
000000' 000000                  ST.EXT: .WORD   0       ;ADDRESS TO EXIT THIS MODULE THRU
                                ;
```

```
                                .SBTTL STRAP ROUTINE


                                ;+
                                ;ENTER HERE FOR SOFTWARE TRAP
                                ;-

000002'                         INLINE <STINT:>
(2)  000002'                                                            STINT:

                                ;+
                                ; SAVE REGISTER AND RETRIEVE PC+2 FROM STACK AND DETERMINE ACTUAL TRAP ADDRESS
                                ;-

000002'                         PUSH    R0
(2)  000002' 010046                                                     MOV     R0,-(SP)
000004'                         LET     R0 := 2(SP)
(4)  000004' 016600  000002                                             MOV     2(SP),R0
000010'                         LET     R0 := -(R0)
(4)  000010' 014000                                                     MOV     -(R0),R0

                                ;+
                                ; IDENTIFY TRAP AND SAVE EVENT CODE ON STACK, RESTORE REGISTER
                                ; AND CLEAN UP STACK AND EXIT ROUTINE
                                ;-

000012'                         LET     R0 := R0 CLR.BY #177400
(6)  000012' 042700  177400                                             BIC     #177400,R0
000016'                         PUSH    R0
(2)  000016' 010046                                                     MOV     R0,-(SP)
000020'                         LET     R0 := 2(SP)
(4)  000020' 016600  000002                                             MOV     2(SP),R0
000024'                         LET     (SP) := (SP)+
(4)  000024' 012616                                                     MOV     (SP)+,(SP)
000026' 000177  177746          JMP     @ST.EXT

                                .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
STINT   000002RG
ST.EXT  000000RG
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
```

```
UIPAR2= 177644          WBUFPA= 000134          $F$DO  = 000340          $IFLEV= 177777          $$LOC  = 000000
UIPAR3= 177646          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOCN= 000000
UIPAR4= 177650          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$REG  = 177777
UIPAR5= 177652          WDFR   = 000116         $F$IF  = 000110         $LSTTA= 000001          $$RETU= 000000
UIPAR6= 177654          WDTO   = 000114         $F$INC= 000210          $NESTL= 177777          $$RTN1= 000000
UIPAR7= 177656          WTINRE= 000352          $F$LOO= 000200          $SAVLE= 177777          $$RTN2= 000000
UIPDR0= 177600          WTWHMI= 000222          $F$NAM= 000160          $TAGLE= 177777          $$SRC  = 000000
UIPDR1= 177602          XFLAG  = 000005         $F$NO  = 000403         $TAGNU= 050000          $$TGSV= 000000
UIPDR2= 177604          XOFF   = 000023         $F$OR  = 000320         $TEMP  = 000402         $$TGS1= 000000
UIPDR3= 177606          XON    = 000021         $F$RTI= 000350          $$ARGC= 000000          $$TGS2= 000000
UIPDR4= 177610          $BGNLE= 177777          $F$RTN= 000300          $$BYTE= 000000          $$TO   = 000000
UIPDR5= 177612          $ERFLG= 000400          $F$SEL= 000140          $$CASE= 000000          $$$TAG= 050000
UIPDR6= 177614          $F$AND= 000310          $F$THE= 000330          $$DST  = 000000          .      = 000032R
UIPDR7= 177616          $F$BAD= 000401          $F$TRU= 000404          $$ELOC= 000000
WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ERFL= 000000
WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$FLAG= 000000
WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FROM= 000000

. ABS.  000000     000
        000032     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:STRAP,DSKZ:STRAP=SPMAC/ML,EQUATE,STRAP
 RUN-TIME: 11 1 .3 SECONDS
 RUN-TIME RATIO: 64/13=4.8
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SYSCLK SYSTEM CLOCK BINDER MODULE
                        .IDENT  /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       SYSCLK
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       BINDER FOR THE FOLLOWING MODULES:
                        ;               CLKCHK - SYSTEM CLOCK CHECK
                        ;               CLKOFF - TURN SYSTEM CLOCK OFF
                        ;               CLKON  - TURN SYSTEM CLOCK ON
                        ;               PRHMS  - PROCESS HOURS, MINUTES AND SECONDS CONVERSION
                        ;               PRPSCNT - PROCESS PASS COUNT TIME
                        ;               HMS    - HOURS, MINUTES AND SECONDS CONVERSION
                        ;               UPDTIM - UPDATE PASS TIME
                        ;               CLRTIM - CLEAR MODULE TIME TABLE
                        ;               CKHUNG - CHECK FOR HUNG OPTION MODULE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY                      REASON
                        ;--
```

```
533
534                             .SBTTL CLKCHK - SYSTEM CLOCK CHECK
535                             .IDENT /V0.0/
536
537                             ;++
538                             ; MODULE NAME:
539                             ;       CLKCHK
540                             ;
541                             ; FUNCTIONAL DESCRIPTION:
542                             ;       THIS MODULE DETERMINES IF A CLOCK IS AVAILABLE ON THE SYSTEM. THIS IS
543                             ;       ACCOMPLISHED BY FIRST CHECKING TO SEE IF A KW11-L AND/OR A KW11-P CLOCK
544                             ;       OPTION MODULES ARE PART OF THE RUNTIME EXERCISER AND SECONDLY,
545                             ;       WHETHER THE CLOCK HARDWARE IS PART OF THE SYSTEM CONFIGURATION.
546                             ;
547                             ;       ONCE A CLOCK HAS BEEN DETERMINED TO BE AVAILABLE THEN THE APPROPRIATE
548                             ;       CLOCK CODE IS SET IN THE SYSTEM CLOCK INDICATOR WORD, THIS INDICATES
549                             ;       THE TYPE OF SYSTEM CLOCK, AND ALSO SYSTEM CLOCK PRESENT BIT IN CONFIGURA-
550                             ;       TION WORD 0 AND FINALLY THE SYSTEM CLOCK MODULE BEGIN ADDRESS
551                             ;       IS PLACED IN THE SYSTEM CLOCK WORD. IF THERE IS NOT A SYSTEM CLOCK
552                             ;       AVAILABLE THEN THE ABOVE WORDS WILL REFLECT THE NO CLOCK CONDITION.
553                             ;
554                             ;       NOTE:   IF THE KW11-L IS AVAILABLE IT WILL BE USED AS THE SYSTEM
555                             ;               CLOCK EVEN IF A KW11-P IS PRESENT.
556                             ;
557                             ;       THIS MODULE WILL BE A ONE TIME ONLY ROUTINE CALLED AT INITIALIZATION
558                             ;       OF DEC/X11.
559                             ;
560                             ; INPUTS:
561                             ;       ADDRESS OF DTABLE
562                             ;
563                             ; IMPLICIT INPUTS:
564                             ;       DT.CF0          - CONFIGURATION WORD 0 LOCATED IN DTABLE
565                             ;       DT.SCT          - SCTAB ADDRESS
566                             ;       SC.CLO          - SYSTEM CLOCK WORD
567                             ;       SC.CKL          - ADDRESS OF POINTER TO KW11-L HEADER ADDRESS
568                             ;       SC.CLP          - ADDRESS OF POINTER TO KW11-P HEADER ADDRESS
569                             ;       SC.SCA          - SYSTEM CLOCK ADDRESS
570                             ;
571                             ; OUTPUTS:
572                             ;       NONE
573                             ;
574                             ; IMPLICIT OUTPUTS:
575                             ;       DT.CF0          - CONFIGURATION WORD 0 LOCATED IN DTABLE
576                             ;       SC.CLO          - SYSTEM CLOCK WORD
577                             ;       SC.SCA          - SYSTEM CLOCK ADDRESS
578                             ;
579                             ; PATHOLOGICAL CONNECTIONS:
580                             ;       NONE
581                             ;
582                             ; SUBORDINATE ROUTINES CALLED:
583                             ;       SAVREG          - SAVE REGISTERS
584                             ;       ENQTQ           - TYPE-QUEUE ENQUEUER
585                             ;       HRDADRCHK       - HARDWARE ADDRESS CHECK
586                             ;       RESREG          - RESTORE REGISTERS
587                             ;
588                             ; FUNCTIONAL SIDE EFFECTS:
```

```
589                             ;       NONE
590                             ;
591                             ; CALLING SEQUENCE:
592                             ;       CALL CLKCHK IN <A>
593                             ;               A=  ADDRESS OF DTABLE
594                             ;
595                             ;       EDIT            DATE            BY              REASON
596                             ;--
597
598
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN  =1
            000001              $LSTTAG =1


                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  CLKCHK          ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  HRDADRCHK       ;HARDWARE ADDRESS CHECK
                                .GLOBL  ENQTQ           ;ENQUEUE INTO TYPE QUEUE
                                .GLOBL  RESREG          ;RESTORE REGISTERS

                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 054523  052123  046505  CC.CM1: .ASCIZ  /SYSTEM CLOCK NOT AVAILABLE%/
000006' 041440  047514  045503
000014' 047040  052117  040440
000022' 040526  046111  041101
000030' 042514  000045
                                .EVEN
```

```
                        .SBTTL  CHECK FOR SYSTEM CLOCK ROUTINE

                        ROUTINE CLKCHK <DTA>
000034'                                                         CLKCHK:
(2)  000034'

                        ;+
                        ; SAVE REGISTERS ON STACK AND RETRIEVE
                        ; ADDRESS OF DTABLE FROM R5 STACK AND ADDRESS OF SCTAB
                        ; FROM DTABLE
                        ;-

                        CALL SAVREG
000034'                                                                 JSR     PC,SAVREG
(3)  000034' 004767  000000G
                        LET  R2  := DTA(R5)
000040'                                                                 MOV     DTA(R5),R2
(4)  000040' 016502  000000
                        LET  R3  := DT.SCT(R2)
000044'                                                                 MOV     DT.SCT(R2),R3
(4)  000044' 016203  000066

                        ;+
                        ; INITIALIZE SYSTEM CLOCK WORD AND SYSTEM CLOCK ADDRESS WORD
                        ;-

                        LET SC.CLO(R3) := #0
000050'                                                                 CLR     SC.CLO(R3)
(4)  000050' 005063  000000
                        LET SC.SCA(R3) := #0
000054'                                                                 CLR     SC.SCA(R3)
(4)  000054' 005063  000012

                        ;+
                        ; DETERMINE IF KW11-L OPTION MODULE
                        ; IS PART OF THE RUNTIME EXERCISER
                        ;-

                        IF SC.CKL(R3) NE #0 THEN
000060'                                                                 TST     SC.CKL(R3)
(6)  000060' 005763  000002                                             BEQ     50002$
(9)  000064' 001423

                        ;+
                        ; GET KW11-L OPTION MODULE HEADER ADDRESS
                        ;-

                                LET R0 := @SC.CKL(R3)
000066'                                                                 MOV     @SC.CKL(R3),R0
(4)  000066' 017300  000002

                        ;+
                        ; GET KW11-L HARDWARE CSR ADDRESS FROM KW11-L
                        ; OPTION MODULE'S HEADER
                        ;-

                                LET R1 := ADR(R0)
000072'                                                                 MOV     ADR(R0),R1
(4)  000072' 016001  000006

                        ;+
                        ; CHECK TO SEE IF KW11-L HARDWARE IS AVAILABLE
                        ;-

                                CALL HRDADRCHK IN <R1>
000076'
```

```
                                                                        MOV     R5,-(SP)
                                                                        MOV     R1,-(R5)
                                                                        JSR     PC,HRDADRCHK
                                                                        MOV     (SP)+,R5
                        ;+
                        ; IF KW11L IS AVAILABLE THEN INDICATE KW11-L IS SYSTEM CLOCK
                        ; IN SYSTEM CLOCK WORD. BIT00 SET MEANS KW11-L IS
                        ; SYSTEM CLOCK.
                        ;-

                                IF.NO.ERROR THEN                        BCS     50003$

                                        LET SC.CLO(R3) := #BIT00        MOV     #BIT00,SC.CLO(R3

                        ;+
                        ; SET SYSTEM CLOCK PRESENT BIT IN CONFIGURATION WORD 0 IN DTABLE
                        ; AND SAVE SYSTEM CLOCK ADDRESS
                        ;-

                                        LET DT.CF0(R2) := DT.CF0(R2) SET.BY #CLKPRES
                                                                        BIS     #CLKPRES,DT.CF0(
                                        LET SC.SCA(R3) := R0            MOV     R0,SC.SCA(R3)

                        ;+
                        ; RETURN
                        ;-

                                        INLINE <BR 100$>                BR 100$

                                ENDIF                                   50003$:

                        ENDIF                                           50002$:


                        ;+
                        ; DETERMINE IF KW11-P OPTION MODULE IS
                        ; PART OF THE RUN TIME EXERCISER
                        ;-

                        IF SC.CKP(R3) NE #0 THEN                        TST     SC.CKP(R3)
                                                                        BEQ     50004$

                        ;+
                        ; GET KW11-P OPTION MODULE HEADER ADDRESS
                        ;-

                                LET R0 := @SC.CKP(R3)                   MOV     @SC.CKP(R3),R0
```

```
                                ;+
                                ; GET KW11-P HARDWARE CSR ADDRESS
                                ; FROM KW11-P OPTION MODULE'S HEADER
                                ;-

                                        LET R1 := ADR(R0)
000146'                                                                 MOV     ADR(R0),R1
(4) 000146' 016001  000006

                                ;+
                                ; CHECK TO SEE IF KW11-P HARDWARE
                                ; IS AVAILABLE
                                ;-

                                        CALL HRDADRCHK IN <R1>
000152'                                                                 MOV     R5,-(SP)
(3) 000152' 010546                                                      MOV     R1,-(R5)
(4) 000154' 010145                                                      JSR     PC,HRDADRCHK
(3) 000156' 004767  000000G                                             MOV     (SP)+,R5
(3) 000162' 012605

                                ;+
                                ; IF ADDRESS WAS AVAILABLE THEN INDICATE KW11-P IS SYSTEM CLOCK
                                ; IN SYSTEM CLOCK WORD
                                ;-

                                        IF.NO.ERROR THEN
000164'                                                                 BCS     50005$
(6) 000164' 103411

                                                LET SC.CLO(R3) := #BIT01
000166'                                                                 MOV     #BIT01,SC.CLO(R3
(4) 000166' 012763  000002  000000

                                ;+
                                ; SET SYSTEM CLOCK PRESENT BIT IN CONFIGURATION WORD 0 IN DTABLE
                                ; AND SAVE SYSTEM CLOCK ADDRESS
                                ;-

                                                LET DT.CF0(R2) := DT.CF0(R2) SET.BY #CLKPRES
000174'                                                                 BIS     #CLKPRES,DT.CF0(
(6) 000174' 052762  000001  000014
                                                LET SC.SCA(R3) := R0
000202'                                                                 MOV     R0,SC.SCA(R3)
(4) 000202' 010063  000012

                                        ELSE
000206'                                                                 BR      50006$
(4) 000206' 000415                                              50005$:
(3) 000210'

                                ;+
                                ; OUTPUT MESSAGE INDICATING NO SYSTEM CLOCK
                                ;-

                                                CALL ENQTQ IN <R2,#MSGSTD,#CC.CM1,#0,#0>
000210'                                                                 MOV     R5,-(SP)
(3) 000210' 010546                                                      MOV     #0,-(R5)
(8) 000212' 012745  000000                                              MOV     #0,-(R5)
(7) 000216' 012745  000000                                              MOV     #CC.CM1,-(R5)
(6) 000222' 012745  000000'                                             MOV     #MSGSTD,-(R5)
(5) 000226' 012745  000000                                              MOV     R2,-(R5)
(4) 000232' 010245
```

```
(3)  000234' 004767  000000G                                                    JSR     PC,ENQTQ
(3)  000240' 012605                                                             MOV     (SP)+,R5
754  000242'                        ENDIF
(4)  000242'                                                            50006$:
755  000242'                    ELSE                                             BR      50007$
(4)  000242' 000415                                                     50004$:
(3)  000244'
756
757                             ;+
758                             ; OUTPUT MESSAGE INDICATING SYSTEM CLOCK IS NOT AVAILABLE
759                             ;-
760
761  000244'                        CALL ENQTQ IN <R2,#0,#CC.CM1,#0,#0>
(3)  000244' 010546                                                             MOV     R5,-(SP)
(8)  000246' 012745  000000                                                     MOV     #0,-(R5)
(7)  000252' 012745  000000                                                     MOV     #0,-(R5)
(6)  000256' 012745  000000'                                                    MOV     #CC.CM1,-(R5)
(5)  000262' 012745  000000                                                     MOV     #0,-(R5)
(4)  000266' 010245                                                             MOV     R2,-(R5)
(3)  000270' 004767  000000G                                                    JSR     PC,ENQTQ
(3)  000274' 012605                                                             MOV     (SP)+,R5
762  000276'                    ENDIF
(4)  000276'                                                            50007$:
763
764  000276'                    INLINE <100$:>                                   100$:
(2)  000276'
765                             ;+
766                             ; RESTORE REGISTERS
767                             ;-
768
769  000276'                    CALL RESREG                                      JSR     PC,RESREG
(3)  000276' 004767  000000G
770
771                             ;+
772                             ; RETURN TO CALLER
773                             ;-
774
775  000302'                    ENDRTN
(3)  000302'                                                            50000$:
(3)  000302'                                                            50001$:
(2)  000302' 000207                                                             RTS     PC
776
777
```

```
                        .SBTTL CLKOFF - SYSTEM CLOCK-OFF
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       CLKOFF
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SHUT OFF THE SYSTEM CLOCK (IF AVAILABLE). THE ROUTINE
                        ;       WILL SAVE THE CLOCK'S CONTROL STATUS REGISTER (CSR) ADDRESS AND IT'S
                        ;       CONTENTS. PLUS TURN THE CLOCK OFF BY CLEARING THE INTERRUPT ENABLE BIT.
                        ;
                        ; INPUTS:
                        ;       DTABLE - ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.SCT - ADDRESS OF SYSTEM CLOCK TABLE
                        ;       DT.CLO - SYSTEM CLOCK WORD
                        ;       SC.CKL - POINTER TO KW11L OPTION MODULE
                        ;       SC.CKP - POINTER TO KW11P OPTION MODULE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       SC.ADR  - SAVE LOCATION FOR CLOCK CSR ADDRESS
                        ;       SC.HLD  - SAVE LOCATION FOR CONTENTS OF CLOCK'S
                        ;                 CSR
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL CLKOFF IN <A>
                        ;               A=  ADDRESS OF DTABLE
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
826
827
828
829
830
831
```

```
.SBTTL COMMON DEFINITIONS AND REFERENCES

;****************
; REFERENCED BY OTHER MODULES
;
.GLOBL  CLKOFF          ;MODULE ENTRY POINT
```

```
                                .SBTTL  CLKOFF ROUTINE

                                ROUTINE CLKOFF <DTA>                                    CLKOFF:

                                ;+
                                ; SAVE REGISTERS ON STACK AND RETRIEVE ADDRESS OF DTABLE AND
                                ; GET ADDRESS OF SYSTEM CLOCK TABLE (SCTAB)
                                ;-

                                PUSH R0,R1                                                      MOV     R0,-(SP)
                                                                                                MOV     R1,-(SP)
                                LET  R1 := DTA(R5)                                              MOV     DTA(R5),R1
                                LET R1 := DT.SCT(R1)                                            MOV     DT.SCT(R1),R1

                                ;+
                                ; IF KW11-L IS SYSTEM CLOCK THEN GET THE KW11-L
                                ; OPTION MODULE HEADER ADDRESS
                                ;-

                                IF #BIT00 SETIN SC.CLO(R1) THEN                                 BIT     #BIT00,SC.CLO(R1
                                                                                                BEQ     50002$
                                        LET R0 := @SC.CKL(R1)                                   MOV     @SC.CKL(R1),R0

                                ELSE                                                            BR      50003$
                                                                                    50002$:

                                ;+
                                ; KW11-P IS SYSTEM CLOCK , GET THE KW11-P
                                ; OPTION MODULE HEADER ADDRESS
                                ;-

                                        LET R0 := @SC.CKP(R1)                                   MOV     @SC.CKP(R1),R0

                                ENDIF                                               50003$:

                                ;+
                                ; GET SYSTEM CLOCK CSR ADDRESS AND SAVE IN
                                ; SC.ADR. R0 CONTAINS THE OPTION
                                ; MODULE HEADER ADDRESS.
                                ;-

                                LET SC.ADR(R1) := ADR(R0)                                       MOV     ADR(R0),SC.ADR(R

                                ;+
```

```
                                ; GET CONTENTS OF SYSTEM CLOCK CSR AND SAVE
                                ; IN SC.HLD
                                ;-

000350'                         LET SC.HLD(R1) := @SC.ADR(R1)
(4) 000350' 017161  000006  000010                                  MOV     @SC.ADR(R1),SC.H
                                ;+
                                ; TURN SYSTEM CLOCK OFF BY CLEARING
                                ; IT'S CSR
                                ;-

000356'                         LET @SC.ADR(R1) := #0
(4) 000356' 005071  000006                                          CLR     @SC.ADR(R1)
                                ;+
                                ; RESTORE REGISTERS
                                ;-

000362'                         POP     R1,R0
(2) 000362' 012601                                                  MOV     (SP)+,R1
(3) 000364' 012600                                                  MOV     (SP)+,R0

                                ;+
                                ; RETURN TO CALLING MODULE
                                ;-

000366'                         ENDRTN
(3) 000366'                                                 50000$:
(3) 000366'                                                 50001$:
(2) 000366' 000207                                                  RTS     PC
```

```
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
```

```
.SBTTL CLKON SYSTEM CLOCK-ON
.IDENT /V0.0/

;++
; MODULE NAME:
;       CLKON
;
; FUNCTIONAL DESCRIPTION:
;       THIS MODULE WILL TURN ON THE SYSTEM CLOCK. THIS WILL BE ACCOMPLISHED
;       BY RESTORING THE CONTENTS OF THE CSR TO IT'S VALUE AT THE TIME
;       THE CLOCK WAS TURNED OFF. THE CONTENTS OF THE CSR IS DEPENDENT UPON
;       THE SYSTEM CLOCK IN USE. THE KW11-L WILL ONLY SET THE INTERRUPT
;       ENABLE BIT WHILE THE KW11-P WILL ALSO SET THE APPROPRIATE MODE AND
;       FREQUENCY BITS.
;
; INPUTS:
;       DTABLE - DATA TABLE ADDRESS
;
; IMPLICIT INPUTS:

;       DT.SCT - ADDRESS OF SYSTEM CLOCK TABLE
;       SC.ADR - STORAGE FOR CLOCK CSR ADDRESS
;       SC.HLD - STORAGE FOR CONTENTS OF CSR

; OUTPUTS:
;       NONE
;
; IMPLICIT OUTPUTS:
;       NONE
;
; PATHOLOGICAL CONNECTIONS:
;       NONE
;
; SUBORDINATE ROUTINES CALLED:
;       NONE
;
; FUNCTIONAL SIDE EFFECTS:
;       NONE
;
; CALLING SEQUENCE:
;       CALL CLKON IN<A>
;               A=   DTABLE ADDRESS
;
;       EDIT            DATE            BY              REASON
;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES


                        ;****************
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  CLKON           ;MODULE ENTRY POINT
```

```
                                .SBTTL  CLKON   ROUTINE

                                ROUTINE CLKON <DTA>
000370'                                                                 CLKON:
(2)  000370'

                                ;+
                                ; SAVE R0 AND RETRIEVE DTABLE FROM R5 AND GET SCTAB INTO R0
                                ; GET THE SYSTEM CLOCK CSR ADDRESS AND
                                ; RESTORE ITS CONTENTS, THIS WILL TURN THE
                                ; CLOCK ON. R0 CONTAINS THE ADDRESS OF SCTAB.
                                ;-

                                PUSH R0
000370'                                                                         MOV     R0,-(SP)
(2)  000370'  010046
                                LET R0 := DTA(R5)
000372'                                                                         MOV     DTA(R5),R0
(4)  000372'  016500  000000
                                LET R0 := DT.SCT(R0)
000376'                                                                         MOV     DT.SCT(R0),R0
(4)  000376'  016000  000066
                                LET  @SC.ADR(R0) := SC.HLD(R0)
000402'                                                                         MOV     SC.HLD(R0),@SC.A
(4)  000402'  016070  000010  000006

                                ;+
                                ; RESTORE REGISTER AND RETURN TO CALLER
                                ;-

                                POP R0
000410'                                                                         MOV     (SP)+,R0
(2)  000410'  012600

                                ENDRTN
000412'                                                         50000$:
(3)  000412'                                                    50001$:
(3)  000412'                                                                    RTS     PC
(2)  000412'  000207
```

```
                        .SBTTL  PRHMS  PROCESS HOURS, MINUTES, AND SECONDS CONVERSION
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       PRHMS
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WHEN CALLED WILL GET THE TOTAL RUNTIME IN SECONDS
                        ;       FROM THE SYSTEM CLOCK OPTION MODULE (KW11-L OR KW11-P) HAVE IT
                        ;       CONVERTED TO HOURS, MINUTES AND SECONDS, AND RETURNS
                        ;       THE CONVERTED TIME TO THE CALLER.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF DTABLE
                        ;       ADDRESS OF CONVERTED TIME STORAGE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.SCT
                        ;       SC.CLO
                        ;       SC.CKL
                        ;       SC.CKP
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       HOURS     \
                        ;       MINUTES     ---IN STORAGE SPECIFIED
                        ;       SECONDS   /
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG  - SAVE REGISTERS
                        ;       HMS     - CONVERSION TO HRS, MINS AND SECS
                        ;       RESREG  - RESTORE REGISTERS
                        ;       LDTIME - LOAD RESULTANT TIME
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL PRHMS IN <A,B>
                        ;                  A = ADDRESS OF DTABLE
                        ;                  B = ADDRESS OF CONVERTED TIME
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
1044
1045                            .SBTTL  COMMON DEFINITIONS AND REFERENCES
1046
1047                            ;***************
1048                            ; REFERENCED BY OTHER MODULES
1049                            ;
1050                            .GLOBL  PRHMS                   ;MODULE ENTRY POINT
1051
1052                            ;***************
1053                            ; GLOBAL REFERENCES
1054                            ;
1055                            .GLOBL  SAVREG                  ;SAVE REGISTERS
1056                            .GLOBL  HMS                     ;HOURS, MINUTES AND SECONDS CONVERSION
1057                            .GLOBL  RESREG                  ;RESTORE REGISTERS
```

```
                                .SBTTL  PROCESS HOURS, MINUTES AND SECONDS CONVERSION ROUTINE

                                ROUTINE PRHMS <DTA,RESULT>
000414'                                                                 PRHMS:
(2) 000414'

                                ;+
                                ; SAVE REGISTERS ON STACK, RETRIEVE ADDRESS OF DTABLE FROM
                                ; R5 STACK AND DETERMINE ADDRESS OF SCTAB.
                                ;-

                                PUSH R0,R1
000414'                                                                 MOV     R0,-(SP)
(2) 000414' 010046                                                      MOV     R1,-(SP)
(3) 000416' 010146
                                LET R0 := DTA(R5)
000420'                                                                 MOV     DTA(R5),R0
(4) 000420' 016500  000000
                                LET R0 := DT.SCT(R0)
000424'                                                                 MOV     DT.SCT(R0),R0
(4) 000424' 016000  000066

                                ;+
                                ; DETERMINE IF KW11-L IS SYSTEM CLOCK BY CHECKING IF
                                ; BIT 0 IS SET IN SC.CLO WORD OF SCTAB
                                ;-

                                IF #BIT00 SETIN SC.CLO(R0) THEN
000430'                                                                 BIT     #BIT00,SC.CLO(R0
(6) 000430' 032760  000001  000000                                     BEQ     50002$
(9) 000436' 001405

                                ;+
                                ; GET ADDRESS OF KW11-L ELAPSED TIME TABLE. TIME TABLE IS THE
                                ; ADDRESS FOLLOWING THE LOCATION CONTAINED IN SC.CKL. THE TABLE
                                ; CONTAINS ELAPSED TIME AND EXTENDED ELAPSED TIME.
                                ;-

                                        LET R1 := SC.CKL(R0) + #2
000440'                                                                 MOV     SC.CKL(R0),R1
(4) 000440' 016001  000002                                              ADD     #2,R1
(6) 000444' 062701  000002

                                ELSE
000450'                                                                 BR      50003$
(4) 000450' 000404
(3) 000452'                                                         50002$:

                                ;+
                                ; MUST BE A KW11-P CLOCK. GET ADDRESS OF KW11-P ELAPSED TIME TABLE.
                                ; THE TABLE IS THE ADDRESS FOLLOWING THE LOCATION CONTAINED IN SC.CKP.
                                ; THE TABLE CONTAINS ELAPSED TIME AND EXTENDED ELAPSED TIME.
                                ;-

                                        LET R1 := SC.CKP(R0) + #2
000452'                                                                 MOV     SC.CKP(R0),R1
(4) 000452' 016001  000004                                              ADD     #2,R1
(6) 000456' 062701  000002
```

```
1102
1103  000462'                       ENDIF
 (4)  000462'                                                    50003$:
1104
1105                                ;+
1106                                ; CALL HMS TO CONVERT TIME TO DECIMAL ASCII HOURS.
1107                                ; MINUTES AND SECONDS.
1108                                ;-
1109
1110  000462'                       CALL HMS IN <(R1),2(R1)> OUT <R1>
 (4)  000462' 162705  000002                                            SUB     #1*2,R5
 (3)  000466' 010546                                                    MOV     R5,-(SP)
 (5)  000470' 016145  000002                                            MOV     2(R1),-(R5)
 (4)  000474' 011145                                                    MOV     (R1),-(R5)
 (3)  000476' 004767  000302                                            JSR     PC,HMS
 (3)  000502' 012605                                                    MOV     (SP)+,R5
 (4)  000504' 012501                                                    MOV     (R5)+,R1
1111
1112
1113                                ;+
1114                                ; LOAD RESULT ADDRESS INTO R1 AND NOW LOAD UP THE TIME
1115                                ;-
1116
1117  000506'                       LET R0 := RESULT(R5)
 (4)  000506' 016500  000002                                            MOV     RESULT(R5),R0
1118  000512'                       CALL LDTIME IN <R1,R0>
 (3)  000512' 010546                                                    MOV     R5,-(SP)
 (5)  000514' 010045                                                    MOV     R0,-(R5)
 (4)  000516' 010145                                                    MOV     R1,-(R5)
 (3)  000520' 004767  000010                                            JSR     PC,LDTIME
 (3)  000524' 012605                                                    MOV     (SP)+,R5
1119
1120                                ;+
1121                                ; RESTORE REGISTERS AND RETURN
1122                                ;-
1123
1124  000526'                       POP R1,R0
 (2)  000526' 012601                                                    MOV     (SP)+,R1
 (3)  000530' 012600                                                    MOV     (SP)+,R0
1125
1126  000532'                       ENDRTN
 (3)  000532'                                                    50000$:
 (3)  000532'                                                    50001$:
 (2)  000532' 000207                                                    RTS     PC
1127
```

```
                                .SBTTL LDTIME - LOAD CONVERTED TIME

                                ROUTINE LDTIME <ADR,RESULT>
                                                                        LDTIME:
                                ;+
                                ; GET CALLING MODULE CONVERTED TIME STORAGE ADDRESS, THEN LOAD UP THE
                                ; STORAGE WITH THE CONVERTED TIME AND INCLUDE COLONS(:) IN APPROPRIATE
                                ; LOCATIONS. ADR(R5) WILL CONTAIN THE STORAGE ADDRESS WHILE RESULT(R5) CONTAINS
                                ; THE RESULT ADDRESS
                                ;-

                                PUSH R0,R1
                                                                        MOV     R0,-(SP)
                                                                        MOV     R1,-(SP)
                                LET R0 := RESULT(R5)
                                                                        MOV     RESULT(R5),R0
                                LET R1 := ADR(R5)
                                                                        MOV     ADR(R5),R1
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0)+ :B= #':
                                                                        MOVB    #':,(R0)+
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0)+ :B= #':
                                                                        MOVB    #':,(R0)+
                                LET (R0)+ :B= (R1)+
                                                                        MOVB    (R1)+,(R0)+
                                LET (R0) :B= (R1)
                                                                        MOVB    (R1),(R0)
                                POP R1,R0
                                                                        MOV     (SP)+,R1
                                                                        MOV     (SP)+,R0
                                ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC
```

```
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
```

```
.SBTTL  PRPSCNT   PROCESS PASS COUNT TIME
.IDENT /V0.0/

;++
; MODULE NAME:
;       PRPSCNT
;
; FUNCTIONAL DESCRIPTION:
;       THIS MODULE WILL BE CALLED IF SYSTEM CLOCK IS AVAILABLE.
;       THE PASS COUNT TIME IS DETERMINED BY FIRST SEARCHING THE MODULE TIME TABLE
;       (IN CLOCK OPTION MODULE) AND THEN SUBSTRACTING PASS COUNT TIME
;       FROM THE SYSTEM ELAPSED TIME. THEN THE PASS COUNT TIME WILL BE CON-
;       VERTED TO HOURS, MINUTES AND SECONDS, AND THEN WILL RETURN CONVERTED TIME TO
;       CALLING MODULE.
;               NOTE - THIS MODULE WILL RETURN THE CONVERTED TIME IN THE
;                      FOLLOWING FORMAT:
;                       HHH:MM:SS
;                       THE COLON (:) WILL BE GENERATED BY THIS MODULE.
;
; INPUTS:
;       DTABLE ADDRESS
;       ADDRESS OF MODULE DOING END OF PASS
;       ADDRESS OF STORAGE FOR CONVERTED TIME
;
; IMPLICIT INPUTS:
;       DT.SCT
;       DT.MLST
;       SC.CLO
;       SC.CKL
;       SC.CKP
; OUTPUTS:
;       NONE
;
; IMPLICIT OUTPUTS:
;       CONVERTED TIME IN HOURS, MINUTES AND SECONDS
;                HHH:MM:SS
;
; PATHOLOGICAL CONNECTIONS:
;       NONE
;
; SUBORDINATE ROUTINES CALLED:
;       SAVREG - SAVE REGISTERS
;       HMS    - CONVERT TIME
;       RESREG - RESTORE REGISTERS
;       LDTIME - LOAD RESULTANT TIME
;
; FUNCTIONAL SIDE EFFECTS:
;       NONE
;
; CALLING SEQUENCE:
;       CALL IN <A,B,C>
;                A - ADDRESS OF DTABLE
;                B - ADDRESS OF MODULE DOING END OF PASS
;                C - ADDRESS OF CONVERTED TIME STORAGE
;
```

```
1210                    ; VERSION:
1211                    ;       0.0
1212                    ;
1213                    ;       EDIT            DATE            BY              REASON
1214                    ;--
```

```
1216
1217
1218                            .SBTTL  COMMON DEFINITIONS AND REFERENCES
1219
1220                            ;****************
1221                            ; REFERENCED BY OTHER MODULES
1222                            ;
1223                            .GLOBL  PRPSCNT                 ;MODULE ENTRY POINT
1224
1225                            ;****************
1226                            ; GLOBAL REFERENCES
1227                            ;
1228                            .GLOBL  HMS                     ;HOURS MINUTES AND SECONDS CONVERSION
1229                            .GLOBL  SAVREG                  ;SAVE REGISTERS
1230                            .GLOBL  RESREG                  ;RESTORE REGISTERS
```

```
                                .SBTTL  PROCESS PASS COUNT TIME ROUTINE


000604'                         ROUTINE PRPSCNT <DTA,BGNADR,RESULT>
(2) 000604'                                                             PRPSCNT:


                                ;+
                                ; SAVE REGISTERS, RETRIEVE ADDRESS OF DTABLE FROM R5 STACK, THEN
                                ; DETERMINE PTR TO MODULE LIST AND ADDRESS OF SCTAB
                                ;-

000604'                         CALL SAVREG
(3) 000604' 004767  000000G                                             JSR     PC,SAVREG

000610'                         LET R1 := DTA(R5)
(4) 000610' 016501  000000                                              MOV     DTA(R5),R1
000614'                         LET R0 := DT.MLST(R1)
(4) 000614' 016100  000032                                              MOV     DT.MLST(R1),R0
000620'                         LET R1 := DT.SCT(R1)
(4) 000620' 016101  000066                                              MOV     DT.SCT(R1),R1

                                ;+
                                ; DETERMINE IF KW11-L IS SYSTEM CLOCK BY CHECKING IF BIT 0 IS SET IN SC.CLO
                                ; WORD OF SCTAB
                                ;-

000624'                         IF #BIT00 SETIN SC.CLO(R1) THEN
(6) 000624' 032761  000001  000000                                      BIT     #BIT00,SC.CLO(R1
(9) 000632' 001403                                                      BEQ     50002$
                                ;+
                                ; GET ADDRESS OF KW11-L ELAPSED TIME TABLE
                                ;-


000634'                                 LET R1 := SC.CKL(R1)
(4) 000634' 016101  000002                                              MOV     SC.CKL(R1),R1
000640'                         ELSE
(4) 000640' 000402                                                      BR      50003$
(3) 000642'                                                             50002$:


                                ;+
                                ; MUST BE A KW11-P CLOCK GET ADDRESS OF KW11-P ELAPSED TIME
                                ; TABLE
                                ;-


000642'                                 LET R1 := SC.CKP(R1)
(4) 000642' 016101  000004                                              MOV     SC.CKP(R1),R1
000646'                         ENDIF
(4) 000646'                                                             50003$:


                                ;+
                                ; GET R2 TO POINT TO MODULE TIME TABLE AND R1 TO POINT TO
                                ; R.T.E ELAPSED TIME
```

```
1276
1277
1278  000646'                        ;-
1279  000654'                        LET R2 := R1 + #6
 (4)  000646' 010102                                                    MOV     R1,R2
 (6)  000650' 062702  000006                                            ADD     #6,R2
                                     LET R1 := R1 + #2
 (6)  000654' 062701  000002                                            ADD     #2,R1
1280
1281                                 ;+
1282                                 ; GET ADDRESS OF MODULE TO BE UPDATED FROM R5 STACK
1283                                 ;-
1284
1285  000660'                        LET R3 := BGNADR(R5)
 (4)  000660' 016503  000002                                            MOV     BGNADR(R5),R3
1286
1287                                 ;+
1288                                 ; SEARCH MODULE LIST UNTIL MODULE LIST POINTER POINTS TO ADDRESS
1289                                 ; IN R3, IF NOT EQUAL INCREMENT MODULE LIST POINTER AND MODULE
1290                                 ; TIME POINTER
1291                                 ;-
1292
1293  000664'                        WHILE (R0) NE R3 DO
 (4)  000664'                                                       50004$:
 (6)  000664' 021003                                                    CMP     (R0),R3
 (9)  000666' 001405                                                    BEQ     50005$
1294  000670'                               LET R0 := R0 + #2
 (6)  000670' 062700  000002                                            ADD     #2,R0
1295  000674'                               LET R2 := R2 + #4
 (6)  000674' 062702  000004                                            ADD     #4,R2
1296  000700'                        ENDDO
 (4)  000700' 000771                                                    BR      50004$
 (3)  000702'                                                       50005$:
1297
1298                                 ;+
1299                                 ; GET ELAPSED TIME AND SUBTRACT MODULES ELAPSED TIME TO DETERMINE
1300                                 ; MODULE'S PASS TIME
1301                                 ;-
1302
1303  000702'                        LET R1 := (R1) - (R2)
 (4)  000702' 011101                                                    MOV     (R1),R1
 (6)  000704' 161201                                                    SUB     (R2),R1
1304
1305                                 ;+
1306                                 ; NOW CALL HMS TO CONVERT THE PASS TIME, EXTENDED TIME WILL
1307                                 ; BE 0.
1308                                 ;-
1309
1310  000706'                        CALL HMS IN <R1,#0> OUT <R1>
 (4)  000706' 162705  000002                                            SUB     #1*2,R5
 (3)  000712' 010546                                                    MOV     R5,-(SP)
 (5)  000714' 012745  000000                                            MOV     #0,-(R5)
 (4)  000720' 010145                                                    MOV     R1,-(R5)
 (3)  000722' 004767  000056                                            JSR     PC,HMS
 (3)  000726' 012605                                                    MOV     (SP)+,R5
 (4)  000730' 012501                                                    MOV     (R5)+,R1
1311
```

```
1312                                    ;+
1313                                    ; LOAD THE RESULT CONVERSION STORAGE AND ALSO LOAD THE TIME
1314                                    ;-
1315
1316  000732'                           LET R0 := RESULT(R5)
 (4)  000732' 016500  000004                                        MOV     RESULT(R5),R0
1317  000736'                           CALL LDTIME IN <R1,R0>
 (3)  000736' 010546                                                MOV     R5,-(SP)
 (5)  000740' 010045                                                MOV     R0,-(R5)
 (4)  000742' 010145                                                MOV     R1,-(R5)
 (3)  000744' 004767  177564                                        JSR     PC,LDTIME
 (3)  000750' 012605                                                MOV     (SP)+,R5
1318
1319                                    ;+
1320                                    ; RESTORE REGISTERS AND RETURN TO CALLER
1321                                    ;-
1322
1323  000752'                           CALL RESREG
 (3)  000752' 004767  000000G                                       JSR     PC,RESREG
1324
1325  000756'                           ENDRTN
 (3)  000756'                                                   50000$:
 (3)  000756'                                                   50001$:
 (2)  000756' 000207                                                RTS     PC
1326
```

```
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
```

```
.SBTTL  HMS    HOURS, MINUTES, AND SECONDS CONVERSION
.IDENT /V0.0/

;++
; MODULE NAME:
;       HMS
;
; FUNCTIONAL DESCRIPTION:
;       THIS MODULE WILL CONVERT THE GIVEN TIME IN SECONDS INTO
;       DECIMAL ASCII HOURS, MINUTES, AND SECONDS. THEN LOAD THE INFORMATION
;       INTO A TABLE AND THE ADDRESS OF THIS TABLE IS RETURNED TO THE CALLER
;                  CONVERTED TIME TABLE:
;                   3 BYTES DEFINED FOR HOURS
;                   2 BYTES DEFINED FOR MINUTES
;                   2 BYTES DEFINED FOR SECONDS
;
;       NOTE: MAXIMUM NUMBER OF HOURS THAT WILL BE CONVERTED WILL
;             BE 999 DECIMAL.
;
; INPUTS:
;       ELAPSED TIME
;       EXTENDED ELAPSED TIME
;
; IMPLICIT INPUTS:
;       NONE
;
; OUTPUTS:
;       ADDRESS OF CONVERTED TIME TABLE
;
; IMPLICIT OUTPUTS:
;       HOURS     \
;       MINUTES    ---CONVERTED TIME TABLE
;       SECONDS   /
;
; PATHOLOGICAL CONNECTIONS:
;       NONE
;
; SUBORDINATE ROUTINES CALLED:
;       SAVREG - SAVE REGISTERS
;       BDACNV - CONVERSION TO DECIMAL ASCII
;       RESREG - RESTORE REGISTERS
;
; FUNCTIONAL SIDE EFFECTS:
;       NONE
;
; CALLING SEQUENCE:
;       CALL HMS IN <A,B> OUT <C>
;                  A - ELAPSED TIME
;                  B - EXTENDED ELAPSED TIME
;                  C - ADDRESS OF CONVERTED TIME TABLE
;
; VERSION:
;       0.0
;
;       EDIT            DATE            BY              REASON
;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  HMS             ;MODULE ENTRY POINT

                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG          ;SAVE REGISTERS
                                .GLOBL  BDACNV          ;BINARY TO DECIMAL ASCII CONVERSION
                                .GLOBL  RESREG          ;RESTORE REGISTERS

                                ;****************
                                ; LOCAL STORAGE
                                ;
000760'                         HM.CTT  :               ;CONVERTED TIME TABLE
000760' 000003                  HM.HRS  :.BLKB 3        ;       HOURS
000763' 000002                  HM.MIN  :.BLKB 2        ;       MINUTES
000765' 000002                  HM.SEC  :.BLKB 2        ;       SECONDS
        000770'                 .EVEN
000770'                         HM.CTE  :               ;END OF CONVERTED TIME TABLE
000770' 000000                  HM.THR  :.WORD 0        ;TOTAL TIME IN HOURS STORAGE
000772' 000000                  HM.TMI  :.WORD 0        ;TOTAL TIME IN MINUTES STORAGE
000774' 000000                  HM.TSE  :.WORD 0        ;TOTAL TIME IN SECONDS STORAGE

000776' 000005                  HM.CNV  :.BLKB 5        ;STORAGE FOR BDACNV CALL
        001004'                 .EVEN
```

```
                                .SBTTL HOURS, MINUTES AND SECONDS CONVERSION ROUTINE


001004'                         ROUTINE HMS <TIME,XTIME,ADRCT>
001004'                                                                 HMS:

                                ;+
                                ; SAVE REGISTERS ON STACK, RETRIEVE TIME (R0) AND EXTENDED TIME (R1) FROM STACK,
                                ; INITIALIZE LOCAL STORAGE
                                ;-

001004'                         CALL SAVREG
001004' 004767  000000G                                                 JSR     PC,SAVREG
001010'                         LET R0 := TIME(R5)
001010' 016500  000000                                                  MOV     TIME(R5),R0
001014'                         LET R1 := XTIME(R5)
001014' 016501  000002                                                  MOV     XTIME(R5),R1
001020'                         LET HM.THR := #0
001020' 005067  177744                                                  CLR     HM.THR
001024'                         LET HM.TMI := #0
001024' 005067  177742                                                  CLR     HM.TMI
001030'                         LET HM.TSE := #0
001030' 005067  177740                                                  CLR     HM.TSE

                                ;+
                                ; PROCESS (EXTENDED TIME BITS) INTO HOURS
                                ;-

001034'                         WHILE R1 NE #0 DO
001034'                                                         50002$:
001034' 005701                                                          TST     R1
001036' 001417                                                          BEQ     50003$

                                ;+
                                ; 18 HRS, 12 MIN AND 16 SEC REQUIRED FOR EXTIME TO BE INCREMENTED
                                ; IF TIME IS LESS THAN 1 HOUR THEN ADD 18 HOURS TO HOURS AND
                                ; ADD 12 MINUTES 16 SECONDS TO R0 (TIME)
                                ;-

001040'                                 IF R0 LO #^D3600 THEN
001040' 020027  007020                                                  CMP     R0,#^D3600
001044' 103007                                                          BHIS    50004$
001046'                                         LET HM.THR := HM.THR + #^D18
001046' 062767  000022  177714                                          ADD     #^D18,HM.THR
001054'                                         LET R0  := R0 + #^D736
001054' 062700  001340                                                  ADD     #^D736,R0
001060'                                         LET R1  := R1 - #1
001060' 005301                                                          DEC     R1
001062'                                 ELSE
001062' 000404                                                          BR      50005$
001064'                                                         50004$:
001064'                                         LET HM.THR := HM.THR + #1
001064' 005267  177700                                                  INC     HM.THR
001070'                                         LET R0  := R0 - #^D3600
001070' 162700  007020                                                  SUB     #^D3600,R0
```

```
1453  001074'                                ;+
 (4)  001074'                                ; PROCESS REMAINING HOURS CONTAINED IN R0 (TIME).
1454  001074'                                ;-
 (4)  001074' 000757                        WHILE R0 HIS #^D3600 DO
 (3)  001076'
1455                                                                    50006$:
1456                                                                            CMP     R0,#^D3600
1457                                                                            BLO     50007$
1458                                                LET HM.THR := HM.THR + #1
1459                                                                            INC     HM.THR
1460  001076'                                       LET R0    := R0 - #^D3600
 (4)  001076'                                                                   SUB     #^D3600,R0
 (6)  001076' 020027  007020               ENDDO
 (9)  001102' 103405                                                            BR      50006$
1461  001104'                                                           50007$:
 (6)  001104' 005267  177660
1462  001110'
 (6)  001110' 162700  007020               ;+
1463  001114'                                ; PROCESS MINUTES FROM REMAINING R0 (TIME)
 (4)  001114' 000770                        ;-
 (3)  001116'
1464                                        WHILE R0 HIS #^D60 DO
1465                                                                    50010$:
1466                                                                            CMP     R0,#^D60
1467                                                                            BLO     50011$
1468                                                LET HM.TMI := HM.TMI + #1
1469  001116'                                                                  INC     HM.TMI
 (4)  001116'                                       LET R0  := R0 - #^D60
 (6)  001116' 020027  000074                                                    SUB     #^D60,R0
 (9)  001122' 103405               ENDDO
1470  001124'                                                                  BR      50010$
 (6)  001124' 005267  177642                                               50011$:
1471  001130'
 (6)  001130' 162700  000074               ;+
1472  001134'                                ; TIME LEFT STICK IN HM.TSE
 (4)  001134' 000770                        ;-
 (3)  001136'
1473                                        LET HM.TSE := R0
1474                                                                            MOV     R0,HM.TSE
1475
1476                                        ;+
1477                                        ; CONVERT HOURS TO DECIMAL ASCII
1478  001136'                               ;-
 (4)  001136' 010067  177632
1479                                        CALL BDACNV IN <HM.THR,#HM.CNV>
1480                                                                            MOV     R5,-(SP)
1481                                                                            MOV     #HM.CNV,-(R5)
1482                                                                            MOV     HM.THR,-(R5)
1483                                                                            JSR     PC,BDACNV
1484  001142'                                                                   MOV     (SP)+,R5
 (3)  001142' 010546
 (5)  001144' 012745  000776'
 (4)  001150' 016745  177614
 (3)  001154' 004767  000000G
 (3)  001160' 012605
1485
```

```
1486                                    ;+
1487                                    ; INITIALIZE R2 TO POINT TO HM.CNV TABLE + 2 AND R3 TO POINT TO HM.HRS.
1488                                    ; THEN MOVE LOWEST THREE BYTES IN HM.CNV TABLE INTO HM.HRS STORAGE IN HM.CTT.
1489                                    ;-
1490
1491  001162'                           LET R2 := #HM.CNV + #2
 (4)  001162' 012702  000776'                                       MOV     #HM.CNV,R2
 (6)  001166' 062702  000002                                       ADD     #2,R2
1492  001172'                           LET R3 := #HM.HRS
 (4)  001172' 012703  000760'                                       MOV     #HM.HRS,R3
1493  001176'                           LET (R3)+ :B= (R2)+
 (4)  001176' 112223                                                MOVB    (R2)+,(R3)+
1494  001200'                           LET (R3)+ :B= (R2)+
 (4)  001200' 112223                                                MOVB    (R2)+,(R3)+
1495  001202'                           LET (R3)+ :B= (R2)+
 (4)  001202' 112223                                                MOVB    (R2)+,(R3)+
1496
1497                                    ;+
1498                                    ; CONVERT MINUTES TO DECIMAL ASCII
1499                                    ;-
1500
1501  001204'                           CALL BDACNV IN <HM.TMI,#HM.CNV>
 (3)  001204' 010546                                                MOV     R5,-(SP)
 (5)  001206' 012745  000776'                                       MOV     #HM.CNV,-(R5)
 (4)  001212' 016745  177554                                        MOV     HM.TMI,-(R5)
 (3)  001216' 004767  000000G                                       JSR     PC,BDACNV
 (3)  001222' 012605                                                MOV     (SP)+,R5
1502
1503                                    ;+
1504                                    ; INITIALIZE R2 TO POINT TO HM.CNV TABLE + 3, R3 TO POINT TO HM.MIN,
1505                                    ; MOVE 2 ENTRIES IN HM.CNV TABLE INTO HM.MIN STORAGE IN HM.CTT.
1506                                    ;-
1507
1508  001224'                           LET R2 := #HM.CNV + #3
 (4)  001224' 012702  000776'                                       MOV     #HM.CNV,R2
 (6)  001230' 062702  000003                                       ADD     #3,R2
1509  001234'                           LET R3 := #HM.MIN
 (4)  001234' 012703  000763'                                       MOV     #HM.MIN,R3
1510  001240'                           LET (R3)+ :B= (R2)+
 (4)  001240' 112223                                                MOVB    (R2)+,(R3)+
1511  001242'                           LET (R3)+ :B= (R2)+
 (4)  001242' 112223                                                MOVB    (R2)+,(R3)+
1512
1513                                    ;+
1514                                    ; FINALLY CONVERT SECONDS TO DECIMAL ASCII
1515                                    ;-
1516
1517  001244'                           CALL BDACNV IN <HM.TSE,#HM.CNV>
 (3)  001244' 010546                                                MOV     R5,-(SP)
 (5)  001246' 012745  000776'                                       MOV     #HM.CNV,-(R5)
 (4)  001252' 016745  177516                                        MOV     HM.TSE,-(R5)
 (3)  001256' 004767  000000G                                       JSR     PC,BDACNV
 (3)  001262' 012605                                                MOV     (SP)+,R5
1518
1519                                    ;+
1520                                    ; INITIALIZE R2 TO POINT TO END OF HM.CNV TABLE + 3, R3 TO POINT
```

```
                        ; TO HM.SEC. MOVE 2 ENTRIES IN HM.CNV TABLE INTO HM.SEC
                        ; STORAGE IN HM.CTT
                        ;-

                        LET R2 := #HM.CNV + #3
001264'                                                         MOV     #HM.CNV,R2
(4) 001264' 012702  000776'                                     ADD     #3,R2
(6) 001270' 062702  000003
                        LET R3 := #HM.SEC
001274'                                                         MOV     #HM.SEC,R3
(4) 001274' 012703  000765'
                        LET (R3)+ :B= (R2)+
001300'                                                         MOVB    (R2)+,(R3)+
(4) 001300' 112223
                        LET (R3)+ :B= (R2)+
001302'                                                         MOVB    (R2)+,(R3)+
(4) 001302' 112223

                        ;+
                        ; LOAD HM.CTT ADDRESS (CONVERTED TIME TABLE) ON R5 STACK FOR
                        ; CALLER
                        ;-

                        LET ADRCT(R5) := #HM.CTT
001304'                                                         MOV     #HM.CTT,ADRCT(R5
(4) 001304' 012765  000760' 000004

                        ;+
                        ; RESTORE REGISTERS AND RETURN TO CALLER
                        ;-

                        CALL RESREG
001312'                                                         JSR     PC,RESREG
(3) 001312' 004767  000000G

                        ENDRTN
001316'                                                 50000$:
(3) 001316'                                             50001$:
(3) 001316'                                                     RTS     PC
(2) 001316' 000207
```

```
1547                            .SBTTL UPDTIM UPDATE MODULE PASS TIME
1548                            .IDENT /V0.0/
1549
1550                            ;++
1551                            ; MODULE NAME:
1552                            ;       UPDTIM
1553                            ;
1554                            ; FUNCTIONAL DESCRIPTION:
1555                            ;       THIS MODULE WILL ONLY BE CALLED IF THERE IS A SYSTEM CLOCK AVAILABLE
1556                            ;       (KW11-L OR KW11-P OPTION MODULE). UPON BEING CALLED IT WILL SEARCH THE MODULE
1557                            ;       LIST UNTIL THE MODULE IS FOUND THAT REQUIRES THE MODULE TIME TO BE UPDATED.
1558                            ;       THE UPDATE IS DONE BY MOVING THE TOTAL ELAPSED TIME AND EXTENDED TIME
1559                            ;       INTO THE APPROPRIATE STORAGE LOCATIONS (CONTAINED IN SYSTEM CLOCK MODULE;
1560                            ;       KW11-L OR KW11-P) FOR THE MODULE BEING UPDATED.
1561                            ;
1562                            ; INPUTS:
1563                            ;       DTABLE ADDRESS
1564                            ;       MODULE BEGIN ADDRESS TO BE UPDATED
1565                            ;
1566                            ; IMPLICIT INPUTS:
1567                            ;       DT.MLST  OFFSET TO MODULE LIST ADDR. IN DTABLE
1568                            ;       DT.SCT   OFFSET TO SCTAB ADDR. IN DTABLE
1569                            ;       SC.CLO
1570                            ;       SC.CKL
1571                            ;       SC.CKP
1572                            ;
1573                            ; OUTPUTS:
1574                            ;       NONE
1575                            ;
1576                            ; IMPLICIT OUTPUTS:
1577                            ;       NONE
1578                            ;
1579                            ; PATHOLOGICAL CONNECTIONS:
1580                            ;       NONE
1581                            ;
1582                            ; SUBORDINATE ROUTINES CALLED:
1583                            ;       SAVREG  - SAVE REGISTERS
1584                            ;       RESREG  - RESTORE REGISTERS
1585                            ;
1586                            ; FUNCTIONAL SIDE EFFECTS:
1587                            ;       NONE
1588                            ;
1589                            ; CALLING SEQUENCE:
1590                            ;       CALL UPDTIM IN <A,B>
1591                            ;                 A - DTABLE ADDRESS
1592                            ;                 B - MODULE BEGIN ADDR. TO BE UPDATED
1593                            ;
1594                            ; VERSION:
1595                            ;       0.0
1596                            ;
1597                            ;       EDIT            DATE            BY              REASON
1598                            ;--
```

```
1600
1601
1602                            .SBTTL COMMON DEFINITIONS AND REFERENCES
1603
1604                            ;***************
1605                            ; REFERENCED BY OTHER MODULES
1606                            ;
1607                            .GLOBL  UPDTIM                  ;MODULE ENTRY POINT
1608
1609                            ;***************
1610                            ; GLOBAL REFERENCES
1611                            ;
1612                            .GLOBL  SAVREG                  ;SAVE REGISTERS
1613                            .GLOBL  RESREG                  ;RESTORE REGISTERS
```

```
                                .SBTTL UPDATE TIME ROUTINE

001320'                         ROUTINE UPDTIME <DTA,UPDMOD>                    UPDTIME:
001320'

                                ;+
                                ; SAVE REGISTERS, GET POINTERS TO MODULE LIST, R.T.E TIME AND
                                ; MODULE TIME TABLE
                                ;-

001320'                         CALL SAVREG                                     JSR     PC,SAVREG
001320' 004767  000000G
001324'                         LET R0 := DTA(R5)                               MOV     DTA(R5),R0
001324' 016500  000000
001330'                         LET R1 := DT.MLST(R0)                           MOV     DT.MLST(R0),R1
001330' 016001  000032
001334'                         LET R2 := DT.SCT(R0)                            MOV     DT.SCT(R0),R2
001334' 016002  000066

                                ;+
                                ; DETERMINE IF SYSTEM CLOCK IS KW11L OR KW11P AND
                                ; GET ADDRESS WITHIN MODULE (CLOCK)
                                ;-
001340'                         IF #BIT00 SETIN SC.CLO(R2) THEN                 BIT     #BIT00,SC.CLO(R2
001340' 032762  000001  000000                                                  BEQ     50002$
001346' 001403
001350'                                 LET R2 := SC.CKL(R2)                    MOV     SC.CKL(R2),R2
001350' 016202  000002
001354'                         ELSE                                            BR      50003$
001354' 000402
001356'                                                                 50002$:
001356'                                 LET R2 := SC.CKP(R2)                    MOV     SC.CKP(R2),R2
001356' 016202  000004
001362'                         ENDIF                                   50003$:
001362'

                                ;+
                                ; GET R3 TO POINT TO MODULE TIME TABLE AND R2
                                ; TO POINT TO R.T.E. ELAPSED TIME
                                ;-

001362'                         LET R3 := R2 + #6                               MOV     R2,R3
001362' 010203                                                                  ADD     #6,R3
001364' 062703  000006
001370'                         LET R2 := R2 + #2                               ADD     #2,R2
001370' 062702  000002

                                ;+
                                ; GET ADDRESS OF MODULE TO BE UPDATED FROM R5 STACK
                                ;-

001374'                         LET R0 := UPDMOD(R5)                            MOV     UPDMOD(R5),R0
001374' 016500  000002
```

```
                                ;+
                                ; SEARCH MODULE LIST UNTIL MODULE LIST POINTER POINTS TO ADDRESS
                                ; IN R0
                                ;-

001400'                         WHILE (R1) NE R0 DO
                                                                50004$:
(4)  001400'                                                            CMP     (R1),R0
(6)  001400' 021100                                                     BEQ     50005$
(9)  001402' 001405
001404'                                 LET R1 := R1 + #2
(6)  001404' 062701  000002                                             ADD     #2,R1
001410'                                 LET R3 := R3 + #4
(6)  001410' 062703  000004                                             ADD     #4,R3
001414'                         ENDDO
(4)  001414' 000771                                                     BR      50004$
(3)  001416'                                                    50005$:

                                ;+
                                ; HAVE FOUND MODULE UPDATE MODULE TIME TO REFLECT
                                ; R.T.E ELAPSED TIME
                                ;-

001416'                         LET (R3) := (R2)
(4)  001416' 011213                                                     MOV     (R2),(R3)
001420'                         LET 2(R3) := 2(R2)
(4)  001420' 016263  000002  000002                                     MOV     2(R2),2(R3)

                                ;+
                                ; RESTORE STACK AND RETURN
                                ;-

001426'                         CALL RESREG
(3)  001426' 004767  000000G                                           JSR     PC,RESREG

001432'                         ENDRTN
(3)  001432'                                                    50000$:
(3)  001432'                                                    50001$:
(2)  001432' 000207                                                     RTS     PC
```

```
1683                    .SBTTL CLRTIM CLEAR TIME FROM SYSTEM CLOCK TIME TABLES
1684                    .IDENT /V0.0/
1685
1686                    ;++
1687                    ; MODULE NAME:
1688                    ;       CLRTIM
1689                    ;
1690                    ; FUNCTIONAL DESCRIPTION:
1691                    ;       THIS MODULE WILL CALL THE APPROPRIATE ROUTINE CONTAINED IN THE SYSTEM
1692                    ;       CLOCK MODULE (KW11L OR KW11P OPTION MODULE) WHICH WILL CLEAR OUT THE R.T.E
1693                    ;       AND EXTENDED ELAPSED TIME WORDS AND ALSO THE MODULE TIME TABLE.
1694                    ;
1695                    ; INPUTS:
1696                    ;       DTABLE ADDRESS
1697                    ;
1698                    ; IMPLICIT INPUTS:
1699                    ;       DT.SCT - SYSTEM CLOCK TABLE ADDRESS
1700                    ;       SC.CLO
1701                    ;       SC.ALC
1702                    ;       SC.APC
1703                    ;
1704                    ; OUTPUTS:
1705                    ;       NONE
1706                    ;
1707                    ; IMPLICIT OUTPUTS:
1708                    ;       NONE
1709                    ;
1710                    ; PATHOLOGICAL CONNECTIONS:
1711                    ;       NONE
1712                    ;
1713                    ; SUBORDINATE ROUTINES CALLED:
1714                    ;       NONE
1715                    ;
1716                    ; FUNCTIONAL SIDE EFFECTS:
1717                    ;       NONE
1718                    ;
1719                    ; CALLING SEQUENCE:
1720                    ;       CALL CLRTIM IN <A>
1721                    ;                       A - DTABLE ADDRESS
1722                    ;
1723                    ; VERSION:
1724                    ;       0.0
1725                    ;
1726                    ;       EDIT            DATE            BY              REASON
1727                    ;--
```

```
1729
1730
1731                            .SBTTL COMMON DEFINITIONS AND REFERENCES
1732
1733
1734                            ;****************
1735                            ; REFERENCED BY OTHER MODULES
1736                            ;
1737                            .GLOBL  CLRTIM                  ;MODULE ENTRY POINT
```

```
                                .SBTTL CLEAR TIME ROUTINE

001434'                         ROUTINE CLRTIM <DTA>                    CLRTIM:
001434'

                                ;+
                                ; SAVE REGISTER, GET ADDRESS OF SCTAB
                                ;-

001434'                         PUSH R0                                         MOV     R0,-(SP)
001434' 010046
001436'                         LET R0 := DTA(R5)                               MOV     DTA(R5),R0
001436' 016500  000000
001442'                         LET R0 := DT.SCT(R0)                            MOV     DT.SCT(R0),R0
001442' 016000  000066

                                ;+
                                ; DETERMINE IF KW11L IS SYSTEM CLOCK, IF IT IS
                                ; GO CLEAR KW11L CLOCK TIME TABLES.
                                ;-

001446'                         IF #BIT00 SETIN SC.CLO(R0) THEN                 BIT     #BIT00,SC.CLO(R0
001446' 032760  000001  000000                                                  BEQ     50002$
001454' 001403

001456'                                 CALL @SC.ALC(R0)                        JSR     PC,@SC.ALC(R0)
001456' 004770  000014

001462'                         ELSE                                            BR      50003$
001462' 000402                                                          50002$:
001464'

                                ;+
                                ; MUST BE KW11P. GO CLEAR KW11P TIME TABLES
                                ;-

001464'                                 CALL @SC.APC(R0)                        JSR     PC,@SC.APC(R0)
001464' 004770  000016

001470'                         ENDIF                                   50003$:
001470'

                                ;+
                                ; RESTORE REGISTER AND RETURN
                                ;-

001470'                         POP R0                                          MOV     (SP)+,R0
001470' 012600

001472'                         ENDRTN                                  50000$:
001472'                                                                 50001$:
001472'                                                                         RTS     PC
001472' 000207
```

```
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
```

```
.SBTTL CKHUNG CHECK FOR HUNG OPTION MODULES
.IDENT /V0.0/

;++
; MODULE NAME:
;       CKHUNG
;
; FUNCTIONAL DESCRIPTION:
;       THIS MODULE WILL CHECK TO SEE IF ANY IOMOD, IOMODP, IOMODR OR IOMODX
;       HAVE NOT COMPLETED AN END OF PASS IN A SPECIFIED PERIOD OF TIME.
;       THE CRITERIA WILL BE CHECKED BY COMPARING THE TIME OF THE MODULES LAST
;       END OF PASS AGAINST THE CURRENT ELAPSED TIME OF THE SYSTEM RUN. THE MODULE TIME
;       IS CONTAINED IN A TABLE WHICH CONTAINS TWO ENTRIES PER MODULE (TIME AND EXTENDED
;       TIME WORDS), THIS TABLE IS LOCATED IN EITHER THE KW11-L OR KW11-P
;       OPTION MODULE DEPENDING ON WHICH IS THE SYSTEM CLOCK. IF A MODULE
;       IS DETERMINED TO BE HUNG THEN THE MODULE IS DROPPED WITH AN APPROPRIATE
;       MESSAGE OUTPUTTED.
;
; INPUTS:
;       NONE
;
; IMPLICIT INPUTS:
;       DT.MTIME
;
; OUTPUTS:
;       NONE
;
; IMPLICIT OUTPUTS:
;       NONE
;
; PATHOLOGICAL CONNECTIONS:
;       DTABLE
;
; SUBORDINATE ROUTINES CALLED:
;       SAVREG    SAVE REGISTERS
;       ENQTQ     ENQUEUE TYPE QUEUE ROUTINE
;       RESREG    RESTORE REGISTERS
;       DUNCHK    CHECK TO SEE IF ALL MODULES DROPPED
;
; FUNCTIONAL SIDE EFFECTS:
;       NONE
;
; CALLING SEQUENCE:
;       CALL CKHUNG
;
; VERSION:
;       0.0
;
;       EDIT            DATE            BY              REASON
;--
```

```
1832
1833
1834                            .SBTTL COMMON DEFINITIONS AND REFERENCES
1835
1836                            ;****************
1837                            ; REFERENCED BY OTHER MODULES
1838                            ;
1839                            .GLOBL  CKHUNG                  ;MODULE ENTRY POINT
1840
1841                            ;****************
1842                            ; GLOBAL REFERENCES
1843                            ;
1844                            .GLOBL  SAVREG                  ;SAVE REGISTERS
1845                            .GLOBL  ENQTQ                   ;ENQUEUE TYPE QUEUE ROUTINE
1846                            .GLOBL  RESREG                  ;RESTORE REGISTERS
1847                            .GLOBL  DUNCHK                  ;CHECK TO SEE IF ALL MODULES ARE DROPPED
1848                            .GLOBL  DTABLE                  ;ADDRESS OF DATA TABLE
1849
1850                            ;****************
1851                            ; LOCAL STORAGE
1852                            ;
1853  001474' 000000            CK.TEMP: .WORD 0                ;TEMP STORAGE FOR TIME COMPARSIONS
```

```
                                .SBTTL CHECK MODULE HUNG ROUTINE


                                ROUTINE CKHUNG                                  CKHUNG:

                                ;+
                                ; SAVE REGISTERS, GET ADDRESSO OF DTABLE AND  SCTAB
                                ;-

                                CALL SAVREG                                             JSR     PC,SAVREG

                                LET R0 := #DTABLE                                       MOV     #DTABLE,R0

                                LET R1 := DT.SCT(R0)                                    MOV     DT.SCT(R0),R1


                                ;+
                                ; DETERMINE IF KW11L IS SYSTEM CLOCK
                                ;-

                                IF #BIT00 SETIN SC.CLO(R1) THEN                         BIT     #BIT00,SC.CLO(R1
                                                                                        BEQ     50002$


                                ;+
                                ; SET R1 EQUAL TO KW11L LABEL IN OPTION MODULE
                                ;-

                                        LET R1 := SC.CKL(R1)                            MOV     SC.CKL(R1),R1


                                ;+
                                ; SET R1 EQUAL TO KW11P LABEL IN OPTION MODULE
                                ;-

                                ELSE                                                    BR      50003$
                                                                                50002$:
                                        LET R1 := SC.CKP(R1)                            MOV     SC.CKP(R1),R1

                                ENDIF                                           50003$:



                                ;+
                                ; POINT R3 TO FIRST ENTRY IN MODULE LIST,
                                ; R2 POINTS TO FIRST ENTRY IN MOD TIME, AND R1 TO R.T.E TIME WORD.
                                ;-
```

```
1900  001534'                       LET R3 := DT.MLST(R0)                                  MOV     DT.MLST(R0),R3
 (4)  001534' 016003  000032
1901  001540'                       LET R2 := R1 + #6                                      MOV     R1,R2
 (4)  001540' 010102                                                                      ADD     #6,R2
 (6)  001542' 062702  000006
1902  001546'                       LET R1 := R1 + #2                                      ADD     #2,R1
 (6)  001546' 062701  000002
1903
1904                                ;+
1905                                ; WHILE R3 IS NOT AT END OF MODULE LIST
1906                                ;-
1907
1908  001552'                       WHILE (R3) NE #ENDLST DO                        50004$:
 (4)  001552'                                                                             CMP     (R3),#ENDLST
 (6)  001552' 021327  000000                                                             BEQ     50005$
 (9)  001556' 001500
1909
1910                                ;+
1911                                ; IF MODULE NOT BKMOD,NBKMOD OR SBKMOD THEN
1912                                ; DETERMINE IF MODULE IS ACTIVE AND NOT DROPPED
1913                                ;-
1914
1915  001560'                               LET R4 := (R3)                                   MOV     (R3),R4
 (4)  001560' 011304
1916
1917                                ;+
1918                                ; MASK OUT UNWANTED BITS
1919                                ;-
1920
1921  001562'                               LET R4 := STAT(R4) CLR.BY #STATBIT               MOV     STAT(R4),R4
 (4)  001562' 016404  000026                                                             BIC     #STATBIT,R4
 (6)  001566' 042704  064757
1922  001572'                               IF #BKMOD NE R4 AND #NBKMOD NE R4 AND #SBKMOD NE R4 THEN
 (6)  001572' 022704  000020                                                             CMP     #BKMOD,R4
 (9)  001576' 001463                                                                     BEQ     50006$
 (6)  001600' 022704  001000                                                             CMP     #NBKMOD,R4
 (9)  001604' 001460                                                                     BEQ     50006$
 (6)  001606' 022704  000000                                                             CMP     #SBKMOD,R4
 (9)  001612' 001455                                                                     BEQ     50006$
1923
1924                                ;+
1925                                ; GET MODULE HEADER ADDRESS
1926                                ;-
1927
1928  001614'                                     LET R4 := (R3)                             MOV     (R3),R4
 (4)  001614' 011304
1929
1930  001616'                                     IF #ACTBIT  SETIN STAT(R4) AND #BIT13 NOTSETIN STAT(R4) THEN
 (6)  001616' 032764  004000  000026                                                    BIT     #ACTBIT,STAT(R4)
 (9)  001624' 001450                                                                     BEQ     50007$
 (6)  001626' 032764  020000  000026                                                    BIT     #BIT13,STAT(R4)
 (9)  001634' 001044                                                                     BNE     50007$
1931
1932
1933                                ;+
1934                                ; DETERMINE WHETHER THE DIFFERENCE OF MODULE TIME AND RTE. TIME
```

```
                        ; IS GREATER THAN THE MAXIMUM ALLOWABLE TIME BETWEEN PASSES
                        ;-
                                        LET CK.TEMP := (R1) - (R2)
001636'                                                         MOV     (R1),CK.TEMP
(4)  001636' 011167  177632                                     SUB     (R2),CK.TEMP
(6)  001642' 161267  177626
                                        IF CK.TEMP HI DT.MTIME(R0) THEN
001646'                                                         CMP     CK.TEMP,DT.MTIME
(6)  001646' 026760  177622  000110                             BLOS    50010$
(9)  001654' 101434

                        ;+
                        ; THEN LOAD MESSAGE WITH MODULE HUNG AND OUTPUT MESSAGE AND DROP
                        ; MODULE
                        ;-
                                        CALL ENQTQ IN <#DTABLE,#MSGHNG,#0,R4,#0>
001656'                                                         MOV     R5,-(SP)
(3)  001656' 010546                                             MOV     #0,-(R5)
(8)  001660' 012745  000000                                     MOV     R4,-(R5)
(7)  001664' 010445                                             MOV     #0,-(R5)
(6)  001666' 012745  000000                                     MOV     #MSGHNG,-(R5)
(5)  001672' 012745  000022                                     MOV     #DTABLE,-(R5)
(4)  001676' 012745  000000G                                    JSR     PC,ENQTQ
(3)  001702' 004767  000000G                                    MOV     (SP)+,R5
(3)  001706' 012605

                        ;+
                        ; CLEAR THE ACTIVE BIT AND SET THE DROPPED BIT IN THE MODULES STATUS WORD
                        ;-

                                        LET STAT(R4) := STAT(R4) SET.BY #BIT13
001710'                                                         BIS     #BIT13,STAT(R4)
(6)  001710' 052764  020000  000026    LET STAT(R4) := STAT(R4) CLR.BY #ACTBIT
001716'                                                         BIC     #ACTBIT,STAT(R4)
(6)  001716' 042764  004000  000026

                        ;+
                        ; NOW SET BIT TO TELL MONITOR THAT A MODULE HAS BEEN DROPPED
                        ;-

                                        LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
001724'                                                         BIS     #CKTIM,DT.ST1(R0
(6)  001724' 052760  100000  000012

                        ;+
                        ; NOW GO SEE IF ANY MODULES ARE STILL RUNNING IF NOT DUNCHK
                        ; WILL DROP THE EXERCISOR.
                        ;-

                                        CALL DUNCHK IN <R0,R4>
001732'                                                         MOV     R5,-(SP)
(3)  001732' 010546                                             MOV     R4,-(R5)
(5)  001734' 010445                                             MOV     R0,-(R5)
(4)  001736' 010045                                             JSR     PC,DUNCHK
(3)  001740' 004767  000000G                                    MOV     (SP)+,R5
(3)  001744' 012605
                                        ENDIF
001746'                                                 50010$:
(4)  001746'
                                    ENDIF
001746'                                                 50007$:
(4)  001746'
```

```
1969  001746'                                ENDIF                                    50006$:
 (4)  001746'
1970
1971                                         ;+
1972                                         ; UPDATE MODULE LIST POINTER AND MODULE TIME POINTER
1973                                         ;-
1974
1975  001746'                                LET R3 := R3 + #2                                ADD     #2,R3
 (6)  001746' 062703  000002
1976  001752'                                LET R2 := R2 + #4                                ADD     #4,R2
 (6)  001752' 062702  000004
1977  001756'                                ENDDO                                            BR      50004$
 (4)  001756' 000675                                                                     50005$:
 (3)  001760'
1978
1979                                         ;+
1980                                         ; RESTORE REGISTERS AND RETURN
1981                                         ;-
1982
1983  001760'                                CALL RESREG                                      JSR     PC,RESREG
 (3)  001760' 004767  000000G
1984
1985  001764'                                ENDRTN                                   50000$:
 (3)  001764'                                                                        50001$:
 (3)  001764'                                                                             RTS     PC
 (2)  001764' 000207
1986
1987          000001                         .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000000
ADRCT = 000004
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BDACNV= ****** G
BGNADR= 000002
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CC.CM1  000000R
CDERCT= 000146
CDWDCT= 000144
CKHUNG  001476RG
CKTIM = 100000
CK.TEM  001474R
CLKCHK  000034RG
CLKOFF  000304RG
CLKON   000370RG
CLKPRE= 000001
CLRTIM  001434RG
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTA   = 000000
DTABLE= ****** G
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000051
DT.WLL= 000052
DUNCHK= ****** G
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ = ****** G
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HMS     001004RG
HM.CNV  000776R
HM.CTE  000770R
HM.CTT  000760R
HM.HRS  000760R
HM.MIN  000763R
HM.SEC  000765R
HM.THR  000770R
HM.TMI  000772R
HM.TSE  000774R
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LDTIME  000534R
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRHMS   000414RG
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PRPSCN  000604RG
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RESULT= 000004
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
```

```
SC.HLD= 000010          TMPIO = 000002          WDFR  = 000116          $F$SEL= 000140          $TSK2 = 050006
SC.SCA= 000012          TQOVF = 000002          WDTO  = 000114          $F$THE= 000330          $TSK3 = 050007
SENDLS= 177777          UIPAR0= 177640          WTINRE= 000352          $F$TRU= 000404          $TSK4 = 050010
SOFCNT= 000042          UIPAR1= 177642          WTWHMI= 000222          $F$UNT= 000130          $$ARGC= 000000
SOFPAS= 000046          UIPAR2= 177644          XFLAG = 000005          $F$WHI= 000120          $$BYTE= 000403
SPACE = 000040          UIPAR3= 177646          XOFF  = 000023          $F$YES= 000402          $$CASE= 000000
SPOINT= 000032          UIPAR4= 177650          XON   = 000021          $IFLEV= 177777          $$DST = 000000
SPVALU= 002200          UIPAR5= 177652          XTIME = 000002          $ISK0 = 000001          $$ELOC= 000402
SR0   = 177572          UIPAR6= 177654          $BGNLE= 177777          $ISK1 = 000001          $$ERFL= 000000
SR1   = 177574          UIPAR7= 177656          $ERFLG= 000400          $ISK2 = 000001          $$FLAG= 000001
SR2   = 177576          UIPDR0= 177600          $F$AND= 000310          $LOCTA= 177777          $$FROM= 000000
SR3   = 172516          UIPDR1= 177602          $F$BAD= 000401          $LSTIN= 000001          $$LOC = 001654R
STAT  = 000026          UIPDR2= 177604          $F$BLA= 000170          $LSTTA= 000001          $$LOCN= 000000
STATBI= 064757          UIPDR3= 177606          $F$CAS= 000150          $NESTL= 177777          $$REG = 177777
STAT1 = 000027          UIPDR4= 177610          $F$DEC= 000220          $NSK0 = 000300          $$RETU= 000000
SUSPND= 000001          UIPDR5= 177612          $F$DO = 000340          $NSK1 = 000120          $$RTN1= 050000
SVR0  = 000062          UIPDR6= 177614          $F$FAL= 000405          $NSK2 = 000110          $$RTN2= 050001
SVR1  = 000064          UIPDR7= 177616          $F$GOO= 000400          $NSK3 = 000110          $$SRC = 000000
SVR2  = 000066          UPDMOD= 000002          $F$IF = 000110          $NSK4 = 000110          $$TGSV= 000000
SVR3  = 000070          UPDTIM  001320RG        $F$INC= 000210          $SAVLE= 177777          $$TGS1= 000000
SVR4  = 000072          WASADR= 000104          $F$LOO= 000200          $SSK0 = 050005          $$TGS2= 000000
SVR5  = 000074          WBSTAT= 000040          $F$NAM= 000160          $TAGLE= 177777          $$TO  = 000000
SVR6  = 000076          WBUFEA= 000136          $F$NO = 000403          $TAGNU= 050011          $$$TAG= 050000
SYSCNT= 000052          WBUFPA= 000134          $F$OR = 000320          $TEMP = 000300          .     = 001766R
SYSERR= 000100          WBUFRQ= 000140          $F$RTI= 000350          $TSK0 = 050004
TIME  = 000000          WBUFSZ= 000142          $F$RTN= 000300          $TSK1 = 050005

. ABS.  000000    000
        001766    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SYSCLK,DSKZ:SYSCLK=SPMAC/ML,EQUATE,SYSCLK
 RUN-TIME: 34 25 .4 SECONDS
 RUN-TIME RATIO: 181/61=2.9
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE TYPQUE (TYPE QUEUE MODULES)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE PACKAGE NAME:
                        ;       TYPQUE
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE PACKAGE CONTAINS THREE ROUTINES. THE FIRST ONE INITIALIZES
                        ;       THE TYPE QUEUE AND THE OTHER TWO ACTUALLY DO THE
                        ;       ENQUEING AND DEQUEING OF ITEMS FROM THE TYPE QUEUE.
                        ;       THE DESCRIPTION IS GIVEN UNDER THE RESPECTIVE MODULES.
                        ;       THESE MODULES ARE:
                        ;
                        ;           1. TQINI - INITIALIZES THE TYPE QUEUE
                        ;           2. ENQTQ - ENQUES ITEMS ON THE TYPE QUEUE
                        ;           3. DEQTQ - DEQUES ITEMS FROM THE TYPE QUEUE.
                        ;
                        ;        THE QUEUE POINTERS ARE CONTAINED
                        ;       IN THIS MODULE FOR USE BY THE ABOVE NAMED MODULES LOCALLY.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS & REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                  $LSTIN=1
        000001                  $LSTTAG=1

                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES:
                                ;
                                .GLOBL TQ.CT                    ;TQ COUNT
                                .GLOBL TQ.INP                   ;TQ IN PTR
                                .GLOBL TQ.OTP                   ;TQ OUT PTR
                                .GLOBL TQINI                    ;ROUTINE TO INIT TYPE QUEUE
                                .GLOBL ENQTQ                    ;ROUTINE TO PUT ITEMS ON TQ
                                .GLOBL DEQTQ                    ;ROUTINE TO DEQUE ITEMS.
                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL GETPSW                   ;GET CALLER'S PSW ROUTINE
                                .GLOBL OV.TQ                    ;TYPE QUEUE BUFFER
                                .GLOBL  OV.HITQ
                                .GLOBL  OV.TQSIZ

                                ;****************
                                ; LOCAL STORAGE
                                ;
000000' 000000                  TQ.VEC: 0                       ;SAVE LOC.4
000002' 000000                  TQ.PSW: 0                       ;LOC. TO SAVE CALLER'S PS WORD
000004' 000000G                 TQ.INP: #OV.TQ                  ;ENQUE POINTER
000006' 000000G                 TQ.OTP: #OV.TQ                  ;DEQUE POINTER
000010' 000000                  TQ.CT:  0                       ;CURRENT ENTRY COUNT
000012' 000000G                 TQ.HI:  #OV.HITQ                ;QUEUE'S HIGH ADDRESS
000014' 000000G                 TQ.MAX: #OV.TQSIZ               ;MAX NO. OF ENTRIES ALLOWED
000016' 000000G                 TQ.ITQ: #OV.TQ
```

```
                        .SBTTL TQINI (FUNCTIONAL DESCRIPTION)

                        ;++
                        ; MODULE NAME:
                        ;       TQINI
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE INITIALIZES THE TYPE QUEUE. IT RESETS THE ENQUE
                        ;       AND DEQUE POINTERS TO THE START OF THE QUEUE, AND SETS THE
                        ;       COUNT TO ZERO
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL TQINI
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;--
```

```
                                .SBTTL TQINI (CODE)


                                ROUTINE TQINI
000020'                                                 TQINI:
000020'
                                ;+
                                ; INITIALIZE IN AND OUT POINTERS AND COUNT
                                ;-


000020'                         LET TQ.INP := TQ.ITQ
000020' 016767  177772  177756                          MOV     TQ.ITQ,TQ.INP
                                LET TQ.OTP := TQ.ITQ
000026' 016767  177764  177752                          MOV     TQ.ITQ,TQ.OTP
                                LET TQ.CT := #0
000034' 005067  177750                                  CLR     TQ.CT
                                ENDRTN
000040'                                                 50000$:
000040'                                                 50001$:
000040' 000207                                          RTS     PC
```

```
                        .SBTTL ENQTQ (FUNCTIONAL DESCRIPTION)

                        ;++
                        ; MODULE NAME:
                        ;       ENQTQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE HANDLES QUEUEING OF ITEMS ONTO THE TYPE
                        ;       QUEUE.
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE ADDRESS
                        ;       2. MESSAGE TYPE CODE
                        ;       3. MESSAGE ADDRESS
                        ;       4. MODULE'S HEADER ADDRESS
                        ;       5. RETURN ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ERR
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       ERROR WORD (DT.ERR)
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       1. GETPSW               ;GET CALLER'S PS WORD
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       ENQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
                        ;       SERVES AS A LOCKING MECHANISM.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL ENQTQ IN <DTADR,TYPCOD,MSGADR,HDRADR,RETADR>
                        ;
                        ;               WHERE DTADR = DATA TABLE ADDRESS
                        ;                     TYPCOD = TYPE CODE
                        ;                     MSGADR = MESSAGE ADDRESS
                        ;                     HDRADR = HEADER ADDRESS
                        ;                     RETADR = RETURN ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;--
```

```
                                .SBTTL ENQTQ (CODE)

                                ROUTINE ENQTQ <DT,CODE,MSG,HDR,RET>
000042'                                                         ENQTQ:
(2)  000042'

                                ;+
                                ; SAVE REGISTER
                                ;-

                                PUSH R0
000042'                                                                 MOV     R0,-(SP)
(2)  000042' 010046

                                ;+
                                ; IF THE COUNT IS EQUAL TO MAX. COUNT,
                                ; THE QUEUE IS FULL AND WE CANNOT MAKE
                                ; FURTHER ENTRIES INTO THE QUEUE. SAVE DTABLE ADDR AND SET
                                ; ERROR FLAG IN THE STATUS INDICATOR
                                ; AND RETURN WITH ERROR.
                                ;-

                                IF TQ.CT GE TQ.MAX THEN
000044'                                                                 CMP     TQ.CT,TQ.MAX
(6)  000044' 026767  177740  177742                                    BLT     50002$
(9)  000052' 002410
                                    LET R0 := DT(R5)
000054'                                                                 MOV     DT(R5),R0
(4)  000054' 016500  000000
                                    LET DT.ERR(R0) := DT.ERR(R0) SET.BY #TQOVF!FATERR
000060'                                                                 BIS     #TQOVF!FATERR,DT
(6)  000060' 052760  100002  000020
                                    POP R0
000066'                                                                 MOV     (SP)+,R0
(2)  000066' 012600
                                    RETURN ERROR
000070'                                                                 SEC
(2)  000070' 000261
(4)  000072' 000447                                                     BR      50001$
                                ENDIF
000074'                                                         50002$:
(4)  000074'

                                ;+
                                ; SAVE CALLER'S PRIORITY AND
                                ; CHANGE PRIORITY LEVEL TO 7
                                ;-

                                CALL GETPSW OUT <TQ.PSW>
000074'                                                                 SUB     #1*2,R5
(4)  000074' 162705  000002                                             JSR     PC,GETPSW
(3)  000100' 004767  000000G                                           MOV     (R5)+,TQ.PSW
(4)  000104' 012567  177672

                                PUSH #PR7
000110'                                                                 MOV     #PR7,-(SP)
(2)  000110' 012746  000340
                                PUSH #1$
000114'                                                                 MOV     #1$,-(SP)
(2)  000114' 012746  000122'
                                INLINE <RTI>
000120'                                                                 RTI
(2)  000120' 000002

                                ;+
                                ; IF QUEUE'S ENQUE POINTER IS EQUAL
                                ; TO QUEUE'S HIGH ADDRESS, RE-
                                ; INITIALIZE THE POINTER TO THE
                                ; START ADDRESS OF THE QUEUE.
```

```
                                        ;-
                                        INLINE <1$:>
                                        IF TQ.INP EQ TQ.HI THEN
                                            LET TQ.INP := #OV.TQ
                                        ENDIF
                                        ;+
                                        ; NOW ENQUE THE ENTRIES.
                                        ;-
                                        LET R0 := TQ.INP
                                        LET (R0)+ := CODE(R5)
                                        LET (R0)+ := MSG(R5)
                                        LET (R0)+ := HDR(R5)
                                        LET (R0)+ := RET(R5)
                                        LET TQ.INP := R0
                                        LET TQ.CT := TQ.CT + #1
                                        ;+
                                        ; RESTORE CALLER'S PS WORD AND RETURN
                                        ;-
                                        PUSH TQ.PSW
                                        PUSH #2$
                                        INLINE <RTI>
                                        INLINE <2$:>
                                        POP R0
                                        ENDRTN
```

```
000122'                                 1$:
000122'
000122'                                 CMP     TQ.INP,TQ.HI
000122' 026767  177656  177662         BNE     50003$
000130' 001003
000132'                                 MOV     #OV.TQ,TQ.INP
000132' 012767  000000G 177644
000140'                         50003$:
000140'
000140'                                 MOV     TQ.INP,R0
000140' 016700  177640
000144'                                 MOV     CODE(R5),(R0)+
000144' 016520  000002
000150'                                 MOV     MSG(R5),(R0)+
000150' 016520  000004
000154'                                 MOV     HDR(R5),(R0)+
000154' 016520  000006
000160'                                 MOV     RET(R5),(R0)+
000160' 016520  000010
000164'                                 MOV     R0,TQ.INP
000164' 010067  177614
000170'                                 INC     TQ.CT
000170' 005267  177614
000174'                                 MOV     TQ.PSW,-(SP)
000174' 016746  177602
000200'                                 MOV     #2$,-(SP)
000200' 012746  000206'
000204'                                 RTI
000204' 000002
000206'                                 2$:
000206'
000206'                                 MOV     (SP)+,R0
000206' 012600
000210'                         50000$:
000210'                                 CLC
000210' 000241
000212'                         50001$:
000212'                                 RTS     PC
000212' 000207
```

```
                        .SBTTL DEQTQ (MODULE FUNCTIONAL DESCRIPTION)


                        ;++
                        ; MODULE NAME:
                        ;       DEQTQ
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE HANDLES DEQUEUEING OF ITEMS FROM THE TYPE
                        ;       QUEUE.
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       1. MESSAGE TYPE CODE
                        ;       2. MESSAGE ADDRESS
                        ;       3. MODULE'S HEADER ADDRESS
                        ;       4. RETURN ADDRESS
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       GETPSW          ;GET CALLER'S PS WORD
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       DEQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
                        ;       SERVES AS A LOCKING MECHANISM.
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL DEQTQ OUT <CODE,MSGADR,HDRADR,RETADR>
                        ;
                        ;               WHERE CODE = TYPE CODE
                        ;                     MSGADR = MESSAGE ADDRESS
                        ;                     HDRADR = HEADER ADDRESS
                        ;                     RETADR = RETURN ADDRESS
                        ;
                        ;
000214'                 ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL DEQTQ (CODE)

                                ROUTINE DEQTQ <CODE,MSG,HDR,RET>
000214'                                                         DEQTQ:
(2)  000214'

                                ;+
                                ; IF THE COUNT IS EQUAL TO ZERO, QUEUE
                                ; IS EMPTY. RETURN WITH ERROR.
                                ;-

                                IF TQ.CT LE #0 THEN
000214'                                                                 TST     TQ.CT
(6)  000214' 005767  177570                                             BGT     50002$
(9)  000220' 003002
                                    RETURN ERROR
000222'                                                                 SEC
(2)  000222' 000261                                                     BR      50001$
(4)  000224' 000450
                                ENDIF
000226'                                                         50002$:
(4)  000226'

                                ;+
                                ; SAVE CALLER'S PRIORITY AND
                                ; CHANGE PRIORITY TO 7
                                ;-

                                CALL GETPSW OUT <TQ.PSW>
000226'                                                                 SUB     #1*2,R5
(4)  000226' 162705  000002                                             JSR     PC,GETPSW
(3)  000232' 004767  000000G                                            MOV     (R5)+,TQ.PSW
(4)  000236' 012567  177540
                                PUSH #PR7
000242'                                                                 MOV     #PR7,-(SP)
(2)  000242' 012746  000340
                                PUSH #3$
000246'                                                                 MOV     #3$,-(SP)
(2)  000246' 012746  000254'
                                INLINE <RTI>
000252'                                                                 RTI
(2)  000252' 000002

                                ;+
                                ; IF THE DEQUE POINTER IS AT HIGH
                                ; ADDRESS, RE-INITIALIZE IT TO THE
                                ; START ADDRESS OF THE QUEUE.
                                ;-

                                INLINE <3$:>
000254'                                                                 3$:
(2)  000254'
                                IF TQ.OTP EQ TQ.HI THEN
000254'                                                                 CMP     TQ.OTP,TQ.HI
(6)  000254' 026767  177526  177530                                     BNE     50003$
(9)  000262' 001003
                                    LET TQ.OTP := #OV.TQ
000264'                                                                 MOV     #OV.TQ,TQ.OTP
(4)  000264' 012767  000000G 177514
                                ENDIF
000272'                                                         50003$:
(4)  000272'

                                ;+
                                ; NOW DEQUE THE ENTRIES.
                                ;-

                                ;+
```

```
                                ; SAVE REGISTERS
                                ;-

                                PUSH R0                         MOV     R0,-(SP)

                                LET R0 := TQ.OTP                MOV     TQ.OTP,R0

                                LET CODE(R5) := (R0)+           MOV     (R0)+,CODE(R5)

                                LET MSG(R5) := (R0)+            MOV     (R0)+,MSG(R5)

                                LET HDR(R5) := (R0)+            MOV     (R0)+,HDR(R5)

                                LET RET(R5) := (R0)+            MOV     (R0)+,RET(R5)

                                LET TQ.OTP := R0                MOV     R0,TQ.OTP

                                ;+
                                ; DECREMENT COUNT
                                ;-

                                LET TQ.CT := TQ.CT - #1         DEC     TQ.CT

                                ;+
                                ; RESTORE CALLER'S PS WORD AND RETURN
                                ;-

                                PUSH TQ.PSW                     MOV     TQ.PSW,-(SP)

                                PUSH #4$                        MOV     #4$,-(SP)

                                INLINE <RTI>                    RTI

                                INLINE <4$:>                    4$:

                                POP R0                          MOV     (SP)+,R0

                                ENDRTN                  50000$: CLC

                                                        50001$: RTS     PC

                                .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CODE   = 000000
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DEQTQ    000214RG
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT     = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
ENQTQ    000042RG
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GETPSW= ****** G
HDR    = 000004
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSG    = 000002
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OV.HIT= ****** G
OV.TQ  = ****** G
OV.TQS= ****** G
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RET    = 000006
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
```

```
SVR4   = 000072          UIPAR5= 177652          WTWHMI= 000222          $F$RTN= 000300          $$BYTE= 000403
SVR5   = 000074          UIPAR6= 177654          XFLAG = 000005          $F$SEL= 000140          $$CASE= 000000
SVR6   = 000076          UIPAR7= 177656          XOFF  = 000023          $F$THE= 000330          $$DST = 000000
SYSCNT= 000052           UIPDR0= 177600          XON   = 000021          $F$TRU= 000404          $$ELOC= 000402
SYSERR= 000100           UIPDR1= 177602          $BGNLE= 177777          $F$UNT= 000130          $$ERFL= 000000
TMPIO  = 000002          UIPDR2= 177604          $ERFLG= 000400          $F$WHI= 000120          $$FLAG= 000001
TQINI    000020RG        UIPDR3= 177606          $F$AND= 000310          $F$YES= 000402          $$FROM= 000001
TQOVF  = 000002          UIPDR4= 177610          $F$BAD= 000401          $IFLEV= 177777          $$LOC = 000262R
TQ.CT    000010RG        UIPDR5= 177612          $F$BLA= 000170          $ISK0 = 000001          $$LOCN= 000000
TQ.HI    000012R         UIPDR6= 177614          $F$CAS= 000150          $LOCTA= 177777          $$REG = 177777
TQ.INP   000004RG        UIPDR7= 177616          $F$DEC= 000220          $LSTIN= 000001          $$RETU= 000001
TQ.ITQ   000016R         VERSIO  000214R         $F$DO = 000340          $LSTTA= 000001          $$RTN1= 050000
TQ.MAX   000014R         WASADR= 000104          $F$FAL= 000405          $NESTL= 177777          $$RTN2= 050001
TQ.OTP   000006RG        WBSTAT= 000040          $F$GOO= 000400          $NSK0 = 000300          $$SRC = 000000
TQ.PSW   000002R         WBUFEA= 000136          $F$IF = 000110          $NSK1 = 000110          $$TGSV= 000000
TQ.VEC   000000R         WBUFPA= 000134          $F$INC= 000210          $SAVLE= 177777          $$TGS1= 000000
UIPAR0= 177640           WBUFRQ= 000140          $F$LOO= 000200          $TAGLE= 177777          $$TGS2= 000000
UIPAR1= 177642           WBUFSZ= 000142          $F$NAM= 000160          $TAGNU= 050004          $$TO  = 000000
UIPAR2= 177644           WDFR  = 000116          $F$NO = 000403          $TEMP = 000300          $$$TAG= 050000
UIPAR3= 177646           WDTO  = 000114          $F$OR = 000320          $TSK0 = 050003          .     = 000350R
UIPAR4= 177650           WTINRE= 000352          $F$RTI= 000350          $$ARGC= 000010

. ABS.  000000    000
        000350    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:TYPQUE,DSKZ:TYPQUE=SPMAC/ML,EQUATE,TYPQUE
 RUN-TIME: 15 5 .3 SECONDS
 RUN-TIME RATIO: 52/20=2.5
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE UNIMAP - LOAD UNIBUS MAP
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       UNIMAP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       LOADS THE UNIBUS MAP SO THAT THE WRITE BUFFER CAN BE
                        ;       ADDRESSED BY UNIBUS DEVICES WHEN 22-BIT ADDRESSING
                        ;       SPACE IS IN USE.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.WLLMT
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       BL.BTMBNK                       ;CURRENT 124K BANK OF BOTTOM OF EXERCISER
                        ;       BL.TOPBNK                       ;CURRENT 124K BANK OF TOP OF EXERCISER
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL UNIMAP IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES
                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN=1
            000001              $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  UNIMAP                  ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SAVE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  BL.BTMBNK               ;CURRENT 124K BANK OF BOTTOM OF RTE
                                .GLOBL  BL.TOPBNK               ;CURRENT 124K BANK OF TOP OF RTE
                                ;
                                ;****************
                                ;
                                ; LOCAL EQUATES
                                ;
            170200              MAPREG=170200                   ;LOWEST ADDRESS OF UNIBUS MAP

```

```
                                .SBTTL UNIMAP ROUTINE

                                ROUTINE UNIMAP <TABL>
000000'                                                                 UNIMAP:
(2)  000000'

                                ;+
                                ; SAVE REGISTERS
                                ;-

000000'                         CALL SAVREG
(3)  000000' 004767  000000G                                            JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

000004'                         LET R0 := TABL(R5)
(4)  000004' 016500  000000                                             MOV     TABL(R5),R0

                                ;+
                                ; IF THE WRITE BUFFER LOW LIMIT IS 4K, IMPLYING THE CURRENT BANK IS THE
                                ; LOWEST 124K BANK OF MEMORY, THEN START
                                ; MAPPING FROM ADDRESS 0.  HOWEVER, IF THIS IS A 128K SYSTEM
                                ; AND IF THE EXERCISER IS CROSSING THE 124K BOUNDARY, BEGIN MAPPING
                                ; FROM THE LOW BUFFER LIMIT.

                                ;-

000010'                         IF DT.WLLMT(R0) EQ #200 THEN
(6)  000010' 026027  000052  000200                                     CMP     DT.WLLMT(R0),#20
(9)  000016' 001014                                                     BNE     50002$
000020'                                 IF DT.SSIZ(R0) EQ #10000 AND BL.TOPBNK NE BL.BTMBNK THEN
(6)  000020' 026027  000046  010000                                     CMP     DT.SSIZ(R0),#100
(9)  000026' 001005                                                     BNE     50003$
(6)  000030' 026767  000000G 000000G                                    CMP     BL.TOPBNK,BL.BTM
(9)  000036' 001401                                                     BEQ     50003$
000040'                                         INLINE <BR 1$>
(2)  000040' 000403                                                     BR 1$
000042'                                 ENDIF
(4)  000042'                                                    50003$:
000042'                                 LET R1 := #0
(4)  000042' 005001                                                     CLR     R1
000044'                                 LET R2 := #0
(4)  000044' 005002                                                     CLR     R2

000046'                         ELSE
(4)  000046' 000417                                                     BR      50004$
(3)  000050'                                                    50002$:

                                ;+
                                ; ELSE CONVERT THE WRITE BUFFER LOW LIMIT TO 22 BITS.  (THE LOW 16 BITS
                                ; WILL BE IN R1, AND THE HIGH 6 BITS WILL BE IN R2.)
                                ; GET THE UPPER SIX BITS INTO THE UPPER SIX BITS OF R1
                                ; THEN SWAP THEM INTO THE LOWER BYTE OF R2
```

```
                        ; THEN LINE THEM UP IN BITS 0 THRU 5 AND MASK OUT UPPER 10 BITS
                        ;-

                        INLINE <1$:>
                                                                        1$:
                                LET R1 := DT.WLLMT(R0) SHIFT #+6
                                                                        MOV     DT.WLLMT(R0),R1
                                                                        ASL     R1
                                                                        ASL     R1
                                                                        ASL     R1
                                                                        ASL     R1
                                                                        ASL     R1
                                                                        ASL     R1
                                LET R2 := SWAP DT.WLLMT(R0)
                                                                        MOV     DT.WLLMT(R0),R2
                                                                        SWAB    R2
                                LET R2 := R2 SHIFT #-2
                                                                        ASR     R2
                                                                        ASR     R2
                                LET R2 := R2 CLR.BY #177400
                                                                        BIC     #177400,R2
                        ENDIF
                                                                50004$:

                        ;+
                        ; NOW MAP, BY SETTIN R3 TO LOWEST MAP REGISTER AND R4 TO MAXIMUM
                        ; NUMBER OF REGISTER.
                        ;-

                        LET R3 := #MAPREG
                                                                        MOV     #MAPREG,R3
                        LET R4 := #^D31
                                                                        MOV     #^D31,R4

                        ;+
                        ; WHILE REGISTERS REMAIN CONTINUE TO MAP
                        ;-

                        WHILE R4 NE #0 DO
                                                                50005$:
                                                                        TST     R4
                                                                        BEQ     50006$

                        ;+
                        ; LOAD REGISTERS AND THEN UPDATE POINTERS
                        ;-

                                LET (R3)+ := R1
                                                                        MOV     R1,(R3)+
                                LET (R3)+ := R2
                                                                        MOV     R2,(R3)+
                                LET R1 := R1 + #20000
                                                                        ADD     #20000,R1
                                LET R2 := R2 + CARRY
                                                                        ADC     R2
```

```
   657  000134'                          LET R4 := R4 - #1
   (6)  000134' 005304                                                  DEC     R4
   658  000136'                          ENDDO
   (4)  000136' 000767                                                  BR      50005$
   (3)  000140'                                                 50006$:
   659
   660
   661                                   ;+
   662                                   ; RESTORE REGISTERS
   663                                   ;-
   664
   665  000140'                          CALL RESREG
   (3)  000140' 004767  000000G                                        JSR     PC,RESREG
   666
   667  000144'                          ENDRTN
   (3)  000144'                                                 50000$:
   (3)  000144'                                                 50001$:
   (2)  000144' 000207                                                  RTS     PC
   668
   669          000001                   .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BL.BTM= ****** G
BL.TOP= ****** G
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPREG= 170200
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TABL   = 000000
```

```
TMPIO = 000002          WASADR= 000104          $F$DEC= 000220          $ISK0 = 000001          $$DST = 000000
TQOVF = 000002          WBSTAT= 000040          $F$DO = 000340          $ISK1 = 000001          $$ELOC= 000402
UIPAR0= 177640          WBUFEA= 000136          $F$FAL= 000405          $LOCTA= 177777          $$ERFL= 000000
UIPAR1= 177642          WBUFPA= 000134          $F$GOO= 000400          $LSTIN= 000001          $$FLAG= 000340
UIPAR2= 177644          WBUFRQ= 000140          $F$IF = 000110          $LSTTA= 000001          $$FROM= 000000
UIPAR3= 177646          WBUFSZ= 000142          $F$INC= 000210          $NESTL= 177777          $$LOC = 000120R
UIPAR4= 177650          WDFR  = 000116          $F$LOO= 000200          $NSK0 = 000300          $$LOCN= 000000
UIPAR5= 177652          WDTO  = 000114          $F$NAM= 000160          $NSK1 = 000120          $$REG = 177777
UIPAR6= 177654          WTINRE= 000352          $F$NO = 000403          $NSK2 = 000110          $$RETU= 000000
UIPAR7= 177656          WTWHMI= 000222          $F$OR = 000320          $SAVLE= 177777          $$RTN1= 050000
UIPDR0= 177600          XFLAG = 000005          $F$RTI= 000350          $SSK0 = 050006          $$RTN2= 050001
UIPDR1= 177602          XOFF  = 000023          $F$RTN= 000300          $TAGLE= 177777          $$SRC = 000000
UIPDR2= 177604          XON   = 000021          $F$SEL= 000140          $TAGNU= 050007          $$TGSV= 000000
UIPDR3= 177606          $BGNLE= 177777          $F$THE= 000330          $TEMP = 000300          $$TGS1= 000000
UIPDR4= 177610          $ERFLG= 000400          $F$TRU= 000404          $TSK0 = 050005          $$TGS2= 000000
UIPDR5= 177612          $F$AND= 000310          $F$UNT= 000130          $TSK1 = 050006          $$TO  = 000000
UIPDR6= 177614          $F$BAD= 000401          $F$WHI= 000120          $$ARGC= 000002          $$$TAG= 050000
UIPDR7= 177616          $F$BLA= 000170          $F$YES= 000402          $$BYTE= 000403          .     = 000146R
UNIMAP  000000RG        $F$CAS= 000150          $IFLEV= 177777          $$CASE= 000000

. ABS.  000000    000
        000146    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:UNIMAP,DSKZ:UNIMAP=SPMAC/ML,EQUATE,UNIMAP
 RUN-TIME: 13 3 .3 SECONDS
 RUN-TIME RATIO: 32/17=1.8
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE UNIPA - GET PHYSICAL ADDRESS FORM UNIBUS MAP
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       UNIPA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       DETERMINES 22-BIT PHYSICAL ADDRESS FROM 18-BIT VIRTUAL
                        ;       ADDRESS AND PROPER UNIBUS MAP REGISTER.
                        ;
                        ; INPUTS:
                        ;       ADDRESS OF I/O TABLE - CONTAINS 16 BIT PA AND 2 EA BITS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       PA
                        ;       EA
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       16 BITS AND EA BITS
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL UNIPA IN <A>
                        ;               A=ADDRESS OF I/O TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                                .SBTTL COMMON DEFINTIONS AND REFERENCES
                                .MCALL STRUCT
000000'                         STRUCT
000000'                         .PRINT  ;SPMAC: VERSION 1.1
                000001          $LSTIN=1
                000001          $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  UNIPA                           ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                          ;SAVE REGISTERS
                                .GLOBL  RESREG                          ;RESTORE REGISTERS
                                ;
                                ;****************
                                ;
                                ; LOCAL EQUATES
                                ;
                170200          MAPREG=170200
                                ;
                                ;****************
                                ;
```

```
                                .SBTTL UNIPA ROUTINE


                                ROUTINE UNIPA <ADDR>
000000'                                                                 UNIPA:
(2)  000000'


                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG
000000'                                                                 JSR     PC,SAVREG
(3)  000000' 004767  000000G


                                ;+
                                ; GET THE ADDRESS OF THE I/O TABLE
                                ;-

000004'                         LET R0 := ADDR(R5)
                                                                        MOV     ADDR(R5),R0
(4)  000004' 016500  000000


                                ;+
                                ; GET THE 18-BIT ADDRESS TO BE CONVERTED.  THE LOW 16 BITS WILL
                                ; BE IN R1 AND THE HIGH 2 BITS WILL BE IN R2 (POSITIONS 4 AND 5).
                                ;-

000010'                         LET R1 := (R0)+
                                                                        MOV     (R0)+,R1
(4)  000010' 012001
000012'                         LET R2 := (R0)+
                                                                        MOV     (R0)+,R2
(4)  000012' 012002


                                ;+
                                ; GET THE HIGH 5 BITS OF THIS ADDRESS TO DETERMINE THE PROPER
                                ; UNIBUS MAP REGISTER.
                                ;-

000014'                         LET R3 := SWAP R1
                                                                        MOV     R1,R3
(6)  000014' 010103
                                                                        SWAB    R3
(6)  000016' 000303
000020'                         LET R3 := R3 SHIFT #-3
                                                                        ASR     R3
(7)  000020' 006203
                                                                        ASR     R3
(7)  000022' 006203
                                                                        ASR     R3
(7)  000024' 006203
000026'                         LET R3 := R3 CLR.BY #177743
                                                                        BIC     #177743,R3
(6)  000026' 042703  177743
000032'                         LET R4 := R2 SHIFT #+1
                                                                        MOV     R2,R4
(4)  000032' 010204
                                                                        ASL     R4
(7)  000034' 006304
000036'                         LET R4 := R4 + R3
                                                                        ADD     R3,R4
(6)  000036' 060304


                                ;+
```

```
                        ; GET THE CONTENTS OF THE PROPER MAP REGISTER.  THE LOW
                        ; 16 BITS WILL BE IN R2 AND THE HIGH 6 BITS WILL BE IN R3.
                        ;-

000040'                 LET R4 := R4 + #MAPREG                          ADD     #MAPREG,R4
(6) 000040' 062704  170200
                        LET R2 := (R4)+                                 MOV     (R4)+,R2
(4) 000044' 012402
                        LET R3 := (R4)                                  MOV     (R4),R3
(4) 000046' 011403


                        ;+
                        ; ADD THE LOW 12 BITS OF THE 18-BIT VIRTUAL ADDRESS TO THE
                        ; VALUE JUST OBTAINED FROM THE MAP REGISTER.  THE RESULT
                        ; IS THE 22-BIT PHYSICAL ADDRESS.
                        ;-

000050'                 LET R1 := R1 CLR.BY #160000                     BIC     #160000,R1
(6) 000050' 042701  160000
                        LET R2 := R2 + R1                               ADD     R1,R2
(6) 000054' 060102
                        LET R3 := R3 + CARRY                            ADC     R3
(6) 000056' 005503


                        ;+
                        ; STORE THE 22-BIT PHYSICAL ADDRESS IN THE I/O TABLE.  THE LOW
                        ; 16 BITS GO IN FIRST, THEN THE HIGH 6.
                        ;-

000060'                 LET (R0)+ := R2                                 MOV     R2,(R0)+
(4) 000060' 010220
                        LET (R0) := R3                                  MOV     R3,(R0)
(4) 000062' 010310


                        ;+
                        ; RESTORE REGISTERS
                        ;-

000064'                 CALL RESREG                                     JSR     PC,RESREG
(3) 000064' 004767  000000G


                        ;+
                        ; COME AGAIN, SOON.
                        ;-

000070'                 ENDRTN                                  50000$:
(3) 000070'                                                     50001$:
(3) 000070'                                                             RTS     PC
(2) 000070' 000207

                000001  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR  = 000000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPREG= 170200
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
```

```
UIPAR0= 177640          WASADR= 000104          $F$BLA= 000170          $F$UNT= 000130          $$ELOC= 000000
UIPAR1= 177642          WBSTAT= 000040          $F$CAS= 000150          $F$WHI= 000120          $$ERFL= 000000
UIPAR2= 177644          WBUFEA= 000136          $F$DEC= 000220          $F$YES= 000402          $$FLAG= 000000
UIPAR3= 177646          WBUFPA= 000134          $F$DO  = 000340          $IFLEV= 177777          $$FROM= 000000
UIPAR4= 177650          WBUFRQ= 000140          $F$FAL= 000405          $LOCTA= 177777          $$LOC = 000000
UIPAR5= 177652          WBUFSZ= 000142          $F$GOO= 000400          $LSTIN= 000001          $$LOCN= 000000
UIPAR6= 177654          WDFR  = 000116          $F$IF  = 000110          $LSTTA= 000001          $$REG = 177777
UIPAR7= 177656          WDTO  = 000114          $F$INC= 000210          $NESTL= 177777          $$RETU= 000000
UIPDR0= 177600          WTINRE= 000352          $F$LOO= 000200          $NSK0 = 000300          $$RTN1= 050000
UIPDR1= 177602          WTWHMI= 000222          $F$NAM= 000160          $SAVLE= 177777          $$RTN2= 050001
UIPDR2= 177604          XFLAG = 000005          $F$NO  = 000403          $TAGLE= 177777          $$SRC = 000000
UIPDR3= 177606          XOFF  = 000023          $F$OR  = 000320          $TAGNU= 050002          $$TGSV= 000000
UIPDR4= 177610          XON   = 000021          $F$RTI= 000350          $TEMP = 000300          $$TGS1= 000000
UIPDR5= 177612          $BGNLE= 177777          $F$RTN= 000300          $$ARGC= 000002          $$TGS2= 000000
UIPDR6= 177614          $ERFLG= 000400          $F$SEL= 000140          $$BYTE= 000000          $$TO  = 000000
UIPDR7= 177616          $F$AND= 000310          $F$THE= 000330          $$CASE= 000000          $$$TAG= 050000
UNIPA   000000RG        $F$BAD= 000401          $F$TRU= 000404          $$DST = 000000          .     = 000072R

. ABS.  000000    000
        000072    001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:UNIPA,DSKZ:UNIPA=SPMAC/ML,EQUATE,UNIPA
RUN-TIME: 12 2 .3 SECONDS
RUN-TIME RATIO: 29/15=1.9
CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE WBFLIM - DETERMINE WRITE BUFFER LIMITS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       WBFLIM
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       CALCULATES LOW AND HIGH WRITE BUFFER LIMITS AND
                        ;       INITIALIZES THE WRITE BUFFER ADDRESS POINTER
                        ;
                        ; INPUTS:
                        ;       DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ADDR
                        ;       DT.ESIZ
                        ;       DT.SSIZ
                        ;       DT.STO
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       DT.WLLMT
                        ;       DT.WHLMT
                        ;       DT.WBUF
                        ;       DT.OFFSET
                        ;
                        ; PATHOLOGICAL CONNECTIONS
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL WBFLIM IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES
                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1
            000001      $LSTIN=1
            000001      $LSTTAG=1

                        ;
                        ;****************
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  WBFLIM                          ;MODULE ENTRY POINT
                        .GLOBL  BL.BTMBNK
                        .GLOBL  BL.TOPBNK
                        ;
                        ;****************
                        .GLOBL  SAVREG                          ;SAVE REGISTERS
                        .GLOBL  RESREG                          ;RESTORE REGISTERS
                        ;
                        ;****************
                        ;
                        ; LOCAL EQUATES
                        ;
            020000      FOURK=20000                             ;4K OF MEMORY
            000200      PAR4K=200                               ;4K OF MEMORY IN PAR FORMAT
            007600      BANK=7600                               ;PAR FORMAT FOR 124K BANK
                        ;
                        ;****************
                        ;
                        ; LOCAL STORAGE
                        ;
000000' 000000          BL.TOPBNK:      .WORD 0                 ;CURRENT 124K BANK OF TOP OF
                                                                ;   MOVABLE PORTION OF EXERCISER
000002' 000000          BL.BTMBNK:      .WORD 0                 ;CURRENT 124K BANK OF BOTTOM OF
                                                                ;   MOVABLE PORTION OF EXERCISER
                        ;
                        ;****************
                        ;
```

```
                                .SBTTL WBFLIM ROUTINE

                                ROUTINE WBFLIM <TABL>                                   WBFLIM:

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG                                             JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := TABL(R5)                                      MOV     TABL(R5),R0

                                ;+
                                ; FIND THE TOP ADDRESS OF THE MOVABLE PORTION OF THE EXERCISER.
                                ; TO DO THIS, FIND THE SIZE OF THE MOVABLE PORTION BY SUBTRACTING 4K FROM
                                ; EXERCISER SIZE, CONVERT IT TO PAR FORMAT
                                ;-

                                LET R1 := DT.ESIZ(R0) - #FOURK                          MOV     DT.ESIZ(R0),R1
                                                                                        SUB     #FOURK,R1
                                LET R1 := R1 SHIFT #-6                                  ASR     R1
                                                                                        ASR     R1
                                                                                        ASR     R1
                                                                                        ASR     R1
                                                                                        ASR     R1
                                                                                        ASR     R1
                                LET R1 := R1 CLR.BY #176000                             BIC     #176000,R1
                                LET R1 := R1 + #1                                       INC     R1

                                ;+
                                ; NOW ADD IT TO THE BASE ADDRESS OF THE MOVABLE PORTION
                                ;-
                                LET R1 := R1 + DT.ADDR(R0)                              ADD     DT.ADDR(R0),R1

                                ;+
                                ; INITIALIZE THE OFFSET VALUE TO ZERO.
                                ;-

                                LET DT.OFFSET(R0) := #0                                 CLR     DT.OFFSET(R0)
```

```
                                ;+
                                ; SEE IF MEMORY MANAGEMENT IS TURNED ON.
                                ;-

000056'                         IF #KTSTAT SETIN DT.ST0(R0) THEN        BIT     #KTSTAT,DT.ST0(R
(6)  000056' 032760  000020  000010                                    BEQ     50002$
(9)  000064' 001566


                                ;+
                                ; MEMORY MANAGEMENT IS ON.
                                ;-


                                ;+
                                ; SEE IF THE MOVABLE PORTION OF THE EXERCISER LIES ACROSS A 124K
                                ; BOUNDARY.  TO DO THIS, DETERMINE THE RESPECTIVE BANKS OF THE
                                ; TOP AND BOTTOM OF THE MOVABLE PORTION OF THE EXERCISER.
                                ;-

000066'                                 LET R2 := #0                    CLR     R2
(4)  000066' 005002
000070'                                 LET BL.TOPBNK := #-1            MOV     #-1,BL.TOPBNK
(4)  000070' 012767  177777  177702
000076'                                 WHILE R2 LO R1 DO               50003$: CMP     R2,R1
(4)  000076'                                                                    BHIS    50004$
(6)  000076' 020201
(9)  000100' 103005
000102'                                         LET R2 := R2 + #BANK    ADD     #BANK,R2
(6)  000102' 062702  007600
000106'                                         LET BL.TOPBNK := BL.TOPBNK + #1
(6)  000106' 005267  177666                                             INC     BL.TOPBNK
000112'                                 ENDDO                           BR      50003$
(4)  000112' 000771
(3)  000114'                                                    50004$:
000114'                                 LET R2 := #0                    CLR     R2
(4)  000114' 005002
000116'                                 LET BL.BTMBNK := #-1            MOV     #-1,BL.BTMBNK
(4)  000116' 012767  177777  177656
000124'                                 WHILE R2 LO DT.ADDR(R0) DO      50005$: CMP     R2,DT.ADDR(R0)
(4)  000124'                                                                    BHIS    50006$
(6)  000124' 020260  000042
(9)  000130' 103005
000132'                                         LET R2 := R2 + #BANK    ADD     #BANK,R2
(6)  000132' 062702  007600
000136'                                         LET BL.BTMBNK := BL.BTMBNK + #1
(6)  000136' 005267  177640                                             INC     BL.BTMBNK
000142'                                 ENDDO                           BR      50005$
(4)  000142' 000770
(3)  000144'                                                    50006$:
```

```
                                ;+
                                ; IF THE BANKS ARE NOT THE SAME, THEN THE EXERCISER IS CROSSING A BOUNDARY.
                                ;-
000144'                                 IF BL.TOPBNK NE BL.BTMBNK THEN
(6) 000144' 026767  177630  177630                                              CMP     BL.TOPBNK,BL.BTM
(9) 000152' 001441                                                              BEQ     50007$


                                ;+
                                ; THE EXERCISER LIES ACROSS A 124K BANK BOUNDARY.  IN THIS CASE, THE
                                ; LOW BUFFER LIMIT IS THE BOTTOM OF THE MOVABLE PORTION OF THE EXERCISER,
                                ; AND THE HIGH BUFFER LIMIT IS EITHER 124K ABOVE THAT OR THE
                                ; TOP OF MEMORY, WHICHEVER IS LOWER.
                                ; IF, HOWEVER, WE LET THE HIGH LIMIT EQUAL THE TOP OF MEMORY,
                                ; WE RE-DEFINE THE LOW LIMIT TO EITHER 124K BELOW THE HIGH LIMIT,
                                ; OR TO 4K, WHICHEVER IS HIGHER.  THIS ASSURES US THAT THERE
                                ; WILL BE A WRITE BUFFER AREA OF 124K AND THAT THE BOTTOM 4K OF THE
                                ; EXERCISER WILL BE PROTECTED.
                                ;-
000154'                                 LET DT.WLLMT(R0) := DT.ADDR(R0)
(4) 000154' 016060  000042  000052                                              MOV     DT.ADDR(R0),DT.W
000162'                                 LET DT.WHLMT(R0) := DT.ADDR(R0) + #BANK
(4) 000162' 016060  000042  000054                                              MOV     DT.ADDR(R0),DT.W
(6) 000170' 062760  007600  000054                                              ADD     #BANK,DT.WHLMT(R
000176'                                 IF DT.WHLMT(R0) HI DT.SSIZ(R0) THEN
(6) 000176' 026060  000054  000046                                              CMP     DT.WHLMT(R0),DT.
(9) 000204' 101420                                                              BLOS    50010$
000206'                                         LET DT.WHLMT(R0) := DT.SSIZ(R0)
(4) 000206' 016060  000046  000054                                              MOV     DT.SSIZ(R0),DT.W
000214'                                         LET DT.WLLMT(R0) := DT.SSIZ(R0) - #7600
(4) 000214' 016060  000046  000052                                              MOV     DT.SSIZ(R0),DT.W
(6) 000222' 162760  007600  000052                                              SUB     #7600,DT.WLLMT(R
000230'                                         IF DT.WLLMT(R0) LO #200 THEN
(6) 000230' 026027  000052  000200                                              CMP     DT.WLLMT(R0),#20
(9) 000236' 103003                                                              BHIS    50011$
000240'                                                 LET DT.WLLMT(R0) := #200
(4) 000240' 012760  000200  000052                                              MOV     #200,DT.WLLMT(R0
000246'                                         ENDIF
(4) 000246'                                                                 50011$:
000246'                                 ENDIF
(4) 000246'                                                                 50010$:

                                ;+
                                ; ALSO, THE OFFSET FROM THE ACTUAL 124K BANK BASE ADDRESS MUST
                                ; BE STORED.  IT EQUALS, IN THIS CASE, THE LOW BUFFER LIMIT.
                                ;-
000246'                                 LET DT.OFFSET(R0) := DT.WLLMT(R0)
(4) 000246' 016060  000052  000070                                              MOV     DT.WLLMT(R0),DT.


000254'                                 ELSE
(4) 000254' 000471                                                              BR      50012$
(3) 000256'                                                                 50007$:
```

```
                                ;+
                                ; OTHERWISE, THE MOVABLE PORTION OF THE EXERCISER IS WITHIN ONE BANK.
                                ;-


                                ;+
                                ; IF THE CURRENT BANK IS THE LOWEST ONE, THEN THE LOW BUFFER LIMIT
                                ; IS THE TOP OF THE BOTTOM 4K OF THE EXERCISER.  OTHERWISE, THE LOW
                                ; BUFFER LIMIT IS THE BOTTOM OF THE CURRENT BANK.
                                ;-

000256'                                 IF BL.BTMBNK EQ #0 THEN
(6)  000256' 005767  177520                                                     TST     BL.BTMBNK
(9)  000262' 001004                                                             BNE     50013$
000264'                                         LET DT.WLLMT(R0) := #PAR4K
(4)  000264' 012760  000200  000052                                            MOV     #PAR4K,DT.WLLMT(
000272'                                 ELSE
(4)  000272' 000413                                                             BR      50014$
(3)  000274'                                                            50013$:
000274'                                         LET R2 := #0
(4)  000274' 005002                                                             CLR     R2
000276'                                         LET DT.WLLMT(R0) := #0
(4)  000276' 005060  000052                                                     CLR     DT.WLLMT(R0)
000302'                                         WHILE R2 LO BL.BTMBNK DO
(4)  000302'                                                            50015$:
(6)  000302' 020267  177474                                                     CMP     R2,BL.BTMBNK
(9)  000306' 103005                                                             BHIS    50016$
000310'                                                 LET DT.WLLMT(R0) := DT.WLLMT(R0) + #BANK
(6)  000310' 062760  007600  000052                                            ADD     #BANK,DT.WLLMT(R
000316'                                                 LET R2 := R2 + #1
(6)  000316' 005202                                                             INC     R2
000320'                                         ENDDO
(4)  000320' 000770                                                             BR      50015$
(3)  000322'                                                            50016$:

000322'                                 ENDIF
(4)  000322'                                                            50014$:


                                ;+
                                ; IF THE CURRENT BANK IS THE HIGHEST ONE, THEN THE HIGH BUFFER LIMIT
                                ; IS THE TOP OF MEMORY.  OTHERWISE, THE HIGH BUFFER LIMIT IS THE TOP OF
                                ; THE CURRENT BANK.
                                ;-

000322'                                 LET R2 := #0
(4)  000322' 005002                                                             CLR     R2
000324'                                 LET R3 := #-1
(4)  000324' 012703  177777                                                     MOV     #-1,R3
000330'                                 WHILE R2 LO DT.SSIZ(R0) DO
(4)  000330'                                                            50017$:
(6)  000330' 020260  000046                                                     CMP     R2,DT.SSIZ(R0)
(9)  000334' 103004                                                             BHIS    50020$
000336'                                                 LET R2 := R2 + #BANK
(6)  000336' 062702  007600                                                     ADD     #BANK,R2
000342'                                                 LET R3 := R3 + #1
(6)  000342' 005203                                                             INC     R3
```

```
744  000344'                                    ENDDO
(4)  000344' 000771                                                    BR      50017$
(3)  000346'                                                 50020$:
745
746  000346'                                    IF BL.TOPBNK EQ R3 THEN
(6)  000346' 026703  177426                                            CMP     BL.TOPBNK,R3
(9)  000352' 001004                                                    BNE     50021$
747  000354'                                         LET DT.WHLMT(R0) := DT.SSIZ(R0)
(4)  000354' 016060  000046  000054                                    MOV     DT.SSIZ(R0),DT.W
748  000362'                                    ELSE
(4)  000362' 000415                                                    BR      50022$
(3)  000364'                                                 50021$:
749  000364'                                         IF BL.BTMBNK EQ #0 THEN
(6)  000364' 005767  177412                                            TST     BL.BTMBNK
(9)  000370' 001004                                                    BNE     50023$
750  000372'                                              LET DT.WHLMT(R0) := #BANK
(4)  000372' 012760  007600  000054                                    MOV     #BANK,DT.WHLMT(R
751  000400'                                         ELSE
(4)  000400' 000406                                                    BR      50024$
(3)  000402'                                                 50023$:
752  000402'                                              LET DT.WHLMT(R0) := DT.WLLMT(R0) + #BANK
(4)  000402' 016060  000052  000054                                    MOV     DT.WLLMT(R0),DT.
(6)  000410' 062760  007600  000054                                    ADD     #BANK,DT.WHLMT(R
753  000416'                                         ENDIF
(4)  000416'                                                 50024$:
754  000416'                                    ENDIF
(4)  000416'                                                 50022$:
```

```
                                ;+
                                ; DETERMINE THE OFFSET FORM THE BASE OF THE CURRENT 124K BANK.
                                ;-

                                        WHILE BL.BTMBMK GT #0 DO
000416'                                                                 50025$:
000416'                                                                         TST     BL.BTMBMK
000416' 005767  177360                                                          BLE     50026$
000422' 003406
                                                LET DT.OFFSET(R0) := DT.OFFSET(R0) + #7600
000424'                                                                         ADD     #7600,DT.OFFSET(
000424' 062760  007600  000070
                                                LET BL.BTMBNK := BL.BTMBNK - #1
000432'                                                                         DEC     BL.BTMBNK
000432' 005367  177344
                                        ENDDO
000436'                                                                         BR      50025$
000436' 000767
000440'                                                                 50026$:


                                ENDIF
000440'                                                                 50012$:
000440'


                                ELSE
000440'                                                                         BR      50027$
000440' 000415                                                          50002$:
000442'

                                ;+
                                ; MEMORY MANAGEMENT IS TURNED OFF
                                ;-

                                ;+
                                ; IN THIS CASE, THE LOW BUFFER LIMIT IS THE TOP OF THE EXERCISER.
                                ;-

                                        LET DT.WLLMT(R0) := R1
000442'                                                                         MOV     R1,DT.WLLMT(R0)
000442' 010160  000052


                                ;+
                                ; IF TOTAL MEMORY IS GREATER THAN 28K, THEN THE HIGH BUFFER LIMIT
                                ; IS 28K.  OTHERWISE, THE HIGH BUFFER LIMIT IS THE TOP OF MEMORY.
                                ;-

                                        IF DT.SSIZ(R0) HIS #1600 THEN
000446'                                                                         CMP     DT.SSIZ(R0),#160
000446' 026027  000046  001600                                                  BLO     50030$
000454' 103404
                                                LET DT.WHLMT(R0) := #1600
000456'                                                                         MOV     #1600,DT.WHLMT(R
000456' 012760  001600  000054
                                        ELSE
000464'                                                                         BR      50031$
000464' 000403                                                          50030$:
000466'
                                                LET DT.WHLMT(R0) := DT.SSIZ(R0)
000466'                                                                         MOV     DT.SSIZ(R0),DT.W
000466' 016060  000046  000054
                                        ENDIF
000474'                                                                 50031$:
000474'


                                ENDIF
000474'
```

```
   (4)  000474'                                                                          50027$:
   794
   795
   796                                ;+
   797                                ; NOW INITIALIZE THE WRITE BUFFER ADDRESS POINTER.  IF THE TOP OF
   798                                ; THE EXERCISER IS AT AN ADDRESS THAT IS GREATER THAN
   799                                ; OR EQUAL TO THE HIGH BUFFER LIMIT MINUS 1/2K, AND THE EXERCISER IS NOT
   800                                ; AT THE BOTTOM OF MEMORY, THEN SET THE POINTER IS THE LOW BUFFER
   801                                ; LIMIT.  OTHERWISE, SET THE POINTER TO THE TOP OF
   802                                ; THE EXERCISER.
   803                                ;-
   804
   805  000474'                       LET R2 := DT.SSIZ(R0) - #20
   (4)  000474' 016002  000046                                                          MOV     DT.SSIZ(R0),R2
   (6)  000500' 162702  000020                                                          SUB     #20,R2
   806  000504'                       IF R1 HIS R2 AND DT.ADDR(R0) HI #PAR4K THEN
   (6)  000504' 020102                                                                  CMP     R1,R2
   (9)  000506' 103410                                                                  BLO     50032$
   (6)  000510' 026027  000042  000200                                                 CMP     DT.ADDR(R0),#PAR
   (9)  000516' 101404                                                                  BLOS    50032$
   807  000520'                               LET DT.WBUF(R0) := DT.WLLMT(R0)
   (4)  000520' 016060  000052  000050                                                 MOV     DT.WLLMT(R0),DT.
   808  000526'                       ELSE
   (4)  000526' 000402                                                                  BR      50033$
   (3)  000530'                                                                         50032$:
   809  000530'                               LET DT.WBUF(R0) := R1
   (4)  000530' 010160  000050                                                          MOV     R1,DT.WBUF(R0)
   810  000534'                       ENDIF
   (4)  000534'                                                                         50033$:
   811
   812
   813                                ;+
   814                                ; RESTORE REGISTERS AND RETURN
   815                                ;-
   816
   817  000534'                       CALL RESREG
   (3)  000534' 004767  000000G                                                        JSR     PC,RESREG
   818
   819  000540'                       ENDRTN
   (3)  000540'                                                                         50000$:
   (3)  000540'                                                                         50001$:
   (2)  000540' 000207                                                                  RTS     PC
   820
   821          000001               .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BANK   = 007600
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
BL.BTM  000002RG
BL.TOP  000000RG
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
FOURK  = 020000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PAR4K  = 000200
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
```

```
SYSERR= 000100            WBFLIM  000004RG        $F$FAL= 000405        $LOCTA= 177777        $$CASE= 000000
TABL  = 000000            WBSTAT= 000040          $F$GOO= 000400        $LSTIN= 000001        $$DST = 000000
TMPIO = 000002            WBUFEA= 000136          $F$IF = 000110        $LSTTA= 000001        $$ELOC= 000402
TQOVF = 000002            WBUFPA= 000134          $F$INC= 000210        $NESTL= 177777        $$ERFL= 000000
UIPAR0= 177640            WBUFRQ= 000140          $F$LOO= 000200        $NSK0 = 000300        $$FLAG= 000001
UIPAR1= 177642            WBUFSZ= 000142          $F$NAM= 000160        $NSK1 = 000110        $$FROM= 000000
UIPAR2= 177644            WDFR  = 000116          $F$NO = 000403        $NSK2 = 000110        $$LOC = 000516R
UIPAR3= 177646            WDTO  = 000114          $F$OR = 000320        $NSK3 = 000120        $$LOCN= 000000
UIPAR4= 177650            WTINRE= 000352          $F$RTI= 000350        $NSK4 = 000110        $$REG = 177777
UIPAR5= 177652            WTWHMI= 000222          $F$RTN= 000300        $SAVLE= 177777        $$RETU= 000000
UIPAR6= 177654            XFLAG = 000005          $F$SEL= 000140        $SSK0 = 050026        $$RTN1= 050000
UIPAR7= 177656            XOFF  = 000023          $F$THE= 000330        $TAGLE= 177777        $$RTN2= 050001
UIPDR0= 177600            XON   = 000021          $F$TRU= 000404        $TAGNU= 050034        $$SRC = 000000
UIPDR1= 177602            $BGNLE= 177777          $F$UNT= 000130        $TEMP = 000300        $$TGSV= 000000
UIPDR2= 177604            $ERFLG= 000400          $F$WHI= 000120        $TSK0 = 050033        $$TGS1= 000000
UIPDR3= 177606            $F$AND= 000310          $F$YES= 000402        $TSK1 = 050031        $$TGS2= 000000
UIPDR4= 177610            $F$BAD= 000401          $IFLEV= 177777        $TSK2 = 050025        $$TO  = 000000
UIPDR5= 177612            $F$BLA= 000170          $ISK0 = 000001        $TSK3 = 050026        $$$TAG= 050000
UIPDR6= 177614            $F$CAS= 000150          $ISK1 = 000001        $TSK4 = 050016        .     = 000542R
UIPDR7= 177616            $F$DEC= 000220          $ISK2 = 000001        $$ARGC= 000002
WASADR= 000104            $F$DO = 000340          $ISK3 = 000001        $$BYTE= 000403

. ABS.  000000     000
        000542     001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:WBFLIM,DSKZ:WBFLIM=SPMAC/ML,EQUATE,WBFLIM
 RUN-TIME: 20 10 .3 SECONDS
 RUN-TIME RATIO: 48/31=1.5
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE WSTBUS - LOAD WORST-CASE UNIBUS PATTERN
                        .IDENT /V0.0/

                        ;++
                        ;
                        ; MODULE NAME:
                        ;       WSTBUS
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       ASSUMES EXERCISER IS IN LOWEST MEMORY AND LOADS ALL
                        ;       OF FREE MEMORY WITH WORST-CASE UNIBUS PATTERNS.
                        ;
                        ; INPUTS:
                        ;       DATA TABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       DT.ST0,DT.SSIZ,DT.EXS
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       SAVREG                          ;SAVE REGISTERS
                        ;       RESREG                          ;RESTORE REGISTERS
                        ;       BADMEM                          ;OUTPUT BAD MEMORY MESSAGE
                        ;       GPA                             ;GET PHYSICAL ADDRESSES
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL WSTBUS IN <A>
                        ;               A=ADDRESS OF DATA TABLE
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;               EDIT            BY              DATE            REASON
                        ;
                        ;--
                        ;
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                 000001         $LSTIN=1
                 000001         $LSTTAG=1

                                ;
                                ;****************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  WSTBUS                          ;MODULE ENTRY POINT
                                ;
                                ;****************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                          ;SAVE REGISTERS
                                .GLOBL  RESREG                          ;RESTORE REGISTERS
                                .GLOBL  BADMEM                          ;OUTPUT BAD MEMORY MESSAGE
                                .GLOBL  GPA                             ;GET PHYSICAL ADDRESS
                                ;
                                ;****************
                                ;
                                ; LOCAL STORAGE
                                ;
000000' 000000                  WB.SAV:         .WORD 0                 ;LOCAL STORAGE WORD
000002' 000000                  WB.DONE:        .WORD 0                 ;CURRENT PATTERN COMPLETE FLAG
000004' 000000                  WB.CNT:         .WORD 0                 ;CURRENT PATTERN WORD COUNT
000006' 000400                  WB.MAX:         .WORD ^D256             ;PATTERN BLOCK SIZE
000010' 000000                  WB.ROTCNT:      .WORD 0                 ;WORD ROTATION COUNTER FOR ROUTINE PAT2
000012' 000000                  WB.TMP: .WORD 0                         ;TEMPORARY STORAGE ON ERROR
000014' 000003                  WB.TBL: .WORD 3                         ;TABLE FOR GPA ROUTINE
000016' 177777                  WB.SD1: .WORD 177777                    ;PATTERN SEEDS SELECTED BY THE
000020' 177776                  WB.SD2: .WORD 177776                    ;       INITALIZATION ROUTINES.  THESE
                                                                        ;       SEEDS CAN BE CHANGED IF IT IS DESIRED
                                                                        ;       TO CHANGE THE PATTERNS THAT ARE
                                ;                                       ;       PRODUCED.
                                ;
                                ;****************
                                ;
                                ;
                                ;       THE MODULE IS SET UP TO PRODUCE THE FOLLOWING PATTERNS:
                                ;
                                ;       1. BUBBLE ALTERNATING WITH ALL 1'S.
                                ;        ...,177777,177776,177777,177775,177777,...
                                ;
                                ;       2. BUBBLE ALTERNATING WITH ALL 0'S.
                                ;       ...,000000,177776,000000,177775,000000,...
                                ;
                                ;       3. TWO ALTERNATING CONSTANTS.  ONE CONSTANT IS ALL 1'S, AND ONE
                                ;          CONSTANT HAS A BUBBLE WHICH SHIFTS AFTER EACH BLOCK.
                                ;             .
                                ;             .
```

```
                        ;          .
                        ;       177777,177776,177777,177776,...
                        ;       177777,177775,177777,177775,...
                        ;          .
                        ;          .
                        ;          .
                        ;
                        ;       4. TWO ALTERNATING CONSTANTS.  ONE CONSTANT IS ALL 0'S, AND
                        ;          ONE CONSTANT HAS A BUBBLE WHICH SHIFTS AFTER EACH BLOCK.
                        ;          .
                        ;          .
                        ;          .
                        ;       000000,177776,000000,177776,...
                        ;       000000,177775,000000,177775,...
                        ;          .
                        ;          .
                        ;          .
                        ;
```

```
                                .SBTTL WSTBUS ROUTINE

                                ROUTINE WSTBUS <TABL>                                   WSTBUS:

                                ;+
                                ; SAVE REGISTERS
                                ;-

                                CALL SAVREG                                             JSR     PC,SAVREG

                                ;+
                                ; SET R0 TO THE START OF THE DATA TABLE
                                ;-

                                LET R0 := TABL(R5)                                      MOV     TABL(R5),R0

                                ;+
                                ; INITIALIZE THE DONE FLAG, THE PATTERN WORD COUNT, AND
                                ; THE ROTATION COUNTER.
                                ;-

                                LET WB.DONE := #1                                       MOV     #1,WB.DONE
                                LET WB.CNT := #0                                        CLR     WB.CNT
                                LET WB.ROTCNT := #0                                     CLR     WB.ROTCNT

                                ;+
                                ; SEE IF MEMORY MANAGEMENT IS IN OPERATION.
                                ;-

                                IF #KTSTAT SETIN DT.STO(R0) THEN                        BIT     #KTSTAT,DT.STO(R
                                                                                        BEQ     50002$

                                ;+
                                ; SAVE THE CONTENTS OF KIPAR6 AND THEN SET IT UP FOR USE AS THE
                                ; MEMORY LOCATION POINTER BY POINTING IT TO THE 4K BANK THAT
                                ; CONTAINS THE TOP OF THE EXERCISER.
                                ;-

                                        PUSH @#KIPAR6                                   MOV     @#KIPAR6,-(SP)
                                        LET R1 := DT.ESIZ(R0) SHIFT #-6                 MOV     DT.ESIZ(R0),R1
                                                                                        ASR     R1
                                                                                        ASR     R1
                                                                                        ASR     R1
```

```
(7)  000076' 006201                                                                    ASR     R1
(7)  000100' 006201                                                                    ASR     R1
(7)  000102' 006201                                                                    ASR     R1
674  000104'                        LET @#KIPAR6 := R1 CLR.BY #176177
(4)  000104' 010137  172354                                                            MOV     R1,@#KIPAR6
(6)  000110' 042737  176177  172354                                                    BIC     #176177,@#KIPAR6
675
676
677                                 ;+
678                                 ; SET R1 TO SELECT KIPAR6 AND POINT TO THE FIRST WORD ABOVE
679                                 ; THE TOP OF THE EXERCISER.
680                                 ;-
681
682  000116'                        LET R1 := DT.ESIZ(R0) CLR.BY #160000
(4)  000116' 016001  000044                                                            MOV     DT.ESIZ(R0),R1
(6)  000122' 042701  160000                                                            BIC     #160000,R1
683  000126'                        LET R1 := R1 SET.BY #140000
(6)  000126' 052701  140000                                                            BIS     #140000,R1
684  000132'                        LET R1 := R1 + #2
(6)  000132' 062701  000002                                                            ADD     #2,R1
685
686
687                                 ;+
688                                 ; WHILE NOT AT THE TOP OF MEMORY...
689                                 ;-
690
691  000136'                        WHILE @#KIPAR6 LO DT.SSIZ(R0) DO
(4)  000136'                                                                   50003$:
(6)  000136' 023760  172354  000046                                                    CMP     @#KIPAR6,DT.SSIZ
(9)  000144' 103014                                                                    BHIS    50004$
692
693                                 ;+
694                                 ; GENERATE PATTERN WORDS, TWO AT A TIME, AND WRITE THEM INTO MEMORY,
695                                 ; ADJUSTING THE PAR- AND REGISTER- CONTENTS AS NECESSARY.
696                                 ;-
697
698  000146'                               WHILE R1 LO #157776 DO
(4)  000146'                                                                   50005$:
(6)  000146' 020127  157776                                                            CMP     R1,#157776
(9)  000152' 103003                                                                    BHIS    50006$
699  000154'                                      CALL GENPAT
(3)  000154' 004767  000142                                                            JSR     PC,GENPAT
700  000160'                               ENDDO
(4)  000160' 000772                                                                    BR      50005$
(3)  000162'                                                                   50006$:
701  000162'                               LET R1 := #140000
(4)  000162' 012701  140000                                                            MOV     #140000,R1
702  000166'                               LET @#KIPAR6 := @#KIPAR6 + #200
(6)  000166' 062737  000200  172354                                                    ADD     #200,@#KIPAR6
703
704  000174'                        ENDDO
(4)  000174' 000760                                                                    BR      50003$
(3)  000176'                                                                   50004$:
705
706
707                                 ;+
```

```
                                ; RESTORE KIPAR6 TO ITS ORIGINAL VALUE.
                                ;-

000176'                                 POP @#KIPAR6
(2) 000176' 012637  172354                                                      MOV     (SP)+,@#KIPAR6

                                ;+
                                ; ELSE MEMORY MANAGEMENT IS OFF.
                                ;-

000202'                         ELSE
(4) 000202' 000444                                                              BR      50007$
(3) 000204'                                                                   50002$:

                                ;+
                                ; USE WB.SAV AS THE HIGH MEMORY LIMIT.  BECAUSE OF
                                ; ADDRESSING LIMITATIONS WITHOUT KT, IF TOTAL MEMORY
                                ; IS GREATER THAN OR EQUAL TO 28K THEN THE HIGH LIMIT IS
                                ; 28K.  OTHERWISE, THE HIGH LIMIT IS THE TOP OF MEMORY.
                                ;-

000204'                                 IF DT.SSIZ(R0) HIS #1600 THEN
(6) 000204' 026027  000046  001600                                              CMP     DT.SSIZ(R0),#160
(9) 000212' 103404                                                              BLO     50010$
000214'                                         LET WB.SAV := #160000
(4) 000214' 012767  160000  177556                                              MOV     #160000,WB.SAV
000222'                                 ELSE
(4) 000222' 000417                                                              BR      50011$
(3) 000224'                                                                   50010$:
000224'                                         LET WB.SAV := DT.SSIZ(R0) SHIFT #+6
(4) 000224' 016067  000046  177546                                              MOV     DT.SSIZ(R0),WB.S
(7) 000232' 006367  177542                                                      ASL     WB.SAV
(7) 000236' 006367  177536                                                      ASL     WB.SAV
(7) 000242' 006367  177532                                                      ASL     WB.SAV
(7) 000246' 006367  177526                                                      ASL     WB.SAV
(7) 000252' 006367  177522                                                      ASL     WB.SAV
(7) 000256' 006367  177516                                                      ASL     WB.SAV
000262'                                 ENDIF
(4) 000262'                                                                   50011$:

                                ;+
                                ; SUBTRACT 2 FROM THE HIGH LIMIT, SO WE CAN WRITE TWO WORDS AT
                                ; A TIME INTO MEMORY WITHOUT TRAPPING OUT AT THE TOP OF CORE.
                                ;-

000262'                                 LET WB.SAV := WB.SAV - #2
(6) 000262' 162767  000002  177510                                              SUB     #2,WB.SAV

                                ;+
                                ; SET UP R1 AS THE MEMORY ADDRESS POINTER
                                ;-
000270'                                 LET R1 := DT.ESIZ(R0) + #2
(4) 000270' 016001  000044                                                      MOV     DT.ESIZ(R0),R1
```

```
(6)  000274' 062701  000002                                                    ADD     #2,R1
746
747
748                                 ;+
749                                 ; WHILE NOT AT TOP OF MEMORY...
750                                 ;-
751
752  000300'                                WHILE R1 LO WB.SAV DO
(4)  000300'                                                          50012$:
(6)  000300' 020167  177474                                                    CMP     R1,WB.SAV
(9)  000304' 103003                                                            BHIS    50013$
753
754
755                                 ;+
756                                 ; CALL THE PATTERN GENERATION ROUTINE.
757                                 ;-
758
759  000306'                                    CALL GENPAT
(3)  000306' 004767  000010                                                    JSR     PC,GENPAT
760
761
762  000312'                                ENDDO
(4)  000312' 000772                                                            BR      50012$
(3)  000314'                                                          50013$:
763  000314'                        ENDIF                                  50007$:
(4)  000314'
764
765
766                                 ;+
767                                 ; RESTORE REGISTERS AND RETURN.
768                                 ;-
769
770  000314'                        CALL RESREG
(3)  000314' 004767  000000G                                                   JSR     PC,RESREG
771
772
773  000320'                        ENDRTN                                 50000$:
(3)  000320'                                                          50001$:
(3)  000320'                                                                   RTS     PC
(2)  000320' 000207
774
775
```

```
                                ;+
                                ; *** SUBROUTINES ***
                                ;-


                                .SBTTL ROUTINE TO CONTROL THE GENERATION OF TWO 16-BIT PATTERN WORDS.

000322'                         ROUTINE GENPAT                                          GENPAT:
(2)  000322'


                                ;+
                                ; IF WE HAVE COMPLETED ONE PATTERN, CLEAR THE DONE FLAG AND SET UP
                                ; TO GET THE NEXT PATTERN ROUTINE.  THEN RE-INITIALIZE THE PATTERN WORDS
                                ; (R3 AND R4) BY COMPLEMENTING ONE SEED AND LEAVING THE OTHER CONSTANT.
                                ;-

000322'                         IF WB.DONE EQ #1 THEN                                           CMP     WB.DONE,#1
(6)  000322' 026727  177454  000001                                                            BNE     50002$
(9)  000330' 001020
000332'                                 LET WB.DONE := #0                                       CLR     WB.DONE
(4)  000332' 005067  177444
000336'                                 IF R2 EQ #PAT1 THEN                                     CMP     R2,#PAT1
(6)  000336' 020227  000432'                                                                    BNE     50003$
(9)  000342' 001003
000344'                                         LET R2 := #PAT2                                 MOV     #PAT2,R2
(4)  000344' 012702  000502'
                                        ELSE                                                    BR      50004$
000350'                                                                                 50003$:
(4)  000350' 000402
(3)  000352'
                                                LET R2 := #PAT1                                 MOV     #PAT1,R2
000352'
(4)  000352' 012702  000432'
                                        ENDIF                                           50004$:
000356'
(4)  000356'

000356'                                 LET WB.SD1 := COMP WB.SD1                               COM     WB.SD1
(6)  000356' 005167  177434
                                        LET R3 := WB.SD1                                        MOV     WB.SD1,R3
000362'
(4)  000362' 016703  177430
                                        LET R4 := WB.SD2                                        MOV     WB.SD2,R4
000366'
(4)  000366' 016704  177426

                                ENDIF                                                   50002$:
000372'
(4)  000372'


                                ;+
                                ; CALL THE PROPER PATTERN ROUTINE.
                                ; R2 = PAT1 OR PAT2
                                ;-

000372'                         CALL (R2)                                                       JSR     PC,(R2)
(3)  000372' 004712


                                ;+
```

```
                        ; MOVE THE TWO PATTERN WORDS INTO MEMORY.  AFTER EACH MOVE, CHECK FOR A
                        ; WRITE ERROR.  IF AN ERROR IS FOUND, GO REPORT IT.
                        ;-

000374'                 LET (R1) := R3
(4) 000374' 010311                                                      MOV     R3,(R1)
000376'                 IF (R1)+ NE R3 THEN
(6) 000376' 022103                                                      CMP     (R1)+,R3
(9) 000400' 001404                                                      BEQ     50005$
000402'                         LET WB.TMP := R3
(4) 000402' 010367  177404                                              MOV     R3,WB.TMP
000406'                         CALL ERROR
(3) 000406' 004767  000160                                              JSR     PC,ERROR
000412'                 ENDIF
(4) 000412'                                                     50005$:
000412'                 LET (R1) := R4
(4) 000412' 010411                                                      MOV     R4,(R1)
000414'                 IF (R1)+ NE R4 THEN
(6) 000414' 022104                                                      CMP     (R1)+,R4
(9) 000416' 001404                                                      BEQ     50006$
000420'                         LET WB.TMP := R4
(4) 000420' 010467  177366                                              MOV     R4,WB.TMP
000424'                         CALL ERROR
(3) 000424' 004767  000142                                              JSR     PC,ERROR
000430'                 ENDIF
(4) 000430'                                                     50006$:

000430'                 ENDRTN
(3) 000430'                                                     50000$:
(3) 000430'                                                     50001$:
(2) 000430' 000207                                                      RTS     PC
```

```
                        .SBTTL ROUTINE TO PRODUCE A ROTATING PATTERN ALTERNATING WITH A CONSTANT.

000432'                 ROUTINE PAT1                                            PAT1:
000432'

                        ;+
                        ; CALCULATE A NEW SECOND PATTERN WORD.  (THE FIRST PATTERN WORD IS
                        ; CONSTANT.)
                        ;-

000432'                 IF #BIT15 SETIN R4 THEN                                         BIT     #BIT15,R4
000432' 032704  100000                                                                  BEQ     50002$
000436' 001402
000440'                         INLINE <SEC>                                            SEC
000440' 000261
000442'                 ELSE                                                            BR      50003$
000442' 000401                                                                  50002$:
000444'                         INLINE <CLC>                                            CLC
000444' 000241
000446'                 ENDIF                                                   50003$:
000446'
000446'                 LET R4 := R4 ROTATE #+1                                         ROL     R4
000446' 006104

                        ;+
                        ; ADD TWO TO THE PATTERN COUNT AND SEE IF WE HAVE REACHED THE MAXIMUM.
                        ; IF YES, THEN CLEAR THE COUNT AND SET THE DONE FLAG.
                        ;-

000450'                 LET WB.CNT := WB.CNT + #2                                       ADD     #2,WB.CNT
000450' 062767  000002  177326
000456'                 IF WB.CNT GE WB.MAX THEN                                        CMP     WB.CNT,WB.MAX
000456' 026767  177322  177322                                                          BLT     50004$
000464' 002405
000466'                         LET WB.CNT := #0                                        CLR     WB.CNT
000466' 005067  177312
000472'                         LET WB.DONE := #1                                       MOV     #1,WB.DONE
000472' 012767  000001  177302
000500'                 ENDIF                                                   50004$:
000500'

                        ENDRTN                                                  50000$:
000500'                                                                         50001$:
000500'                                                                                 RTS     PC
000500' 000207
```

```
                                .SBTTL ROUTINE TO PRODUCE A BLOCK OF TWO ALTERNATING CONSTANT VALUES.


                                ;+
                                ; AFTER A BLOCK IS COMPLETE, ONE OF THE WORDS IS CHANGED (THE OTHER
                                ; ALWAYS REMAINS CONSTANT) AND ANOTHER BLOCK IS PRODUCED.  A
                                ; ROTATING PATTERN IS USED IN THE WORD THAT IS CHANGED.  THE PATTERN IS
                                ; COMPLETE WHEN THE CHANGING WORD HAS ROTATED ALL THE WAY AROUND
                                ; TO THE STARTING POINT.
                                ;-

000502'                         ROUTINE PAT2
000502'                                                                 PAT2:


                                ;+
                                ; ADD TWO TO THE PATTERN WORD COUNT.
                                ; IF THE PATTERN WORD COUNT HAS REACHED THE MAXIMUM, THEN ROTATE THE
                                ; CHANGING WORD.  IF THE WORD HAS ROTATED COMPLETELY, THEN
                                ; SET THE DONE FLAG.
                                ;-

000502'                         LET WB.CNT := WB.CNT + #2
                                                                        ADD     #2,WB.CNT
000502' 062767  000002  177274
                                IF WB.CNT GE WB.MAX THEN
000510'                                                                 CMP     WB.CNT,WB.MAX
000510' 026767  177270  177270                                          BLT     50002$
000516' 002424
000520'                                 LET WB.CNT := #0
                                                                        CLR     WB.CNT
000520' 005067  177260
                                        IF #BIT15 SETIN R4 THEN
000524'                                                                 BIT     #BIT15,R4
000524' 032704  100000                                                  BEQ     50003$
000530' 001402
000532'                                         INLINE <SEC>
                                                                        SEC
000532' 000261
                                        ELSE
000534'                                                                 BR      50004$
000534' 000401
                                                                50003$:
000536'
                                        INLINE <CLC>
000536'                                                                 CLC
000536' 000241
                                        ENDIF
000540'                                                         50004$:
000540'
                                        LET R4 := R4 ROTATE #+1
000540'                                                                 ROL     R4
000540' 006104
                                        LET WB.ROTCNT := WB.ROTCNT + #1
000542'                                                                 INC     WB.ROTCNT
000542' 005267  177242
                                        IF WB.ROTCNT EQ #^D16 THEN
000546'                                                                 CMP     WB.ROTCNT,#^D16
000546' 026727  177236  000020                                          BNE     50005$
000554' 001005
                                                LET WB.ROTCNT := #0
000556'                                                                 CLR     WB.ROTCNT
000556' 005067  177226
                                                LET WB.DONE := #1
000562'                                                                 MOV     #1,WB.DONE
000562' 012767  000001  177212
                                        ENDIF
000570'                                                         50005$:
000570'
                                ENDIF                           50002$:
000570'
```

```
   904  000570'                        ENDRTN
   (3)  000570'                                                    50000$:
   (3)  000570'                                                    50001$:
   (2)  000570' 000207                                                     RTS     PC
   905
   906
```

```
908
909                                     .SBTTL ROUTINE TO REPORT MEMORY WRITE ERRORS.
910
911
912                                     ;+
913                                     ; THIS ROUTINE PROCESSES BAD MEMORY TRANSFERS.  IT POINTS TO THE BAD
914                                     ; ADDRESS, PUTS THAT ADDRESS IN A TABLE, AND CALLS GPA TO
915                                     ; CONVERT THAT ADDRESS FROM VIRTUAL TO PHYSICAL.  THEN BADMEM IS CALLED
916                                     ; TO OUTPUT A MESSAGE.  FINALLY, THE ADDRESS POINTER IS RESTORED TO
917                                     ; THE NEXT MEMORY LOCATION.
918                                     ;-
919
920  000572'                            ROUTINE ERROR                                       ERROR:
(2)  000572'
921
922  000572'                            INLINE <TST     -(R1)>                                      TST     -(R1)
(2)  000572' 005741
923  000574'                            LET WB.TBL := R1                                            MOV     R1,WB.TBL
(4)  000574' 010167  177214
924  000600'                            CALL GPA IN <R0,#WB.TBL>                                    MOV     R5,-(SP)
(3)  000600' 010546                                                                                 MOV     #WB.TBL,-(R5)
(5)  000602' 012745  000014'                                                                        MOV     R0,-(R5)
(4)  000606' 010045                                                                                 JSR     PC,GPA
(3)  000610' 004767  000000G                                                                        MOV     (SP)+,R5
(3)  000614' 012605
925  000616'                            CALL BADMEM IN <R0,#WB.TBL+2,WB.TMP,(R1)>                   MOV     R5,-(SP)
(3)  000616' 010546                                                                                 MOV     (R1),-(R5)
(7)  000620' 011145                                                                                 MOV     WB.TMP,-(R5)
(6)  000622' 016745  177164                                                                         MOV     #WB.TBL+2,-(R5)
(5)  000626' 012745  000016'                                                                        MOV     R0,-(R5)
(4)  000632' 010045                                                                                 JSR     PC,BADMEM
(3)  000634' 004767  000000G                                                                        MOV     (SP)+,R5
(3)  000640' 012605
926  000642'                            INLINE <TST     (R1)+>                                      TST     (R1)+
(2)  000642' 005721
927  000644'                            ENDRTN                                              50000$:
(3)  000644'                                                                            50001$:
(3)  000644'                                                                                RTS     PC
(2)  000644' 000207
928
929
930          000001                     .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BADMEM= ****** G
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERROR  000572R
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
GENPAT 000322R
GPA   = ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PAT1   000432R
PAT2   000502R
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
```

```
SYSCNT= 000052          WBSTAT= 000040          $BGNLE= 177777          $F$WHI= 000120          $TSK4 = 050006
SYSERR= 000100          WBUFEA= 000136          $ERFLG= 000400          $F$YES= 000402          $$ARGC= 000000
TABL  = 000000          WBUFPA= 000134          $F$AND= 000310          $IFLEV= 177777          $$BYTE= 000403
TMPIO = 000002          WBUFRQ= 000140          $F$BAD= 000401          $ISK0 = 000001          $$CASE= 000000
TQOVF = 000002          WBUFSZ= 000142          $F$BLA= 000170          $ISK1 = 000001          $$DST = 000000
UIPAR0= 177640          WB.CNT  000004R         $F$CAS= 000150          $LOCTA= 177777          $$ELOC= 000402
UIPAR1= 177642          WB.DON  000002R         $F$DEC= 000220          $LSTIN= 000001          $$ERFL= 000000
UIPAR2= 177644          WB.MAX  000006R         $F$DO = 000340          $LSTTA= 000001          $$FLAG= 000001
UIPAR3= 177646          WB.ROT  000010R         $F$FAL= 000405          $NESTL= 177777          $$FROM= 000000
UIPAR4= 177650          WB.SAV  000000R         $F$GOO= 000400          $NSK0 = 000300          $$LOC = 000554R
UIPAR5= 177652          WB.SD1  000016R         $F$IF = 000110          $NSK1 = 000110          $$LOCN= 000000
UIPAR6= 177654          WB.SD2  000020R         $F$INC= 000210          $NSK2 = 000110          $$REG = 177777
UIPAR7= 177656          WB.TBL  000014R         $F$LOO= 000200          $NSK3 = 000120          $$RETU= 000000
UIPDR0= 177600          WB.TMP  000012R         $F$NAM= 000160          $SAVLE= 177777          $$RTN1= 050000
UIPDR1= 177602          WDFR  = 000116          $F$NO = 000403          $SSK0 = 050013          $$RTN2= 050001
UIPDR2= 177604          WDTO  = 000114          $F$OR = 000320          $TAGLE= 177777          $$SRC = 000000
UIPDR3= 177606          WSTBUS  000022RG        $F$RTI= 000350          $TAGNU= 050002          $$TGSV= 000000
UIPDR4= 177610          WTINRE= 000352          $F$RTN= 000300          $TEMP = 000300          $$TGS1= 000000
UIPDR5= 177612          WTWHMI= 000222          $F$SEL= 000140          $TSK0 = 050002          $$TGS2= 000000
UIPDR6= 177614          XFLAG = 000005          $F$THE= 000330          $TSK1 = 050005          $$TO  = 000004
UIPDR7= 177616          XOFF  = 000023          $F$TRU= 000404          $TSK2 = 050013          $$$TAG= 050000
WASADR= 000104          XON   = 000021          $F$UNT= 000130          $TSK3 = 050005          .     = 000646R

. ABS.  000000      000
        000646      001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:WSTBUS,DSKZ:WSTBUS=SPMAC/ML,EQUATE,WSTBUS
 RUN-TIME: 20 10 .4 SECONDS
 RUN-TIME RATIO: 48/31=1.5
 CORE USED:  14K  (27 PAGES)
```

```
                                    .TITLE  KBDRV - KEYBOARD DRIVER ROUTINE
                                    .IDENT  /V0.0/
                                    ;++
                                    ; MODULE PACKAGE NAME:
                                    ;       KBDRV
                                    ;
                                    ; FUNCTIONAL DECSRIPTION:
                                    ;       THIS MODULE CONTAINS THE TWO ROUTINES: KBINI AND KBINT, WHICH TOGETHER
                                    ;       MAKE UP THE KEYBOARD DRIVER.
                                    ;
                                    ;
                                    ;--
```

```
                                .SBTTL  KBDRV - COMMON DEFINITIONS AND REFERENCES


                                        .MCALL  STRUCT
000000'                                 STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
        000001                          $LSTIN = 1
        000001                          $LSTTAG = 1


                                ;
                                ;***************
                                ;
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  KB.INP
                                .GLOBL  KB.RBP
                                .GLOBL  KB.RBF
                                .GLOBL  KB.XFLAG
                                .GLOBL  KBINT
                                .GLOBL  KBINI
                                ;
                                ;***************
                                ;
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  DTABLE                  ;DTABLE ADDRESS
                                .GLOBL  CTRLOF                  ;ENABLE/DISABLE TERMINAL OUTPUT FLAG
                                .GLOBL  OV.KBBUF                ;KEYBOARD BUFFER OVERLAY
                                .GLOBL  OV.KBSIZ                ;KEYBOARD BUFFER SIZE
                                .GLOBL  OV.HIKB                 ;HIGH BUFFER ADDRESS
                                .GLOBL  PCTRLC                  ;PROCESS A CTRLC
                                .GLOBL  DX.KFL
                                .GLOBL  DX.R5                   ;MONITOR R5
                                .GLOBL  DX.SP                   ;MONITOR R6
                                .GLOBL  MD.BSY                  ;MESSAGE BUSY FLAG
                                ;
                                ;***************
                                ;
                                ; DL11  DEVICE REGISTER DEFINITIONS
                                ;
                                .GLOBL  RCSR                    ;RECEIVER CONTROL AND STATUS REGISTER
                                .GLOBL  RBUF                    ;RECEIVER DATA BUFFER REGISTER
                                .GLOBL  XCSR                    ;TRANSMITTER CONTROL AND STATUS REGISTER
                                ;
                                ;***************
                                ;
                                ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
                                ;
000000' 000000                  KB.XFLAG: .WORD 0               ;XOFF FLAG WORD INDICATOR
000002' 000000                  KB.CNT: .WORD   0               ;COUNT OF PREVIOUSLY RUBBED OUT CHARACTERS
000004' 000000                  KB.INP: .WORD   0               ;POINTER TO KEYBOARD INPUT BUFFER
000006' 000000                  KB.RBP: .WORD   0               ;POINTER TO NEXT CHAR TO BE RUBBED OUT
000010' 000000                  KB.RBF: .WORD   0               ;NUMBER OF RUBOUTS RECEIVED
```

```
   576  000012' 000000G                KB.KBSIZ: .WORD OV.KBSIZ        ;KEYBOARD BUFFER SIZE
   577  000014' 000000G                KB.HIKB: .WORD  OV.HIKB         ;KEYBOARD BUFFER HIGH LIMIT
   578  000016' 000000G                KB.KBBUF: .WORD OV.KBBUF        ;KEYBOARD BUFFER ADDRESS
   579
```

```
                        .SBTTL  KBINI - KEYBOARD I/O INITIATOR ROUTINE
                        ;++
                        ; ROUTINE NAME:
                        ;       KBINI
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THE I/O INITIATOR ROUTINE PERFORMS THE NECESSARY
                        ;       SET-UP FUNCTIONS, SUCH AS CLEARING THE KEYBOARD BUFFER AND RESETTING
                        ;       THE APPROPRIATE POINTERS TO ALLOW THE NEXT KEYBOARD COMMAND TO BE
                        ;       ENTERED. A RETURN TO THE CALLING ROUTINE IS THEN EXECUTED.
                        ;
                        ; INPUTS:
                        ;       1. ADDRESS OF DATA TABLE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. ADDRESS OF NEXT CHAR TO BE ECHOED (DATA TABLE + DT.KBECH)
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       KEYBOARD CSR
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL KBINI IN <DTABLE>
                        ;
                        ; VERSION:
                        ;        0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                        .SBTTL  KBINI - KBINI ROUTINE

                                ROUTINE KBINI <DTA>
000020'                                                         KBINI:
(2)  000020'

                        ;+
                        ; DO THE NECESSARY SETUP STUFF, WHICH CONSISTS OF GETTING THE DATA TABLE
                        ; ADDRESS
                        ;-
                                PUSH    R0,R1
000020'                                                                 MOV     R0,-(SP)
(2)  000020' 010046                                                     MOV     R1,-(SP)
(3)  000022' 010146
                                LET R1 := DTA(R5)
000024'                                                                 MOV     DTA(R5),R1
(4)  000024' 016501  000000

                        ;+
                        ; INITIALIZE THE INPUT, ECHO, AND RUBOUT POINTERS,  AS
                        ; WELL AS THE RUBOUT FLAG
                        ;-
                                LET R0 := KB.KBBUF
000030'                                                                 MOV     KB.KBBUF,R0
(4)  000030' 016700  177762
                                LET KB.INP := R0
000034'                                                                 MOV     R0,KB.INP
(4)  000034' 010067  177744
                                LET DT.KBECH(R1) := R0
000040'                                                                 MOV     R0,DT.KBECH(R1)
(4)  000040' 010061  000024
                                LET KB.RBP := #0
000044'                                                                 CLR     KB.RBP
(4)  000044' 005067  177736
                                LET KB.RBF := #0
000050'                                                                 CLR     KB.RBF
(4)  000050' 005067  177734

                        ;+
                        ; ZERO THE KEYBOARD INPUT BUFFER
                        ;-
                                LET R1 := KB.KBSIZ
000054'                                                                 MOV     KB.KBSIZ,R1
(4)  000054' 016701  177732
                                WHILE R1 GT #0 DO
000060'                                                         50002$:
(4)  000060'                                                            TST     R1
(6)  000060' 005701                                                     BLE     50003$
(9)  000062' 003403
                                    LET (R0)+ :B= #0
000064'                                                                 CLRB    (R0)+
(4)  000064' 105020
                                    LET R1 := R1 - #1
000066'                                                                 DEC     R1
(6)  000066' 005301
                                ENDDO
000070'                                                                 BR      50002$
(4)  000070' 000773                                             50003$:
(3)  000072'

                        ;+
                        ; RESTORE THE REGISTERS, ENABLE INTERRUPTS
                        ; AND RETURN TO CALLER
                        ;-
                                POP     R1,R0
000072'                                                                 MOV     (SP)+,R1
(2)  000072' 012601                                                     MOV     (SP)+,R0
(3)  000074' 012600
                                LET @RCSR := #<IE>
000076'
```

```
    (4)  000076' 012777  000100  000000G          MOV     #<IE>,@RCSR
    660  000104'                                 ENDRTN
    (3)  000104'                                50000$:
    (3)  000104'                                50001$:
    (2)  000104' 000207                                 RTS     PC
```

```
                        .SBTTL  KBINT - KEYBOARD INTERRUPT SERVICE ROUTINE
                        ;++
                        ; ROUTINE NAME:
                        ;       KBINT
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THE INTERRUPT SERVICE ROUTINE IS ENTERED DIRECTLY
                        ;       VIA THE HARDWARE INTERRUPT VECTOR AND PERFORMS THE NEC-
                        ;       ESSARY FUNCTIONS TO  HANDLE THE INTERRUPT. UPON RECEIVING
                        ;       A LINE TERMINATOR EVENT, CONTROL IS PASSED TO THE APPROPRIATE
                        ;       SERVICE ROUTINE, OR THE EVENT-PENDING FLAG IS SET.
                        ;       ALSO, ALL LOWER CASE INPUT IS CONVERTED TO UPPER CASE
                        ;
                        ; INPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       NONE
                        ;
                        ; OUTPUTS:
                        ;       EVENT CODE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       1. CTRLOF                                       ;CONTROL-O FLAG
                        ;       2. DX.KFL
                        ;       3. DTABLE
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       IF XOFF (^S) IS GENERATED BY KEYBOARD, THE  RESULT WILL
                        ;       BE TO DISABLE TTDRV UNTIL XON (^Q) IS GENERATED.
                        ;
                        ; CALLING SEQUENCE:
                        ;       ENTERED DIRECTLY VIA THE HARDWARE INTERRUPT MECHANISM
                        ;
                        ; VERSION:
                        ;        0.0
                        ;
                        ;       EDIT            BY              DATE            REASON
                        ;--
```

```
                                .SBTTL  KBINT - ROUTINE KBINT


                                ;+
                                ; DISABLE INTERRUPTS, GET THE CHARACTER ONTO THE STACK
                                ; AND STRIP-OFF THE ASCII PARITY BIT
                                ;-

000106'                         IROUTINE KBINT                                          KBINT:
(2)  000106'
000106'                         LET @RCSR := @RCSR CLR.BY #<IE>                                 BIC     #<IE>,@RCSR
(6)  000106' 042777  000100  000000G
000114'                         LET -(SP) :B= @RBUF                                     MOVB    @RBUF,-(SP)
(4)  000114' 117746  000000G
000120'                         LET (SP) := (SP) CLR.BY #177600                         BIC     #177600,(SP)
(6)  000120' 042716  177600

                                ;+
                                ; CONVERT LOWER CASE TO UPPER CASE
                                ;-

000124'                         IFB (SP) GE #141 ANDB (SP) LE #172 THEN                 CMPB    (SP),#141
(6)  000124' 121627  000141                                                             BLT     50002$
(9)  000130' 002405                                                                     CMPB    (SP),#172
(6)  000132' 121627  000172                                                             BGT     50002$
(9)  000136' 003002
000140'                                 LET (SP) :B= (SP) CLR.BY #BIT05                 BICB    #BIT05,(SP)
(6)  000140' 142716  000040
000144'                         ENDIF                                                   50002$:
(4)  000144'

                                ;+
                                ; IF XOFF(^S) THEN SET XFLAG
                                ;-

000144'                         IFB (SP) EQ #<XOFF> THEN                                CMPB    (SP),#<XOFF>
(6)  000144' 121627  000023                                                             BNE     50003$
(9)  000150' 001005
000152'                                 LET KB.XFLAG := #1                              MOV     #1,KB.XFLAG
(4)  000152' 012767  000001  177620
000160'                                 INLINE <JMP 2$>                                 JMP 2$
(2)  000160' 000167  000426
000164'                         ENDIF                                                   50003$:
(4)  000164'

                                ;+
                                ; IF XON(^Q) THEN CLEAR XFLAG AD IF MESSAGE IN PROGRESS
                                ; SET TTRANSMITTER INTERRUPT ENABLE
                                ;-

000164'                         IFB (SP) EQ #<XON> THEN                                 CMPB    (SP),#<XON>
(6)  000164' 121627  000021                                                             BNE     50004$
(9)  000170' 001012
000172'                                 LET KB.XFLAG := #0                              CLR     KB.XFLAG
(4)  000172' 005067  177602
000176'                                 IF MD.BSY NE #0 THEN
```

```
(6)  000176' 005767  000000G                                                    TST     MD.BSY
(9)  000202' 001403                                                             BEQ     50005$
748  000204'                        LET @XCSR := @XCSR SET.BY #<IE>
(6)  000204' 052777  000100  000000G                                           BIS     #<IE>,@XCSR
749  000212'                    ENDIF
(4)  000212'                                                                50005$:
750  000212'                    INLINE <JMP 2$>                                  JMP 2$
(2)  000212' 000167  000374
751  000216'                ENDIF                                            50004$:
(4)  000216'
752
753                             ;+
754                             ; IF THE CHARACTER JUST ENTERED IS A CONTROL-C, THEN RAISE THE
755                             ; PRIORITY TO 7, SWITCH TO MONITOR STACKS AND DISPATCH
756                             ; CONTROL TO PROCESS "CONTROL C" ROUTINE.
757                             ;-
758  000216'                    IFB (SP) EQ #<CTRLC> THEN                        CMPB    (SP),#<CTRLC>
(6)  000216' 121627  000003                                                     BNE     50006$
(9)  000222' 001017
759  000224'                        PUSH #PRI7                                   MOV     #PRI7,-(SP)
(2)  000224' 012746  000340
760  000230'                        PUSH #1$                                     MOV     #1$,-(SP)
(2)  000230' 012746  000236'
761  000234'                        INLINE <RTI>                                 RTI
(2)  000234' 000002
762  000236'                INLINE <1$:>                                     1$:
(2)  000236'
763  000236'                        LET R5 := DX.R5                              MOV     DX.R5,R5
(4)  000236' 016705  000000G
764  000242'                        LET SP := DX.SP                              MOV     DX.SP,SP
(4)  000242' 016706  000000G
765  000246'                        CALL PCTRLC IN <#DTABLE>                     MOV     R5,-(SP)
(3)  000246' 010546                                                             MOV     #DTABLE,-(R5)
(4)  000250' 012745  000000G                                                    JSR     PC,PCTRLC
(3)  000254' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000260' 012605
766  000262'                    ENDIF                                        50006$:
(4)  000262'
767
768                             ;+
769                             ; IF THE CHARACTER JUST ENTERED IS A CARRIAGE RETURN OR A LINE FEED, THEN
770                             ;-
771  000262'                    IFB (SP) EQ #<CR> ORB (SP) EQ #<LF> THEN         CMPB    (SP),#<CR>
(6)  000262' 121627  000015                                                     BEQ     50007$
(8)  000266' 001403                                                             CMPB    (SP),#<LF>
(6)  000270' 121627  000012                                                     BNE     50010$
(9)  000274' 001006                                                         50007$:
(6)  000276'
772
773                             ;+
774                             ; IF THERE HAVE BEEN RUBOUTS, THEN LOAD THE BUFFER WITH RUBOUT CHARACTER
775                             ; AND UPDATE INPUT BUFFER POINTER
776                             ;-
777  000276'                        IF KB.RBF NE #0 THEN                         TST     KB.RBF
(6)  000276' 005767  177506                                                     BEQ     50011$
(9)  000302' 001402
```

```
778  000304'                                    INLINE <JSR PC,22222$>                                  JSR PC,22222$
(2)  000304' 004767  000432
779  000310'                                    ENDIF
(4)  000310'                                                                                       50011$:
780
781                                             ;+
782                                             ; GO LOAD BUFFER
783                                             ;-
784
785  000310'                                    INLINE <BR      11111$>                                 BR      11111$
(2)  000310' 000546
786  000312'                                    ENDIF
(4)  000312'                                                                                       50010$:
787
788                                             ;+
789                                             ; IF THE CHARACTER IS A CONTROL-U, THEN SAVE IT IN THE
790                                             ; INPUT BUFFER, DEFER SERVICING AND THEN SET THE KEYBOARD FLAG.
791                                             ;-
792
793  000312'                                    IFB (SP) EQ #<CTRLU> THEN                               CMPB    (SP),#<CTRLU>
(6)  000312' 121627  000025
(9)  000316' 001010                                                                                     BNE     50012$
794  000320'                                    IF KB.RBF NE #0 THEN                                    TST     KB.RBF
(6)  000320' 005767  177464
(9)  000324' 001402                                                                                     BEQ     50013$
795  000326'                                            INLINE <JSR PC,22222$>                          JSR PC,22222$
(2)  000326' 004767  000410
796  000332'                                    ENDIF
(4)  000332'                                                                                       50013$:
797
798                                             ;+
799                                             ; GO LOAD UP INPUT CHARACTER INTO BUFFER
800                                             ;-
801
802  000332'                                    INLINE <JSR PC,33333$>                                  JSR PC,33333$
(2)  000332' 004767  000430
803
804                                             ;+
805                                             ; NOW LOAD UP <CR> AND <LF> AND LEAVE
806                                             ;-
807
808  000336'                                    INLINE <BR 11111$>                                      BR 11111$
(2)  000336' 000533
809  000340'                                    ENDIF
(4)  000340'                                                                                       50012$:
810                                             ;+
811                                             ; IF THE CHARACTER IS A CONTROL-O THEN SIMPLY TOGGLE THE
812                                             ; ENABLE/DISABLE FLAG AND UPDATE THE STACK
813                                             ;-
814
815  000340'                                    IFB (SP) EQ #<CTRLO> THEN                               CMPB    (SP),#<CTRLO>
(6)  000340' 121627  000017
(9)  000344' 001011                                                                                     BNE     50014$
816  000346'                                    IF CTRLOF NE #0 THEN                                    TST     CTRLOF
(6)  000346' 005767  000000G
(9)  000352' 001403                                                                                     BEQ     50015$
```

```
817  000354'                                                                                    CLR     CTRLOF
(4)  000354' 005067  000000G                                  LET CTRLOF := #0
818  000360'                                                                                    BR      50016$
(4)  000360' 000402                                           ELSE
(3)  000362'                                                                          50015$:
819  000362'                                                                                    INC     CTRLOF
(6)  000362' 005267  000000G                                  LET CTRLOF := CTRLOF + #1
820  000366'                                                                                    
(4)  000366'                                                  ENDIF
821  000366'                                                                          50016$:
(4)  000366' 000511                                   ELSE                                     BR      50017$
(3)  000370'                                                                          50014$:
822                                                   ;+
823                                                   ; IF THE CHARACTER IS A RUBOUT, DO THE NECESSARY  STUFF
824                                                   ;-
825
826  000370'                                                                                    CMPB    (SP),#<RUBOUT>
(6)  000370' 121627  000177                                   IFB (SP) EQ #<RUBOUT> THEN       BNE     50020$
(9)  000374' 001075
827
828                                                   ;+
829                                                   ; MAKE SURE THERE ARE CHARACTERS TO RUBOUT!!!!!!!!!
830                                                   ;-
831
832  000376'                                                                                    CMP     KB.INP,KB.KBBUF
(6)  000376' 026767  177402  177412                           IF KB.INP GT KB.KBBUF THEN       BLE     50021$
(9)  000404' 003465
833  000406'                                                                                    TST     KB.RBF
(6)  000406' 005767  177376                           IF KB.RBF EQ #0 OR KB.RBP GE KB.KBBUFF THEN
(8)  000412' 001404                                                                           BEQ     50022$
(6)  000414' 026767  177366  177374                                                           CMP     KB.RBP,KB.KBBUFF
(9)  000422' 002456                                                                           BLT     50023$
(6)  000424'                                                                          50022$:
834                                                   ;+
835                                                   ; IS IT THE FIRST RUBOUT?
836                                                   ;-
837  000424'                                                                                    TST     KB.RBF
(6)  000424' 005767  177360                                   IF KB.RBF EQ #0 THEN             BNE     50024$
(9)  000430' 001012
838                                                   ;+
839                                                   ; FOR THE FIRST RUBOUT, SET UP THE RUBOUT FLAG
840                                                   ; AND RUBOUT POINTER, AND STORE THE RUBOUT CHAR
841                                                   ; IN THE INPUT BUFFER
842                                                   ;-
843
844  000432'                                                                                    INC     KB.RBF
(6)  000432' 005267  177352                                   LET KB.RBF := KB.RBF + #1
845  000436'                                                                                    JSR PC,33333$
(2)  000436' 004767  000324                           INLINE <JSR PC,33333$>
846  000442'                                                                                    MOV     KB.INP,KB.RBP
(4)  000442' 016767  177336  177336                           LET KB.RBP := KB.INP - #2        SUB     #2,KB.RBP
(6)  000450' 162767  000002  177330
847  000456'                                                  ENDIF
(4)  000456'                                                                          50024$:
848
849                                                   ;+
```

```
                                ; DELETE THE EXISTING CHARACTER, BACK UP THE RUBOUT POINTER, AND IF
                                ; THERE ARE ANY PREVIOUSLY RUBBED OUT CHARACTERS IN THE BUFFER,
                                ; MOVE THE RUBOUT POINTER PAST THEM.
                                ;-
                                                LET @KB.INP :B= @KB.RBP
                                                                        MOVB    @KB.RBP,@KB.INP
                                                IF KB.INP LT KB.HIKB THEN
                                                                        CMP     KB.INP,KB.HIKB
                                                                        BGE     50025$
                                                        LET KB.INP := KB.INP + #1
                                                                        INC     KB.INP
                                                ENDIF
                                                                50025$:
                                                LET KB.RBP := KB.RBP - #1
                                                                        DEC     KB.RBP

                                                IFB @KB.RBP EQ #RUBOUT THEN
                                                                        CMPB    @KB.RBP,#RUBOUT
                                                                        BNE     50026$
                                                        LET KB.CNT := #0
                                                                        CLR     KB.CNT
                                                        REPEAT
                                                                50027$:
                                                                LET KB.RBP := KB.RBP - #1
                                                                        DEC     KB.RBP
                                                                LET KB.CNT := KB.CNT + #1
                                                                        INC     KB.CNT
                                                        UNTILB @KB.RBP EQ #RUBOUT
                                                                        CMPB    @KB.RBP,#RUBOUT
                                                                        BNE     50027$
                                                        REPEAT
                                                                50030$:
                                                                LET KB.RBP := KB.RBP - #1
                                                                        DEC     KB.RBP
                                                                LET KB.CNT := KB.CNT - #1
                                                                        DEC     KB.CNT
                                                        UNTIL KB.CNT EQ #0
                                                                        TST     KB.CNT
                                                                        BNE     50030$
                                                ENDIF
                                                                50026$:

                                                INLINE <BR 2$>
                                                                        BR 2$

                                        ENDIF
                                                                50023$:
                                ENDIF
                                                                50021$:
                                ;+
                                ; THERE AREN'T ANY CHARACTERS TO RUBOUT, STORE A RUBOUT AND LOAD CRLF
                                ; NOTE: THE LAST ENTRY IN KEYBOARD BUFFER IS OVERWRITTEN BY A <CR>
                                        INLINE <JSR PC,33333$>
                                                                        JSR PC,33333$
                                        INLINE <BR 11111$>
                                                                        BR 11111$
```

```
                                        ELSE
(4)  000566' 000411                                                                     BR      50031$
(3)  000570'                                                                    50020$:

                                    ;+
                                    ; IF IT IS NOT THE FIRST RUBOUT CHARACTER THEN CLEAR THE RUBOUT FLAG
                                    ; AND LOAD RUBOUT CHARACTER INTO BUFFER
                                    ;-

                                            IF KB.RBF NE #0 THEN
(6)  000570' 005767  177214                                                             TST     KB.RBF
(9)  000574' 001404                                                                     BEQ     50032$
                                                LET KB.RBF := #0
(4)  000576' 005067  177206                                                             CLR     KB.RBF
                                                    INLINE <JSR PC,22222$>
(2)  000602' 004767  000134                                                             JSR PC,22222$
                                            ENDIF
(4)  000606'                                                                    50032$:

                                    ;+
                                    ; IT IS NOT THE FIRST NONRUBOUT CHAR, SO SIMPLY
                                    ; STORE THE CHAR IN THE INPUT BUFFER, UPDATE THE
                                    ; POINTER AND DO A NICE CLEAN RETURN
                                    ;-

                                            INLINE <JSR PC,33333$>
(2)  000606' 004767  000154                                                             JSR PC,33333$
                                        ENDIF
(4)  000612'                                                                    50031$:
                                    ENDIF
(4)  000612'                                                                    50017$:

                                    ;+
                                    ; ENABLE INTERRUPTS AND RETURN
                                    ;-

                                    INLINE <2$:>
(2)  000612'                                                                            2$:
                                    LET SP := SP + #2
(6)  000612' 062706  000002                                                             ADD     #2,SP
                                    LET @RCSR := #<IE>
(4)  000616' 012777  000100  000000G                                                   MOV     #<IE>,@RCSR
                                    ENDRTI
(3)  000624'                                                                    50000$:
(3)  000624'                                                                    50001$:
(2)  000624' 000002                                                                     RTI

                                    ;+
                                    ; LINE TERMINATOR EXIT
                                    ; EITHER A <CR> OR <LF> TERMINATOR, OR A CONTROL-U HAS BEEN RECEIVED-
                                    ; SET THE FLAG
                                    ; AND LOAD <CR> AND <LF> INTO BUFFER, ALONG WITH UPDATING STACK
                                    ;-

                                    INLINE <11111$:>
(2)  000626'                                                                    11111$:
```

```
                        LET DX.KFL := #1                                MOV     #1,DX.KFL

                        PUSH R0,R1                                      MOV     R0,-(SP)
                                                                        MOV     R1,-(SP)

                        IF 4(SP) EQ #<LF> THEN                          CMP     4(SP),#<LF>
                                                                        BNE     50033$

                                LET R0 := #<LF>                         MOV     #<LF>,R0

                                LET R1 := #<CR>                         MOV     #<CR>,R1

                        ELSE                                            BR      50034$
                                                                50033$:
                                LET R0 := #<CR>                         MOV     #<CR>,R0

                                LET R1 := #<LF>                         MOV     #<LF>,R1

                        ENDIF                                   50034$:

                        LET @KB.INP :B= R0                              MOVB    R0,@KB.INP

                        IF KB.INP LT KB.HIKB THEN                       CMP     KB.INP,KB.HIKB
                                                                        BGE     50035$

                                LET KB.INP := KB.INP + #1               INC     KB.INP

                        ENDIF                                   50035$:

                        LET @KB.INP :B= R1                              MOVB    R1,@KB.INP

                        IF KB.INP LT KB.HIKB THEN                       CMP     KB.INP,KB.HIKB
                                                                        BGE     50036$

                                LET KB.INP := KB.INP + #1               INC     KB.INP

                        ENDIF                                   50036$:


                ;+
                ; NOW CLEAN UP STACK, RESTORING REGISTERS AND REMOVING INPUT CHARACTER.
                ;-

                        POP R1,R0                                       MOV     (SP)+,R1
                                                                        MOV     (SP)+,R0

                        INLINE <TST (SP)+>                              TST (SP)+

                        INLINE <RTI>                                    RTI



                ;+
                ; LOAD RUBOUT INTO BUFFER AND UPDATE POINTER
                ;-
```

```
950
951  000742'                          INLINE <22222$:>                        22222$:
(2)  000742'
952  000742'                                  LET @KB.INP :B= #<RUBOUT>       MOVB    #<RUBOUT>,@KB.IN
(4)  000742' 112777  000177  177034
953  000750'                                  IF KB.INP LT KB.HIKB THEN       CMP     KB.INP,KB.HIKB
(6)  000750' 026767  177030  177036                                          BGE     50037$
(9)  000756' 002002
954  000760'                                          LET KB.INP := KB.INP + #1   INC     KB.INP
(6)  000760' 005267  177020
955  000764'                                  ENDIF
(4)  000764'                                                                50037$:
956  000764'                                  INLINE <RTS PC>                 RTS PC
(2)  000764' 000207
957
958                                   ;+
959                                   ; LOAD CHARACTER INTO BUFFER
960                                   ;-
961
962  000766'                          INLINE <33333$:>                        33333$:
(2)  000766'
963  000766'                                  LET @KB.INP :B= 2(SP)           MOVB    2(SP),@KB.INP
(4)  000766' 116677  000002  177010
964  000774'                                  IF KB.INP LT KB.HIKB THEN       CMP     KB.INP,KB.HIKB
(6)  000774' 026767  177004  177012                                          BGE     50040$
(9)  001002' 002002
965  001004'                                          LET KB.INP := KB.INP + #1   INC     KB.INP
(6)  001004' 005267  176774
966  001010'                                  ENDIF
(4)  001010'                                                                50040$:
967  001010'                                  INLINE <RTS PC>                 RTS PC
(2)  001010' 000207
968           000001                  .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLOF= ****** G
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTA   = 000000
DTABLE= ****** G
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
DX.KFL= ****** G
DX.R5 = ****** G
DX.SP = ****** G
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KBINI   000020RG
KBINT   000106RG
KB.CNT  000002R
KB.HIK  000014R
KB.INP  000004RG
KB.KBB  000016R
KB.KBS  000012R
KB.RBF  000010RG
KB.RBP  000006RG
KB.XFL  000000RG
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MD.BSY= ****** G
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OV.HIK= ****** G
OV.KBB= ****** G
OV.KBS= ****** G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PCTRLC= ****** G
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUF  = ****** G
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RCSR  = ****** G
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
```

```
SR0    = 177572          UIPAR6= 177654          $ERFLG= 000400          $ISK1 = 000001          $TSK5 = 050030
SR1    = 177574          UIPAR7= 177656          $F$AND= 000310          $ISK2 = 000001          $$ARGC= 000000
SR2    = 177576          UIPDR0= 177600          $F$BAD= 000401          $ISK3 = 000001          $$BYTE= 000403
SR3    = 172516          UIPDR1= 177602          $F$BLA= 000170          $ISK4 = 000001          $$CASE= 000000
STAT   = 000026          UIPDR2= 177604          $F$CAS= 000150          $LOCTA= 177777          $$DST = 000000
STATBI= 064757           UIPDR3= 177606          $F$DEC= 000220          $LSTIN= 000001          $$ELOC= 000402
STAT1  = 000027          UIPDR4= 177610          $F$DO = 000340          $LSTTA= 000001          $$ERFL= 000000
SUSPND= 000001           UIPDR5= 177612          $F$FAL= 000405          $NESTL= 177777          $$FLAG= 000001
SVR0   = 000062          UIPDR6= 177614          $F$GOO= 000400          $NSK0 = 000110          $$FROM= 000000
SVR1   = 000064          UIPDR7= 177616          $F$IF = 000110          $NSK1 = 000110          $$LOC = 001002R
SVR2   = 000066          WASADR= 000104          $F$INC= 000210          $NSK2 = 000110          $$LOCN= 000000
SVR3   = 000070          WBSTAT= 000040          $F$LOO= 000200          $NSK3 = 000110          $$REG = 177777
SVR4   = 000072          WBUFEA= 000136          $F$NAM= 000160          $NSK4 = 000110          $$RETU= 000000
SVR5   = 000074          WBUFPA= 000134          $F$NO = 000403          $NSK5 = 000110          $$RTN1= 050000
SVR6   = 000076          WBUFRQ= 000140          $F$OR = 000320          $NSK6 = 000130          $$RTN2= 050001
SYSCNT= 000052           WBUFSZ= 000142          $F$RTI= 000350          $SAVLE= 177777          $$SRC = 000000
SYSERR= 000100           WDFR   = 000116         $F$RTN= 000300          $SSK0 = 050003          $$TGSV= 000000
TMPIO  = 000002          WDTO   = 000114         $F$SEL= 000140          $TAGLE= 177777          $$TGS1= 000000
TQOVF  = 000002          WTINRE= 000352          $F$THE= 000330          $TAGNU= 050041          $$TGS2= 000000
UIPAR0= 177640           WTWHMI= 000222          $F$TRU= 000404          $TEMP = 050040          $$TO   = 000001
UIPAR1= 177642           XCSR   = ****** G       $F$UNT= 000130          $TSK0 = 050040          $$$TAG= 050000
UIPAR2= 177644           XFLAG  = 000005         $F$WHI= 000120          $TSK1 = 050031          .      = 001012R
UIPAR3= 177646           XOFF   = 000023         $F$YES= 000402          $TSK2 = 050032
UIPAR4= 177650           XON    = 000021         $IFLEV= 177777          $TSK3 = 050023
UIPAR5= 177652           $BGNLE= 177777          $ISK0 = 000001          $TSK4 = 050026

. ABS.  000000     000
        001012     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:KBDRV,DSKZ:KBDRV=SPMAC/ML,EQUATE,KBDRV
 RUN-TIME: 24 15 .4 SECONDS
 RUN-TIME RATIO: 60/39=1.5
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE LPDRV  (LINE PRINTER OUTPUT DRIVER MODULE)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       LP
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE CONTAINS TWO MODULES - LPINI AND LPINT.
                        ;       THE LPINI IS THE INITIALIZATION ROUTINE FOR THE
                        ;       OUTPUT INTERRUPT HANDLING ROUTINE - LPINT.
                        ;       LPINT ACTUALLY PROCESSES ALL MESSAGES TO THE DEVICE.
                        ;
                        ;       A DESCRIPTION IS GIVEN IN EACH MODULE.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       -----   ----------  ----------  --------------------
                        ;
                        ;--
```

```
                        .PRINT  ;SPMAC: VERSION 1.1
                        ;****************                ******************************
                        ;               GLOBAL REFERENCES
                        .GLOBL LPCSR                    ;DEVICE REGISTER -    CONTROL & STATUS
                        .GLOBL LPBUF                    ;DEVICE REGISTER -    DATA BUFFER
                        .GLOBL CTRLOF                   ;FLAG - OUTPUT SUPPRESS
                        ;****************               ******************************
                        ;               REFERENCED BY OTHER MODULES
                        .GLOBL LPINT                    ;MODULE ENTRY POINT - LP HANDLER
                        .GLOBL LPINI                    ;MODULE ENTRY POINT - LP INITIATOR
                        ;****************               ******************************
                        ;               LOCAL EQUATES
                        LP.PCNT =       45              ;CHARACTER -    '%' PERCENT
                        LP.BKSLSH =     134             ;CHARACTER -    '\' BACKSLASH
                        ;****************               ******************************
                        ;               LOCAL STORAGE
000000' 000000          LP.FCH: 0                       ;CHARACTER -    FILL
000002' 000000          LP.PFC: 0                       ;COUNT -        FILL, PERMANENT
000004' 000000          LP.FCF: 0                       ;COUNT, FLAG -  FILL (FLAG SET NE 0)
000006' 000000          LP.EXT: 0                       ;POINTER -      LOCATION TO JUMP OUT INDIRECTLY
000010' 000000          LP.PTR: 0                       ;POINTER -      MESSAGE
000012' 000000          LP.TP:  0                       ;POINTER -      TEMPORARY BUFFER
000014' 000000          LP.COD: 0                       ;CODE -         MESSAGE
000016'                 LP.FLG:                         ;FLAGS -        PERCENT AND CONTROL-U
000016'     000         LP.PCF: .BYTE   0               ;FLAG -         PERCENT
000017'     000         LP.CUF: .BYTE   0               ;FLAG -         CONTROL-U
000020'     015         LP.PCB: .BYTE   15              ;+BUFFER -      PERCENT '%'             <CR>+
000021'     012                 .BYTE   12              ;+BUFFER        ECHO                    <LF>+
000022'     001                 .BYTE   1               ;+BUFFER -      TERMINATOR              <+1>+
000023'     136         LP.CUB: .BYTE   136             ;#BUFFER -      CONTROL-U                <^>#
000024'     125                 .BYTE   125             ;#BUFFER        ECHO                     <U>#
000025'     001         LP.BEN: .BYTE   1               ;#BUFFER -      TERMINATOR              <+1>#
                                                        ;FLAG -         (LP.BEN) END SPECIAL BUFFER

                        .EVEN
```

```
                        ;++
                        ; MODULE NAME:
                        ;       LPINI
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THE DEVICE OUTPUT STARTER MODULE (LPINI) SETS UP FOR
                        ;       THE DEVICE OUTPUT INTERRUPT HANDLING MODULE (LPINT).
                        ;
                        ; INPUTS:
                        ;       1. DATA TABLE ADDRESS
                        ;       2. MESSAGE ADDRESS
                        ;       3. MESSAGE TYPE CODE
                        ;       4. RETURN ADDRESS (DRIVER WILL JUMP TO AT END OF MESSAGE).
                        ;
                        ; IMPLICIT INPUTS:
                        ;       1. FILL CHARACTER
                        ;       2. FILL COUNT
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       NONE
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;       NONE
                        ;
                        ; SUBORDINATE MODULES CALLED:
                        ;       NONE
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL LPINI IN <DTADDR,TYPCOD,MSGADR,RETADR>
                        ;
                        ;               WHERE DTADDR = ADDRESS OF DATA TABLE
                        ;                     TYPCOD = TYPE CODE
                        ;                     MSGADR = ADDRESS OF MESSAGE
                        ;                     RETADR = RETURN ADDRESS THE DRIVER WILL JUMP TO
                        ;                              AT THE COMPLETION OF THE MESSAGE.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       -----   --------    ----------  ------------------
                        ;
                        ;--
```

```
                                ;+
                                ; INITIALIZE MESSAGE TYPE, MESSAGE
                                ; ADDRESS, FILL COUNT, AND FILL CHARACTER.
                                ; STUFF RETURN ADDRESS AT LP.EXT,
                                ; ENABLE INTERRUPTS AND RETURN.
                                ;-

                                ROUTINE LPINI <DTBL,MSGCOD,MSGADR,RETADR>
000026'                                                                 LPINI:
(2)  000026'                            PUSH R0                                 MOV     R0,-(SP)
000026'                         LET R0 := DTBL(R5)                              MOV     DTBL(R5),R0
(2)  000026' 010046
000030'                         LET LP.FCH :B= DT.FCH(R0)                       MOVB    DT.FCH(R0),LP.FC
(4)  000030' 016500  000000
000034'                         LET LP.PFC :B= DT.FCN(R0)                       MOVB    DT.FCN(R0),LP.PF
(4)  000034' 116067  000037  177736
000042'                         LET LP.COD := MSGCOD(R5)                        MOV     MSGCOD(R5),LP.CO
(4)  000042' 116067  000036  177732
000050'                         LET LP.PTR := MSGADR(R5)                        MOV     MSGADR(R5),LP.PT
(4)  000050' 016567  000002  177736
000056'                         LET LP.EXT := RETADR(R5)                        MOV     RETADR(R5),LP.EX
(4)  000056' 016567  000004  177724
000064'                         POP R0                                          MOV     (SP)+,R0
(4)  000064' 016567  000006  177714
000072'
(2)  000072' 012600

                                ;+
                                ; NOW FIRST CLEAR INTERRUPTS IN CASE THEY WERE STILL ENABLED (BY
                                ; THE LINE PRINTER OPTION MODULE OR SOME OTHER THING
                                ;-

000074'                         LET @LPCSR := @LPCSR CLR.BY #IE                 BIC     #IE,@LPCSR
(6)  000074' 042777  000100  000000G
000102'                         LET @LPCSR := @LPCSR SET.BY #IE                 BIS     #IE,@LPCSR
(6)  000102' 052777  000100  000000G
000110'                 ENDRTN                                          50000$:
(3)  000110'                                                            50001$:
(3)  000110'                                                                    RTS     PC
(2)  000110' 000207
```

```
;++
; MODULE NAME:
;       LPINT
;
; FUNCTIONAL DESCRIPTION:
;       THIS MODULE HANDLES ALL OUTPUT MESSAGES TO THE DEVICE.
;
;       THERE ARE TWO KINDS OF MESSAGES :
;       (1) STANDARD            (ERROR MSG, PROMPT MSG, ETC.)
;       (2) NON-STANDARD        (ECHOING OF INPUT STRINGS, ETC.)
;
; INPUTS:
;       NONE
;
; IMPLICIT INPUTS:
;       NONE
;
; OUTPUTS:
;       NONE
; IMPLICIT OUTPUTS:
;       NONE
;
; PATHOLOGICAL CONNECTIONS:
;       CONTROL-O FLAG
;
; SUBORDINATE ROUTINES CALLED:
;       NONE
;
; FUNCTIONAL SIDE EFFECTS:
;       NONE
;
; CALLING SEQUENCE:
;
;       THE ROUTINE IS ENTERED VIA HARDWARE INTERRUPT
;
; VERSION:
;       0.0
;
;       EDIT    DATE        BY          REASON
;       -----   --------    ----------  --------------------
;
;--
```

```
                        IROUTINE        LPINT
                        ;+
                        ;DISABLE INTERRUPTS.
                        ;-
                        LET @LPCSR := @LPCSR CLR.BY #IE
                        ;+
                        ; IS FILL COUNT-FLAG NOT EQUAL TO ZERO ?
                        ; TRUE -        OUTPUT THE FILLER-CHARACTER
                        ;                DECREMENT FILL COUNT-FLAG
                        ;                SET INTERRUPTS, RETURN
                        ; FALSE -       CONTINUE.
                        ;-
                        IF LP.FCF NE #0 THEN

                                LET @LPBUF :B= #NULL

                                LET LP.FCF := LP.FCF - #1

                        ELSE

                                ;+
                                ; IS OUTPUT OF <^U> OR <%> BUFFER COMPLETED ?
                                ; TRUE -        RESET <^U> AND <%> FLAGS
                                ;                RESTORE MESSAGE POINTER
                                ; FALSE -       NO ACTION
                                ;-
                                IF LP.FLG NE #0 ANDB @LP.PTR EQ LP.BEN THEN

                                        LET LP.FLG := #0

                                        LET LP.PTR := LP.TP

                                ENDIF

                                ;+
                                ; IS INPUT CHARACTER A NULL ?
                                ; TRUE -        THIS IS THE END OF THE MESSAGE
                                ;                RETURN BY 'JMP @LP.EXT'.
                                ; FALSE -       NO ACTION
                                ;-
                                IFB @LP.PTR EQ #NULL THEN

                                        IF LP.COD EQ #MSGPOP THEN

                                           INLINE <CMP (SP)+,(SP)+>

                                        ENDIF
```

```
000112'                 LPINT:
000112' 042777 000100 000000G           BIC     #IE,@LPCSR
000120' 005767 177660                   TST     LP.FCF
000124' 001406                          BEQ     50002$
000126' 112777 000000 000000G           MOVB    #NULL,@LPBUF
000134' 005367 177644                   DEC     LP.FCF
000140' 000524                          BR      50003$
000142'                 50002$:
000142' 005767 177650                   TST     LP.FLG
000146' 001411                          BEQ     50004$
000150' 127767 177634 177647            CMPB    @LP.PTR,LP.BEN
000156' 001005                          BNE     50004$
000160' 005067 177632                   CLR     LP.FLG
000164' 016767 177622 177616            MOV     LP.TP,LP.PTR
000172'                 50004$:
000172' 127727 177612 000000            CMPB    @LP.PTR,#NULL
000200' 001007                          BNE     50005$
000202' 026727 177606 000002            CMP     LP.COD,#MSGPOP
000210' 001001                          BNE     50006$
000212' 022626                          CMP (SP)+,(SP)+
000214'
```

```
(4)  000214'                                                                    50006$:
754  000214'                                    INLINE <JMP @LP.EXT;LEAVE,FINISHED MESSAGE>
(2)  000214' 000177  177566                                                         JMP @LP.EXT;LEAVE,FINISH
755  000220'                                    ENDIF
(4)  000220'                                                                    50005$:
756                                             ;+
757                                             ; NON-STANDARD MESSAGE CODE HANDLING
758                                             ;  SPECIAL CONDITIONS- CONTROL U,RUBOUT
759                                             ;-
760  000220'                                    IF LP.COD LT #0 THEN
(6)  000220' 005767  177570                                                         TST     LP.COD
(9)  000224' 002033                                                                 BGE     50007$
761                                                     ;+
762                                                     ; IS INPUT CHARACTER IS A <^U> CONTROL U ?
763                                                     ; TRUE -        SET ^U-FLAG
764                                                     ;               SAVE REST OF MESSAGE POINTER
765                                                     ;               POINT TO ^U-BUFFER
766                                                     ; FALSE -       NO ACTION
767                                                     ;-
768  000226'                                            IFB @LP.PTR EQ #CTRLU THEN
(6)  000226' 127727  177556  000025                                                 CMPB    @LP.PTR,#CTRLU
(9)  000234' 001013                                                                 BNE     50010$
769  000236'                                                    LET LP.CUF :B= #1
(4)  000236' 112767  000001  177553                                                 MOVB    #1,LP.CUF
770  000244'                                                   LET LP.TP := LP.PTR + #1
(4)  000244' 016767  177540  177540                                                 MOV     LP.PTR,LP.TP
(6)  000252' 005267  177534                                                         INC     LP.TP
771  000256'                                                   LET LP.PTR := #LP.CUB
(4)  000256' 012767  000023' 177524                                                 MOV     #LP.CUB,LP.PTR
772  000264'                                            ENDIF
(4)  000264'                                                                    50010$:
773                                                     ;+
774                                                     ; IS INPUT CHARACTER IS A RUBOUT ?
775                                                     ; TRUE -        OUTPUT BACKSLASH
776                                                     ; FALSE -       OUTPUT CHARACTER
777                                                     ;-
778  000264'                                            IFB @LP.PTR EQ #RUBOUT THEN
(6)  000264' 127727  177520  000177                                                 CMPB    @LP.PTR,#RUBOUT
(9)  000272' 001004                                                                 BNE     50011$
779  000274'                                                    LET @LPBUF :B= #LP.BKSLSH
(4)  000274' 112777  000134  000000G                                                MOVB    #LP.BKSLSH,@LPBU
780  000302'                                            ELSE
(4)  000302' 000403                                                                 BR      50012$
(3)  000304'                                                                    50011$:
781  000304'                                                   LET @LPBUF :B= @LP.PTR
(4)  000304' 117777  177500  000000G                                                MOVB    @LP.PTR,@LPBUF
782  000312'                                            ENDIF
(4)  000312'                                                                    50012$:
783                                             ;+
784                                             ; STANDARD MESSAGE CODE HANDLING
785                                             ;  SPECIAL CONDITIONS- PERCENT,SURPRESS
786                                             ;-
787  000312'                                    ELSE
(4)  000312' 000426                                                                 BR      50013$
(3)  000314'                                                                    50007$:
788                                                     ;+
```

```
789
790
791
792
793
794
795  000314'
(6)  000314' 127727  177470  000045
(9)  000322' 001013
796  000324'
(4)  000324' 112767  000001  177464
797  000332'
(4)  000332' 016767  177452  177452
(6)  000340' 005267  177446
798  000344'
(4)  000344' 012767  000020' 177436
799  000352'
(4)  000352'
800
801
802
803
804
805  000352'
(6)  000352' 026727  000000G 000001
(9)  000360' 001403
806  000362'
(4)  000362' 117777  177422  000000G
807  000370'
(4)  000370'
808  000370'
(4)  000370'
809
810
811
812
813
814  000370'
(6)  000370' 127767  177414  177402
(9)  000376' 001003
815  000400'
(4)  000400' 016767  177376  177376
816  000406'
(4)  000406'
817
818
819
820  000406'
(6)  000406' 005267  177376
821  000412'
(4)  000412'
822
823
824
825
826  000412'
```

```
                ; IS INPUT CHARACTER IS A <%> PERCENT ?
                ; TRUE -        SET %-FLAG
                ;               SAVE REST OF MESSAGE POINTER
                ;               POINT TO %-BUFFER
                ; FALSE -       NO ACTION
                ;-
                IFB @LP.PTR EQ #LP.PCNT THEN

                        LET LP.PCF :B= #1

                        LET LP.TP := LP.PTR + #1


                        LET LP.PTR := #LP.PCB

                ENDIF

                ;+
                ; IS MESSAGE SURPRESS FLAG NOT SET ?
                ; TRUE -        OUTPUT CHARACTER
                ; FALSE -       NO ACTION
                ;-
                IF CTRLOF NE #1 THEN

                        LET @LPBUF :B= @LP.PTR

                ENDIF

        ENDIF

        ;+
        ; IS INPUT CHARACTER THE "FILL-CHARACTER" ?
        ; TRUE -        PRESET FILL COUNT-FLAG
        ; FALSE -       NO ACTION
        ;-
        IFB @LP.PTR EQ LP.FCH THEN

                LET LP.FCF := LP.PFC

        ENDIF

        ;+
        ; INCREMENT MESSAGE CHARACTER POINTER
        ;-
        LET LP.PTR := LP.PTR + #1

ENDIF

;+
;ENABLE INTERRUPTS. BUT FIRST RAISE PRIORITY TO 4
;-

PUSH #PRI4
```

```
                CMPB    @LP.PTR,#LP.PCNT
                BNE     50014$

                MOVB    #1,LP.PCF

                MOV     LP.PTR,LP.TP
                INC     LP.TP

                MOV     #LP.PCB,LP.PTR

50014$:


                CMP     CTRLOF,#1
                BEQ     50015$

                MOVB    @LP.PTR,@LPBUF

50015$:

50013$:


                CMPB    @LP.PTR,LP.FCH
                BNE     50016$

                MOV     LP.PFC,LP.FCF

50016$:


                INC     LP.PTR

50003$:
```

```
(2)  000412' 012746  000200
827  000416'                          PUSH #1000$                                          MOV     #PRI4,-(SP)
(2)  000416' 012746  000424'
828  000422'                          INLINE <RTI>                                         MOV     #1000$,-(SP)
(2)  000422' 000002
829  000424'                          INLINE <1000$:>                                      RTI
(2)  000424'
830  000424'                          LET @LPCSR := @LPCSR SET.BY #IE                      1000$:
(6)  000424' 052777  000100  000000G                                                     BIS     #IE,@LPCSR
831  000432'                          ENDRTI
(3)  000432'                                                                        50000$:
(3)  000432'                                                                        50001$:
(2)  000432' 000002                                                                  RTI
832          000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLOF= ****** G
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTBL   = 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPBUF  = ****** G
LPCSR  = ****** G
LPINI    000026RG
LPINT    000112RG
LPSTAT= 000001
LP.BEN   000025R
LP.BKS= 000134
LP.COD   000014R
LP.CUB   000023R
LP.CUF   000017R
LP.EXT   000006R
LP.FCF   000004R
LP.FCH   000000R
LP.FLG   000016R
LP.PCB   000020R
LP.PCF   000016R
LP.PCN= 000045
LP.PFC   000002R
LP.PTR   000010R
LP.TP    000012R
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGADR= 000004
MSGCKD= 000010
MSGCKS= 000011
MSGCOD= 000002
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRI0   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1   = 000056
RES2   = 000060
RETADR= 000006
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
```

```
SPVALU= 002200          UIPAR3= 177646          XFLAG = 000005          $F$TRU= 000404          $$ARGC= 000000
SR0   = 177572          UIPAR4= 177650          XOFF  = 000023          $F$UNT= 000130          $$BYTE= 000402
SR1   = 177574          UIPAR5= 177652          XON   = 000021          $F$WHI= 000120          $$CASE= 000000
SR2   = 177576          UIPAR6= 177654          $BGNLE= 177777          $F$YES= 000402          $$DST = 000000
SR3   = 172516          UIPAR7= 177656          $ERFLG= 000400          $IFLEV= 177777          $$ELOC= 000402
STAT  = 000026          UIPDR0= 177600          $F$AND= 000310          $ISK0 = 000001          $$ERFL= 000000
STATBI= 064757          UIPDR1= 177602          $F$BAD= 000401          $ISK1 = 000001          $$FLAG= 000001
STAT1 = 000027          UIPDR2= 177604          $F$BLA= 000170          $ISK2 = 000001          $$FROM= 000000
SUSPND= 000001          UIPDR3= 177606          $F$CAS= 000150          $LOCTA= 177777          $$LOC = 000376R
SVR0  = 000062          UIPDR4= 177610          $F$DEC= 000220          $LSTIN= 000001          $$LOCN= 000000
SVR1  = 000064          UIPDR5= 177612          $F$DO = 000340          $LSTTA= 000001          $$REG = 177777
SVR2  = 000066          UIPDR6= 177614          $F$FAL= 000405          $NESTL= 177777          $$RETU= 000000
SVR3  = 000070          UIPDR7= 177616          $F$GOO= 000400          $NSK0 = 000350          $$RTN1= 050000
SVR4  = 000072          WASADR= 000104          $F$IF = 000110          $NSK1 = 000110          $$RTN2= 050001
SVR5  = 000074          WBSTAT= 000040          $F$INC= 000210          $NSK2 = 000110          $$SRC = 000000
SVR6  = 000076          WBUFEA= 000136          $F$LOO= 000200          $NSK3 = 000110          $$TGSV= 000000
SYSCNT= 000052          WBUFPA= 000134          $F$NAM= 000160          $SAVLE= 177777          $$TGS1= 000000
SYSERR= 000100          WBUFRQ= 000140          $F$NO = 000403          $TAGLE= 177777          $$TGS2= 000000
TMPIO = 000002          WBUFSZ= 000142          $F$OR = 000320          $TAGNU= 050017          $$TO  = 000000
TQOVF = 000002          WDFR  = 000116          $F$RTI= 000350          $TEMP = 000350          $$$TAG= 050000
UIPAR0= 177640          WDTO  = 000114          $F$RTN= 000300          $TSK0 = 050003          .     = 000434R
UIPAR1= 177642          WTINRE= 000352          $F$SEL= 000140          $TSK1 = 050016
UIPAR2= 177644          WTWHMI= 000222          $F$THE= 000330          $TSK2 = 050015

. ABS.  000000      000
        000434      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:LPDRV,DSKZ:LPDRV=SPMAC/ML,EQUATE,LPDRV
 RUN-TIME: 16 7 .4 SECONDS
 RUN-TIME RATIO: 40/24=1.6
 CORE USED:  14K  (27 PAGES)
```

TABLE OF CONTENTS

```
                        .TITLE TTDRV  (TERMINAL  DRIVER MODULE)
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       TTDRV
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE CONTAINS TWO MODULES - TTINI AND TTINT.
                        ;       THE TTINI IS THE INITIALIZATION ROUTINE FOR THE
                        ;       TERMINAL INTERRUPT HANDLING ROUTINE - TTINT.
                        ;       TTINT ACTUALLY PROCESSES ALL MESSAGES TO THE DEVICE.
                        ;
                        ;       A DESCRIPTION IS GIVEN IN EACH MODULE.
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT    DATE        BY          REASON
                        ;       -----   ----------  ----------  --------------------
                        ;
                        ;--
```

```
                        .PRINT  ;SPMAC: VERSION 1.1
                        ;****************                ******************************
                        ;               GLOBAL REFERENCES
                        .GLOBL XCSR                     ;DEVICE REGISTER -    CONTROL & STATUS
                        .GLOBL XBUF                     ;DEVICE REGISTER -    DATA BUFFER
                        .GLOBL KB.XFLAG         ;XOFF FLAG WORD
                        .GLOBL CTRLOF                   ;FLAG - OUTPUT SUPPRESS
                        ;****************               ******************************
                        ;               REFERENCED BY OTHER MODULES
                        .GLOBL TTINT                    ;MODULE ENTRY POINT - TT HANDLER
                        .GLOBL TTINI                    ;MODULE ENTRY POINT - TT INITIATOR
                        ;****************               ******************************
                        ;               LOCAL EQUATES
000045                  TT.PCNT =       45              ;CHARACTER -    '%' PERCENT
000134                  TT.BKSLSH =     134             ;CHARACTER -    '\' BACKSLASH
                        ;****************               ******************************
                        ;               LOCAL STORAGE
000000' 000000          TT.FCH: 0                       ;CHARACTER -    FILL
000002' 000000          TT.PFC: 0                       ;COUNT -        FILL, PERMANENT
000004' 000000          TT.FCF: 0                       ;COUNT, FLAG -  FILL (FLAG SET NE 0)
000006' 000000          TT.EXT: 0                       ;POINTER -      LOCATION TO JUMP OUT INDIRECTLY
000010' 000000          TT.PTR: 0                       ;POINTER -      MESSAGE
000012' 000000          TT.TP:  0                       ;POINTER -      TEMPORARY BUFFER
000014' 000000          TT.COD: 0                       ;CODE -         MESSAGE
000016'                 TT.FLG:                         ;FLAGS -        PERCENT AND CONTROL-U
000016'     000         TT.PCF: .BYTE   0               ;FLAG -         PERCENT
000017'     000         TT.CUF: .BYTE   0               ;FLAG -         CONTROL-U
000020'     015         TT.PCB: .BYTE   15              ;+BUFFER -      PERCENT '%'             <CR>+
000021'     012                 .BYTE   12              ;+BUFFER        ECHO                    <LF>+
000022'     001                 .BYTE   1               ;+BUFFER -      TERMINATOR              <+1>+
000023'     136         TT.CUB: .BYTE   136             ;#BUFFER -      CONTROL-U                <^>#
000024'     125                 .BYTE   125             ;#BUFFER        ECHO                     <U>#
000025'     001         TT.BEN: .BYTE   1               ;#BUFFER -      TERMINATOR              <+1>#
                                                        ;FLAG -         (TT.BEN) END SPECIAL BUFFER

                        .EVEN
```

```
585
586                     ;++
587                     ; MODULE NAME:
588                     ;       TTINI
589                     ;
590                     ; FUNCTIONAL DESCRIPTION:
591                     ;       THE DEVICE OUTPUT STARTER MODULE (TTINI) SETS UP FOR
392                     ;       THE DEVICE OUTPUT INTERRUPT HANDLING MODULE (TTINT).
593                     ;
594                     ; INPUTS:
595                     ;       1. DATA TABLE ADDRESS
596                     ;       2. MESSAGE ADDRESS
597                     ;       3. MESSAGE TYPE CODE
598                     ;       4. RETURN ADDRESS (DRIVER WILL JUMP TO AT END OF MESSAGE).
599                     ;
600                     ; IMPLICIT INPUTS:
601                     ;       1. FILL CHARACTER
602                     ;       2. FILL COUNT
603                     ;
604                     ; OUTPUTS:
605                     ;       NONE
606                     ;
607                     ; IMPLICIT OUTPUTS:
608                     ;       NONE
609                     ;
610                     ; PATHOLOGICAL CONNECTIONS:
611                     ;       NONE
612                     ;
613                     ; SUBORDINATE MODULES CALLED:
614                     ;       NONE
615                     ;
616                     ; FUNCTIONAL SIDE EFFECTS:
617                     ;       NONE
618                     ;
619                     ; CALLING SEQUENCE:
620                     ;       CALL TTINI IN <DTADDR,TYPCOD,MSGADR,RETADR>
621                     ;
622                     ;               WHERE DTADDR = ADDRESS OF DATA TABLE
623                     ;                     TYPCOD = TYPE CODE
624                     ;                     MSGADR = ADDRESS OF MESSAGE
625                     ;                     RETADR = RETURN ADDRESS THE DRIVER WILL JUMP TO
626                     ;                              AT THE COMPLETION OF THE MESSAGE.
627                     ;
628                     ; VERSION:
629                     ;       0.0
630                     ;
631                     ;       EDIT    DATE        BY          REASON
632                     ;       -----   --------    ----------  ------------------
633                     ;--
```

```
                                ;+
                                ; INITIALIZE MESSAGE TYPE, MESSAGE
                                ; ADDRESS, FILL COUNT, AND FILL CHARACTER.
                                ; STUFF RETURN ADDRESS AT TT.EXT,
                                ; ENABLE INTERRUPTS AND RETURN.
                                ;-

                                ROUTINE TTINI <DTBL,MSGCOD,MSGADR,RETADR>

                                        PUSH R0

                                        LET R0 := DTBL(R5)

                                        LET TT.FCH :B= DT.FCH(R0)

                                        LET TT.PFC :B= DT.FCN(R0)

                                        LET TT.COD := MSGCOD(R5)

                                        LET TT.PTR := MSGADR(R5)

                                        LET TT.EXT := RETADR(R5)

                                        POP R0

                                        LET @XCSR := @XCSR SET.BY #IE

                                ENDRTN
```

```
TTINI:
        MOV     R0,-(SP)
        MOV     DTBL(R5),R0
        MOVB    DT.FCH(R0),TT.FC
        MOVB    DT.FCN(R0),TT.PF
        MOV     MSGCOD(R5),TT.CO
        MOV     MSGADR(R5),TT.PT
        MOV     RETADR(R5),TT.EX
        MOV     (SP)+,R0
        BIS     #IE,@XCSR
50000$:
50001$:
        RTS     PC
```

```
663
664                             ;++
665                             ; MODULE NAME:
666                             ;       TTINT
667                             ;
668                             ; FUNCTIONAL DESCRIPTION:
669                             ;       THIS MODULE HANDLES ALL OUTPUT MESSAGES TO THE DEVICE.
670                             ;
671                             ;       THERE ARE TWO KINDS OF MESSAGES :
672                             ;       (1) STANDARD            (ERROR MSG, PROMPT MSG, ETC.)
673                             ;       (2) NON-STANDARD        (ECHOING OF INPUT STRINGS, ETC.)
674                             ;
675                             ; INPUTS:
676                             ;       NONE
677                             ;
678                             ; IMPLICIT INPUTS:
679                             ;       NONE
680                             ;
681                             ; OUTPUTS:
682                             ;       NONE
683                             ; IMPLICIT OUTPUTS:
684                             ;       NONE
685                             ;
686                             ; PATHOLOGICAL CONNECTIONS:
687                             ;       CONTROL-O FLAG
688                             ;       XOFF FLAG WORD
689                             ;
690                             ; SUBORDINATE ROUTINES CALLED:
691                             ;       NONE
692                             ;
693                             ; FUNCTIONAL SIDE EFFECTS:
694                             ;       IF XOFF FLAG IS DETECTED CONTROL IS RETURNED TO THE INTERRUPTED
695                             ;       CODE IMMEDIATELY WITHOUT RE-ENABLING INTERRUPTS.
696                             ;       THIS WILL BE DONE BY THE KBDRV MODULE WHEN A XON IS DETECTED.
697                             ;
698                             ; CALLING SEQUENCE:
699                             ;
700                             ;       THE ROUTINE IS ENTERED VIA HARDWARE INTERRUPT
701                             ;
702                             ; VERSION:
703                             ;       0.0
704                             ;
705                             ;       EDIT    DATE        BY          REASON
706                             ;       -----   --------    ----------  --------------------
707                             ;
708                             ;--
```

```
                        IROUTINE        TTINT
                                                                TTINT:
                        ;+
                        ;DISABLE INTERRUPTS.
                        ;-
                        LET @XCSR := @XCSR CLR.BY #IE
                                                                        BIC     #IE,@XCSR
                        ;+
                        ; IF XOFF FLAG SET
                        ; TRUE -    RTI
                        ; ELSE -    CONTINUE
                        ;-

                        IF KB.XFLAG NE #0 THEN
                                                                        TST     KB.XFLAG
                                                                        BEQ     50002$
                                INLINE <RTI>
                                                                        RTI
                        ENDIF
                                                                50002$:

                        ;+
                        ; IS FILL COUNT-FLAG NOT EQUAL TO ZERO ?
                        ; TRUE -        OUTPUT THE FILLER-CHARACTER
                        ;                DECREMENT FILL COUNT-FLAG
                        ;                SET INTERRUPTS, RETURN
                        ; FALSE -       CONTINUE.
                        ;-
                        IF TT.FCF NE #0 THEN
                                                                        TST     TT.FCF
                                                                        BEQ     50003$
                                LET @XBUF :B= #NULL
                                                                        MOVB    #NULL,@XBUF
                                LET TT.FCF := TT.FCF - #1
                                                                        DEC     TT.FCF
                        ELSE
                                                                        BR      50004$
                                                                50003$:
                                ;+
                                ; IS OUTPUT OF <^U> OR <%> BUFFER COMPLETED ?
                                ; TRUE -        RESET <^U> AND <%> FLAGS
                                ;                RESTORE MESSAGE POINTER
                                ; FALSE -       NO ACTION
                                ;-
                                IF TT.FLG NE #0 ANDB @TT.PTR EQ TT.BEN THEN
                                                                        TST     TT.FLG
                                                                        BEQ     50005$
                                                                        CMPB    @TT.PTR,TT.BEN
                                                                        BNE     50005$
                                        LET TT.FLG := #0
                                                                        CLR     TT.FLG
                                        LET TT.PTR := TT.TP
                                                                        MOV     TT.TP,TT.PTR
                                ENDIF
                                                                50005$:
```

```
                        ;+
                        ; IS INPUT CHARACTER A NULL ?
                        ; TRUE -        THIS IS THE END OF THE MESSAGE
                        ;                RETURN BY 'JMP @TT.EXT'.
                        ; FALSE -       NO ACTION
                        ;-
                        IFB @TT.PTR EQ #NULL THEN
000174'                                                         CMPB    @TT.PTR,#NULL
000174' 127727  177610  000000                                  BNE     50006$
000202' 001007
                                IF TT.COD EQ #MSGPOP THEN
000204'                                                         CMP     TT.COD,#MSGPOP
000204' 026727  177604  000002                                  BNE     50007$
000212' 001001
                                   INLINE <CMP (SP)+,(SP)+>
000214'                                                         CMP (SP)+,(SP)+
000214' 022626
                                ENDIF
000216'                                                 50007$:
000216'
                                INLINE <JMP @TT.EXT;LEAVE,FINISHED MESSAGE>
000216' 000177  177564                                          JMP @TT.EXT;LEAVE,FINISH
                        ENDIF
000222'                                                 50006$:
000222'
                        ;+
                        ; NON-STANDARD MESSAGE CODE HANDLING
                        ;  SPECIAL CONDITIONS- CONTROL U,RUBOUT
                        ;-
                        IF TT.COD LT #0 THEN
000222'                                                         TST     TT.COD
000222' 005767  177566                                          BGE     50010$
000226' 002033
                                ;+
                                ; IS INPUT CHARACTER IS A <^U> CONTROL U ?
                                ; TRUE -        SET ^U-FLAG
                                ;                SAVE REST OF MESSAGE POINTER
                                ;                POINT TO ^U-BUFFER
                                ; FALSE -       NO ACTION
                                ;-
                                IFB @TT.PTR EQ #CTRLU THEN
000230'                                                         CMPB    @TT.PTR,#CTRLU
000230' 127727  177554  000025                                  BNE     50011$
000236' 001013
                                        LET TT.CUF :B= #1
000240'                                                         MOVB    #1,TT.CUF
000240' 112767  000001  177551
                                        LET TT.TP := TT.PTR + #1
000246'                                                         MOV     TT.PTR,TT.TP
000246' 016767  177536  177536                                  INC     TT.TP
000254' 005267  177532
                                        LET TT.PTR := #TT.CUB
000260'                                                         MOV     #TT.CUB,TT.PTR
000260' 012767  000023' 177522
                                ENDIF
000266'                                                 50011$:
000266'
                                ;+
                                ; IS INPUT CHARACTER IS A RUBOUT ?
                                ; TRUE -        OUTPUT BACKSLASH
                                ; FALSE -       OUTPUT CHARACTER
                                ;-
                                IFB @TT.PTR EQ #RUBOUT THEN
000266'                                                         CMPB    @TT.PTR,#RUBOUT
000266' 127727  177516  000177                                  BNE     50012$
000274' 001004
                                        LET @XBUF :B= #TT.BKSLSH
000276'                                                         MOVB    #TT.BKSLSH,@XBUF
000276' 112777  000134  000000G
```

```
787  000304'                                                    ELSE
(4)  000304' 000403                                                                    BR      50013$
(3)  000306'                                                                50012$:
788  000306'                                                                LET @XBUF :B= @TT.PTR
(4)  000306' 117777  177476  000000G                                                  MOVB    @TT.PTR,@XBUF
789  000314'                                                    ENDIF
(4)  000314'                                                                50013$:
790                                                         ;+
791                                                         ; STANDARD MESSAGE CODE HANDLING
792                                                         ;  SPECIAL CONDITIONS- PERCENT,SURPRESS
793                                                         ;-
794  000314'                                                ELSE
(4)  000314' 000426                                                                    BR      50014$
(3)  000316'                                                                50010$:
795                                                                 ;+
796                                                                 ; IS INPUT CHARACTER IS A <%> PERCENT ?
797                                                                 ; TRUE -        SET %-FLAG
798                                                                 ;               SAVE REST OF MESSAGE POINTER
799                                                                 ;               POINT TO %-BUFFER
800                                                                 ; FALSE -       NO ACTION
801                                                                 ;-
802  000316'                                                        IFB @TT.PTR EQ #TT.PCNT THEN
(6)  000316' 127727  177466  000045                                                    CMPB    @TT.PTR,#TT.PCNT
(9)  000324' 001013                                                                    BNE     50015$
803  000326'                                                                LET TT.PCF :B= #1
(4)  000326' 112767  000001  177462                                                   MOVB    #1,TT.PCF
804  000334'                                                                LET TT.TP := TT.PTR + #1
(4)  000334' 016767  177450  177450                                                   MOV     TT.PTR,TT.TP
(6)  000342' 005267  177444                                                           INC     TT.TP
805  000346'                                                                LET TT.PTR := #TT.PCB
(4)  000346' 012767  000020' 177434                                                   MOV     #TT.PCB,TT.PTR
806  000354'                                                        ENDIF
(4)  000354'                                                                50015$:
807                                                                 ;+
808                                                                 ; IS MESSAGE SURPRESS FLAG NOT SET ?
809                                                                 ; TRUE -        OUTPUT CHARACTER
810                                                                 ; FALSE -       NO ACTION
811                                                                 ;-
812  000354'                                                        IF CTRLOF NE #1 THEN
(6)  000354' 026727  000000G 000001                                                   CMP     CTRLOF,#1
(9)  000362' 001403                                                                    BEQ     50016$
813  000364'                                                                LET @XBUF :B= @TT.PTR
(4)  000364' 117777  177420  000000G                                                  MOVB    @TT.PTR,@XBUF
814  000372'                                                        ENDIF
(4)  000372'                                                                50016$:
815  000372'                                                ENDIF
(4)  000372'                                                                50014$:
816                                                         ;+
817                                                         ; IS INPUT CHARACTER THE "FILL-CHARACTER" ?
818                                                         ; TRUE -        PRESET FILL COUNT-FLAG
819                                                         ; FALSE -       NO ACTION
820                                                         ;-
821  000372'                                                IFB @TT.PTR EQ TT.FCH THEN
(6)  000372' 127767  177412  177400                                                   CMPB    @TT.PTR,TT.FCH
(9)  000400' 001003                                                                    BNE     50017$
822  000402'                                                                LET TT.FCF := TT.PFC
```

```
(4)  000402' 016767  177374  177374                    MOV     TT.PFC,TT.FCF
823  000410'                                 ENDIF
(4)  000410'                                                50017$:
824                                          ;+
825                                          ; INCREMENT MESSAGE CHARACTER POINTER
826                                          ;-
827  000410'                                 LET TT.PTR := TT.PTR + #1
(6)  000410' 005267  177374                              INC     TT.PTR
828  000414'                                 ENDIF
(4)  000414'                                                50004$:
829                                          ;+
830                                          ;ENABLE INTERRUPTS. BUT FIRST RAISE PRIORITY TO 4
831                                          ;-
832
833  000414'                                 PUSH #PRI4
(2)  000414' 012746  000200                              MOV     #PRI4,-(SP)
834  000420'                                 PUSH #1000$
(2)  000420' 012746  000426'                             MOV     #1000$,-(SP)
835  000424'                                 INLINE <RTI>
(2)  000424' 000002                                      RTI
836  000426'                                 INLINE <1000$:>
(2)  000426'                                             1000$:
837  000426'                                 LET @XCSR := @XCSR SET.BY #IE
(6)  000426' 052777  000100  000000G                     BIS     #IE,@XCSR
838  000434'                                 ENDRTI
(3)  000434'                                                50000$:
(3)  000434'                                                50001$:
(2)  000434' 000002                                      RTI
839          000001                         .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLOF= ****** G
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTBL  = 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KB.XFL= ****** G
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGADR= 000004
MSGCKD= 000010
MSGCKS= 000011
MSGCOD= 000002
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RES1  = 000056
RES2  = 000060
RETADR= 000006
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
```

```
TMPIO = 000002          UIPAR4= 177650          XCSR  = ****** G        $F$THE= 000330          $TSK2 = 050016
TQOVF = 000002          UIPAR5= 177652          XFLAG = 000005          $F$TRU= 000404          $$ARGC= 000000
TTINI   000026RG        UIPAR6= 177654          XOFF  = 000023          $F$UNT= 000130          $$BYTE= 000402
TTINT   000104RG        UIPAR7= 177656          XON   = 000021          $F$WHI= 000120          $$CASE= 000000
TT.BEN  000025R         UIPDR0= 177600          $BGNLE= 177777          $F$YES= 000402          $$DST = 000000
TT.BKS= 000134          UIPDR1= 177602          $ERFLG= 000400          $IFLEV= 177777          $$ELOC= 000402
TT.COD  000014R         UIPDR2= 177604          $F$AND= 000310          $ISK0 = 000001          $$ERFL= 000000
TT.CUB  000023R         UIPDR3= 177606          $F$BAD= 000401          $ISK1 = 000001          $$FLAG= 000001
TT.CUF  000017R         UIPDR4= 177610          $F$BLA= 000170          $ISK2 = 000001          $$FROM= 000000
TT.EXT  000006R         UIPDR5= 177612          $F$CAS= 000150          $LOCTA= 177777          $$LOC = 000400R
TT.FCF  000004R         UIPDR6= 177614          $F$DEC= 000220          $LSTIN= 000001          $$LOCN= 000000
TT.FCH  000000R         UIPDR7= 177616          $F$DO = 000340          $LSTTA= 000001          $$REG = 177777
TT.FLG  000016R         WASADR= 000104          $F$FAL= 000405          $NESTL= 177777          $$RETU= 000000
TT.PCB  000020R         WBSTAT= 000040          $F$GOO= 000400          $NSK0 = 000350          $$RTN1= 050000
TT.PCF  000016R         WBUFEA= 000136          $F$IF = 000110          $NSK1 = 000110          $$RTN2= 050001
TT.PCN= 000045          WBUFPA= 000134          $F$INC= 000210          $NSK2 = 000110          $$SRC = 000000
TT.PFC  000002R         WBUFRQ= 000140          $F$LOO= 000200          $NSK3 = 000110          $$TGSV= 000000
TT.PTR  000010R         WBUFSZ= 000142          $F$NAM= 000160          $SAVLE= 177777          $$TGS1= 000000
TT.TP   000012R         WDFR  = 000116          $F$NO = 000403          $TAGLE= 177777          $$TGS2= 000000
UIPAR0= 177640          WDTO  = 000114          $F$OR = 000320          $TAGNU= 050020          $$TO  = 000000
UIPAR1= 177642          WTINRE= 000352          $F$RTI= 000350          $TEMP = 000350          $$$TAG= 050000
UIPAR2= 177644          WTWHMI= 000222          $F$RTN= 000300          $TSK0 = 050004          .     = 000436R
UIPAR3= 177646          XBUF  = ****** G        $F$SEL= 000140          $TSK1 = 050017

. ABS.  000000      000
        000436      001


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:TTDRV,DSKZ:TTDRV=SPMAC/ML,EQUATE,TTDRV
 RUN-TIME: 17 7 .3 SECONDS
 RUN-TIME RATIO: 40/24=1.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SIZPLA - SIZE AND POLL SYSTEM - NO SUPPORT OF ANY CPU OPTIONS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SIZPLA
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SIZE MEMORY TO 28K AND WRITE
                        ;       GOOD PARITY IN THAT AREA. THIS ROUTINE WILL NOT CHECK FOR
                        ;       ANY CPU OPTIONS AND WILL INDICATE THERE ARE NONE. THE "MODQ" LIST WILL
                        ;       BE SORTED TO DEVELOP A BACKGROUND MODULE LIST (QUEUE).
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CONFIGURATION WORD 0
                        ;       EXERCISER SIZE
                        ;       SYSTEM SIZE
                        ;       MODULE LIST
                        ;       BACKGROUND LIST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       SYSTEM SIZE
                        ;       EXERCISE SIZE
                        ;       CONFIGURATION WORD 0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SIZPOL IN <A>
                        ;                 A - DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES

                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1
        000001          $LSTIN = 1
        000001          $LSTTAG = 1


                        ;****************
                        ; GLOBAL REFERENCES
                        ;
                        .GLOBL  SAVREG                  ;SARE REGISTERS
                        .GLOBL  RESREG                  ;RESTORE REGISTERS
                        .GLOBL  HRDADRCHK               ;ILLEGAL ADDRESS CHECK


                        ;****************
                        ; LOCAL EQUATES
                        ;
        004000          SS.0KC  = 4000                  ;ONE K CHUNK
        000034          SS.28   = ^D28                  ;28 DECIMAL
        160000          SS.RFS  = 160000                ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE

                        ;****************
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  SIZPOL                  ;MODULE ENTRY POINT


                        ;++++++++++++++++++++++++++++++
                        ;    OVERLAY REGION GLOBALS
                        ;++++++++++++++++++++++++++++++
                        .GLOBL  OV.KBBUF
                        .GLOBL  OV.HIKB
                        .GLOBL  OV.KBSIZ
                        .GLOBL  OV.CQ
                        .GLOBL  OV.HICQ
                        .GLOBL  OV.CQSIZ
                        .GLOBL  OV.TQ
                        .GLOBL  OV.HITQ
                        .GLOBL  OV.TQSIZ
```

```
                                .SBTTL KERNEL - DETERMINE MEMORY SIZE


                                ROUTINE SIZPOL <DTABLE>
000000'                                                                 SIZPOL:
(2)  000000'
                                ;+
                                ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
                                ; R1
                                ;-

                                CALL SAVREG
000000'                                                                 JSR     PC,SAVREG
(3)  000000' 004767  000000G
                                LET R2 := DTABLE(R5)
000004'                                                                 MOV     DTABLE(R5),R2
(4)  000004' 016502  000000
                                LET R1 := #0
000010'                                                                 CLR     R1
(4)  000010' 005001


                                ;+
                                ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
                                ; AND LOAD IT INTO EXERCISER SIZE WORD
                                ;-

                                LET DT.ESIZ(R2) := 0
000012'                                                                 MOV     0,DT.ESIZ(R2)
(4)  000012' 016762  000000  000044


                                ;+
                                ; DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
                                ; THAN 28K.
                                ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
                                ;-

                                LET R0 := #0
000020'                                                                 CLR     R0
(4)  000020' 005000
                                LET R3 := #0
000022'                                                                 CLR     R3
(4)  000022' 005003


                                ;+
                                ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K IS FOUND
                                ;-

                                REPEAT
000024'                                                         50002$:
(3)  000024'
                                        LET R0 := R0 + #1
000024'                                                                 INC     R0
(6)  000024' 005200
                                        LET R3 := R3 + #SS.0KC
000026'                                                                 ADD     #SS.0KC,R3
(6)  000026' 062703  004000
                                        CALL HRDADRCHK IN <R3>
000032'                                                                 MOV     R5,-(SP)
(3)  000032' 010546                                                     MOV     R3,-(R5)
(4)  000034' 010345                                                     JSR     PC,HRDADRCHK
(3)  000036' 004767  000000G                                           MOV     (SP)+,R5
(3)  000042' 012605
                                        IF.ERROR THEN
000044'                                                                 BCC     50003$
(6)  000044' 103001
                                                INLINE <BR 1$>
000046'
```

```
                                                                                    BR 1$

                        ENDIF                                                50003$:

UNTIL R0 EQ #SS.28                                                                  CMP     R0,#SS.28
                                                                                    BNE     50002$


;+
; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
;-

INLINE <1$:>                                                                        1$:

IF R0 EQ #SS.28 THEN                                                                CMP     R0,#SS.28
                                                                                    BNE     50004$

        CALL HRDADRCHK IN <#SS.RFS>                                                 MOV     R5,-(SP)
                                                                                    MOV     #SS.RFS,-(R5)
                                                                                    JSR     PC,HRDADRCHK
                                                                                    MOV     (SP)+,R5

        IF.NO.ERROR THEN                                                            BCS     50005$

                LET R3 := R3 + #10000                                               ADD     #10000,R3

        ENDIF                                                                50005$:

ENDIF                                                                        50004$:


;+
; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
; SIZE WORD.
;-

LET CARRY := #0                                                                     CLC

LET DT.SSIZ(R2) := R3 ROTATE -6                                                     MOV     R3,DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
                                                                                    ROR     DT.SSIZ(R2)
```

```
                                .SBTTL KERNEL - WRITE GOOD PARITY
                                ;+
                                ; SET UP POINTER TO POINT TO LOC. 0 AND SAVE BUSERR VECTOR
                                ;-

000144'                         LET R0 := #0                                    CLR     R0
(4)  000144' 005000
000146'                         PUSH @#4                                        MOV     @#4,-(SP)
(2)  000146' 013746  000004
000152'                         LET @#4 := #2$                                  MOV     #2$,@#4
(4)  000152' 012737  000164' 000004


                                ;+
                                ; NOW WRITE GOOD PARITY
                                ;-
000160'                         INLINE<3$:>                                     3$:
(2)  000160'
000160'                         LET (R0)+ := (R0)                               MOV     (R0),(R0)+
(4)  000160' 011020
000162'                         INLINE <BR 3$>                                  BR 3$
(2)  000162' 000776


                                ;+
                                ; WILL TRAP HERE WHEN TOP OF MEMORY IS FOUND
                                ; CLEAN UP STACK AND RESTORE BUSERR
                                ;-

000164'                         INLINE <2$:>                                    2$:
(2)  000164'
000164'                         INLINE <CMP (SP)+,(SP)+>                        CMP (SP)+,(SP)+
(2)  000164' 022626
000166'                         POP @#4                                         MOV     (SP)+,@#4
(2)  000166' 012637  000004


                                ;+
                                ; LOAD NCPUOP BIT INTO R1
                                ;-

000172'                         LET R1 := R1 SET.BY #NCPUOP                     BIS     #NCPUOP,R1
(6)  000172' 052701  000020


                                ;+
                                ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD 0
                                ;-

000176'                         IF @#42 NE #0 THEN                              TST     @#42
(6)  000176' 005737  000042                                                     BEQ     50006$
(9)  000202' 001402
000204'                                 LET R1 := R1 SET.BY #AUTO               BIS     #AUTO,R1
(6)  000204' 052701  000010
000210'                         ENDIF                                   50006$:
(4)  000210'
```

```
  712                                   ;+
  713                                   ; LOAD UP CONFIGURATION WORD 0
  714                                   ;-
  715
  716  000210'                          LET DT.CF0(R2) := DT.CF0(R2) SET.BY R1
  (6)  000210' 050162  000014                                                    BIS     R1,DT.CF0(R2)
  717
```

```
                                .SBTTL KERNEL - SORT MODULE LIST


                                ;+
                                ; COUNT THE MODULES
                                ;-

                                        LET R0 := #0                                    CLR     R0
000214'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4)  000214' 005000
000216'                                 WHILE (R1) NE #ENDLST DO                50007$:
(4)  000216' 016201  000032                                                             CMP     (R1),#ENDLST
000222'                                                                                 BEQ     50010$
(4)  000222'
(6)  000222' 021127  000000                     LET R0 := R0 + #1                       INC     R0
(9)  000226' 001404
000230'                                         LET R1 := R1 + #2                       ADD     #2,R1
(6)  000230' 005200
000232'                                 ENDDO                                           BR      50007$
(6)  000232' 062701  000002
000236'                                                                         50010$:
(4)  000236' 000771
(3)  000240'

                                ;+
                                ; START THE SORT OPERATION, AND SORT "BKMODS" TO BOTTOM OF MODQ LIST
                                ; DEVELOPING A "BKMOD QUEUE".
                                ;-

000240'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4)  000240' 016201  000032

                                ;+
                                ; WORK THRU THE LIST UNTIL THE END IS REACHED
                                ;-

000244'                                 WHILE R0 GT #0 DO                       50011$:
(4)  000244'                                                                            TST     R0
(6)  000244' 005700                                                                     BLE     50012$
(9)  000246' 003432                             REPEAT
000250'                                                                         50013$:
(3)  000250'

                                ;+
                                ; IS THE CURRENT ENTRY A BKMOD MODULE?
                                ;-

000250'                                         LET R4 := (R1)                          MOV     (R1),R4
(4)  000250' 011104
000252'                                         IF #BKMOD SETIN STAT(R4) THEN           BIT     #BKMOD,STAT(R4)
(6)  000252' 032764  000020  000026                                                    BEQ     50014$
(9)  000260' 001414

                                ;+
                                ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
                                ;-
```

```
                                                    LET R3 := 2(R1)
000262'                                                                 MOV     2(R1),R3
(4) 000262' 016103  000002

                        ;+
                        ; MAKE SURE ITS NOT THE END OF THE LIST
                        ;-

                                                    IF R3 NE #ENDLST THEN
000266'                                                                 CMP     R3,#ENDLST
(6) 000266' 020327  000000                                              BEQ     50015$
(9) 000272' 001407

                        ;+
                        ; IF ITS NOT A BKMOD, SWAP THEM
                        ;-

                                                        IF #BKMOD NOTSETIN STAT(R3) THEN
000274'                                                                 BIT     #BKMOD,STAT(R3)
(6) 000274' 032763  000020  000026                                      BNE     50016$
(9) 000302' 001003
                                                            LET 2(R1) := (R1)
000304'                                                                 MOV     (R1),2(R1)
(4) 000304' 011161  000002
                                                            LET (R1) := R3
000310'                                                                 MOV     R3,(R1)
(4) 000310' 010311
                                                        ENDIF
000312'                                                                 50016$:
(4) 000312'
                                                    ENDIF
000312'                                                                 50015$:
(4) 000312'
                                                ENDIF
000312'                                                                 50014$:
(4) 000312'

                        ;+
                        ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
                        ;-

                                                LET R1 := R1 + #2
000312'                                                                 ADD     #2,R1
(6) 000312' 062701  000002
                                            UNTIL (R1) EQ #ENDLST
000316'                                                                 CMP     (R1),#ENDLST
(3) 000316' 021127  000000                                              BNE     50013$
(6) 000322' 001352

                        ;+
                        ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
                        ;-

                                                LET R0 := R0 - #1
000324'                                                                 DEC     R0
(6) 000324' 005300
                                                LET R1 := DT.MLST(R2)
000326'                                                                 MOV     DT.MLST(R2),R1
(4) 000326' 016201  000032
                                        ENDDO
000332'                                                                 BR      50011$
(4) 000332' 000744                                              50012$:
(3) 000334'

                        ;+
                        ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
                        ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
                        ; A BACKGROUND LIST)  INTO THE DATA TABLE
                        ;-
```

```
                                        LET R1 := DT.MLST(R2)                                   MOV     DT.MLST(R2),R1
                                    ;+
                                    ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
                                    ;-
                                        LET DT.BLST(R2) := #0                                   CLR     DT.BLST(R2)
                                        LET R3 := #-1                                           MOV     #-1,R3
                                        WHILE (R1) NE #ENDLST AND R3 NE #0 DO           50017$:
                                                                                                CMP     (R1),#ENDLST
                                                                                                BEQ     50020$
                                                                                                TST     R3
                                                                                                BEQ     50020$
                                            LET R4 := (R1)                                      MOV     (R1),R4
                                            IF #BKMOD SETIN STAT(R4) THEN                       BIT     #BKMOD,STAT(R4)
                                                                                                BEQ     50021$
                                    ;+
                                    ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
                                    ;-
                                                LET DT.BLST(R2) := R1                           MOV     R1,DT.BLST(R2)
                                                LET R1 := #ENDLST                               MOV     #ENDLST,R1
                                                LET R3 := #0                                    CLR     R3
                                            ELSE                                                BR      50022$
                                                                                        50021$:
                                    ;+
                                    ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
                                    ;-
                                                LET R1 := R1 + #2                               ADD     #2,R1
                                            ENDIF                                       50022$:
                                        ENDDO                                                   BR      50017$
                                                                                        50020$:
                                    ;+
                                    ; NOW RETURN TO THE CALLER
                                    ;-
                                        CALL RESREG                                             JSR     PC,RESREG
```

```
000422'                                 ENDRTN
(3)  000422'                                                                    50000$:
(3)  000422'                                                                    50001$:
(2)  000422' 000207                                                                     RTS     PC

                                ;+
                                ; LABEL TYPE QUEUE HIGH LIMIT
                                ;-

                                ;+
                                ; ALLOW STORAGE FOR TQ
                                ;-

000424' 000120                  .BLKB   ^D<80>
000544'                         INLINE <OV.TQEND:>
(2)  000544'                                                                            OV.TQEND:



                                ;************************************
                                ; NOTE:  THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
                                ;   (THAT IS, AFTER SIZPOL AND TQ.END LABELS)

                                ;++++++++++++++++++++++++++++++++++++++
                                ;    OVERLAY REGIONS EQUATES
                                ;++++++++++++++++++++++++++++++++++++++
        000000'                 OV.KBBUF = SIZPOL                       ;KEYBOARD BUFFER STARTING ADDRESS
        000122'                 OV.CQ= OV.KBBUF + ^D<82>                ;CONTROL QUEUE STARTING ADDRESS
        000244'                 OV.TQ = OV.CQ + ^D<82>                  ;TYPE QUEUE STARTING ADDRESS
        000120'                 OV.HIKB = OV.CQ - 2                     ;KEYBOARD BUFFER HIGH LIMIT
        000122                  OV.KBSIZ = OV.CQ - OV.KBBUF             ;KEYBOARD BUFFER SIZE
        000242'                 OV.HICQ = OV.TQ - 2                     ;CONTROL QUEUE HIGH LIMIT
        000024                  OV.CQSIZ = <OV.TQ - OV.CQ>/4            ;CONTROL QUEUE SIZE
        000300                  OV.A = OV.TQEND - OV.TQ                 ;GET THE TQ SIZE
        000000                  OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
        000544'                 OV.HITQ = OV.TQEND - OV.REM             ;TYPE QUEUE HIGH LIMIT
        000030                  OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10      ;TYPE QUEUE SIZE

        000001                          .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KTERRO= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPQP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OV.A  = 000300
OV.CQ = 000122RG
OV.CQS= 000024 G
OV.HIC= 000242RG
OV.HIK= 000120RG
OV.HIT= 000544RG
OV.KBB= 000000RG
OV.KBS= 000122 G
OV.REM= 000000
OV.TQ = 000244RG
OV.TQE  000544R
OV.TQS= 000030 G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SIZPOL  000000RG
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
SS.RFS= 160000
SS.OKC= 004000
```

```
SS.28 = 000034
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000120
$NSK2 = 000110
$NSK3 = 000110
$NSK4 = 000110
$NSK5 = 000110
$SAVLE= 177777
$SSK0 = 050020
$TAGLE= 177777
$TAGNU= 050023
$TEMP = 000300
$TSK0 = 050017
$TSK1 = 050020
$TSK2 = 050022
$TSK3 = 050014
$TSK4 = 050015
$TSK5 = 050016
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 000372R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 000544R

. ABS.  000000     000
        000544     001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:SIZPLA,DSKZ:SIZPLA=SPMAC/ML,EQUATE,SIZPLA
RUN-TIME: 18 8 .4 SECONDS
RUN-TIME RATIO: 47/27=1.7
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SIZPLB SIZE AND POLL SYSTEM - FOR NON-KT MACHINES
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SIZPLB
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SIZE THE SYSTEM AND DETERMINE IF PARITY
                        ;       OR ECC IS PRESENT AND ALSO SORT THE
                        ;       MODQ LIST TO GENERATE A BACKGROUND LIST(QUEUE).
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CONFIGURATION WORD 0
                        ;       EXERCISER SIZE
                        ;       SYSTEM SIZE
                        ;       PARITY TABLE ADDRESS
                        ;       MODULE LIST
                        ;       BACKGROUND LIST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       PARITY TABLE ADDRESS
                        ;       SYSTEM SIZE
                        ;       EXERCISE SIZE
                        ;       CONFIGURATION WORD 0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SIZPOL IN <A>
                        ;                 A - DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
                        000001  $LSTIN = 1
                        000001  $LSTTAG = 1


                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  SAVREG                  ;SARE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  HTPAER                  ;PARITY VECTOR
                                .GLOBL  HRDADRCHK               ;ILLEGAL ADDRESS CHECK
                                .GLOBL  CPUCPE                  ;CPU ERROR REGISTER
                                .GLOBL  CCNTRL                  ;CACHE CONTROL REGISTER
                                .GLOBL  KONTRL                  ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER

                                ;****************
                                ; LOCAL EQUATES
                                ;
                        172100  SS.PCSR = 172100                ;START OF PARITY CSRS
                        172136  SS.PCLS = 172136                ;LAST PARITY CSR
                        004000  SS.0KC  = 4000                  ;ONE K CHUNK
                        000034  SS.28   = ^D28                  ;28 DECIMAL
                        177600  SS.IOP  = 177600                ;I/O PAGE
                        160000  SS.RFS  = 160000                ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  SIZPOL                  ;MODULE ENTRY POINT



                                ;++++++++++++++++++++++++++++++
                                ;    OVERLAY REGION GLOBALS
                                ;++++++++++++++++++++++++++++++
                                .GLOBL  OV.KBBUF
                                .GLOBL  OV.HIKB
                                .GLOBL  OV.KBSIZ
                                .GLOBL  OV.CQ
                                .GLOBL  OV.HICQ
                                .GLOBL  OV.CQSIZ
                                .GLOBL  OV.TQ
                                .GLOBL  OV.HITQ
                                .GLOBL  OV.TQSIZ
```

```
                                .SBTTL KERNEL - DETERMINE PROCESSOR TYPE



                                ROUTINE SIZPOL <DTABLE>
000000'                                                                 SIZPOL:
(2)  000000'
                                ;+
                                ; STUFF PARITY VECTOR
                                ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
                                ; R1
                                ;-



                                LET @#114 := #HTPAER
000000'                                                                 MOV     #HTPAER,@#114
(4)  000000' 012737  000000G 000114
                                LET @#116 := #PRI7
000006'                                                                 MOV     #PRI7,@#116
(4)  000006' 012737  000340  000116
                                CALL SAVREG
000014'                                                                 JSR     PC,SAVREG
(3)  000014' 004767  000000G
                                LET R2 := DTABLE(R5)
000020'                                                                 MOV     DTABLE(R5),R2
(4)  000020' 016502  000000
                                LET R1 := #0
000024'                                                                 CLR     R1
(4)  000024' 005001

                                ;+
                                ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI-11, FIRST CHECK FOR
                                ; CPU ERROR REGISTER
                                ;-

                                CALL HRDADRCHK IN <CPUCPE>
000026'                                                                 MOV     R5,-(SP)
(3)  000026' 010546                                                     MOV     CPUCPE,-(R5)
(4)  000030' 016745  000000G                                           JSR     PC,HRDADRCHK
(3)  000034' 004767  000000G                                           MOV     (SP)+,R5
(3)  000040' 012605

                                IF.NO.ERROR THEN
000042'                                                                 BCS     50002$
(6)  000042' 103422
                                ;+
                                ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
                                ; 11/60
                                ;-

                                        PUSH @#10
000044'                                                                 MOV     @#10,-(SP)
(2)  000044' 013746  000010
                                        LET @#10 := #1$
000050'                                                                 MOV     #1$,@#10
(4)  000050' 012737  000070' 000010
                                        INLINE <MED>
000056'                                                                 MED
(2)  000056' 076600
                                        INLINE <RDSERV>
000060'                                                                 RDSERV
(2)  000060' 000101
                                        LET R1 := R1 SET.BY #PDP60
000062'                                                                 BIS     #PDP60,R1
(6)  000062' 052701  004000
                                        INLINE <BR 3$>
000066'                                                                 BR 3$
(2)  000066' 000405
```

```
                                INLINE <1$:>
                                                                        1$:
                                ;+
                                ; SET RETURN TO 2$ TO RESTORE PSW
                                ;-
                                        LET (SP) := #2$
                                                                        MOV     #2$,(SP)
                                        INLINE <RTI>
                                                                        RTI
                                ;+
                                ; MUST BE AN 11/70, BECAUSE OF TRAP
                                ;-
                                INLINE <2$:>
                                                                        2$:
                                        LET R1 := R1 SET.BY #PDP70
                                                                        BIS     #PDP70,R1
                                ;+
                                ; RESTORE RESERVED INSTRUCTION TRAP
                                ;-
                                INLINE <3$:>
                                                                        3$:
                                        POP @#10
                                                                        MOV     (SP)+,@#10
                                ELSE
                                                                        BR      50003$
                                                                50002$:
                                ;+
                                ; CHECK TO SEE IF LSI11
                                ;-
                                        CALL HRDADRCHK IN <#PSW>
                                                                        MOV     R5,-(SP)
                                                                        MOV     #PSW,-(R5)
                                                                        JSR     PC,HRDADRCHK
                                                                        MOV     (SP)+,R5
                                        IF.ERROR THEN
                                                                        BCC     50004$
                                                LET R1 := R1 SET.BY #PDPLSI
                                                                        BIS     #PDPLSI,R1
                                        ENDIF
                                                                50004$:
                                ENDIF
                                                                50003$:
```

```
                        .SBTTL KERNEL - DETERMINE IF CACHE OPTION AVAILABLE

                        ;+
                        ; DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING CACHE
                        ; CONTROL REGISTER
                        ;-

                        CALL HRDADRCHK IN <CCNTRL>
                                                                        MOV     R5,-(SP)
(3)  000132' 010546                                                     MOV     CCNTRL,-(R5)
(4)  000134' 016745  000000G                                            JSR     PC,HRDADRCHK
(3)  000140' 004767  000000G                                            MOV     (SP)+,R5
(3)  000144' 012605

                        IF.NO.ERROR THEN
(6)  000146' 103402                                                     BCS     50005$
                                LET R1 := R1 SET.BY #CAPRES
(6)  000150' 052701  000004                                             BIS     #CAPRES,R1
                        ENDIF
(4)  000154'                                                    50005$:
```

```
                        .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT

                        ;+
                        ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
                        ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
                        ; IN PARITY TABLE.
                        ;-

000154'                 LET R0 := DT.PTA(R2)
(4) 000154' 016200 000064                                       MOV     DT.PTA(R2),R0

                        LET R3 := #SS.PCSR
000160'                                                         MOV     #SS.PCSR,R3
(4) 000160' 012703 172100
                        WHILE R3 LE #SS.PCLS DO
000164'                                                 50006$:
(4) 000164'                                                     CMP     R3,#SS.PCLS
(6) 000164' 020327 172136                                       BGT     50007$
(9) 000170' 003035
                                CALL HRDADRCHK IN <R3>
000172'                                                         MOV     R5,-(SP)
(3) 000172' 010546                                              MOV     R3,-(R5)
(4) 000174' 010345                                              JSR     PC,HRDADRCHK
(3) 000176' 004767 000000G                                      MOV     (SP)+,R5
(3) 000202' 012605
                                IF.NO.ERROR THEN
000204'                                                         BCS     50010$
(6) 000204' 103414
                                        LET (R0)+ := R3
000206'                                                         MOV     R3,(R0)+
(4) 000206' 010320

                        ;+
                        ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
                        ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
                        ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
                        ;-
                                LET (R3) := (R3) SET.BY #BIT01
000210'                                                         BIS     #BIT01,(R3)
(6) 000210' 052713 000002
                                IF #BIT01 SETIN (R3) THEN
000214'                                                         BIT     #BIT01,(R3)
(6) 000214' 032713 000002                                       BEQ     50011$
(9) 000220' 001403
                                        LET R1 := R1 SET.BY #ECCMEM
000222'                                                         BIS     #ECCMEM,R1
(6) 000222' 052701 000100
                                ELSE
000226'                                                         BR      50012$
(4) 000226' 000402
(3) 000230'                                             50011$:
                                        LET R1 := R1 SET.BY #PARPRES
000230'                                                         BIS     #PARPRES,R1
(6) 000230' 052701 002000
                                ENDIF
000234'                                                 50012$:
(4) 000234'
                                LET (R3) := #0
000234'                                                         CLR     (R3)
(4) 000234' 005013
                        ENDIF
000236'                                                 50010$:
(4) 000236'

                        ;+
                        ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
                        ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
                        ; PARITY CSR + 2
```

```
                                        ;-
                                                IF #PDP70 SETIN R1 THEN

                                                        LET R1 := R1 SET.BY #PARPRES

                                                        LET R3 := R3 + #4

                                                ELSE

                                                        LET R3 := R3 + #2

                                                ENDIF

                                        ENDDO
```

```
000236'                                                 BIT     #PDP70,R1
000236' 032701  010000                                  BEQ     50013$
000242' 001405
000244'                                                 BIS     #PARPRES,R1
000244' 052701  002000
000250'                                                 ADD     #4,R3
000250' 062703  000004
000254'                                                 BR      50014$
000254' 000402
000256'                                         50013$:
000256'                                                 ADD     #2,R3
000256' 062703  000002
000262'                                         50014$:
000262'
000262'                                                 BR      50006$
000262' 000740
000264'                                         50007$:
```

```
                        .SBTTL KERNEL - DETERMINE SYSTEM SIZE

                        ;+
                        ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
                        ; AND LOAD IT INTO EXERCISER SIZE WORD
                        ;-

000264'                 LET DT.ESIZ(R2) := 0                                    MOV     0,DT.ESIZ(R2)
(4) 000264' 016762 000000 000044

                        ;+
                        ; DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
                        ; THAN 28K.
                        ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
                        ;-

000272'                 LET R0 := #0                                            CLR     R0
(4) 000272' 005000
000274'                 LET R3 := #0                                            CLR     R3
(4) 000274' 005003

                        ;+
                        ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K OF MEMORY IS FOUND
                        ;-

000276'                 REPEAT                                          50015$:
(3) 000276'
000276'                         LET R0 := R0 + #1                               INC     R0
(6) 000276' 005200
000300'                         LET R3 := R3 + #SS.0KC                          ADD     #SS.0KC,R3
(6) 000300' 062703 004000
000304'                         CALL HRDADRCHK IN <R3>                          MOV     R5,-(SP)
(3) 000304' 010546                                                              MOV     R3,-(R5)
(4) 000306' 010345                                                              JSR     PC,HRDADRCHK
(3) 000310' 004767 000000G                                                      MOV     (SP)+,R5
(3) 000314' 012605
000316'                         IF.ERROR THEN                                   BCC     50016$
(6) 000316' 103001
000320'                                 INLINE <BR 4$>                          BR 4$
(2) 000320' 000403
000322'                         ENDIF
(4) 000322'                                                             50016$:
000322'                 UNTIL R0 EQ #SS.28                                      CMP     R0,#SS.28
(3) 000322' 020027 000034                                                       BNE     50015$
(6) 000326' 001363

                        ;+
                        ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
                        ;-

000330'                 INLINE <4$:>                                            4$:
(2) 000330'
000330'                 IF R0 EQ #SS.28 THEN                                    CMP     R0,#SS.28
(6) 000330' 020027 000034                                                       BNE     50017$
(9) 000334' 001011
000336'                         CALL HRDADRCHK IN <#SS.RFS>
```

```
(3)  000336' 010546                                                          MOV     R5,-(SP)
(4)  000340' 012745  160000                                                  MOV     #SS.RFS,-(R5)
(3)  000344' 004767  000000G                                                 JSR     PC,HRDADRCHK
(3)  000350' 012605                                                          MOV     (SP)+,R5
782  000352'                        IF.NO.ERROR THEN
(6)  000352' 103402                                                          BCS     50020$
783  000354'                                LET R3 := R3 + #10000
(6)  000354' 062703  010000                                                  ADD     #10000,R3
784  000360'                        ENDIF
(4)  000360'                                                         50020$:
785  000360'                    ENDIF
(4)  000360'                                                         50017$:
786
787                             ;+
788                             ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
789                             ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
790                             ; SIZE WORD.
791                             ;-
792
793  000360'                    LET CARRY := #0
(4)  000360' 000241                                                          CLC
794  000362'                    LET DT.SSIZ(R2) := R3 ROTATE -6
(4)  000362' 010362  000046                                                  MOV     R3,DT.SSIZ(R2)
(7)  000366' 006062  000046                                                  ROR     DT.SSIZ(R2)
(7)  000372' 006062  000046                                                  ROR     DT.SSIZ(R2)
(7)  000376' 006062  000046                                                  ROR     DT.SSIZ(R2)
(7)  000402' 006062  000046                                                  ROR     DT.SSIZ(R2)
(7)  000406' 006062  000046                                                  ROR     DT.SSIZ(R2)
(7)  000412' 006062  000046                                                  ROR     DT.SSIZ(R2)
795
```

```
                        .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT

                        ;+
                        ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
                        ;-

000416'                 IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
                                                                        BIT     #PARPRES,R1
(6) 000416' 032701  002000                                              BNE     50021$
(8) 000422' 001003                                                      BIT     #ECCMEM,R1
(6) 000424' 032701  000100                                              BEQ     50022$
(9) 000430' 001424                                              50021$:
(6) 000432'

                        ;+
                        ; IF CACHE PRESENT, SET BIT 0 AND 1 IN CACHE CONTROL REGISTER
                        ; THIS WILL TURN CACHE OFF
                        ;-

000432'                         IF #CAPRES SETIN R1 THEN
                                                                        BIT     #CAPRES,R1
(6) 000432' 032701  000004                                              BEQ     50023$
(9) 000436' 001406
000440'                                 LET KONTRL := #3
                                                                        MOV     #3,KONTRL
(4) 000440' 012767  000003  000000G
000446'                                 LET @CCNTRL := KONTRL
                                                                        MOV     KONTRL,@CCNTRL
(4) 000446' 016777  000000G 000000G
000454'                         ENDIF
(4) 000454'                                                     50023$:

                        ;+
                        ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
                        ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
                        ; OF MEMORY
                        ;-


                        ;+
                        ; SET UP POINTER AND SAVE BUSERR VECTOR
                        ;-

000454'                         LET R0 := #0
                                                                        CLR     R0
(4) 000454' 005000
000456'                         PUSH @#4
                                                                        MOV     @#4,-(SP)
(2) 000456' 013746  000004
000462'                         LET @#4 := #6$
                                                                        MOV     #6$,@#4
(4) 000462' 012737  000474' 000004


                        ;+
                        ; NOW WRITE GOOD PARITY
                        ;-
                        INLINE<5$:>
000470'                                                                 5$:
(2) 000470'
000470'                         LET (R0)+ := (R0)
                                                                        MOV     (R0),(R0)+
(4) 000470' 011020
000472'                         INLINE <BR 5$>
```

```
(2)  000472' 000776                                                                  BR 5$
838
839
840                                 ;+
841                                 ; WILL TRAP HERE WHEN THROUGH, CLEAN UP STACK AND RESTORE BUSERR
842                                 ;-
843
844  000474'                        INLINE <6$:>                                     6$:
(2)  000474'
845  000474'                        INLINE <CMP (SP)+,(SP)+>                         CMP (SP)+,(SP)+
(2)  000474' 022626
846  000476'                        POP @#4                                          MOV     (SP)+,@#4
(2)  000476' 012637  000004
847
848
849  000502'                        ENDIF
(4)  000502'                                                                  50022$:
850
851                                 ;+
852                                 ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD0
853                                 ;-
854
855  000502'                        IF @#42 NE #0 THEN                               TST     @#42
(6)  000502' 005737  000042                                                         BEQ     50024$
(9)  000506' 001402
856  000510'                                LET R1 := R1 SET.BY #AUTO                BIS     #AUTO,R1
(6)  000510' 052701  000010
857  000514'                        ENDIF
(4)  000514'                                                                  50024$:
858
859                                 ;+
860                                 ; CLEAN UP R1 BEFORE LOADING CONFIGURATION WORD 0
861                                 ; LOAD UP CONFIGURATION WORD 0
862                                 ;-
863
864  000514'                        LET R1 := R1 CLR.BY #SAM                         BIC     #SAM,R1
(6)  000514' 042701  075464
865  000520'                        LET DT.CF0(R2) := DT.CF0(R2) SET.BY R1           BIS     R1,DT.CF0(R2)
(6)  000520' 050162  000014
866
```

```
                                .SBTTL KERNEL - SORT MODULE LIST


                                ;+
                                ; COUNT THE MODULES
                                ;-

000524'                                 LET R0 := #0                                    CLR     R0
(4) 000524' 005000
000526'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4) 000526' 016201  000032
000532'                                 WHILE (R1) NE #ENDLST DO                50025$:
(4) 000532'                                                                             CMP     (R1),#ENDLST
(6) 000532' 021127  000000                                                             BEQ     50026$
(9) 000536' 001404
000540'                                         LET R0 := R0 + #1                       INC     R0
(6) 000540' 005200
000542'                                         LET R1 := R1 + #2                       ADD     #2,R1
(6) 000542' 062701  000002
000546'                                 ENDDO                                           BR      50025$
(4) 000546' 000771
(3) 000550'                                                                     50026$:


                                ;+
                                ; START THE SORT OPERATION, AND SORT "BKMODS" TO BOTTOM OF
                                ; MODQ LIST, DEVELOPING A "BKMOD QUEUE".
                                ;-

000550'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4) 000550' 016201  000032


                                ;+
                                ; WORK THRU THE LIST UNTIL THE END IS REACHED
                                ;-

000554'                                 WHILE R0 GT #0 DO                       50027$:
(4) 000554'                                                                             TST     R0
(6) 000554' 005700                                                                      BLE     50030$
(9) 000556' 003432
000560'                                         REPEAT                          50031$:
(3) 000560'


                                ;+
                                ; IS THE CURRENT ENTRY A BKMOD MODULE?
                                ;-

000560'                                         LET R4 := (R1)                          MOV     (R1),R4
(4) 000560' 011104
000562'                                         IF #BKMOD SETIN STAT(R4) THEN           BIT     #BKMOD,STAT(R4)
(6) 000562' 032764  000020  000026                                                      BEQ     50032$
(9) 000570' 001414


                                ;+
                                ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
                                ;-
```

```
                                                    LET R3 := 2(R1)
000572'                                                                 MOV     2(R1),R3
(4) 000572' 016103  000002
                        ;+
                        ; MAKE SURE ITS NOT THE END OF THE LIST
                        ;-
                                                    IF R3 NE #ENDLST THEN
000576'                                                                 CMP     R3,#ENDLST
(6) 000576' 020327  000000                                              BEQ     50033$
(9) 000602' 001407
                        ;+
                        ; IF ITS NOT A BKMOD, SWAP THEM
                        ;-
                                                        IF #BKMOD NOTSETIN STAT(R3) THEN
000604'                                                                 BIT     #BKMOD,STAT(R3)
(6) 000604' 032763  000020  000026                                      BNE     50034$
(9) 000612' 001003
                                                            LET 2(R1) := (R1)
000614'                                                                 MOV     (R1),2(R1)
(4) 000614' 011161  000002
                                                            LET (R1) := R3
000620'                                                                 MOV     R3,(R1)
(4) 000620' 010311
                                                        ENDIF
000622'                                                         50034$:
(4) 000622'
                                                    ENDIF
000622'                                                         50033$:
(4) 000622'
                                                ENDIF
000622'                                                         50032$:
(4) 000622'
                        ;+
                        ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
                        ;-
                                                LET R1 := R1 + #2
000622'                                                                 ADD     #2,R1
(6) 000622' 062701  000002
                                            UNTIL (R1) EQ #ENDLST
000626'                                                                 CMP     (R1),#ENDLST
(3) 000626' 021127  000000                                              BNE     50031$
(6) 000632' 001352
                        ;+
                        ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
                        ;-
                                                LET R0 := R0 - #1
000634'                                                                 DEC     R0
(6) 000634' 005300
                                                LET R1 := DT.MLST(R2)
000636'                                                                 MOV     DT.MLST(R2),R1
(4) 000636' 016201  000032
                                        ENDDO
000642'                                                                 BR      50027$
(4) 000642' 000744                                              50030$:
(3) 000644'
                        ;+
                        ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
                        ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
                        ; A BACKGROUND LIST)  INTO THE DATA TABLE
                        ;-
```

```
                                        LET R1 := DT.MLST(R2)
000644'                                                                 MOV     DT.MLST(R2),R1
(4) 000644' 016201  000032
                                    ;+
                                    ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
                                    ;-
                                        LET DT.BLST(R2) := #0
000650'                                                                 CLR     DT.BLST(R2)
(4) 000650' 005062  000034
                                        LET R3 := #-1
000654'                                                                 MOV     #-1,R3
(4) 000654' 012703  177777
                                        WHILE (R1) NE #ENDLST AND R3 NE #0 DO
000660'                                                         50035$:
(4) 000660'
000660' 021127  000000                                                  CMP     (R1),#ENDLST
000664' 001420                                                          BEQ     50036$
000666' 005703                                                          TST     R3
000670' 001416                                                          BEQ     50036$
                                            LET R4 := (R1)
000672'                                                                 MOV     (R1),R4
(4) 000672' 011104
                                            IF #BKMOD SETIN STAT(R4) THEN
000674'                                                                 BIT     #BKMOD,STAT(R4)
(6) 000674' 032764  000020  000026                                      BEQ     50037$
(9) 000702' 001406
                                    ;+
                                    ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
                                    ;-
                                                LET DT.BLST(R2) := R1
000704'                                                                 MOV     R1,DT.BLST(R2)
(4) 000704' 010162  000034
                                                LET R1 := #ENDLST
000710'                                                                 MOV     #ENDLST,R1
(4) 000710' 012701  000000
                                                LET R3 := #0
000714'                                                                 CLR     R3
(4) 000714' 005003
                                            ELSE
000716'                                                                 BR      50040$
(4) 000716' 000402
000720'                                                         50037$:
                                    ;+
                                    ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
                                    ;-
                                                LET R1 := R1 + #2
000720'                                                                 ADD     #2,R1
(6) 000720' 062701  000002
                                            ENDIF
000724'                                                         50040$:
(4) 000724'
                                        ENDDO
000724'                                                                 BR      50035$
(4) 000724' 000755
000726'                                                         50036$:
                                    ;+
                                    ; NOW RETURN TO THE CALLER
                                    ;-
                                        CALL RESREG
000726'                                                                 JSR     PC,RESREG
(3) 000726' 004767  000000G
```

```
  981  000732'                                ENDRTN
  (3)  000732'                                                                    50000$:
  (3)  000732'                                                                    50001$:
  (2)  000732' 000207                                                                     RTS     PC
  982
  983                                ;+
  984                                ; LABEL TYPE QUEUE HIGH LIMIT
  985                                ;-
  986
  987  000734'                       INLINE <OV.TQEND:>                                   OV.TQEND:
  (2)  000734'
  988
  989
  990
  991
  992
  993                                ;************************************
  994                                ; NOTE:  THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
  995                                ;   (THAT IS, AFTER SIZPOL AND TQ.END LABELS)
  996
  997                                ;++++++++++++++++++++++++++++++++++++++
  998                                ;    OVERLAY REGIONS EQUATES
  999                                ;++++++++++++++++++++++++++++++++++++++
 1000          000000'               OV.KBBUF = SIZPOL                       ;KEYBOARD BUFFER STARTING ADDRESS
 1001          000122'               OV.CQ= OV.KBBUF + ^D<82>                ;CONTROL QUEUE STARTING ADDRESS
 1002          000314'               OV.TQ = OV.CQ + ^D<122>                 ;TYPE QUEUE STARTING ADDRESS
 1003          000120'               OV.HIKB = OV.CQ - 2                     ;KEYBOARD BUFFER HIGH LIMIT
 1004          000122                OV.KBSIZ = OV.CQ - OV.KBBUF             ;KEYBOARD BUFFER SIZE
 1005          000312'               OV.HICQ = OV.TQ - 2                     ;CONTROL QUEUE HIGH LIMIT
 1006          000036                OV.CQSIZ = <OV.TQ - OV.CQ>/4            ;CONTROL QUEUE SIZE
 1007          000420                OV.A = OV.TQEND - OV.TQ                 ;GET THE TQ SIZE
 1008          000000                OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
 1009          000734'               OV.HITQ = OV.TQEND - OV.REM             ;TYPE QUEUE HIGH LIMIT
 1010          000042                OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10      ;TYPE QUEUE SIZE
 1011
 1012          000001                        .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CPUCPE= ****** G
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
HTPAER= ****** G
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KTERR0= 000040
KTPRES= 000400
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 075600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OV.A  = 000420
OV.CQ = 000122RG
OV.CQS= 000036 G
OV.HIC= 000312RG
OV.HIK= 000120RG
OV.HIT= 000734RG
OV.KBB= 000000RG
OV.KBS= 000122 G
OV.REM= 000000
OV.TQ = 000314RG
OV.TQE  000734R
OV.TQS= 000042 G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRIO  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
RSVALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SIZPOL  000000RG
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
SR0   = 177572
SR1   = 177574
```

```
SR2    = 177576
SR3    = 172516
SS.IOP= 177600
SS.PCL= 172136
SS.PCS= 172100
SS.RFS= 160000
SS.0KC= 004000
SS.28  = 000034
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR   = 000116
WDTO   = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG  = 000005
XOFF   = 000023
XON    = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO  = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF  = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO  = 000403
$F$OR  = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0  = 000001
$ISK1  = 000001
$ISK2  = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0  = 000300
$NSK1  = 000120
$NSK2  = 000110
$NSK3  = 000110
$NSK4  = 000110
$NSK5  = 000110
$SAVLE= 177777
$SSK0  = 050036
$TAGLE= 177777
$TAGNU= 050041
$TEMP  = 000300
$TSK0  = 050035
$TSK1  = 050036
$TSK2  = 050040
$TSK3  = 050032
$TSK4  = 050033
$TSK5  = 050034
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST  = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC  = 000702R
$$LOCN= 000000
$$REG  = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC  = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO   = 000000
$$$TAG= 050000
.      = 000734R

. ABS.  000000     000
        000734     001

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:SIZPLB,DSKZ:SIZPLB=SPMAC/ML,EQUATE,SIZPLB
RUN-TIME: 23 14 .4 SECONDS
RUN-TIME RATIO: 59/38=1.5
CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SIZPLC SIZE AND POLL SYSTEM - KT OPTIONS
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SIZPLC
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
                        ;       OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
                        ;       WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
                        ;       A BACKGROUND MODULE LIST (QUEUE)
                        ;       NOTE: IT WILL NOT SIZE FOR 22-BIT ADDRESSING
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CONFIGURATION WORD 0
                        ;       EXERCISER SIZE
                        ;       SYSTEM SIZE
                        ;       PARITY TABLE ADDRESS
                        ;       MODULE LIST
                        ;       BACKGROUND LIST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       PARITY TABLE ADDRESS
                        ;       SYSTEM SIZE
                        ;       EXERCISE SIZE
                        ;       CONFIGURATION WORD 0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       KTSET - SET UP KT REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SIZPOL IN <A>
                        ;                A - DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN = 1
            000001              $LSTTAG = 1


                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  HTPAER                  ;PARITY VECTOR
                                .GLOBL  HTKT                    ;KT VECTOR
                                .GLOBL  SAVREG                  ;SARE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  KTSET                   ;SET UP KT
                                .GLOBL  HRDADRCHK               ;ILLEGAL ADDRESS CHECK
                                .GLOBL  CPUCPE                  ;11/70 CPU ERROR REGISTER
                                .GLOBL  CCNTRL                  ;CACHE CONTROL REGISTER
                                .GLOBL  KONTRL                  ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER

                                ;****************
                                ; LOCAL EQUATES
                                ;
            172100              SS.PCSR = 172100                ;START OF PARITY CSRS
            172136              SS.PCLS = 172136                ;LAST PARITY CSR
            004000              SS.0KC  = 4000                  ;ONE K CHUNK
            000034              SS.28   = ^D28                  ;28 DECIMAL
            000040              SS.1KP  = 40                    ;1K IN PAR FORMAT
            120000              SS.120K = 120000                ;FIRST ADDRESS IN PAR5 MAP
            177600              SS.IOP  = 177600                ;I/O PAGE
            160000              SS.RFS  = 160000                ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  SIZPOL                  ;MODULE ENTRY POINT



                                ;++++++++++++++++++++++++++++++++
                                ;    OVERLAY REGION GLOBALS
                                ;++++++++++++++++++++++++++++++++
                                .GLOBL  OV.KBBUF
                                .GLOBL  OV.HIKB
                                .GLOBL  OV.KBSIZ
                                .GLOBL  OV.CQ
                                .GLOBL  OV.HICQ
                                .GLOBL  OV.CQSIZ
                                .GLOBL  OV.TQ
                                .GLOBL  OV.HITQ
                                .GLOBL  OV.TQSIZ
```

```
                                .SBTTL KERNEL - DETERMINE PROCESSOR TYPE



000000'                         ROUTINE SIZPOL <DTABLE>                                 SIZPOL:
(2) 000000'

                                LET @#114 := #HTPAER                                            MOV     #HTPAER,@#114
(4) 000000' 012737  000000G 000114
                                LET @#116 := #PRI7                                              MOV     #PRI7,@#116
(4) 000006' 012737  000340  000116

                                LET @#250 := #HTKT                                              MOV     #HTKT,@#250
(4) 000014' 012737  000000G 000250
                                LET @#252 := #PRI7                                              MOV     #PRI7,@#252
(4) 000022' 012737  000340  000252

                                ;+
                                ; STUFF PARITY AND KT VECTORS
                                ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
                                ; R1
                                ;-

000030'                         CALL SAVREG                                                     JSR     PC,SAVREG
(3) 000030' 004767  000000G
                                LET R2 := DTABLE(R5)                                            MOV     DTABLE(R5),R2
(4) 000034' 016502  000000
                                LET R1 := #0                                                    CLR     R1
(4) 000040' 005001

                                ;+
                                ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
                                ;-

000042'                         CALL HRDADRCHK IN <CPUCPE>                                      MOV     R5,-(SP)
(3) 000042' 010546                                                                              MOV     CPUCPE,-(R5)
(4) 000044' 016745  000000G                                                                     JSR     PC,HRDADRCHK
(3) 000050' 004767  000000G                                                                     MOV     (SP)+,R5
(3) 000054' 012605

000056'                         IF.NO.ERROR THEN                                                BCS     50002$
(6) 000056' 103422

                                ;+
                                ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
                                ; 11/60
                                ;-

000060'                                 PUSH @#10                                               MOV     @#10,-(SP)
(2) 000060' 013746  000010
                                        LET @#10 := #1$                                         MOV     #1$,@#10
(4) 000064' 012737  000104' 000010
000072'                                 INLINE <MED>                                            MED
(2) 000072' 076600
000074'                                 INLINE <RDSERV>                                         RDSERV
(2) 000074' 000101
```

```
                                        LET R1 := R1 SET.BY #PDP60
000076'                                                                 BIS     #PDP60,R1
(6) 000076' 052701  004000
                                        INLINE <BR 3$>
000102'                                                                 BR 3$
(2) 000102' 000405

                                INLINE <1$:>
000104'                                                                 1$:
(2) 000104'

                                ;+
                                ; SET RETURN TO 2$ TO RESTORE PSW
                                ;-

                                        LET (SP) := #2$
000104'                                                                 MOV     #2$,(SP)
(4) 000104' 012716  000112'
                                        INLINE <RTI>
000110'                                                                 RTI
(2) 000110' 000002

                                ;+
                                ; MUST BE AN 11/70, BECAUSE OF TRAP
                                ;-

                                INLINE <2$:>
000112'                                                                 2$:
(2) 000112'
                                        LET R1 := R1 SET.BY #PDP70
000112'                                                                 BIS     #PDP70,R1
(6) 000112' 052701  010000

                                ;+
                                ; RESTORE RESERVED INSTRUCTION TRAP
                                ;-

                                INLINE <3$:>
000116'                                                                 3$:
(2) 000116'
                                        POP @#10
000116'                                                                 MOV     (SP)+,@#10
(2) 000116' 012637  000010
                                ELSE
000122'                                                                 BR      50003$
(4) 000122' 000411
                                                                50002$:
000124'

                                ;+
                                ; CHECK TO SEE IF LSI11
                                ;-

                                        CALL HRDADRCHK IN <#PSW>
000124'                                                                 MOV     R5,-(SP)
(3) 000124' 010546                                                      MOV     #PSW,-(R5)
(4) 000126' 012745  177776                                              JSR     PC,HRDADRCHK
(3) 000132' 004767  000000G                                             MOV     (SP)+,R5
(3) 000136' 012605
                                        IF.ERROR THEN
000140'                                                                 BCC     50004$
(6) 000140' 103002
                                                LET R1 := R1 SET.BY #PDPLSI
000142'                                                                 BIS     #PDPLSI,R1
(6) 000142' 052701  020000

                                        ENDIF
000146'                                                         50004$:
(4) 000146'
                                ENDIF
000146'                                                         50003$:
(4) 000146'
```

```
                        .SBTTL KERNEL - DETERMINE CPU OPTIONS

                        ;+
                        ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
                        ; SEE IF EXTENDED KT" IS PRESENT
                        ;-

000146'                 CALL HRDADRCHK IN <#SR0>
(3)  000146' 010546                                             MOV     R5,-(SP)
(4)  000150' 012745  177572                                     MOV     #SR0,-(R5)
(3)  000154' 004767  000000G                                    JSR     PC,HRDADRCHK
(3)  000160' 012605                                             MOV     (SP)+,R5
000162'                 IF.NO.ERROR THEN
(6)  000162' 103413                                             BCS     50005$
000164'                         LET R1 := R1 SET.BY #KTPRES
(6)  000164' 052701  000400                                     BIS     #KTPRES,R1
000170'                         CALL HRDADRCHK IN <#SR3>
(3)  000170' 010546                                             MOV     R5,-(SP)
(4)  000172' 012745  172516                                     MOV     #SR3,-(R5)
(3)  000176' 004767  000000G                                    JSR     PC,HRDADRCHK
(3)  000202' 012605                                             MOV     (SP)+,R5
000204'                         IF.NO.ERROR THEN
(6)  000204' 103402                                             BCS     50006$
000206'                                 LET R1 := R1 SET.BY #KTXTND
(6)  000206' 052701  040000                                     BIS     #KTXTND,R1

                        ;+
                        ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
                        ; CACHE CONTROL REGISTER
                        ;-

000212'                 CALL HRDADRCHK IN <CCNTRL>
(3)  000212' 010546                                             MOV     R5,-(SP)
(4)  000214' 016745  000000G                                    MOV     CCNTRL,-(R5)
(3)  000220' 004767  000000G                                    JSR     PC,HRDADRCHK
(3)  000224' 012605                                             MOV     (SP)+,R5
000226'                 IF.NO.ERROR THEN
(6)  000226' 103402                                             BCS     50007$
000230'                         LET R1 := R1 SET.BY #CAPRES
(6)  000230' 052701  000004                                     BIS     #CAPRES,R1
000234'                 ENDIF
(4)  000234'                                                50007$:
```

```
                        .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT

                        ;+
                        ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
                        ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
                        ; IN PARITY TABLE.
                        ;-

000234'                 LET R0 := DT.PTA(R2)
(4) 000234' 016200 000064                                       MOV     DT.PTA(R2),R0

                        LET R3 := #SS.PCSR
000240'                                                         MOV     #SS.PCSR,R3
(4) 000240' 012703 172100
                        WHILE R3 LE #SS.PCLS DO
000244'                                                 50010$:
(4) 000244'                                                     CMP     R3,#SS.PCLS
(6) 000244' 020327 172136                                       BGT     50011$
(9) 000250' 003035
                                CALL HRDADRCHK IN <R3>
000252'                                                         MOV     R5,-(SP)
(3) 000252' 010546                                              MOV     R3,-(R5)
(4) 000254' 010345                                              JSR     PC,HRDADRCHK
(3) 000256' 004767 000000G                                      MOV     (SP)+,R5
(3) 000262' 012605
                                IF.NO.ERROR THEN
000264'                                                         BCS     50012$
(6) 000264' 103414
                                        LET (R0)+ := R3
000266'                                                         MOV     R3,(R0)+
(4) 000266' 010320

                        ;+
                        ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
                        ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
                        ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
                        ;-
                                LET (R3) := (R3) SET.BY #BIT01
000270'                                                         BIS     #BIT01,(R3)
(6) 000270' 052713 000002
                                IF #BIT01 SETIN (R3) THEN
000274'                                                         BIT     #BIT01,(R3)
(6) 000274' 032713 000002                                       BEQ     50013$
(9) 000300' 001403
                                        LET R1 := R1 SET.BY #ECCMEM
000302'                                                         BIS     #ECCMEM,R1
(6) 000302' 052701 000100
                                ELSE
000306'                                                         BR      50014$
(4) 000306' 000402
(3) 000310'                                             50013$:
                                        LET R1 := R1 SET.BY #PARPRES
000310'                                                         BIS     #PARPRES,R1
(6) 000310' 052701 002000
                                ENDIF
000314'                                                 50014$:
(4) 000314'
                                LET (R3) := #0
000314'                                                         CLR     (R3)
(4) 000314' 005013
                        ENDIF
000316'                                                 50012$:
(4) 00031E'

                        ;+
                        ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
                        ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
                        ; PARITY CSR + 2
                        ;-
```

```
761  000316'                          IF #PDP70 SETIN R1 THEN
(6)  000316' 032701  010000                                          BIT     #PDP70,R1
(9)  000322' 001405                                                  BEQ     50015$
762  000324'                                 LET R1 := R1 SET.BY #PARPRES
(6)  000324' 052701  002000                                          BIS     #PARPRES,R1
763  000330'                                 LET R3 := R3 + #4
(6)  000330' 062703  000004                                          ADD     #4,R3
764  000334'                          ELSE
(4)  000334' 000402                                                  BR      50016$
(3)  000336'                                                 50015$:
765  000336'                                 LET R3 := R3 + #2
(6)  000336' 062703  000002                                          ADD     #2,R3
766  000342'                          ENDIF
(4)  000342'                                                 50016$:
767  000342'                     ENDDO
(4)  000342' 000740                                                  BR      50010$
(3)  000344'                                                 50011$:
768
769
```

```
                                .SBTTL KERNEL - DETERMINE SYSTEM SIZE

                                ;+
                                ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
                                ; AND LOAD IT INTO EXERCISER SIZE WORD
                                ;-

000344'                         LET DT.ESIZ(R2) := 0
(4) 000344' 016762 000000 000044                                        MOV     0,DT.ESIZ(R2)


                                ;+
                                ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
                                ; THAN 28K.
                                ;-

000352'                         IF #KTPRES NOTSETIN R1 THEN
(6) 000352' 032701 000400                                               BIT     #KTPRES,R1
(9) 000356' 001053                                                      BNE     50017$

                                ;+
                                ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
                                ;-

000360'                                 LET R0 := #0
(4) 000360' 005000                                                      CLR     R0
000362'                                 LET R3 := #0
(4) 000362' 005003                                                      CLR     R3

                                ;+
                                ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
                                ;-

000364'                                 REPEAT
(3) 000364'                                                     50020$:
000364'                                         LET R0 := R0 + #1
(6) 000364' 005200                                                      INC     R0
000366'                                         LET R3 := R3 + #SS.0KC
(6) 000366' 062703 004000                                               ADD     #SS.0KC,R3
000372'                                         CALL HRDADRCHK IN <R3>
(3) 000372' 010546                                                      MOV     R5,-(SP)
(4) 000374' 010345                                                      MOV     R3,-(R5)
(3) 000376' 004767 000000G                                              JSR     PC,HRDADRCHK
(3) 000402' 012605                                                      MOV     (SP)+,R5
000404'                                         IF.ERROR THEN
(6) 000404' 103001                                                      BCC     50021$
000406'                                                 INLINE <BR 4$>
(2) 000406' 000403                                                      BR 4$
000410'                                         ENDIF
(4) 000410'                                                     50021$:
000410'                                 UNTIL R0 EQ #SS.28
(3) 000410' 020027 000034                                               CMP     R0,#SS.28
(6) 000414' 001363                                                      BNE     50020$

                                ;+
                                ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
```

```
                                ;-
                                INLINE <4$:>
000416'                                                                 4$:
                                        IF R0 EQ #SS.28 THEN
000416'                                                                 CMP     R0,#SS.28
000416' 020027  000034                                                  BNE     50022$
                                                CALL HRDADRCHK IN <#SS.RFS>
000424'                                                                 MOV     R5,-(SP)
000424' 010546                                                          MOV     #SS.RFS,-(R5)
000426' 012745  160000                                                  JSR     PC,HRDADRCHK
000432' 004767  000000G                                                 MOV     (SP)+,R5
000436' 012605
                                                IF.NO.ERROR THEN
000440'                                                                 BCS     50023$
000440' 103402
                                                        LET R3 := R3 + #10000
000442'                                                                 ADD     #10000,R3
000442' 062703  010000
                                                ENDIF
000446'                                                         50023$:
                                        ENDIF
000446'                                                         50022$:
                                ;+
                                ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
                                ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
                                ; SIZE WORD.
                                ;-
                                        LET CARRY := #0
000446'                                                                 CLC
000446' 000241
                                        LET DT.SSIZ(R2) := R3 ROTATE -6
000450'                                                                 MOV     R3,DT.SSIZ(R2)
000450' 010362  000046                                                  ROR     DT.SSIZ(R2)
000454' 006062  000046                                                  ROR     DT.SSIZ(R2)
000460' 006062  000046                                                  ROR     DT.SSIZ(R2)
000464' 006062  000046                                                  ROR     DT.SSIZ(R2)
000470' 006062  000046                                                  ROR     DT.SSIZ(R2)
000474' 006062  000046                                                  ROR     DT.SSIZ(R2)
000500' 006062  000046
                                ELSE
000504'                                                                 BR      50024$
000504' 000435
000506'                                                         50017$:
                                ;+
                                ; KT IS PRESENT, FIRST GO MAP APRS'
                                ;-
                                        CALL KTSET IN <R2>
000506'                                                                 MOV     R5,-(SP)
000506' 010546                                                          MOV     R2,-(R5)
000510' 010245                                                          JSR     PC,KTSET
000512' 004767  000000G                                                 MOV     (SP)+,R5
000516' 012605
                                ;+
                                ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
                                ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
                                ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
                                ; 1K STEPS
```

```
                                        LET @#KIPAR5 := #SS.1KP
000520'                                                         MOV     #SS.1KP,@#KIPAR5
(4) 000520' 012737  000040  172352
842
                                        LET @#SR0 := @#SR0 SET.BY #BIT00
000526'                                                         BIS     #BIT00,@#SR0
(6) 000526' 052737  000001  177572
844
                                        REPEAT
000534'                                                 50025$:
(3) 000534'
                                                LET @#KIPAR5 := @#KIPAR5 + #SS.1KP
000534'                                                         ADD     #SS.1KP,@#KIPAR5
(6) 000534' 062737  000040  172352
                                                CALL HRDADRCHK IN <#SS.120K>
000542'                                                         MOV     R5,-(SP)
(3) 000542' 010546                                              MOV     #SS.120K,-(R5)
(4) 000544' 012745  120000                                      JSR     PC,HRDADRCHK
(3) 000550' 004767  000000G                                     MOV     (SP)+,R5
(3) 000554' 012605
                                                IF.ERROR THEN
000556'                                                         BCC     50026$
(6) 000556' 103001
                                                        INLINE <BR 5$>
000560'                                                         BR 5$
(2) 000560' 000404
                                                ENDIF
000562'                                                 50026$:
(4) 000562'
                                        UNTIL @#KIPAR5 EQ #SS.IOP
000562'                                                         CMP     @#KIPAR5,#SS.IOP
(3) 000562' 023727  172352  177600                              BNE     50025$
(6) 000570' 001361
852
                                ;+
                                ; LOAD SYSTEM SIZE WORD WITH CONTENTS OF KIPAR5
                                ;-
856
                                INLINE <5$:>
000572'                                                         5$:
(2) 000572'
                                        LET DT.SSIZ(R2) := @#KIPAR5
000572'                                                         MOV     @#KIPAR5,DT.SSIZ
(4) 000572' 013762  172352  000046
                                ENDIF
000600'                                                 50024$:
(4) 000600'
860
```

```
                        .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT

                        ;+
                        ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
                        ;-

                        IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
000600'                                                                 BIT     #PARPRES,R1
(6)  000600' 032701  002000                                             BNE     50027$
(8)  000604' 001003                                                     BIT     #ECCMEM,R1
(6)  000606' 032701  000100                                             BEQ     50030$
(9)  000612' 001500
(6)  000614'                                                    50027$:

                        ;+
                        ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER
                        ; THIS WILL TURN CACHE OFF
                        ;-

                                IF #CAPRES SETIN R1 THEN
000614'                                                                 BIT     #CAPRES,R1
(6)  000614' 032701  000004                                             BEQ     50031$
(9)  000620' 001406
                                        LET KONTRL := #3
000622'                                                                 MOV     #3,KONTRL
(4)  000622' 012767  000003  000000G
                                        LET @CCNTRL := KONTRL
000630'                                                                 MOV     KONTRL,@CCNTRL
(4)  000630' 016777  000000G 000000G
                                ENDIF
000636'                                                         50031$:
(4)  000636'

                        ;+
                        ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
                        ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
                        ; OF MEMORY, FIRST DETERMINE IF KT IS PRESENT THEN SET
                        ; UP KT REGISTERS.
                        ;-

                                IF #KTPRES SETIN R1 THEN
000636'                                                                 BIT     #KTPRES,R1
(6)  000636' 032701  000400                                             BEQ     50032$
(9)  000642' 001426
                                        CALL KTSET
000644'                                                                 JSR     PC,KTSET
(3)  000644' 004767  000000G

                        ;+
                        ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
                        ; SET UP KIPAR5 TO MAP TO 0
                        ;-

                                        LET @#KIPDR6 := #0
000650'                                                                 CLR     @#KIPDR6
(4)  000650' 005037  172314
                                        LET @#KIPAR5 := #0
000654'                                                                 CLR     @#KIPAR5
(4)  000654' 005037  172352

                        ;+
                        ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
                        ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
                        ; SET UP R0 TO USE PAR5
                        ;-
```

```
                                        PUSH @#4                                        MOV     @#4,-(SP)
000660'                                 LET @#4 := #8$                                  MOV     #8$,@#4
(2) 000660' 013746  000004              PUSH @#250                                      MOV     @#250,-(SP)
000664'                                 LET @#250 := #7$                                MOV     #7$,@#250
(4) 000664' 012737  000760' 000004      LET @#SR0 := @#SR0 SET.BY #BIT00                BIS     #BIT00,@#SR0
000672'                                 LET R0 := #SS.120K                              MOV     #SS.120K,R0
(2) 000672' 013746  000250          ELSE                                               BR      50033$
000676'                          50032$:
(4) 000676' 012737  000740' 000250
000704'                         ;+
(6) 000704' 052737  000001  177572  ; IF KT NOT PRESENT THEN SET UP POINTER AND SAVE BUSERR VECTOR
000712'                         ;-
(4) 000712' 012700  120000
000716'                                 LET R0 := #0                                    CLR     R0
(4) 000716' 000406                      PUSH @#4                                        MOV     @#4,-(SP)
000720'                                 LET @#4 := #10$                                 MOV     #10$,@#4
(4) 000720' 005000                  ENDIF
000722'                          50033$:
(2) 000722' 013746  000004
000726'                         ;+
(4) 000726' 012737  001006' 000004  ; NOW WRITE GOOD PARITY
000734'                         ;-
(4) 000734'                     INLINE<6$:>                                         6$:
000734'                                 LET (R0)+ := (R0)                               MOV     (R0),(R0)+
(2) 000734'                             INLINE <BR 6$>                              BR 6$
000734'
(4) 000734' 011020
000736'                         ;+
(2) 000736' 000776              ; COME HERE BY WAY OF KT TRAP.
                                ; UPDATE PAR5 TO NEXT 4K BLOCK AND RESET R0 TO START
                                ; OF BLOCK. LOAD RETURN PC WITH 6$ AND RTI
                                ;-

                                INLINE <7$:>                                        7$:
000740'                                 LET @#KIPAR5 := @#KIPAR5 + #200                 ADD     #200,@#KIPAR5
(2) 000740'                             LET R0 := #SS.120K                          MOV     #SS.120K,R0
000740'                                 LET (SP) := #6$                             MOV     #6$,(SP)
(6) 000740' 062737  000200  172352      INLINE <RTI>                                RTI
000746'
(4) 000746' 012700  120000
000752'
(4) 000752' 012716  000734'
000756'
(2) 000756' 000002
```

```
                                ;+
                                ; COME HERE BY WAY OF ILLEGAL ADDRESS TRAP.
                                ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
                                ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
                                ;-

000760'                         INLINE <8$:>                                    8$:
(2)  000760'
000760'                                 LET @#SR0 := @#SR0 CLR.BY #BIT00        BIC     #BIT00,@#SR0
(6)  000760' 042737  000001  177572
000766'                                 LET (SP) := #9$                         MOV     #9$,(SP)
(4)  000766' 012716  000774'
000772'                                 INLINE <RTI>                            RTI
(2)  000772' 000002

000774'                         INLINE <9$:>                                    9$:
(2)  000774'
000774'                                 POP @#250                               MOV     (SP)+,@#250
(2)  000774' 012637  000250
001000'                                 POP @#4                                 MOV     (SP)+,@#4
(2)  001000' 012637  000004
001004'                                 INLINE <BR 11$>                         BR 11$
(2)  001004' 000403

                                ;+
                                ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
                                ; STACK AND RESTORE BUSERR
                                ;-

001006'                         INLINE <10$:>                                   10$:
(2)  001006'
001006'                                 INLINE <CMP (SP)+,(SP)+>                CMP (SP)+,(SP)+
(2)  001006' 022626
001010'                                 POP @#4                                 MOV     (SP)+,@#4
(2)  001010' 012637  000004

001014'                         ENDIF                                   50030$:
(4)  001014'


                                ;+
                                ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD0
                                ;-

001014'                         INLINE <11$:>                                   11$:
(2)  001014'
001014'                         IF @#42 NE #0 THEN                              TST     @#42
(6)  001014' 005737  000042                                                     BEQ     50034$
(9)  001020' 001402
001022'                                 LET R1 := R1 SET.BY #AUTO               BIS     #AUTO,R1
(6)  001022' 052701  000010
001026'                         ENDIF                                   50034$:
(4)  001026'

```

```
                                ;+
                                ; CLEAN UP R1 BEFORE LOADING CONFIGURATION WORD 0
                                ; LOAD UP CONFIGURATION WORD 0
                                ;-

001026'                         LET R1 := R1 CLR.BY #JACK                       BIC     #JACK,R1
(6) 001026' 042701  035060
001032'                         LET DT.CF0(R2) :=  DT.CF0(R2) SET.BY R1         BIS     R1,DT.CF0(R2)
(6) 001032' 050162  000014
```

```
                        .SBTTL KERNEL - SORT MODULE LIST


                        ;+
                        ; COUNT THE MODULES
                        ;-

                                LET R0 := #0                                    CLR     R0
001036'                                                                         
(4) 001036' 005000
                                LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
001040'
(4) 001040' 016201  000032
                                WHILE (R1) NE #ENDLST DO                50035$:
001044'                                                                         CMP     (R1),#ENDLST
(4) 001044'                                                                     BEQ     50036$
(6) 001044' 021127  000000
(9) 001050' 001404
                                        LET R0 := R0 + #1                       INC     R0
001052'
(6) 001052' 005200
                                        LET R1 := R1 + #2                       ADD     #2,R1
001054'
(6) 001054' 062701  000002
                                ENDDO                                           BR      50035$
001060'
(4) 001060' 000771                                                      50036$:
(3) 001062'

                        ;+
                        ; START THE SORT OPERATION, AND SORT "BKMODS" TO BOTTOM OF MODQ
                        ; LIST, DEVELOPING A "BKMOD QUEUE".
                        ;-

                                LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
001062'
(4) 001062' 016201  000032

                        ;+
                        ; WORK THRU THE LIST UNTIL THE END IS REACHED
                        ;-

                                WHILE R0 GT #0 DO                       50037$:
001066'                                                                         TST     R0
(4) 001066'                                                                     BLE     50040$
(6) 001066' 005700
(9) 001070' 003432
                                        REPEAT                          50041$:
001072'
(3) 001072'

                        ;+
                        ; IS THE CURRENT ENTRY A BKMOD MODULE?
                        ;-

                                        LET R4 := (R1)                          MOV     (R1),R4
001072'
(4) 001072' 011104
                                        IF #BKMOD SETIN STAT(R4) THEN           BIT     #BKMOD,STAT(R4)
001074'                                                                         BEQ     50042$
(6) 001074' 032764  000020  000026
(9) 001102' 001414

                        ;+
                        ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
                        ;-
```

```
1026  001104'                                                    LET R3 := 2(R1)
 (4)  001104' 016103  000002                                                        MOV     2(R1),R3
1027
1028                                    ;+
1029                                    ; MAKE SURE ITS NOT THE END OF THE LIST
1030                                    ;-
1031
1032  001110'                                                    IF R3 NE #ENDLST THEN
 (6)  001110' 020327  000000                                                        CMP     R3,#ENDLST
 (9)  001114' 001407                                                               BEQ     50043$
1033
1034                                    ;+
1035                                    ; IF ITS NOT A BKMOD, SWAP THEM
1036                                    ;-
1037
1038  001116'                                                          IF #BKMOD NOTSETIN STAT(R3) THEN
 (6)  001116' 032763  000020  000026                                                BIT     #BKMOD,STAT(R3)
 (9)  001124' 001003                                                               BNE     50044$
1039  001126'                                                               LET 2(R1) := (R1)
 (4)  001126' 011161  000002                                                       MOV     (R1),2(R1)
1040  001132'                                                               LET (R1) := R3
 (4)  001132' 010311                                                               MOV     R3,(R1)
1041  001134'                                                          ENDIF
 (4)  001134'                                                                                   50044$:
1042  001134'                                                    ENDIF
 (4)  001134'                                                                                   50043$:
1043  001134'                                              ENDIF
 (4)  001134'                                                                                   50042$:
1044
1045                                    ;+
1046                                    ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
1047                                    ;-
1048
1049  001134'                                                         LET R1 := R1 + #2
 (6)  001134' 062701  000002                                                       ADD     #2,R1
1050  001140'                                                   UNTIL (R1) EQ #ENDLST
 (3)  001140' 021127  000000                                                       CMP     (R1),#ENDLST
 (6)  001144' 001352                                                               BNE     50041$
1051
1052                                    ;+
1053                                    ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
1054                                    ;-
1055
1056  001146'                                                   LET R0 := R0 - #1
 (6)  001146' 005300                                                               DEC     R0
1057  001150'                                                   LET R1 := DT.MLST(R2)
 (4)  001150' 016201  000032                                                       MOV     DT.MLST(R2),R1
1058  001154'                                             ENDDO
 (4)  001154' 000744                                                               BR      50037$
 (3)  001156'                                                                                   50040$:
1059
1060                                    ;+
1061                                    ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
1062                                    ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
1063                                    ; A BACKGROUND LIST)  INTO THE DATA TABLE
1064                                    ;-
```

```
                        LET R1 := DT.MLST(R2)
001156'                                                         MOV     DT.MLST(R2),R1
(4) 001156' 016201  000032

                    ;+
                    ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
                    ;-

                        LET DT.BLST(R2) := #0
001162'                                                         CLR     DT.BLST(R2)
(4) 001162' 005062  000034
                        LET R3 := #-1
001166'                                                         MOV     #-1,R3
(4) 001166' 012703  177777
                        WHILE (R1) NE #ENDLST AND R3 NE #0 DO
001172'                                                 50045$:
(4) 001172'                                                     CMP     (R1),#ENDLST
(6) 001172' 021127  000000                                      BEQ     50046$
(9) 001176' 001420                                              TST     R3
(6) 001200' 005703                                              BEQ     50046$
(9) 001202' 001416
                            LET R4 := (R1)
001204'                                                         MOV     (R1),R4
(4) 001204' 011104
                            IF #BKMOD SETIN STAT(R4) THEN
001206'                                                         BIT     #BKMOD,STAT(R4)
(6) 001206' 032764  000020  000026                              BEQ     50047$
(9) 001214' 001406

                    ;+
                    ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
                    ;-

                                LET DT.BLST(R2) := R1
001216'                                                         MOV     R1,DT.BLST(R2)
(4) 001216' 010162  000034
                                LET R1 := #ENDLST
001222'                                                         MOV     #ENDLST,R1
(4) 001222' 012701  000000
                                LET R3 := #0
001226'                                                         CLR     R3
(4) 001226' 005003
                            ELSE
001230'                                                         BR      50050$
(4) 001230' 000402                                      50047$:
(3) 001232'

                    ;+
                    ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
                    ;-

                                LET R1 := R1 + #2
001232'                                                         ADD     #2,R1
(6) 001232' 062701  000002
                            ENDIF
001236'                                                 50050$:
(4) 001236'
                        ENDDO
001236'                                                         BR      50045$
(4) 001236' 000755                                      50046$:
(3) 001240'

                    ;+
                    ; NOW RETURN TO THE CALLER
                    ;-

                        CALL RESREG
001240'                                                         JSR     PC,RESREG
(3) 001240' 004767  000000G
```

```
                        ENDRTN
                                                                50000$:
                                                                50001$:
                                                                        RTS     PC

                ;+
                ; LABEL TYPE QUEUE HIGH LIMIT
                ;-

                INLINE <OV.TQEND:>
                                                                        OV.TQEND:




                ;************************************
                ; NOTE:  THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
                ;   (THAT IS, AFTER SIZPOL AND TQ.END LABELS)

                ;++++++++++++++++++++++++++++++++++++
                ;    OVERLAY REGIONS EQUATES
                ;++++++++++++++++++++++++++++++++++++
000000'         OV.KBBUF = SIZPOL                       ;KEYBOARD BUFFER STARTING ADDRESS
000122'         OV.CQ= OV.KBBUF + ^D<82>                ;CONTROL QUEUE STARTING ADDRESS
000364'         OV.TQ = OV.CQ + ^D<162>                 ;TYPE QUEUE STARTING ADDRESS
000120'         OV.HIKB = OV.CQ - 2                     ;KEYBOARD BUFFER HIGH LIMIT
000122          OV.KBSIZ = OV.CQ - OV.KBBUF             ;KEYBOARD BUFFER SIZE
000362'         OV.HICQ = OV.TQ - 2                     ;CONTROL QUEUE HIGH LIMIT
000050          OV.CQSIZ = <OV.TQ - OV.CQ>/4            ;CONTROL QUEUE SIZE
000662          OV.A = OV.TQEND - OV.TQ                 ;GET THE TQ SIZE
000002          OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
001244'         OV.HITQ = OV.TQEND - OV.REM             ;TYPE QUEUE HIGH LIMIT
000066          OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10      ;TYPE QUEUE SIZE

000001                  .END
```

```
ACSR   = 000102
ACTBIT= 004000
ADDR22= 001000
ADR    = 000006
APTFER= 000004
APTPRE= 000200
ASB    = 000106
ASSEMB= 000010
ASTAT  = 000104
AUTO   = 000010
AUTOST= 020000
AWAS   = 000110
BIT0   = 000001
BIT00  = 000001
BIT01  = 000002
BIT02  = 000004
BIT03  = 000010
BIT04  = 000020
BIT05  = 000040
BIT06  = 000100
BIT07  = 000200
BIT08  = 000400
BIT09  = 001000
BIT1   = 000002
BIT10  = 002000
BIT11  = 004000
BIT12  = 010000
BIT13  = 020000
BIT14  = 040000
BIT15  = 100000
BIT2   = 000004
BIT3   = 000010
BIT4   = 000020
BIT5   = 000040
BIT6   = 000100
BIT7   = 000200
BIT8   = 000400
BIT9   = 001000
BKDEF  = 000002
BKMOD  = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM  = 100000
CLKPRE= 000001
CONFIG= 000056
CPUCPE= ****** G
CQOVF  = 000001
CR     = 000015
CSRA   = 000100
CSRC   = 000102
CTRLC  = 000003
CTRLO  = 000017
CTRLU  = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP  = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC  = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP  = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1  = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
HTKT   = ****** G
HTPAER= ****** G
ICONT  = 000036
ICOUNT= 000040
IDNUM  = 000122
IE     = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT   = 000030
INTR   = 000120
IOMOD  = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK   = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KTERRO= 000040
KTPRES= 000400
KTSET  = ****** G
KTSTAT= 000020
KTXTND= 040000
LF     = 000012
LPSTAT= 000001
MAPSTA= 000200
MED    = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL   = 000000
OV.A   = 000662
OV.CQ  = 000122RG
OV.CQS= 000050 G
OV.HIC= 000362RG
OV.HIK= 000120RG
OV.HIT= 001244RG
OV.KBB= 000000RG
OV.KBS= 000122 G
OV.REM= 000002
OV.TQ  = 000364RG
OV.TQE  001246R
OV.TQS= 000066 G
OWEN   = 024020
PAERR  = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60  = 004000
PDP70  = 010000
PRIO   = 000000
PRI1   = 000040
PRI4   = 000200
PRI5   = 000240
PRI6   = 000300
PRI7   = 000340
PR0    = 000000
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSW    = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1   = 000056
RES2   = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT  = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM    = 075464
SAVREG= ****** G
SBADR  = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SIZPOL  000000RG
SOFCNT= 000042
SOFPAS= 000046
SPACE  = 000040
SPOINT= 000032
SPVALU= 002200
```

```
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
SS.IOP= 177600
SS.PCL= 172136
SS.PCS= 172100
SS.RFS= 160000
SS.0KC= 004000
SS.1KP= 000040
SS.120= 120000
SS.28 = 000034
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR   = 000116
WDTO   = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG  = 000005
XOFF   = 000023
XON    = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO  = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF  = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO  = 000403
$F$OR  = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000120
$NSK2 = 000110
$NSK3 = 000110
$NSK4 = 000110
$NSK5 = 000110
$SAVLE= 177777
$SSK0 = 050046
$TAGLE= 177777
$TAGNU= 050051
$TEMP  = 000300
$TSK0  = 050045
$TSK1  = 050046
$TSK2  = 050050
$TSK3  = 050042
$TSK4  = 050043
$TSK5  = 050044
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST  = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC  = 001214R
$$LOCN= 000000
$$REG  = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC  = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO   = 000000
$$$TAG= 050000
.      = 001246R

. ABS.  000000    000
        001246    001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SIZPLC,DSKZ:SIZPLC=SPMAC/ML,EQUATE,SIZPLC
 RUN-TIME: 28 19 .4 SECONDS
 RUN-TIME RATIO: 118/48=2.4
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SIZPLD SIZE AND POLL SYSTEM - KT OPTIONS AND 11/60 SUPPORT
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SIZPLD
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
                        ;       OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
                        ;       WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
                        ;       A BACKGROUND MODULE LIST (QUEUE)
                        ;       NOTE: IT WILL NOT SIZE FOR 22-BIT ADDRESSING BUT WILL SIZE FOR 11/60 CPU
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CONFIGURATION WORD 0
                        ;       EXERCISER SIZE
                        ;       SYSTEM SIZE
                        ;       PARITY TABLE ADDRESS
                        ;       MODULE LIST
                        ;       BACKGROUND LIST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       PARITY TABLE ADDRESS
                        ;       SYSTEM SIZE
                        ;       EXERCISE SIZE
                        ;       CONFIGURATION WORD 0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       KTSET - SET UP KT REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SIZPOL IN <A>
                        ;                   A - DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                                .SBTTL COMMON DEFINITIONS AND REFERENCES

                                .MCALL STRUCT
000000'                         STRUCT
(1) 000000'                     .PRINT  ;SPMAC: VERSION 1.1
            000001              $LSTIN = 1
            000001              $LSTTAG = 1


                                ;****************
                                ; GLOBAL REFERENCES
                                ;
                                .GLOBL  HTKT
                                .GLOBL  HTPAER
                                .GLOBL  SAVREG                  ;SARE REGISTERS
                                .GLOBL  RESREG                  ;RESTORE REGISTERS
                                .GLOBL  KTSET                   ;SET UP KT
                                .GLOBL  HRDADRCHK               ;ILLEGAL ADDRESS CHECK
                                .GLOBL  CPUCPE                  ;11/70 CPU ERROR REGISTER
                                .GLOBL  CCNTRL                  ;CACHE CONTROL REGISTER
                                .GLOBL  KONTRL                  ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER

                                ;****************
                                ; LOCAL EQUATES
                                ;
            172100              SS.PCSR = 172100                ;START OF PARITY CSRS
            172136              SS.PCLS = 172136                ;LAST PARITY CSR
            004000              SS.0KC  = 4000                  ;ONE K CHUNK
            000034              SS.28   = ^D28                  ;28 DECIMAL
            000040              SS.1KP  = 40                    ;1K IN PAR FORMAT
            120000              SS.120K = 120000                ;FIRST ADDRESS IN PAR5 MAP
            177600              SS.IOP  = 177600                ;I/O PAGE
            160000              SS.RFS  = 160000                ;REVERSED FOR DIAGNOSTIC USE IN I/O PAGE

                                ;****************
                                ; REFERENCED BY OTHER MODULES
                                ;
                                .GLOBL  SIZPOL                  ;MODULE ENTRY POINT



                                ;++++++++++++++++++++++++++++++
                                ;    OVERLAY REGION GLOBALS
                                ;++++++++++++++++++++++++++++++
                                .GLOBL  OV.KBBUF
                                .GLOBL  OV.HIKB
                                .GLOBL  OV.KBSIZ
                                .GLOBL  OV.CQ
                                .GLOBL  OV.HICQ
                                .GLOBL  OV.CQSIZ
                                .GLOBL  OV.TQ
                                .GLOBL  OV.HITQ
                                .GLOBL  OV.TQSIZ
```

```
                                .SBTTL KERNEL - DETERMINE PROCESSOR TYPE


                                ROUTINE SIZPOL <DTABLE>                                 SIZPOL:
                                ;+
                                ; STUFF PARITY AND KT VECTORS
                                ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
                                ; R1
                                ;-


                                LET @#114 := #HTPAER                            MOV     #HTPAER,@#114
(4)  000000' 012737  000000G 000114
                                LET @#116 := #PRI7                              MOV     #PRI7,@#116
(4)  000006' 012737  000340  000116

                                LET @#250 := #HTKT                              MOV     #HTKT,@#250
(4)  000014' 012737  000000G 000250
                                LET @#252 := PRI7                               MOV     PRI7,@#252
(4)  000022' 016737  000340  000252

                                CALL SAVREG                                     JSR     PC,SAVREG
(3)  000030' 004767  000000G
                                LET R2 := DTABLE(R5)                            MOV     DTABLE(R5),R2
(4)  000034' 016502  000000
                                LET R1 := #0                                    CLR     R1
(4)  000040' 005001


                                ;+
                                ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
                                ;-

                                CALL HRDADRCHK IN <CPUCPE>                      MOV     R5,-(SP)
(3)  000042' 010546                                                             MOV     CPUCPE,-(R5)
(4)  000044' 016745  000000G                                                    JSR     PC,HRDADRCHK
(3)  000050' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000054' 012605

                                IF.NO.ERROR THEN                                BCS     50002$
(6)  000056' 103422
                                ;+
                                ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
                                ; 11/60
                                ;-

                                        PUSH @#10                               MOV     @#10,-(SP)
(2)  000060' 013746  000010
                                        LET @#10 := #1$                         MOV     #1$,@#10
(4)  000064' 012737  000104' 000010
                                        INLINE <MED>                            MED
(2)  000072' 076600
                                        INLINE <RDSERV>                         RDSERV
(2)  000074' 000101
```

```
                                        LET R1 := R1 SET.BY #PDP60
(6)  000076' 052701  004000                                                 BIS     #PDP60,R1
                                        INLINE <BR 3$>
(2)  000102' 000405                                                         BR 3$

                        INLINE <1$:>
(2)  000104'                                                                1$:

                        ;+
                        ; SET RETURN TO 2$ TO RESTORE PSW
                        ;-

                                        LET (SP) := #2$
(4)  000104' 012716  000112'                                                MOV     #2$,(SP)
                                        INLINE <RTI>
(2)  000110' 000002                                                         RTI

                        ;+
                        ; MUST BE AN 11/70
                        ;-

                        INLINE <2$:>
(2)  000112'                                                                2$:
                                        LET R1 := R1 SET.BY #PDP70
(6)  000112' 052701  010000                                                 BIS     #PDP70,R1

                        ;+
                        ; RESTORE RESERVED INSTRUCTION TRAP
                        ;-

                        INLINE <3$:>
(2)  000116'                                                                3$:
                                        POP @#10
(2)  000116' 012637  000010                                                 MOV     (SP)+,@#10
                        ELSE
(4)  000122' 000411                                                         BR      50003$
(3)  000124'                                                        50002$:

                        ;+
                        ; CHECK TO SEE IF LSI11
                        ;-

                                        CALL HRDADRCHK IN <#PSW>
(3)  000124' 010546                                                         MOV     R5,-(SP)
(4)  000126' 012745  177776                                                 MOV     #PSW,-(R5)
(3)  000132' 004767  000000G                                                JSR     PC,HRDADRCHK
(3)  000136' 012605                                                         MOV     (SP)+,R5
                                        IF.ERROR THEN
(6)  000140' 103002                                                         BCC     50004$
                                                LET R1 := R1 SET.BY #PDPLSI
(6)  000142' 052701  020000                                                 BIS     #PDPLSI,R1

                                        ENDIF
(4)  000146'                                                        50004$:
                        ENDIF
(4)  000146'                                                        50003$:
```

694

```
                        .SBTTL KERNEL - DETERMINE CPU OPTIONS

                        ;+
                        ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
                        ; SEE IF EXTENDED KT" IS PRESENT
                        ;-

000146'                 CALL HRDADRCHK IN <#SR0>
                                                                MOV     R5,-(SP)
(3)  000146' 010546                                             MOV     #SR0,-(R5)
(4)  000150' 012745  177572                                     JSR     PC,HRDADRCHK
(3)  000154' 004767  000000G                                    MOV     (SP)+,R5
(3)  000160' 012605
000162'                 IF.NO.ERROR THEN
                                                                BCS     50005$
(6)  000162' 103413
000164'                         LET R1 := R1 SET.BY #KTPRES
                                                                BIS     #KTPRES,R1
(6)  000164' 052701  000400
000170'                         CALL HRDADRCHK IN <#SR3>
                                                                MOV     R5,-(SP)
(3)  000170' 010546                                             MOV     #SR3,-(R5)
(4)  000172' 012745  172516                                     JSR     PC,HRDADRCHK
(3)  000176' 004767  000000G                                    MOV     (SP)+,R5
(3)  000202' 012605
000204'                         IF.NO.ERROR THEN
                                                                BCS     50006$
(6)  000204' 103402
000206'                                 LET R1 := R1 SET.BY #KTXTND
                                                                BIS     #KTXTND,R1
(6)  000206' 052701  040000

                                ENDIF
000212'                                                         50006$:
(4)  000212'
000212'                 ENDIF                                   50005$:
(4)  000212'

                        ;+
                        ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
                        ; CACHE CONTROL REGISTER
                        ;-


000212'                 CALL HRDADRCHK IN <CCNTRL>
                                                                MOV     R5,-(SP)
(3)  000212' 010546                                             MOV     CCNTRL,-(R5)
(4)  000214' 016745  000000G                                    JSR     PC,HRDADRCHK
(3)  000220' 004767  000000G                                    MOV     (SP)+,R5
(3)  000224' 012605
000226'                 IF.NO.ERROR THEN
                                                                BCS     50007$
(6)  000226' 103402
000230'                         LET R1 := R1 SET.BY #CAPRES
                                                                BIS     #CAPRES,R1
(6)  000230' 052701  000004
000234'                 ENDIF                                   50007$:
(4)  000234'
```

```
                        .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT

                        ;+
                        ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
                        ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
                        ; IN PARITY TABLE.
                        ;-

000234'                 LET R0 := DT.PTA(R2)
(4) 000234' 016200 000064                                       MOV     DT.PTA(R2),R0

000240'                 LET R3 := #SS.PCSR
(4) 000240' 012703 172100                                       MOV     #SS.PCSR,R3
000244'                 WHILE R3 LE #SS.PCLS DO
(4) 000244'                                                 50010$:
(6) 000244' 020327 172136                                       CMP     R3,#SS.PCLS
(9) 000250' 003035                                              BGT     50011$
000252'                         CALL HRDADRCHK IN <R3>
(3) 000252' 010546                                              MOV     R5,-(SP)
(4) 000254' 010345                                              MOV     R3,-(R5)
(3) 000256' 004767 000000G                                      JSR     PC,HRDADRCHK
(3) 000262' 012605                                              MOV     (SP)+,R5
000264'                         IF.NO.ERROR THEN
(6) 000264' 103414                                              BCS     50012$
000266'                                 LET (R0)+ := R3
(4) 000266' 010320                                              MOV     R3,(R0)+

                        ;+
                        ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
                        ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
                        ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
                        ;-
000270'                                 LET (R3) := (R3) SET.BY #BIT01
(6) 000270' 052713 000002                                       BIS     #BIT01,(R3)
000274'                                 IF #BIT01 SETIN (R3) THEN
(6) 000274' 032713 000002                                       BIT     #BIT01,(R3)
(9) 000300' 001403                                              BEQ     50013$
000302'                                         LET R1 := R1 SET.BY #ECCMEM
(6) 000302' 052701 000100                                       BIS     #ECCMEM,R1
000306'                                 ELSE
(4) 000306' 000402                                              BR      50014$
000310'                                                     50013$:
000310'                                         LET R1 := R1 SET.BY #PARPRES
(6) 000310' 052701 002000                                       BIS     #PARPRES,R1
000314'                                 ENDIF
(4) 000314'                                                 50014$:
000314'                                 LET (R3) := #0
(4) 000314' 005013                                              CLR     (R3)
000316'                         ENDIF
(4) 000316'                                                 50012$:

                        ;+
                        ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
                        ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
                        ; PARITY CSR + 2
                        ;-
```

```
                        IF #PDP70 SETIN R1 THEN

                                LET R1 := R1 SET.BY #PARPRES

                                LET R3 := R3 + #4

                        ELSE

                                LET R3 := R3 + #2

                        ENDIF

                ENDDO
```

```
762  000316'
(6)  000316' 032701  010000          BIT     #PDP70,R1
(9)  000322' 001405                  BEQ     50015$
763  000324'
(6)  000324' 052701  002000          BIS     #PARPRES,R1
764  000330'
(6)  000330' 062703  000004          ADD     #4,R3
765  000334'
(4)  000334' 000402                  BR      50016$
(3)  000336'                 50015$:
766  000336'
(6)  000336' 062703  000002          ADD     #2,R3
767  000342'
(4)  000342'                 50016$:
768  000342'                         BR      50010$
(4)  000342' 000740
(3)  000344'                 50011$:
769
770
```

```
                                .SBTTL KERNEL - DETERMINE SYSTEM SIZE

                                ;+
                                ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
                                ; AND LOAD IT INTO EXERCISER SIZE WORD
                                ;-

000344'                         LET DT.ESIZ(R2) := 0
(4) 000344' 016762 000000 000044                                        MOV     0,DT.ESIZ(R2)

                                ;+
                                ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
                                ; THAN 28K.
                                ;-

000352'                         IF #KTPRES NOTSETIN R1 THEN
(6) 000352' 032701 000400                                               BIT     #KTPRES,R1
(9) 000356' 001053                                                      BNE     50017$

                                ;+
                                ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
                                ;-

000360'                                 LET R0 := #0
(4) 000360' 005000                                                      CLR     R0
000362'                                 LET R3 := #0
(4) 000362' 005003                                                      CLR     R3

                                ;+
                                ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
                                ;-

000364'                                 REPEAT
(3) 000364'                                                     50020$:
000364'                                         LET R0 := R0 + #1
(6) 000364' 005200                                                      INC     R0
000366'                                         LET R3 := R3 + #SS.0KC
(6) 000366' 062703 004000                                               ADD     #SS.0KC,R3
000372'                                         CALL HRDADRCHK IN <R3>
(3) 000372' 010546                                                      MOV     R5,-(SP)
(4) 000374' 010345                                                      MOV     R3,-(R5)
(3) 000376' 004767 000000G                                              JSR     PC,HRDADRCHK
(3) 000402' 012605                                                      MOV     (SP)+,R5
000404'                                         IF.ERROR THEN
(6) 000404' 103001                                                      BCC     50021$
000406'                                                 INLINE <BR 4$>
(2) 000406' 000403                                                      BR 4$
000410'                                         ENDIF
(4) 000410'                                                     50021$:
000410'                                 UNTIL R0 EQ #SS.28
(3) 000410' 020027 000034                                               CMP     R0,#SS.28
(6) 000414' 001363                                                      BNE     50020$

                                ;+
                                ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
```

```
                        ;-
                        INLINE <4$:>
                                                                        4$:
                                IF R0 EQ #SS.28 THEN
                                                                        CMP     R0,#SS.28
                                                                        BNE     50022$
                                        CALL HRDADRCHK IN <#SS.RFS>
                                                                        MOV     R5,-(SP)
                                                                        MOV     #SS.RFS,-(R5)
                                                                        JSR     PC,HRDADRCHK
                                                                        MOV     (SP)+,R5
                                        IF.NO.ERROR THEN
                                                                        BCS     50023$
                                                LET R3 := R3 + #10000
                                                                        ADD     #10000,R3
                                        ENDIF
                                                                    50023$:
                                ENDIF
                                                                    50022$:

                        ;+
                        ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
                        ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
                        ; SIZE WORD.
                        ;-

                                LET CARRY := #0
                                                                        CLC
                                LET DT.SSIZ(R2) := R3 ROTATE -6
                                                                        MOV     R3,DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                        ELSE
                                                                        BR      50024$
                                                                    50017$:

                        ;+
                        ; KT IS PRESENT, FIRST GO MAP PAR'S AND PDR'S
                        ;-
                                CALL KTSET IN <R2>
                                                                        MOV     R5,-(SP)
                                                                        MOV     R2,-(R5)
                                                                        JSR     PC,KTSET
                                                                        MOV     (SP)+,R5

                        ;+
                        ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
                        ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
                        ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
                        ; 1K STEPS
```

```
  842  000520'                                 LET @#KIPAR5 := #SS.1KP
  (4)  000520' 012737  000040  172352                                                  MOV     #SS.1KP,@#KIPAR5
  843
  844  000526'                                 LET @#SR0 := @#SR0 SET.BY #BIT00
  (6)  000526' 052737  000001  177572                                                  BIS     #BIT00,@#SR0
  845
  846  000534'                                 REPEAT
  (3)  000534'                                                                 50025$:
  847  000534'                                         LET @#KIPAR5 := @#KIPAR5 + #SS.1KP
  (6)  000534' 062737  000040  172352                                                  ADD     #SS.1KP,@#KIPAR5
  848  000542'                                         CALL HRDADRCHK IN <#SS.120K>
  (3)  000542' 010546                                                                  MOV     R5,-(SP)
  (4)  000544' 012745  120000                                                          MOV     #SS.120K,-(R5)
  (3)  000550' 004767  000000G                                                        JSR     PC,HRDADRCHK
  (3)  000554' 012605                                                                  MOV     (SP)+,R5
  849  000556'                                         IF.ERROR THEN
  (6)  000556' 103001                                                                  BCC     50026$
  850  000560'                                                 INLINE <BR 5$>
  (2)  000560' 000404                                                                  BR 5$
  851  000562'                                         ENDIF
  (4)  000562'                                                                 50026$:
  852  000562'                                 UNTIL @#KIPAR5 EQ #SS.IOP
  (3)  000562' 023727  172352  177600                                                  CMP     @#KIPAR5,#SS.IOP
  (6)  000570' 001361                                                                  BNE     50025$
  853
  854                                  ;+
  855                                  ; LOAD SYSTEM SIZE WORD WITH CONTENTS OF KIPAR5
  856                                  ;-
  857
  858  000572'                         INLINE <5$:>
  (2)  000572'                                                                         5$:
  859  000572'                                 LET DT.SSIZ(R2) := @#KIPAR5
  (4)  000572' 013762  172352  000046                                                  MOV     @#KIPAR5,DT.SSIZ
  860  000600'                         ENDIF
  (4)  000600'                                                                 50024$:
  861
```

```
                        .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT

                        ;+
                        ; DETERMINE IF PARITY OP ECC MEMORY IS PRESENT.
                        ;-

                        IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
000600'                                                                 BIT     #PARPRES,R1
(6) 000600' 032701  002000                                              BNE     50027$
(8) 000604' 001003                                                      BIT     #ECCMEM,R1
(6) 000606' 032701  000100                                              BEQ     50030$
(9) 000612' 001500                                                  50027$:
(6) 000614'

                        ;+
                        ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER
                        ; THIS WILL TURN CACHE OFF
                        ;-

                                IF #CAPRES SETIN R1 THEN
000614'                                                                 BIT     #CAPRES,R1
(6) 000614' 032701  000004                                              BEQ     50031$
(9) 000620' 001406
                                        LET KONTRL := #3
000622'                                                                 MOV     #3,KONTRL
(4) 000622' 012767  000003  000000G
                                        LET @CCNTRL := KONTRL
000630'                                                                 MOV     KONTRL,@CCNTRL
(4) 000630' 016777  000000G 000000G
                                ENDIF
000636'                                                             50031$:
(4) 000636'

                        ;+
                        ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
                        ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
                        ; OF MEMORY, FIRST KT PRESENT THEN SET
                        ; UP KT REGISTERS.
                        ;-

                                IF #KTPRES SETIN R1 THEN
000636'                                                                 BIT     #KTPRES,R1
(6) 000636' 032701  000400                                              BEQ     50032$
(9) 000642' 001426
                                        CALL KTSET
000644'                                                                 JSR     PC,KTSET
(3) 000644' 004767  000000G

                        ;+
                        ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
                        ; SET UP KIPAR5 TO MAP TO 0
                        ;-

                                        LET @#KIPDR6 := #0
000650'                                                                 CLR     @#KIPDR6
(4) 000650' 005037  172314
                                        LET @#KIPAR5 := #0
000654'                                                                 CLR     @#KIPAR5
(4) 000654' 005037  172352

                        ;+
                        ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
                        ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
                        ; SET UP R0 TO USE PAR5
                        ;-
```

```
                                        PUSH @#4                                        MOV     @#4,-(SP)
000660'                                 LET @#4 := #8$                                  MOV     #8$,@#4
(2) 000660' 013746  000004              PUSH @#250                                      MOV     @#250,-(SP)
000664'                                 LET @#250 := #7$                                MOV     #7$,@#250
(4) 000664' 012737  000760' 000004      LET @#SR0 := @#SR0 SET.BY #BIT00                BIS     #BIT00,@#SR0
000672'                                 LET R0 := #SS.120K                              MOV     #SS.120K,R0
(2) 000672' 013746  000250
000676'                                 ELSE                                            BR      50033$
(4) 000676' 012737  000740' 000250  50032$:
000704'
(6) 000704' 052737  000001  177572  ;+
000712'                             ; IF NOT KT THEN SET UP POINTER AND SAVE BUSERR VECTOR
(4) 000712' 012700  120000          ;-
000716'                                 LET R0 := #0                                    CLR     R0
(4) 000716' 000406                      PUSH @#4                                        MOV     @#4,-(SP)
(3) 000720'                             LET @#4 := #10$                                 MOV     #10$,@#4
000720'                                 ENDIF
(4) 000720' 005000                  50033$:
000722'
(2) 000722' 013746  000004          ;+
000726'                             ; NOW WRITE GOOD PARITY
(4) 000726' 012737  001006' 000004  ;-
000734'                             INLINE<6$:>                                         6$:
(4) 000734'                             LET (R0)+ := (R0)                               MOV     (R0),(R0)+
000734'                                 INLINE <BR 6$>                                  BR 6$
(4) 000734' 011020
000736'                             ;+
(2) 000736' 000776                  ; UPDATE PAR5 TO NEXT 4K BLOCK AND RESET R0 TO START
000740'                             ; OF BLOCK. LOAD RETURN PC WITH 6$ AND RTI
(2) 000740'                         ;-
000740'                             INLINE <7$:>                                        7$:
(6) 000740' 062737  000200  172352      LET @#KIPAR5 := @#KIPAR5 + #200                 ADD     #200,@#KIPAR5
000746'                                 LET R0 := #SS.120K                              MOV     #SS.120K,R0
(4) 000746' 012700  120000              LET (SP) := #6$                                 MOV     #6$,(SP)
000752'                                 INLINE <RTI>                                    RTI
(4) 000752' 012716  000734'
000756'
(2) 000756' 000002
```

```
                                ;+
                                ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
                                ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
                                ;-
000760'                         INLINE <8$:>                                    8$:
000760'
000760'                                 LET @#SR0 := @#SR0 CLR.BY #BIT00        BIC     #BIT00,@#SR0
000760' 042737  000001  177572
                                
000766'                                 LET (SP) := #9$                         MOV     #9$,(SP)
000766' 012716  000774'
                                        INLINE <RTI>                            RTI
000772'
000772' 000002
                                
                                INLINE <9$:>                                    9$:
000774'
000774'                                 POP @#250                               MOV     (SP)+,@#250
000774' 012637  000250
                                        POP @#4                                 MOV     (SP)+,@#4
001000'
001000' 012637  000004
                                        INLINE <BR 11$>                         BR 11$
001004'
001004' 000403
                                
                                ;+
                                ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
                                ; STACK AND RESTORE BUSERR
                                ;-
                                
                                INLINE <10$:>                                   10$:
001006'
001006'                                 INLINE <CMP (SP)+,(SP)+>                CMP (SP)+,(SP)+
001006'
001006' 022626
                                        POP @#4                                 MOV     (SP)+,@#4
001010'
001010' 012637  000004
                                
                                ENDIF                                   50030$:
001014'
001014'
                                
                                ;+
                                ; IF SCRIPTING, SET AUTO BIT IN CONFIGURATION WORD0
                                ;-
                                
                                INLINE <11$:>                                   11$:
001014'
001014'                         IF @#42 NE #0 THEN                              TST     @#42
001014'                                                                         BEQ     50034$
001014' 005737  000042
001020' 001402
                                        LET R1 := R1 SET.BY #AUTO               BIS     #AUTO,R1
001022'
001022' 052701  000010
                                ENDIF                                   50034$:
001026'
001026'
                                
                                ;+
```

```
                                ; CLEAN UP R1 BEFORE LOADING CONFIGURATION WORD 0
                                ; LOAD UP CONFIGURATION WORD 0
                                ;-

001026'                         LET R1 := R1 CLR.BY #RICHARD                    BIC     #RICHARD,R1
(6)  001026' 042701  031060
001032'                         LET DT.CF0(R2) := DT.CF0(R2) SET.BY R1          BIS     R1,DT.CF0(R2)
(6)  001032' 050162  000014
```

```
                                .SBTTL KERNEL - SORT MODULE LIST


                                ;+
                                ; COUNT THE MODULES
                                ;-

001036'                                 LET R0 := #0                                    CLR     R0
(4)  001036' 005000
001040'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4)  001040' 016201  000032
001044'                                 WHILE (R1) NE #ENDLST DO                50035$:
(4)  001044'                                                                            CMP     (R1),#ENDLST
(6)  001044' 021127  000000                                                             BEQ     50036$
(9)  001050' 001404
001052'                                         LET R0 := R0 + #1                       INC     R0
(6)  001052' 005200
001054'                                         LET R1 := R1 + #2                       ADD     #2,R1
(6)  001054' 062701  000002
001060'                                 ENDDO                                           BR      50035$
(4)  001060' 000771                                                             50036$:
(3)  001062'


                                ;+
                                ; START THE SORT OPERATION
                                ;-

001062'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
(4)  001062' 016201  000032


                                ;+
                                ; WORK THRU THE LIST UNTIL THE END IS REACHED
                                ;-

001066'                                 WHILE R0 GT #0 DO                       50037$:
(4)  001066'                                                                            TST     R0
(6)  001066' 005700                                                                     BLE     50040$
(9)  001070' 003432
001072'                                         REPEAT                          50041$:
(3)  001072'


                                ;+
                                ; IS THE CURRENT ENTRY A BKMOD MODULE?
                                ;-

001072'                                         LET R4 := (R1)                          MOV     (R1),R4
(4)  001072' 011104
001074'                                         IF #BKMOD SETIN STAT(R4) THEN           BIT     #BKMOD,STAT(R4)
(6)  001074' 032764  000020  000026                                                     BEQ     50042$
(9)  001102' 001414


                                ;+
                                ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
                                ;-

001104'                                                 LET R3 := 2(R1)
```

```
 (4)  001104' 016103  000002                                                    MOV     2(R1),R3
1026
1027                                    ;+
1028                                    ; MAKE SURE ITS NOT THE END OF THE LIST
1029                                    ;-
1030
1031  001110'                                       IF R3 NE #ENDLST THEN
 (6)  001110' 020327  000000                                                    CMP     R3,#ENDLST
 (9)  001114' 001407                                                            BEQ     50043$
1032
1033                                    ;+
1034                                    ; IF ITS NOT A BKMOD, SWAP THEM
1035                                    ;-
1036
1037  001116'                                             IF #BKMOD NOTSETIN STAT(R3) THEN
 (6)  001116' 032763  000020  000026                                           BIT     #BKMOD,STAT(R3)
 (9)  001124' 001003                                                            BNE     50044$
1038  001126'                                                 LET 2(R1) := (R1)
 (4)  001126' 011161  000002                                                    MOV     (R1),2(R1)
1039  001132'                                                 LET (R1) := R3
 (4)  001132' 010311                                                            MOV     R3,(R1)
1040  001134'                                             ENDIF
 (4)  001134'                                                           50044$:
1041  001134'                                       ENDIF
 (4)  001134'                                                           50043$:
1042  001134'                                 ENDIF
 (4)  001134'                                                           50042$:
1043
1044                                    ;+
1045                                    ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
1046                                    ;-
1047
1048  001134'                                       LET R1 := R1 + #2
 (6)  001134' 062701  000002                                                    ADD     #2,R1
1049  001140'                                 UNTIL (R1) EQ #ENDLST
 (3)  001140' 021127  000000                                                    CMP     (R1),#ENDLST
 (6)  001144' 001352                                                            BNE     50041$
1050
1051                                    ;+
1052                                    ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
1053                                    ;-
1054
1055  001146'                                       LET R0 := R0 - #1
 (6)  001146' 005300                                                            DEC     R0
1056  001150'                                       LET R1 := DT.MLST(R2)
 (4)  001150' 016201  000032                                                    MOV     DT.MLST(R2),R1
1057  001154'                                 ENDDO
 (4)  001154' 000744                                                            BR      50037$
 (3)  001156'                                                           50040$:
1058
1059                                    ;+
1060                                    ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
1061                                    ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
1062                                    ; A BACKGROUND LIST)  INTO THE DATA TABLE
1063                                    ;-
1064
```

```
                        LET R1 := DT.MLST(R2)                                   MOV     DT.MLST(R2),R1

                ;+
                ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
                ;-

                        LET DT.BLST(R2) := #0                                   CLR     DT.BLST(R2)

                        LET R3 := #-1                                           MOV     #-1,R3

                        WHILE (R1) NE #ENDLST AND R3 NE #0 DO           50045$:
                                                                                CMP     (R1),#ENDLST
                                                                                BEQ     50046$
                                                                                TST     R3
                                                                                BEQ     50046$

                                LET R4 := (R1)                                  MOV     (R1),R4

                                IF #BKMOD SETIN STAT(R4) THEN                   BIT     #BKMOD,STAT(R4)
                                                                                BEQ     50047$

                ;+
                ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
                ;-

                                        LET DT.BLST(R2) := R1                   MOV     R1,DT.BLST(R2)

                                        LET R1 := #ENDLST                       MOV     #ENDLST,R1

                                        LET R3 := #0                            CLR     R3

                                ELSE                                            BR      50050$
                                                                        50047$:

                ;+
                ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
                ;-

                                        LET R1 := R1 + #2                       ADD     #2,R1

                                ENDIF                                   50050$:

                        ENDDO                                                   BR      50045$
                                                                        50046$:

                ;+
                ; NOW RETURN TO THE CALLER
                ;-

                        CALL RESREG                                             JSR     PC,RESREG

                        ENDRTN
```

```
  (3)  001244'                                                          50000$:
  (3)  001244'                                                          50001$:
  (2)  001244' 000207                                                           RTS     PC
1100
1101                                ;+
1102                                ; LABEL TYPE QUEUE HIGH LIMIT
1103                                ;-
1104
1105  001246'                       INLINE <OV.TQEND:>                                OV.TQEND:
 (2)  001246'
1106
1107
1108
1109
1110
1111                                ;************************************
1112                                ; NOTE:  THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
1113                                ;   (THAT IS, AFTER SIZPOL AND TQ.END LABELS)
1114
1115                                ;++++++++++++++++++++++++++++++++++++
1116                                ;    OVERLAY REGIONS EQUATES
1117                                ;++++++++++++++++++++++++++++++++++++
1118          000000'               OV.KBBUF = SIZPOL                       ;KEYBOARD BUFFER STARTING ADDRESS
1119          000122'               OV.CQ= OV.KBBUF + ^D<82>                ;CONTROL QUEUE STARTING ADDRESS
1120          000364'               OV.TQ = OV.CQ + ^D<162>                 ;TYPE QUEUE STARTING ADDRESS
1121          000120'               OV.HIKB = OV.CQ - 2                     ;KEYBOARD BUFFER HIGH LIMIT
1122          000122                OV.KBSIZ = OV.CQ - OV.KBBUF             ;KEYBOARD BUFFER SIZE
1123          000362'               OV.HICQ = OV.TQ - 2                     ;CONTROL QUEUE HIGH LIMIT
1124          000050                OV.CQSIZ = <OV.TQ - OV.CQ>/4            ;CONTROL QUEUE SIZE
1125          000662                OV.A = OV.TQEND - OV.TQ                 ;GET THE TQ SIZE
1126          000002                OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
1127          001244'               OV.HITQ = OV.TQEND - OV.REM             ;TYPE QUEUE HIGH LIMIT
1128          000066                OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10      ;TYPE QUEUE SIZE
1129
1130          000001                        .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CPUCPE= ****** G
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
HTKT  = ****** G
HTPAER= ****** G
ICONT = 000036
ICOUNT= 000040
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KTERRO= 000040
KTPRES= 000400
KTSET = ****** G
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NOAPTY= 000002
NULL  = 000000
OV.A  = 000662
OV.CQ = 000122RG
OV.CQS= 000050 G
OV.HIC= 000362RG
OV.HIK= 000120RG
OV.HIT= 001244RG
OV.KBB= 000000RG
OV.KBS= 000122 G
OV.REM= 000002
OV.TQ = 000364RG
OV.TQE  001246R
OV.TQS= 000066 G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SIZPOL  000000RG
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
SPVALU= 002200
```

```
SR0    = 177572
SR1    = 177574
SR2    = 177576
SR3    = 172516
SS.IOP= 177600
SS.PCL= 172136
SS.PCS= 172100
SS.RFS= 160000
SS.0KC= 004000
SS.1KP= 000040
SS.120= 120000
SS.28  = 000034
STAT   = 000026
STATBI= 064757
STAT1  = 000027
SUSPND= 000001
SVR0   = 000062
SVR1   = 000064
SVR2   = 000066
SVR3   = 000070
SVR4   = 000072
SVR5   = 000074
SVR6   = 000076
SYSCNT= 000052
SYSERR= 000100
TMPIO  = 000002
TQOVF  = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR   = 000116
WDTO   = 000114
WTINRE= 000352
WTWHMI= 000222
XFLAG  = 000005
XOFF   = 000023
XON    = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO  = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF  = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO  = 000403
$F$OR  = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404
$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0  = 000001
$ISK1  = 000001
$ISK2  = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0  = 000300
$NSK1  = 000120
$NSK2  = 000110
$NSK3  = 000110
$NSK4  = 000110
$NSK5  = 000110
$SAVLE= 177777
$SSK0  = 050046
$TAGLE= 177777
$TAGNU= 050051
$TEMP  = 000300
$TSK0  = 050045
$TSK1  = 050046
$TSK2  = 050050
$TSK3  = 050042
$TSK4  = 050043
$TSK5  = 050044
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST  = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC  = 001214R
$$LOCN= 000000
$$REG  = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC  = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO   = 000000
$$$TAG= 050000
.      = 001246R

. ABS.  000000     000
        001246     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SIZPLD,DSKZ:SIZPLD=SPMAC/ML,EQUATE,SIZPLD
 RUN-TIME: 29 19 .4 SECONDS
 RUN-TIME RATIO: 281/49=5.6
 CORE USED:  14K  (27 PAGES)
```

```
                        .TITLE  SIZPLE SIZE AND POLL SYSTEM - 11/70 SUPPORT ALONG WITH 22-BIT ADDRESSING
                        .IDENT /V0.0/

                        ;++
                        ; MODULE NAME:
                        ;       SIZPLE
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;       THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
                        ;       OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
                        ;       WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
                        ;       A BACKGROUND MODULE LIST (QUEUE)
                        ;       NOTE: SUPPORT 11/70 AND 22-BIT ADDRESSING
                        ;
                        ; INPUTS:
                        ;       DTABLE ADDRESS
                        ;
                        ; IMPLICIT INPUTS:
                        ;       CONFIGURATION WORD 0
                        ;       EXERCISER SIZE
                        ;       SYSTEM SIZE
                        ;       PARITY TABLE ADDRESS
                        ;       MODULE LIST
                        ;       BACKGROUND LIST
                        ;
                        ; OUTPUTS:
                        ;       NONE
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       PARITY TABLE ADDRESS
                        ;       SYSTEM SIZE
                        ;       EXERCISE SIZE
                        ;       CONFIGURATION WORD 0
                        ;
                        ; PATHOLOGICAL CONNECTIONS:
                        ;
                        ; SUBORDINATE ROUTINES CALLED:
                        ;       SAVREG - SAVE REGISTERS
                        ;       RESREG - RESTORE REGISTERS
                        ;       KTSET - SET UP KT REGISTERS
                        ;       HRDADRCHK - ILLEGAL ADDRESS CHECK
                        ;       ICSRSZ - INDIRECT ECC CSR SIZING
                        ;
                        ; FUNCTIONAL SIDE EFFECTS:
                        ;       NONE
                        ;
                        ; CALLING SEQUENCE:
                        ;       CALL SIZPOL IN <A>
                        ;                 A - DTABLE ADDRESS
                        ;
                        ; VERSION:
                        ;       0.0
                        ;
                        ;       EDIT            DATE            BY              REASON
                        ;--
```

```
                        .SBTTL COMMON DEFINITIONS AND REFERENCES

                        .MCALL STRUCT
000000'                 STRUCT
(1) 000000'             .PRINT  ;SPMAC: VERSION 1.1
            000001      $LSTIN = 1
            000001      $LSTTAG = 1


                        ;****************
                        ; GLOBAL REFERENCES
                        ;
                        .GLOBL  HTPAER
                        .GLOBL  HTKT
                        .GLOBL  SAVREG              ;SARE REGISTERS
                        .GLOBL  RESREG              ;RESTORE REGISTERS
                        .GLOBL  KTSET               ;SET UP KT
                        .GLOBL  HRDADRCHK           ;ILLEGAL ADDRESS CHECK
                        .GLOBL  ICSRSZ              ;INDIRECT ECC CSR SIZING
                        .GLOBL  CPUCPE              ;11/70 CPU ERROR REGISTER
                        .GLOBL  CCNTRL              ;CACHE CONTROL REGISTER
                        .GLOBL  KONTRL              ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER

                        ;****************
                        ; LOCAL EQUATES
                        ;
            172100      SS.PCSR = 172100            ;START OF PARITY CSRS
            172136      SS.PCLS = 172136            ;LAST PARITY CSR
            004000      SS.0KC  = 4000              ;ONE K CHUNK
            000034      SS.28   = ^D28              ;28 DECIMAL
            000040      SS.1KP  = 40                ;1K IN PAR FORMAT
            120000      SS.120K = 120000            ;FIRST ADDRESS IN PAR5 MAP
            177600      SS.IOP  = 177600            ;I/O PAGE
            160000      SS.RFS  = 160000            ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE

                        ;****************
                        ; REFERENCED BY OTHER MODULES
                        ;
                        .GLOBL  SIZPOL              ;MODULE ENTRY POINT




                        ;++++++++++++++++++++++++++++++
                        ;    OVERLAY REGION GLOBALS
                        ;++++++++++++++++++++++++++++++
                        .GLOBL  OV.KBBUF
                        .GLOBL  OV.HIKB
                        .GLOBL  OV.KBSIZ
                        .GLOBL  OV.CQ
                        .GLOBL  OV.HICQ
                        .GLOBL  OV.CQSIZ
                        .GLOBL  OV.TQ
                        .GLOBL  OV.HITQ
                        .GLOBL  OV.TQSIZ
```

```
                                .SBTTL KERNEL - DETERMINE PROCESSOR TYPE



000000'                         ROUTINE SIZPOL <DTABLE>
(2)  000000'                                                                    SIZPOL:
                                ;+
                                ; STUFF PARITY AND KT VECTORS
                                ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
                                ; R1
                                ;-


000000'                         LET @#114 := #HTPAER                            MOV     #HTPAER,@#114
(4)  000000' 012737  000000G 000114
                                LET @#116 := #PRI7                              MOV     #PRI7,@#116
000006'
(4)  000006' 012737  000340  000116

000014'                         LET @#250 := #HTKT                              MOV     #HTKT,@#250
(4)  000014' 012737  000000G 000250
                                LET @#252 := #PRI7                              MOV     #PRI7,@#252
000022'
(4)  000022' 012737  000340  000252

000030'                         CALL SAVREG                                     JSR     PC,SAVREG
(3)  000030' 004767  000000G
                                LET R2 := DTABLE(R5)                            MOV     DTABLE(R5),R2
000034'
(4)  000034' 016502  000000
                                LET R1 := #0                                    CLR     R1
000040'
(4)  000040' 005001


                                ;+
                                ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
                                ;-

000042'                         CALL HRDADRCHK IN <CPUCPE>                      MOV     R5,-(SP)
(3)  000042' 010546                                                             MOV     CPUCPE,-(R5)
(4)  000044' 016745  000000G                                                    JSR     PC,HRDADRCHK
(3)  000050' 004767  000000G                                                    MOV     (SP)+,R5
(3)  000054' 012605

000056'                         IF.NO.ERROR THEN                                BCS     50002$
(6)  000056' 103422
                                ;+
                                ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
                                ; 11/60
                                ;-

000060'                                 PUSH @#10                               MOV     @#10,-(SP)
(2)  000060' 013746  000010
                                        LET @#10 := #1$                         MOV     #1$,@#10
000064'
(4)  000064' 012737  000104' 000010
                                        INLINE <MED>                            MED
000072'
(2)  000072' 076600
                                        INLINE <RDSERV>                         RDSERV
000074'
(2)  000074' 000101
```

```
                                        LET R1 := R1 SET.BY #PDP60
(6)  000076' 052701  004000                                                 BIS     #PDP60,R1
                                        INLINE <BR 3$>
(2)  000102' 000405                                                         BR 3$

                                INLINE <1$:>
(2)  000104'                                                                1$:

                                ;+
                                ; SET RETURN TO 2$ TO RESTORE PSW
                                ;-

                                        LET (SP) := #2$
(4)  000104' 012716  000112'                                                MOV     #2$,(SP)
                                        INLINE <RTI>
(2)  000110' 000002                                                         RTI

                                ;+
                                ; MUST BE AN 11/70, BECAUSE OF TRAP
                                ;-

                                INLINE <2$:>
(2)  000112'                                                                2$:
                                        LET R1 := R1 SET.BY #PDP70
(6)  000112' 052701  010000                                                 BIS     #PDP70,R1

                                ;+
                                ; RESTORE RESERVED INSTRUCTION TRAP
                                ;-

                                INLINE <3$:>
(2)  000116'                                                                3$:
                                        POP @#10
(2)  000116' 012637  000010                                                 MOV     (SP)+,@#10
                                ELSE
(4)  000122' 000411                                                         BR      50003$
(3)  000124'                                                        50002$:

                                ;+
                                ; CHECK TO SEE IF LSI11
                                ;-

                                        CALL HRDADRCHK IN <#PSW>
(3)  000124' 010546                                                         MOV     R5,-(SP)
(4)  000126' 012745  177776                                                 MOV     #PSW,-(R5)
(3)  000132' 004767  000000G                                                JSR     PC,HRDADRCHK
(3)  000136' 012605                                                         MOV     (SP)+,R5
                                        IF.ERROR THEN
(6)  000140' 103002                                                         BCC     50004$
                                                LET R1 := R1 SET.BY #PDPLSI
(6)  000142' 052701  020000                                                 BIS     #PDPLSI,R1

                                        ENDIF
(4)  000146'                                                        50004$:
                                ENDIF
(4)  000146'                                                        50003$:
```

696

```
                                .SBTTL KERNEL - DETERMINE CPU OPTIONS

                                ;+
                                ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
                                ; SEE IF EXTENDED KT" IS PRESENT
                                ;-

000146'                         CALL HRDADRCHK IN <#SR0>
(3)  000146' 010546                                                     MOV     R5,-(SP)
(4)  000150' 012745  177572                                             MOV     #SR0,-(R5)
(3)  000154' 004767  000000G                                            JSR     PC,HRDADRCHK
(3)  000160' 012605                                                     MOV     (SP)+,R5
000162'                         IF.NO.ERROR THEN
(6)  000162' 103424                                                     BCS     50005$
000164'                                 LET R1 := R1 SET.BY #KTPRES
(6)  000164' 052701  000400                                             BIS     #KTPRES,R1
000170'                                 CALL HRDADRCHK IN <#SR3>
(3)  000170' 010546                                                     MOV     R5,-(SP)
(4)  000172' 012745  172516                                             MOV     #SR3,-(R5)
(3)  000176' 004767  000000G                                            JSR     PC,HRDADRCHK
(3)  000202' 012605                                                     MOV     (SP)+,R5
000204'                                 IF.NO.ERROR THEN
(6)  000204' 103413                                                     BCS     50006$
000206'                                         LET R1 := R1 SET.BY #KTXTND
(6)  000206' 052701  040000                                             BIS     #KTXTND,R1

                                ;+
                                ; DETERMINE IF 22-ADDRESSING BY LOADING PAR5 WITH 16 BIT
                                ; WORD AND READ PAR5 TO SEE IF WORD READ IS SAME AS WRITTEN
                                ;-

000212'                                 LET @#KIPAR5 := #100000
(4)  000212' 012737  100000  172352                                     MOV     #100000,@#KIPAR5
000220'                                 IF @#KIPAR5 EQ #100000 THEN
(6)  000220' 023727  172352  100000                                     CMP     @#KIPAR5,#100000
(9)  000226' 001002                                                     BNE     50007$
000230'                                         LET R1 := R1 SET.BY #ADDR22
(6)  000230' 052701  001000                                             BIS     #ADDR22,R1
000234'                                 ENDIF
(4)  000234'                                                    50007$:
000234'                         ENDIF
(4)  000234'                                                    50006$:
000234'                         ENDIF
(4)  000234'                                                    50005$:

                                ;+
                                ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
                                ; CACHE CONTROL REGISTER
                                ;-



000234'                         CALL HRDADRCHK IN <CCNTRL>
(3)  000234' 010546                                                     MOV     R5,-(SP)
(4)  000236' 016745  000000G                                            MOV     CCNTRL,-(R5)
```

```
(3)  000242' 004767  000000G                                        JSR     PC,HRDADRCHK
(3)  000246' 012605                                                 MOV     (SP)+,R5
733  000250'                    IF.NO.ERROR THEN
(6)  000250' 103402                                                 BCS     50010$
734  000252'                            LET R1 := R1 SET.BY #CAPRES
(6)  000252' 052701  000004                                         BIS     #CAPRES,R1
735  000256'                    ENDIF
(4)  000256'                                                    50010$:
736
```

```
                        .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT

                        ;+
                        ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
                        ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
                        ; IN PARITY TABLE.
                        ;-

000256'                 LET R0 := DT.PTA(R2)                                    MOV     DT.PTA(R2),R0
(4) 000256' 016200 000064

000262'                 LET R3 := #SS.PCSR                                      MOV     #SS.PCSR,R3
(4) 000262' 012703 172100
000266'                 WHILE R3 LE #SS.PCLS DO                         50011$:
(4) 000266'                                                                     CMP     R3,#SS.PCLS
(6) 000266' 020327 172136                                                       BGT     50012$
(9) 000272' 003035
000274'                         CALL HRDADRCHK IN <R3>                          MOV     R5,-(SP)
(3) 000274' 010546                                                              MOV     R3,-(R5)
(4) 000276' 010345                                                              JSR     PC,HRDADRCHK
(3) 000300' 004767 000000G                                                      MOV     (SP)+,R5
(3) 000304' 012605
000306'                         IF.NO.ERROR THEN                                BCS     50013$
(6) 000306' 103414
000310'                                 LET (R0)+ := R3                         MOV     R3,(R0)+
(4) 000310' 010320

                        ;+
                        ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
                        ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
                        ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
                        ;-
000312'                                 LET (R3) := (R3) SET.BY #BIT01          BIS     #BIT01,(R3)
(6) 000312' 052713 000002
000316'                                 IF #BIT01 SETIN (R3) THEN               BIT     #BIT01,(R3)
(6) 000316' 032713 000002                                                       BEQ     50014$
(9) 000322' 001403
000324'                                         LET R1 := R1 SET.BY #ECCMEM     BIS     #ECCMEM,R1
(6) 000324' 052701 000100
000330'                                 ELSE                                    BR      50015$
(4) 000330' 000402                                                      50014$:
(3) 000332'
000332'                                         LET R1 := R1 SET.BY #PARPRES    BIS     #PARPRES,R1
(6) 000332' 052701 002000
000336'                                 ENDIF                           50015$:
(4) 000336'
000336'                                 LET (R3) := #0                          CLR     (R3)
(4) 000336' 005013
000340'                         ENDIF                                   50013$:
(4) 000340'

                        ;+
                        ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
                        ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
                        ; PARITY CSR + 2
                        ;-
```

```
773  000340'                        IF #PDP70 SETIN R1 THEN
(6)  000340' 032701  010000                                                  BIT     #PDP70,R1
(9)  000344' 001405                                                          BEQ     50016$
774  000346'                                LET R1 := R1 SET.BY #PARPRES
(6)  000346' 052701  002000                                                  BIS     #PARPRES,R1
775  000352'                                LET R3 := R3 + #4
(6)  000352' 062703  000004                                                  ADD     #4,R3
776  000356'                        ELSE
(4)  000356' 000402                                                          BR      50017$
(3)  000360'                                                              50016$:
777  000360'                                LET R3 := R3 + #2
(6)  000360' 062703  000002                                                  ADD     #2,R3
778  000364'                        ENDIF
(4)  000364'                                                              50017$:
779  000364'                    ENDDO
(4)  000364' 000740                                                          BR      50011$
(3)  000366'                                                              50012$:
780
781                             ;+
782                             ; DETERMINE IF WE SHOULD CHECK FOR INDIRECT CSRS ON THE 11/70
783                             ;-
784
785  000366'                    IF #PDP70 SETIN R1 AND @DT.PTA(R2) EQ #0 THEN
(6)  000366' 032701  010000                                                  BIT     #PDP70,R1
(9)  000372' 001410                                                          BEQ     50020$
(6)  000374' 005772  000064                                                  TST     @DT.PTA(R2)
(9)  000400' 001005                                                          BNE     50020$
786  000402'                            CALL ICSRSZ IN <R2>
(3)  000402' 010546                                                          MOV     R5,-(SP)
(4)  000404' 010245                                                          MOV     R2,-(R5)
(3)  000406' 004767  000000G                                                 JSR     PC,ICSRSZ
(3)  000412' 012605                                                          MOV     (SP)+,R5
787  000414'                    ENDIF
(4)  000414'                                                              50020$:
788
789
```

```
                                .SBTTL KERNEL - DETERMINE SYSTEM SIZE

                                ;+
                                ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
                                ; AND LOAD IT INTO EXERCISER SIZE WORD
                                ;-

000414'                         LET DT.ESIZ(R2) := 0
(4)  000414' 016762  000000  000044                                    MOV     0,DT.ESIZ(R2)

                                ;+
                                ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
                                ; THAN 28K.
                                ;-

000422'                         IF #KTPRES NOTSETIN R1 THEN
(6)  000422' 032701  000400                                            BIT     #KTPRES,R1
(9)  000426' 001053                                                    BNE     50021$

                                ;+
                                ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
                                ;-

000430'                                 LET R0 := #0
(4)  000430' 005000                                                    CLR     R0
000432'                                 LET R3 := #0
(4)  000432' 005003                                                    CLR     R3

                                ;+
                                ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
                                ;-

000434'                                 REPEAT
(3)  000434'                                                   50022$:
000434'                                         LET R0 := R0 + #1
(6)  000434' 005200                                                    INC     R0
000436'                                         LET R3 := R3 + #SS.0KC
(6)  000436' 062703  004000                                            ADD     #SS.0KC,R3
000442'                                         CALL HRDADRCHK IN <R3>
(3)  000442' 010546                                                    MOV     R5,-(SP)
(4)  000444' 010345                                                    MOV     R3,-(R5)
(3)  000446' 004767  000000G                                           JSR     PC,HRDADRCHK
(3)  000452' 012605                                                    MOV     (SP)+,R5
000454'                                         IF.ERROR THEN
(6)  000454' 103001                                                    BCC     50023$
000456'                                                 INLINE <BR 4$>
(2)  000456' 000403                                                    BR 4$
000460'                                         ENDIF
(4)  000460'                                                   50023$:
000460'                                 UNTIL R0 EQ #SS.28
(3)  000460' 020027  000034                                            CMP     R0,#SS.28
(6)  000464' 001363                                                    BNE     50022$

                                ;+
                                ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
                                ;-
```

```
                        INLINE <4$:>                                    4$:
                                IF R0 EQ #SS.28 THEN                    CMP     R0,#SS.28
                                                                        BNE     50024$
                                        CALL HRDADRCHK IN <#SS.RFS>     MOV     R5,-(SP)
                                                                        MOV     #SS.RFS,-(R5)
                                                                        JSR     PC,HRDADRCHK
                                                                        MOV     (SP)+,R5
                                        IF.NO.ERROR THEN                BCS     50025$
                                                LET R3 := R3 + #10000   ADD     #10000,R3
                                        ENDIF
                                                                50025$:
                                ENDIF
                                                                50024$:

                        ;+
                        ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
                        ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
                        ; SIZE WORD.
                        ;-

                                LET CARRY := #0                         CLC
                                LET DT.SSIZ(R2) := R3 ROTATE -6         MOV     R3,DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                                                                        ROR     DT.SSIZ(R2)
                        ELSE                                            BR      50026$
                                                                50021$:

                        ;+
                        ; KT IS PRESENT, FIRST GO MAP APRS
                        ;-
                                CALL KTSET IN <R2>                      MOV     R5,-(SP)
                                                                        MOV     R2,-(R5)
                                                                        JSR     PC,KTSET
                                                                        MOV     (SP)+,R5

                        ;+
                        ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
                        ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
                        ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
                        ; 1K STEPS

                                LET @#KIPAR5 := #SS.1KP
```

```
(4)  000570' 012737  000040  172352                                                    MOV     #SS.1KP,@#KIPAR5
861  000576'
(6)  000576' 052737  000001  177572                                                    BIS     #BIT00,@#SR0
862  000604'
(6)  000604' 032701  001000                                                          BIT     #ADDR22,R1
(9)  000610' 001403                                                                  BEQ     50027$
863  000612'
(6)  000612' 052737  000020  172516                                                    BIS     #BIT04,@#SR3
864  000620'
(4)  000620'                                                                         50027$:
865
866  000620'                                                                         50030$:
(3)  000620'
867  000620'
(6)  000620' 062737  000040  172352                                                    ADD     #SS.1KP,@#KIPAR5
868  000626'
(3)  000626' 010546                                                                  MOV     R5,-(SP)
(4)  000630' 012745  120000                                                          MOV     #SS.120K,-(R5)
(3)  000634' 004767  000000G                                                         JSR     PC,HRDADRCHK
(3)  000640' 012605                                                                  MOV     (SP)+,R5
869  000642'
(6)  000642' 103001                                                                  BCC     50031$
870  000644'
(2)  000644' 000404                                                                  BR 5$
871  000646'                                                                         50031$:
(4)  000646'
872  000646'
(3)  000646' 023727  172352  177600                                                    CMP     @#KIPAR5,#SS.IOP
(6)  000654' 001361                                                                  BNE     50030$
873
874
875
876
877
878  000656'                                                                         5$:
(2)  000656'
879  000656'
(4)  000656' 013762  172352  000046                                                    MOV     @#KIPAR5,DT.SSIZ
880  000664'
(4)  000664'                                                                         50026$:
881
```

```
                        LET @#SR0 := @#SR0 SET.BY #BIT00

                        IF #ADDR22 SETIN R1 THEN

                                LET @#SR3 := @#SR3 SET.BY #BIT04

                        ENDIF


                        REPEAT

                                LET @#KIPAR5 := @#KIPAR5 + #SS.1KP

                                CALL HRDADRCHK IN <#SS.120K>




                                IF.ERROR THEN

                                        INLINE <BR 5$>

                                ENDIF

                        UNTIL @#KIPAR5 EQ #SS.IOP


;+
; LOAD SYSTEM SIZE WORD WITH CONTENTS OF KIPAR5
;-

INLINE <5$:>

                        LET DT.SSIZ(R2) := @#KIPAR5

ENDIF
```

```
                        .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT

                        ;+
                        ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
                        ;-

000664'                 IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
(6) 000664' 032701 002000                                       BIT     #PARPRES,R1
(8) 000670' 001003                                              BNE     50032$
(6) 000672' 032701 000100                                       BIT     #ECCMEM,R1
(9) 000676' 001506                                              BEQ     50033$
(6) 000700'                                             50032$:

                        ;+
                        ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER
                        ; THIS WILL TURN CACHE OFF
                        ;-

000700'                         IF #CAPRES SETIN R1 THEN
(6) 000700' 032701 000004                                       BIT     #CAPRES,R1
(9) 000704' 001406                                              BEQ     50034$
000706'                                 LET KONTRL := #3
(4) 000706' 012767 000003 000000G                               MOV     #3,KONTRL
000714'                                 LET @CCNTRL := KONTRL
(4) 000714' 016777 000000G 000000G                              MOV     KONTRL,@CCNTRL
000722'                         ENDIF
(4) 000722'                                             50034$:

                        ;+
                        ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
                        ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
                        ; OF MEMORY, FIRST DETERMINE IF KT IS PRESENT THEN SET
                        ; UP KT REGISTERS.
                        ;-

000722'                         IF #KTPRES SETIN R1 THEN
(6) 000722' 032701 000400                                       BIT     #KTPRES,R1
(9) 000726' 001434                                              BEQ     50035$
000730'                                 CALL KTSET
(3) 000730' 004767 000000G                                      JSR     PC,KTSET

                        ;+
                        ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
                        ; SET UP KIPAR5 TO MAP TO 0
                        ;-

                                        LET @#KIPDR6 := #0
000734'                                                         CLR     @#KIPDR6
(4) 000734' 005037 172314
                                        LET @#KIPAR5 := #0
000740'                                                         CLR     @#KIPAR5
(4) 000740' 005037 172352

                        ;+
                        ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
                        ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
                        ; SET UP R0 TO USE PAR5
                        ;-
```

```
                                        PUSH @#4                                        MOV     @#4,-(SP)
000744'
(2)  000744' 013746  000004
                                        LET @#4 := #8$                                  MOV     #8$,@#4
000750'
(4)  000750' 012737  001060' 000004
                                        PUSH @#250                                      MOV     @#250,-(SP)
000756'
(2)  000756' 013746  000250
                                        LET @#250 := #7$                                MOV     #7$,@#250
000762'
(4)  000762' 012737  001040' 000250
                                        LET @#SR0 := @#SR0 SET.BY #BIT00                BIS     #BIT00,@#SR0
000770'
(6)  000770' 052737  000001  177572
                                        IF #ADDR22 SETIN R1 THEN                        BIT     #ADDR22,R1
000776'
(6)  000776' 032701  001000                                                             BEQ     50036$
(9)  001002' 001403


                                                LET @#SR3 := @#SR3 SET.BY #BIT04
001004'                                                                                 BIS     #BIT04,@#SR3
(6)  001004' 052737  000020  172516
                                        ENDIF
001012'                                                                         50036$:
(4)  001012'
                                        LET R0 := #SS.120K
001012'                                                                                 MOV     #SS.120K,R0
(4)  001012' 012700  120000
                                ELSE                                                    BR      50037$
001016'
(4)  001016' 000406                                                             50035$:
(3)  001020'

                        ;+
                        ; IF KT NOT PRESENT THEN SET UP POINTER AND SAVE BUSERR VECTOR
                        ;-

                                        LET R0 := #0                                    CLR     R0
001020'
(4)  001020' 005000
                                        PUSH @#4                                        MOV     @#4,-(SP)
001022'
(2)  001022' 013746  000004
                                        LET @#4 := #10$                                 MOV     #10$,@#4
001026'
(4)  001026' 012737  001106' 000004

                                ENDIF                                           50037$:
001034'
(4)  001034'

                        ;+
                        ; NOW WRITE GOOD PARITY
                        ;-
001034'                 INLINE<6$:>                                             6$:
(2)  001034'
                                        LET (R0)+ := (R0)                               MOV     (R0),(R0)+
001034'
(4)  001034' 011020
                                        INLINE <BR 6$>                                  BR 6$
001036'
(2)  001036' 000776

                        ;+
                        ; COME HERE BY WAY OF KT TRAP
                        ; UPDATE PAR5 TO NEXT 4K BLOCK AND RESET R0 TO START
                        ; OF BLOCK. LOAD RETURN PC WITH 6$ AND RTI
```

```
                                ;-
                                INLINE <7$:>
                                        LET @#KIPAR5 := @#KIPAR5 + #200
                                        LET R0 := #SS.120K
                                        LET (SP) := #6$
                                        INLINE <RTI>

                                ;+
                                ; COME HERE BY WAY OF ILLEGAL ADDRESS TRAP
                                ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
                                ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
                                ;-
                                INLINE <8$:>
                                        LET @#SR0 := @#SR0 CLR.BY #BIT00
                                        LET (SP) := #9$
                                        INLINE <RTI>

                                INLINE <9$:>
                                        POP @#250
                                        POP @#4
                                        INLINE <BR 11$>

                                ;+
                                ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
                                ; STACK AND RESTORE BUSERR
                                ;-

                                INLINE <10$:>
                                        INLINE <CMP (SP)+,(SP)+>
                                        POP @#4

                                ENDIF

                                ;+
                                ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD0
```

```
001040'                                 7$:
001040'
001040'                                 ADD     #200,@#KIPAR5
001040' 062737  000200  172352
001046'                                 MOV     #SS.120K,R0
001046' 012700  120000
001052'                                 MOV     #6$,(SP)
001052' 012716  001034'
001056'                                 RTI
001056' 000002


001060'                                 8$:
001060'
001060'                                 BIC     #BIT00,@#SR0
001060' 042737  000001  177572

001066'                                 MOV     #9$,(SP)
001066' 012716  001074'
001072'                                 RTI
001072' 000002

001074'                                 9$:
001074'
001074'                                 MOV     (SP)+,@#250
001074' 012637  000250
001100'                                 MOV     (SP)+,@#4
001100' 012637  000004
001104'                                 BR 11$
001104' 000403


001106'                                 10$:
001106'
001106'                                 CMP (SP)+,(SP)+
001106' 022626
001110'                                 MOV     (SP)+,@#4
001110' 012637  000004

001114'                                 50033$:
001114'


```

```
                                ;-
                                
001114'                         INLINE <11$:>                                   11$:
(2)  001114'
001114'                         IF @#42 NE #0 THEN                              TST     @#42
(6)  001114' 005737  000042                                                     BEQ     50040$
(9)  001120' 001402
001122'                                 LET R1 := R1 SET.BY #AUTO               BIS     #AUTO,R1
(6)  001122' 052701  000010
001126'                         ENDIF                                   50040$:
(4)  001126'
                                
                                ;+
                                ; LOAD UP CONFIGURATION WORD 0
                                ; CLEAN UP R1 BEFORE LOADING DT.CF0
                                ;-
                                
001126'                         LET R1 := R1 CLR.BY #OWEN                       BIC     #OWEN,R1
(6)  001126' 042701  024020
001132'                         LET DT.CF0(R2) := DT.CF0(R2) SET.BY R1          BIS     R1,DT.CF0(R2)
(6)  001132' 050162  000014
                                
```

```
                                .SBTTL KERNEL - SORT MODULE LIST


                                ;+
                                ; COUNT THE MODULES
                                ;-

001136'                                 LET R0 := #0                                    CLR     R0
 (4)  001136' 005000
001140'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
 (4)  001140' 016201  000032
001144'                                 WHILE (R1) NE #ENDLST DO                50041$:
 (4)  001144'                                                                           CMP     (R1),#ENDLST
 (6)  001144' 021127  000000                                                            BEQ     50042$
 (9)  001150' 001404
001152'                                         LET R0 := R0 + #1                       INC     R0
 (6)  001152' 005200
001154'                                         LET R1 := R1 + #2                       ADD     #2,R1
 (6)  001154' 062701  000002
001160'                                 ENDDO                                           BR      50041$
 (4)  001160' 000771                                                            50042$:
 (3)  001162'

                                ;+
                                ; START THE SORT OPERATION, AND SORT "BKMOD" TO BOTTOM OF MODQ LIST,
                                ; DEVELOPING A "BKMOD QUEUE"
                                ;-

001162'                                 LET R1 := DT.MLST(R2)                           MOV     DT.MLST(R2),R1
 (4)  001162' 016201  000032

                                ;+
                                ; WORK THRU THE LIST UNTIL THE END IS REACHED
                                ;-

001166'                                 WHILE R0 GT #0 DO                       50043$:
 (4)  001166'                                                                           TST     R0
 (6)  001166' 005700                                                                    BLE     50044$
 (9)  001170' 003432
001172'                                         REPEAT                          50045$:
 (3)  001172'

                                ;+
                                ; IS THE CURRENT ENTRY A BKMOD MODULE?
                                ;-

001172'                                         LET R4 := (R1)                          MOV     (R1),R4
 (4)  001172' 011104
001174'                                         IF #BKMOD SETIN STAT(R4) THEN           BIT     #BKMOD,STAT(R4)
 (6)  001174' 032764  000020  000026                                                   BEQ     50046$
 (9)  001202' 001414

                                ;+
                                ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
                                ;-
```

```
                                                        LET R3 := 2(R1)
001204'                                                                         MOV     2(R1),R3
(4)  001204' 016103  000002
                        ;+
                        ; MAKE SURE ITS NOT THE END OF THE LIST
                        ;-
                                                        IF R3 NE #ENDLST THEN
001210'                                                                         CMP     R3,#ENDLST
(6)  001210' 020327  000000                                                     BEQ     50047$
(9)  001214' 001407
                        ;+
                        ; IF ITS NOT A BKMOD, SWAP THEM
                        ;-
                                                                IF #BKMOD NOTSETIN STAT(R3) THEN
001216'                                                                         BIT     #BKMOD,STAT(R3)
(6)  001216' 032763  000020  000026                                             BNE     50050$
(9)  001224' 001003
                                                                        LET 2(R1) := (R1)
001226'                                                                         MOV     (R1),2(R1)
(4)  001226' 011161  000002
                                                                        LET (R1) := R3
001232'                                                                         MOV     R3,(R1)
(4)  001232' 010311
                                                                ENDIF
001234'                                                                 50050$:
(4)  001234'
                                                        ENDIF
001234'                                                                 50047$:
(4)  001234'
                                                ENDIF
001234'                                                                 50046$:
(4)  001234'
                        ;+
                        ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
                        ;-
                                                LET R1 := R1 + #2
001234'                                                                         ADD     #2,R1
(6)  001234' 062701  000002
                                        UNTIL (R1) EQ #ENDLST
001240'                                                                         CMP     (R1),#ENDLST
(3)  001240' 021127  000000                                                     BNE     50045$
(6)  001244' 001352
                        ;+
                        ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
                        ;-
                                        LET R0 := R0 - #1
001246'                                                                         DEC     R0
(6)  001246' 005300
                                        LET R1 := DT.MLST(R2)
001250'                                                                         MOV     DT.MLST(R2),R1
(4)  001250' 016201  000032
                                ENDDO
001254'                                                                         BR      50043$
(4)  001254' 000744                                                     50044$:
(3)  001256'
                        ;+
                        ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
                        ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
                        ; A BACKGROUND LIST)  INTO THE DATA TABLE
                        ;-
```

```
1094
1095  001256'                               LET R1 := DT.MLST(R2)                                   MOV     DT.MLST(R2),R1
 (4)  001256' 016201  000032
1096
1097                                        ;+
1098                                        ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
1099                                        ;-
1100
1101  001262'                               LET DT.BLST(R2) := #0                                   CLR     DT.BLST(R2)
 (4)  001262' 005062  000034
1102  001266'                               LET R3 := #-1                                           MOV     #-1,R3
 (4)  001266' 012703  177777
1103  001272'                               WHILE (R1) NE #ENDLST AND R3 NE #0 DO
 (4)  001272'                                                                                  50051$:
 (6)  001272' 021127  000000                                                                          CMP     (R1),#ENDLST
 (9)  001276' 001420                                                                                  BEQ     50052$
 (6)  001300' 005703                                                                                  TST     R3
 (9)  001302' 001416                                                                                  BEQ     50052$
1104  001304'                                       LET R4 := (R1)                                  MOV     (R1),R4
 (4)  001304' 011104
1105  001306'                                       IF #BKMOD SETIN STAT(R4) THEN                   BIT     #BKMOD,STAT(R4)
 (6)  001306' 032764  000020  000026                                                                 BEQ     50053$
 (9)  001314' 001406
1106
1107                                        ;+
1108                                        ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
1109                                        ;-
1110
1111  001316'                                               LET DT.BLST(R2) := R1                   MOV     R1,DT.BLST(R2)
 (4)  001316' 010162  000034
1112  001322'                                               LET R1 := #ENDLST                       MOV     #ENDLST,R1
 (4)  001322' 012701  000000
1113  001326'                                               LET R3 := #0                            CLR     R3
 (4)  001326' 005003
1114  001330'                                       ELSE                                            BR      50054$
 (4)  001330' 000402                                                                           50053$:
 (3)  001332'
1115
1116                                        ;+
1117                                        ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
1118                                        ;-
1119
1120  001332'                                               LET R1 := R1 + #2                       ADD     #2,R1
 (6)  001332' 062701  000002
1121  001336'                                       ENDIF                                       50054$:
 (4)  001336'
1122  001336'                               ENDDO                                                   BR      50051$
 (4)  001336' 000755                                                                           50052$:
 (3)  001340'
1123
1124                                        ;+
1125                                        ; NOW RETURN TO THE CALLER
1126                                        ;-
1127
1128  001340'                               CALL RESREG                                             JSR     PC,RESREG
 (3)  001340' 004767  000000G
```

```
1129  001344'                          ENDRTN
 (3)  001344'                                                          50000$:
 (3)  001344'                                                          50001$:
 (2)  001344' 000207                                                           RTS     PC
1130
1131                                ;+
1132                                ; LABEL TYPE QUEUE HIGH LIMIT
1133                                ;-
1134
1135  001346'                       INLINE <OV.TQEND:>                                OV.TQEND:
 (2)  001346'
1136
1137
1138
1139
1140
1141                                ;*************************************
1142                                ; NOTE:  THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
1143                                ;   (THAT IS, AFTER SIZPOL AND TQ.END LABELS)
1144
1145                                ;+++++++++++++++++++++++++++++++++++++
1146                                ;    OVERLAY REGIONS EQUATES
1147                                ;+++++++++++++++++++++++++++++++++++++
1148          000000'               OV.KBBUF = SIZPOL                      ;KEYBOARD BUFFER STARTING ADDRESS
1149          000122'               OV.CQ= OV.KBBUF + ^D<82>               ;CONTROL QUEUE STARTING ADDRESS
1150          000364'               OV.TQ = OV.CQ + ^D<162>                ;TYPE QUEUE STARTING ADDRESS
1151          000120'               OV.HIKB = OV.CQ - 2                    ;KEYBOARD BUFFER HIGH LIMIT
1152          000122                OV.KBSIZ = OV.CQ - OV.KBBUF            ;KEYBOARD BUFFER SIZE
1153          000362'               OV.HICQ = OV.TQ - 2                    ;CONTROL QUEUE HIGH LIMIT
1154          000050                OV.CQSIZ = <OV.TQ - OV.CQ>/4           ;CONTROL QUEUE SIZE
1155          000762                OV.A = OV.TQEND - OV.TQ                ;GET THE TQ SIZE
1156          000002                OV.REM = OV.A - <<OV.A/10>*10>         ;GET ANY REMAINDER
1157          001344'               OV.HITQ = OV.TQEND - OV.REM            ;TYPE QUEUE HIGH LIMIT
1158          000076                OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10      ;TYPE QUEUE SIZE
1159
1160          000001                        .END
```

```
ACSR  = 000102
ACTBIT= 004000
ADDR22= 001000
ADR   = 000006
APTFER= 000004
APTPRE= 000200
ASB   = 000106
ASSEMB= 000010
ASTAT = 000104
AUTO  = 000010
AUTOST= 020000
AWAS  = 000110
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BKDEF = 000002
BKMOD = 000020
BKMODE= 040000
BKSLSH= 000134
CAPRES= 000004
CASTAT= 000004
CCNTRL= ****** G
CDERCT= 000146
CDWDCT= 000144
CKTIM = 100000
CLKPRE= 000001
CONFIG= 000056
CPUCPE= ****** G
CQOVF = 000001
CR    = 000015
CSRA  = 000100
CSRC  = 000102
CTRLC = 000003
CTRLO = 000017
CTRLU = 000025
DCEVNT= 000011
DEFRTN= 000400
DIAGMC= 000000
DROPMO= 100000
DSEVNT= 000014
DTABLE= 000000
DT.ADD= 000042
DT.AP = 000100
DT.APK= 000076
DT.BLS= 000034
DT.CF0= 000014
DT.CF1= 000016
DT.ERR= 000020
DT.ESI= 000044
DT.EVN= 000000
DT.EXS= 000060
DT.FCH= 000037
DT.FCN= 000036
DT.HMX= 000104
DT.KBE= 000024
DT.KBP= 000026
DT.KBR= 000022
DT.KBU= 000030
DT.MLS= 000032
DT.MTI= 000110
DT.OFF= 000070
DT.PAS= 000074
DT.PC = 000002
DT.PFL= 000062
DT.PSW= 000004
DT.PTA= 000064
DT.RCS= 000102
DT.REL= 000040
DT.SCT= 000066
DT.SMX= 000106
DT.SP = 000006
DT.SSI= 000046
DT.ST0= 000010
DT.ST1= 000012
DT.SWR= 000056
DT.SYP= 000072
DT.WBU= 000050
DT.WHL= 000054
DT.WLL= 000052
DVID1 = 000014
ECCMEM= 000100
ECCSTA= 000010
ENBEOP= 010000
ENBNUL= 000001
ENDLST= 000000
EOPBIT= 000001
ERRTYP= 000106
EVNTBE= 000200
EVNTHD= 000200
EVNTKT= 000203
EVNTPE= 000202
EVNTRE= 000201
FATERR= 100000
HRDADR= ****** G
HRDCNT= 000044
HRDPAS= 000050
HTKT  = ****** G
HTPAER= ****** G
ICONT = 000036
ICOUNT= 000040
ICSRSZ= ****** G
IDNUM = 000122
IE    = 000100
INDPAR= 000040
INHDRP= 040000
INHEPR= 020000
INHREL= 001000
INHRRE= 000400
INIT  = 000030
INTR  = 000120
IOMOD = 100000
IOMODP= 102000
IOMODR= 112000
IOMODX= 110000
JACK  = 035060
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316
KONTRL= ****** G
KTERRO= 000040
KTPRES= 000400
KTSET = ****** G
KTSTAT= 000020
KTXTND= 040000
LF    = 000012
LPSTAT= 000001
MAPSTA= 000200
MED   = 076600
MEMPAS= 040000
MODEXH= 004000
MODHOL= 002000
MODSEL= 001000
MSGCKD= 000010
MSGCKS= 000011
MSGDER= 000005
MSGDRP= 000017
MSGECH= 177777
MSGEOP= 000013
MSGHDR= 000004
MSGHNG= 000022
MSGHRD= 000007
MSGMAP= 000021
MSGNUL= 177775
MSGPOP= 000002
MSGPRM= 177776
MSGRES= 000001
MSGSFT= 000006
MSGSKE= 000003
MSGSMB= 000015
MSGSMH= 000014
MSGSMS= 000016
MSGSTD= 000000
MSGSYS= 000012
MSGVEC= 000020
NBKMOD= 001000
NCPUOP= 000020
NDAPTY= 000002
NULL  = 000000
OV.A  = 000762
OV.CQ = 000122RG
OV.CQS= 000050 G
OV.HIC= 000362RG
OV.HIK= 000120RG
OV.HIT= 001344RG
OV.KBB= 000000RG
OV.KBS= 000122 G
OV.REM= 000002
OV.TQ = 000364RG
OV.TQE  001346R
OV.TQS= 000076 G
OWEN  = 024020
PAERR = 000010
PARPRE= 002000
PARSTA= 000100
PASCNT= 000034
PDPLSI= 020000
PDP60 = 004000
PDP70 = 010000
PRI0  = 000000
PRI1  = 000040
PRI4  = 000200
PRI5  = 000240
PRI6  = 000300
PRI7  = 000340
PR0   = 000000
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
RANNUM= 000054
RBUFEA= 000130
RBUFPA= 000126
RBUFSZ= 000132
RBUFVA= 000124
RDSERV= 000101
RDWHMI= 000022
RELERR= 000020
RELMOD= 020000
RELTIM= 010000
RESREG= ****** G
RES1  = 000056
RES2  = 000060
RICHAR= 031060
RPTDAT= 002000
RSTRT = 000112
RUBOUT= 000177
RUNMOD= 100000
R5VALU= 001740
SAM   = 075464
SAVREG= ****** G
SBADR = 000102
SBKMOD= 000000
SBKSEL= 010000
SC.ADR= 000006
SC.ALC= 000014
SC.APC= 000016
SC.CKL= 000002
SC.CKP= 000004
SC.CLO= 000000
SC.HLD= 000010
SC.SCA= 000012
SENDLS= 177777
SIZPOL  000000RG
SOFCNT= 000042
SOFPAS= 000046
SPACE = 000040
SPOINT= 000032
```

```
SPVALU= 002200
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
SS.IOP= 177600
SS.PCL= 172136
SS.PCS= 172100
SS.RFS= 160000
SS.0KC= 004000
SS.1KP= 000040
SS.120= 120000
SS.28 = 000034
STAT  = 000026
STATBI= 064757
STAT1 = 000027
SUSPND= 000001
SVR0  = 000062
SVR1  = 000064
SVR2  = 000066
SVR3  = 000070
SVR4  = 000072
SVR5  = 000074
SVR6  = 000076
SYSCNT= 000052
SYSERR= 000100

TMPIO = 000002
TQOVF = 000002
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
WASADR= 000104
WBSTAT= 000040
WBUFEA= 000136
WBUFPA= 000134
WBUFRQ= 000140
WBUFSZ= 000142
WDFR  = 000116
WDTO  = 000114

WTINRE= 000352
WTWHMI= 000222
XFLAG = 000005
XOFF  = 000023
XON   = 000021
$BGNLE= 177777
$ERFLG= 000400
$F$AND= 000310
$F$BAD= 000401
$F$BLA= 000170
$F$CAS= 000150
$F$DEC= 000220
$F$DO = 000340
$F$FAL= 000405
$F$GOO= 000400
$F$IF = 000110
$F$INC= 000210
$F$LOO= 000200
$F$NAM= 000160
$F$NO = 000403
$F$OR = 000320
$F$RTI= 000350
$F$RTN= 000300
$F$SEL= 000140
$F$THE= 000330
$F$TRU= 000404

$F$UNT= 000130
$F$WHI= 000120
$F$YES= 000402
$IFLEV= 177777
$ISK0 = 000001
$ISK1 = 000001
$ISK2 = 000001
$LOCTA= 177777
$LSTIN= 000001
$LSTTA= 000001
$NESTL= 177777
$NSK0 = 000300
$NSK1 = 000120
$NSK2 = 000110
$NSK3 = 000110
$NSK4 = 000110
$NSK5 = 000110
$SAVLE= 177777
$SSK0 = 050052
$TAGLE= 177777
$TAGNU= 050055
$TEMP = 000300
$TSK0 = 050051
$TSK1 = 050052
$TSK2 = 050054
$TSK3 = 050046

$TSK4 = 050047
$TSK5 = 050050
$$ARGC= 000002
$$BYTE= 000403
$$CASE= 000000
$$DST = 000000
$$ELOC= 000402
$$ERFL= 000000
$$FLAG= 000001
$$FROM= 000000
$$LOC = 001314R
$$LOCN= 000000
$$REG = 177777
$$RETU= 000000
$$RTN1= 050000
$$RTN2= 050001
$$SRC = 000000
$$TGSV= 000000
$$TGS1= 000000
$$TGS2= 000000
$$TO  = 000000
$$$TAG= 050000
.     = 001346R

. ABS.  000000     000
        001346     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKZ:SIZPLE,DSKZ:SIZPLE=SPMAC/ML,EQUATE,SIZPLE
 RUN-TIME: 30 21 .4 SECONDS
 RUN-TIME RATIO: 257/53=4.8
 CORE USED:  14K  (27 PAGES)
```

```
 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


                         * * * X E R S P L   R u n   L o g * * *

12:27:16 LPDAT  [XERLSJ  XERSPL version 102(2263)/3(61) running on MTA061, 25-Sep-78 12:27:16]
12:27:16 LPDAT  [XERSJS  Starting Job XMONA0, Seq #4205, request created at 25-Sep-78 11:13:24]
12:27:17 LPMSG  [XERSTF  Starting File DSKZ:XMONA0.SEQ<057>[400,3341]]
12:28:44 LPMSG  [XERFPF  Finished Printing File DSKZ:XMONA0.SEQ<057>[400,3341]]
12:28:45 LPSUM  Spooler runtime 39 Seconds, 437 KCS, 4010 disk reads, 1194 pages printed


/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**


/TO:ML21-4:CARPEN -- Distribution to ML21-4, slot 133


 **END** User CARPENTER,SA [400,4372]  Job  XMONA0 Seq. 4205 Date 25-Sep-78 12:28:45 Monitor IPC-F 603 [6B7] **END**
```