



IDENTIFICATION

SEQ 0001

PRODUCT CODE: AC-F080A-MC  
PRODUCT NAME: CXMONA0 DEC/X11 MONITOR LIBRARY  
PRODUCT DATE: SEPTEMBER 1978  
MAINTAINER: DEC/X11 Support Group

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

copyright (c) 1973,1978 digital equipment corporation





.MAIN. MACY11 30A(1052) 20-SEP-78 17:26

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0003

3	COMMON EQUATE MODULE
310	MONITOR TYPE CODE EQUATES
523	MONITOR TABLE
564	MODULE LIST WITH MONITOR TYPE CODES

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521

```
.TITLE MNSPEC - MONITOR SPECIFICATION FILE  
.SBTTL MONITOR TYPE CODE EQUATES  
A      =      000001      ;SMALLEST MONITOR  
B      =      000002      ;NO MEMORY MANAGEMENT  
C      =      000004      ;MEMORY MANAGEMENT  
D      =      000010      ;11/60  
E      =      000020      ;11/70  
F      =      000040      ;B WITH APT  
G      =      000100      ;C WITH APT  
H      =      000200      ;D WITH APT  
I      =      000400      ;E WITH APT
```

523  
524  
525 000000' 000101  
526  
527 000002' 000001  
528  
529 000004' 000102  
530  
531 000006' 000002  
532  
533 000010' 000103  
534  
535 000012' 000004  
536  
537 000014' 000104  
538  
539 000016' 000010  
540  
541 000020' 000105  
542  
543 000022' 000020  
544  
545 000024' 000106  
546  
547 000026' 000040  
548  
549 000030' 000107  
550  
551 000032' 000100  
552  
553 000034' 000110  
554  
555 000036' 000200  
556  
557 000040' 000111  
558  
559 000042' 000400  
560  
561 000044' 377  
562

.SBTTL MONITOR TABLE

.ASCIZ /A/  
.EVEN  
.WORD A  
.  
.ASCIZ /B/  
.EVEN  
.WORD B  
.  
.ASCIZ /C/  
.EVEN  
.WORD C  
.  
.ASCIZ /D/  
.EVEN  
.WORD D  
.  
.ASCIZ /E/  
.EVEN  
.WORD E  
.  
.ASCIZ /F/  
.EVEN  
.WORD F  
.  
.ASCIZ /G/  
.EVEN  
.WORD G  
.  
.ASCIZ /H/  
.EVEN  
.WORD H  
.  
.ASCIZ /I/  
.EVEN  
.WORD I  
.  
.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.
;-
564
565
566
567
568
569
570
571
572 000045' 104 041505 000130 .ASCIZ /DECX/
573 .EVEN
574 000052' 000777 .WORD A!B!C!D!E!F!G!H!I
575
576 000054' 050101 046524 042117 .ASCIZ /APTMOD/
577 000062' 000 .EVEN
578 000064' 000740 .WORD FIG!H!I
579
580
581 000066' 042502 051123 000126 .ASCIZ /BESRV/
582 .EVEN
583 000074' 000777 .WORD A!B!C!D!E!F!G!H!I
584
585 000076' 046102 043517 030066 .ASCIZ /BLOG60/
586 000104' 000 .EVEN
587 000106' 000106' .WORD DIH
588
589 000110' 046102 043517 030067 .ASCIZ /BLOG70/
590 000116' 000 .EVEN
591 000120' 000420 .WORD E!I
592
593
594 000122' 050107 000101 .ASCIZ /GPA/
595 .EVEN
596 000126' 000777 .WORD A!B!C!D!E!F!G!H!I
597
598 000130' 052113 051105 000122 .ASCIZ /KTERR/
599 .EVEN
600 000136' 000777 .WORD A!B!C!D!E!F!G!H!I
601
602 000140' 052113 042523 000124 .ASCIZ /KTSET/
603 .EVEN
604 000146' 000734 .WORD C!D!E!G!H!I
605
606 000150' 040520 042522 051122 .ASCIZ /PARERR/
607 000156' 000 .EVEN
608 000160' 000160' .WORD A!B!C!D!E!F!G!H!I
609
610 000162' 041520 040504 040524 .ASCIZ /PCDATA/
611 000170' 000 .EVEN
612 000172' 000777 .WORD A!B!C!D!E!F!G!H!I
613
614 000174' 041520 051124 041514 .ASCIZ /PCTRLC/
```

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

664	000350'	000				.EVEN	
665	000352'	000352'				.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

710						
711	000526'	047503	047516	000106	.ASCIZ	/CONOF/
712					.EVEN	
713	000534'	000734			.WORD	CIDIE!G!H!I
714						
715	000536'	050104	047505	000120	.ASCIZ	/DPEOP/
716					.EVEN	
717	000544'	000777			.WORD	A!B!C!D!E!F!G!H!I
718						
719	000546'	050104	052123	052122	.ASCIZ	/DPSTRT/
	000554'	000				
720		000556'			.EVEN	
721	000556'	000777			.WORD	A!B!C!D!E!F!G!H!I
722						
723	000560'	051104	046520	042117	.ASCIZ	/DRPMOD/
	000566'	000				
724		000570'			.EVEN	
725	000570'	000777			.WORD	A!B!C!D!E!F!G!H!I
726						
727	000572'	052504	041516	045510	.ASCIZ	/DUNCHK/
	000600'	000				
728		000602'			.EVEN	
729	000602'	000777			.WORD	A!B!C!D!E!F!G!H!I
730						
731	000604'	051105	042522	000103	.ASCIZ	/ERREC/
732					.EVEN	
733	000612'	000777			.WORD	A!B!C!D!E!F!G!H!I
734						
735	000614'	044506	046114	046115	.ASCIZ	/FILLML/
	000622'	000				
736		000624'			.EVEN	
737	000624'	000001			.WORD	A
738						
739	000626'	044506	046114	051515	.ASCIZ	/FILLMS/
	000634'	000				
740		000636'			.EVEN	
741	000636'	000776			.WORD	B!C!D!E!F!G!H!I
742						
743	000640'	042507	050124	053523	.ASCIZ	/GETPSW/
	000646'	000				
744		000650'			.EVEN	
745	000650'	000777			.WORD	A!B!C!D!E!F!G!H!I
746	000652'	051110	040504	051104	.ASCIZ	/HRDADR/
	000660'	000				
747		000662'			.EVEN	
748	000662'	000777			.WORD	A!B!C!D!E!F!G!H!I
749						
750	000664'	052110	040522	000120	.ASCIZ	/HTRAP/
751					.EVEN	
752	000672'	000777			.WORD	A!B!C!D!E!F!G!H!I
753						
754	000674'	041511	051123	047517	.ASCIZ	/ICSROO/
	000702'	000				
755		000704'			.EVEN	
756	000704'	000420			.WORD	E!I
757						



758	000706'	041511	051123	055123	.ASCIZ	/ICRSRZ/
	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'	040516	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'	042520	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/

856	001250'	000				.EVEN
857	001252'	001252'				.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/

304	001430'	000				.EVEN	
905	001432'	001432'				.WORD	A!B!C!D!E!F!G!H!I
906							
907	001434'	042522	041514	046124		.ASCIZ	/RELCTL/
	001442'	000					
908		001444'				.EVEN	
909	001444'	000734				.WORD	C!D!E!G!H!I
910							
911	001446'	047522	047516	000106		.ASCIZ	/RONOF/
912						.EVEN	
913	001454'	000777				.WORD	A!B!C!D!E!F!G!H!I
914							
915	001456'	040523	051126	051505		.ASCIZ	/SAVRES/
	001464'	000					
916		001466'				.EVEN	
917	001466'	000777				.WORD	A!B!C!D!E!F!G!H!I
918							
919	001470'	042523	042114	051505		.ASCIZ	/SELDES/
	001476'	000					
920		001500'				.EVEN	
921	001500'	000777				.WORD	A!B!C!D!E!F!G!H!I
922							
923	001502'	052123	040522	000120		.ASCIZ	/STRAP/
924						.EVEN	
925	001510'	000777				.WORD	A!B!C!D!E!F!G!H!I
926							
927	001512'	054523	041523	045514		.ASCIZ	/SYSCLK/
	001520'	000					
928		001522'				.EVEN	
929	001522'	000776				.WORD	B!C!D!E!F!G!H!I
930							
931	001524'	054524	050520	042525		.ASCIZ	/TYPQUE/
	001532'	000					
932		001534'				.EVEN	
933	001534'	000777				.WORD	A!B!C!D!E!F!G!H!I
934							
935	001536'	047125	046511	050101		.ASCIZ	/UNIMAP/
	001544'	000					
936		001546'				.EVEN	
937	001546'	000420				.WORD	E!I
938							
939	001550'	047125	050111	000101		.ASCIZ	/UNIPA/
940						.EVEN	
941	001556'	000420				.WORD	E!I
942							
943	001560'	041127	046106	046511		.ASCIZ	/WBFLIM/
	001566'	000					
944		001570'				.EVEN	
945	001570'	000777				.WORD	A!B!C!D!E!F!G!H!I
946							
947	001572'	051527	041124	051525		.ASCIZ	/WSTBUS/
	001600'	000					
948		001602'				.EVEN	
949	001602'	000777				.WORD	A!B!C!D!E!F!G!H!I
950							

```
951 ;+
952 ; THE FOLLOWING MODULES ARE LISTED LAST BECAUSE THEY MUST NOT
953 ; LIE WITHIN THE BOTTOM 4K OF THE EXERCISER.
954 ;-
955
956 001604' 041113 051104 000126 .ASCIZ /KDRV/
957 .EVEN
958 001612' 000777 .WORD A!B!C!D!E!F!G!H!I
959
960 001614' 050114 051104 000126 .ASCIZ /LPDRV/
961 .EVEN
962 001622' 000776 .WORD B!C!D!E!F!G!H!I
963
964 001624' 052124 051104 000126 .ASCIZ /TTDRV/
965 .EVEN
966 001632' 000777 .WORD A!B!C!D!E!F!G!H!I
967
968 001634' 044523 050132 040514 .ASCIZ /SIZPLA/
001642' 000
969 001644' 001644' .EVEN
970 001644' 000001 .WORD A
971
972 001646' 044523 050132 041114 .ASCIZ /SIZPLB/
001654' 000
973 001656' 001656' .EVEN
974 001656' 000042 .WORD B!F
975
976 001660' 044523 050132 041514 .ASCIZ /SIZPLC/
001666' 000
977 001670' 001670' .EVEN
978 001670' 000104 .WORD C!G
979
980 001672' 044523 050132 042114 .ASCIZ /SIZPLD/
001700' 000
981 001702' 001702' .EVEN
982 001702' 000210 .WORD D!H
983
984 001704' 044523 050132 042514 .ASCIZ /SIZPLE/
001712' 000
985 001714' 001714' .EVEN
986 001714' 000420 .WORD E!I
987
988 001716' 377 .BYTE -1
989 000001 .END
```

A = 000001	CSRA = 000100	ENBNUL= 000001	LPSTAT= 000001	PSW = 177776
ACSR = 000102	CSRC = 000102	ENDLST= 000003	MAPSTA= 000200	RANNUM= 000054
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	MED = 076600	RBUFEA= 000130
ADDR22= 001000	CTRL0 = 000017	ERRTYP= 000103	MEMPAS= 040000	RBUFPA= 000126
ADR = 000005	CTRLU = 000025	EVNTBE= 000200	MODEXH= 000000	RBUFSZ= 000132
APTFER= 000004	D = 000010	EVNTHD= 000200	MODHOL= 002000	RBUFVA= 000124
APTPRE= 000200	DCEVNT= 000011	EVNTKT= 000203	MODSEL= 000000	RDSERV= 000101
ASB = 000106	DEFRTN= 000400	EVNTPE= 000202	MSGCKD= 000010	RDWHMI= 000022
ASTAT = 000104	DROPMJ= 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= 031060
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	MSGPDP= 000002	RSVALU= 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	INHRRR= 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.CPK= 000004
BIT13 = 020000	DT.MTI= 000110	IOMODP= 102000	MSGVEC= 000020	SC.CLD= 000000
BIT14 = 040000	DT.OFF= 000070	IOMODR= 112000	NBKMOD= 000000	SC.HLD= 000010
BIT15 = 100000	DT.PAS= 000074	IOMODX= 110000	NCPUOP= 000020	SC.SCA= 000012
BIT2 = 000004	DT.PC = 000002	JACK = 035060	NDAPTY= 000002	SENDLS= 177777
BIT3 = 000010	DT.PFL= 000062	KIPAR0= 172340	NULL = 000000	SOFCNT= 000042
BIT4 = 000020	DT.PSW= 000004	KIPAR1= 172342	OWEN = 020020	SOFPAS= 000046
BIT5 = 000040	DT.PTA= 000064	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT6 = 000100	DT.RCS= 000102	KIPAR3= 172346	PAPRE= 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= 000105	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
CDWDCY= 000144	DT.WLL= 000052	KIPDR7= 172316	PRO = 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
IMPIO = 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)

3	COMMON EQUATE MODULE
531	DATABASE - MODULE HEADER
573	DATABASE - TRAP AND VECTOR GLOBAL DEFINITIONS
594	DATABASE - STUFF TRAP AND VECTOR AREAS
566	DATABASE - DATA TABLE GLOBAL DEFINITIONS
572	DATABASE - DATA TABLE VALUES GLOBAL DEFINITIONS
678	DATABASE - DATA TABLE ALLOCATION
727	DATABASE - PARITY TABLE -- PARTAB -- GLOBAL DEFINITION
734	DATABASE - PARITY TABLE -- PARTAB -- ALLOCATION
745	DATABASE - SYSTEM CLOCK TABLE -- SCTAB -- GLOBAL DEFINITIONS
763	DATABASE - SYSTEM CLOCK TABLE AND OFFSET ADDRESSES
785	DATABASE - DEVICE REGISTER GLOBAL DEFINITIONS
808	DATABASE - DEVICE REGISTER DEFINITIONS
819	DATABASE - CPU RELATED REGISTERS GLOBAL DEFINITIONS
843	DATABASE - CPU RELATED REGISTER DEFINITIONS
867	DATABASE - MISCELLANEOUS STORAGE GLOBAL DEFINITIONS
885	DATABASE - MISCELLANEOUS STORAGE
890	KERNEL - MODULE HEADER
947	KERNEL - COMMON DEFINITIONS AND REFERENCES
952	001236 .PRINT ;SPMAC: VERSION 1.1
1059	KERNEL - START UP ROUTINE
1222	KERNEL - RESTART ROUTINE
1285	KERNEL - INITIALIZATION ROUTINE
1381	KERNEL - CAPTURE EVENT: DISPATCH TO SERVICE ROUTINE
1467	KERNEL - CAPTURE EVENT: RETURN FROM SERVICING EVENT
1537	KERNEL - DE-QUEUE NEXT MODULE IN CONTROL QUEUE ROUTINE
1723	KERNEL - DE-QUEUE NEXT MODULE: PASS CPU CONTROL TO MODULE



```
508 .TITLE DECX-11 CONTROL MCDULE
509 .IDENT /V0.0/
510
511
512
513
514
515 ;++
516 ; MODULE NAME:
517 ;     DECX-11
518 ;
519 ; MODULE DESCRIPTION:
520 ;     THIS MODULE IS THE DEC/X-11 MONITOR CONTROL MODULE AND CONTAINS
521 ;     BOTH THE MONITOR'S DATABASE AND KERNEL ROUTINES
522 ;
523 ;
524 ;
525 ; VERSION:
526 ;
527 ;     EDIT           DATE           BY           REASON
528 ;
529 ;--
```

```
531 .SBTTL DATABASE - MODULE HEADER
532
533
534 ;++
535 ; MODULE NAME:
536 ; MONITOR DATABASE AREA
537 ;
538 ; FUNCTIONAL DESCRIPTION:
539 ; THE MONITOR'S DATABASE CONTAINS ALL OF THE PARAMETERS AND VALUES
540 ; THAT ARE USED BY BOTH THE MONITOR'S KERNEL ROUTINE AND THE
541 ; SUBORDINATE MONITOR ROUTINES.
542 ;
543 ; INPUTS:
544 ; NONE
545 ;
546 ; IMPLICIT INPUTS:
547 ; NONE
548 ;
549 ; OUTPUTS:
550 ; NONE
551 ;
552 ; IMPLICIT OUTPUTS:
553 ; NONE
554 ;
555 ; PATHOLOGICAL CONNECTIONS:
556 ; NONE
557 ;
558 ; SUBORDINATE ROUTINES CALLED:
559 ; NONE
560 ;
561 ; FUNCTIONAL SIDE EFFECTS:
562 ; NONE
563 ;
564 ; CALLING SEQUENCE:
565 ; NONE
566 ;
567 ; VERSION:
568 ; 0.0
569 ;
570 ; EDIT BY DATE REASON
571 ;--
```

```
573          .SBTTL DATABASE - TRAP AND VECTOR GLOBAL DEFINITIONS
574
575
576
577
578          .GLOBL HTBUSS
579          .GLOBL HTREIN
580          .GLOBL RPIRQ
581          .GLOBL PWRFL
582          .GLOBL PWRDOWN
583          .GLOBL PBDVEC
584          .GLOBL STINT
585          .GLOBL KBINT
586          .GLOBL TTINT
587          .GLOBL MEMPAR
588          .GLOBL MEMGMT
589          .GLOBL AP.PB
590          .GLOBL $ENDAD
591          .GLOBL CAST
592
```

```

594 .SBTTL DATABASE - STUFF TRAP AND VECTOR AREAS
595
596
597
598         000000         .ASECT
599         000000         .=0           ;START THIS AT ABSOLUTE 0
600
601
602
603 ;+
604 ; LOAD TRAP AND VECTOR AREA
605 ;-
606 000000 105000         .WORD 105000         ;(RESERVED)
607 000002 000000         .WORD 0
608 000004 000000G      TIMEOUT: .WORD HTBUSS         ;TIME OUT AND OTHER ERRORS
609 000006 000340         .WORD PRI7
610 000010 000000G      RESERV:  .WORD HTREIN         ;RESERVED INSTRUCTION VECTOR
611 000012 000340         .WORD PRI7
612 000014 000016         .+2           ;HALT IF T BIT SETS
613 000016 000000         HALT
614 000020 000000G      IOTINS:  .WORD RPIRQ         ;IOT INSTRUCTION VECTOR
615 000022 000340         .WORD PRI7
616 000024 000000G      PWRFL:   .WORD PWRDOWN        ;POWER FAILURE VECTOR
617 000026 000340         .WORD PRI7
618 000030 000000G      EMTINS:  .WORD PBDVEC         ;EMT INSTRUCTION VECTOR
619 000032 000000         .WORD PRI0
620 000034 000000G      TRPINS:  .WORD STINT         ;TRAP INSTRUCTION
621 000036 000000         .WORD PRI0
622 000040 000000         .WORD 0           ;40
623 000042 000000         .WORD 0           ;42
624 000044 000000G      APTVEC:  .WORD AP.PB         ;44 - APT PARAMETER BLOCK
625 000046 000000G         .WORD $ENDAD         ;46
626 000050 000000         .WORD 0           ;50
627 000052 000000         .WORD 0           ;52
628 000054 000000G         .WORD CAST         ;54
629 000056 000001         .WORD 1           ;56
630
631         .=60
632 000060 000000G      KBVEC:   .WORD KBINT         ;KEYBOARD INTERRUPT VECTOR
633 000062 000040         .WORD PRI1
634 000064 000000G      TTVEC:   .WORD TTINT         ;TERMINAL INTERRUPT VECTOR
635 000066 000040         .WORD PRI1
636
637
638 ;+
639 ; 70 TO 176 AND 204 TO 776 HAVE .+2 AND HALT. THESE VECTOR LOCATIONS
640 ; MAY BE STUFFED BY CERTAIN MONITORS WITH THE ADDRESS OF THE BAD
641 ; VECTOR SERVICE ROUTINE. THE PARITY AND KT VECTORS (114 & 250) MAY
642 ; ALSO BE STUFFED WITH PARITY AND KT TRAP ROUTINES.
643 ;-
644
645         000022         .REPT ^D<18>
646         .NLIST
647         .+2
648         HALT
649         .LIST

```

```
650 .ENDR
651
652 . =200
653 000200 000167 001774 START: JMP DX.STRT ;GO TO START ADDRESS
654
655 000137 .REPT ^D<95>
656
657 .NLIST
658 .+2
659 HALT
660 .LIST
661 .ENDR
662
663 001000 . =1000
664 001000 000167 001174 RESTART: JMP DX.STRT
```

666  
667  
668  
669  
670

.SBTTL DATABASE - DATA TABLE GLOBAL DEFINITIONS

.GLOBL DTABLE

672 .SBTTL DATABASE - DATA TABLE VALUES GLOBAL DEFINITIONS  
673  
674  
675 .GLOBL OV.KBBUF ;ADDRESS OF KEYBOARD INPUT BUFFER  
676 .GLOBL AC.MODQ ;MODULE-QUEUE-LIST ADDRESS

.SBTTL DATABASE - DATA TABLE ALLOCATION

```

678
679
680
681          ;+
682          ; DATA TABLE ALLOCATION
683          ; -
684
685
686
687          DTABLE:
688          001004 000000          .WORD 0          ;DT.EVNT - EVENT CODE
689          001006 000000          .WORD 0          ;DT.PC - PC+ OF TRAP
690          001010 000000          .WORD 0          ;DT.PSW - PSW AT TIME OF TRAP
691          001012 000000          .WORD 0          ;DT.SP - STACK POINTER AT TIME OF TRAP
692          001014 000000          .WORD 0          ;DT.ST0 - STATUS INDICATOR 0
693          001016 000000          .WORD 0          ;DT.ST1 - STATUS INDICATOR 1
694          001020 000000          .WORD 0          ;DT.CF0 - CONFIGURATION WORD 0
695          001022 000000          .WORD 0          ;DT.CF1 - CONFIGURATION WORD 1
696          001024 000000          .WORD 0          ;DT.ERR - ERROR WORD
697          001026 000000          .WORD 0          ;DT.KBRSP - ADDRESS OF KEYBOARD RESPONSE BUFFER
698          001030 000000          .WORD 0          ;DT.KBECH - ADDRESS OF KEYBOARD ECHO BUFFER
699          001032 000000          .WORD 0          ;DT.KBPRM - ADDRESS OF KEYBOARD PROMPT
700          001034 000000G          .WORD OV.KBBUFF ;DT.KBUF - ADDRESS OF KEYBOARD BUFFER POINTER
701          001036 000000G          .WORD AC.MODQ   ;DT.MLST - ADDRESS OF MODULE LIST
702          001040 000000          .WORD 0          ;DT.BLST - ADDRESS OF MODULE LIST
703          001042 004             .BYTE 4          ;DT.FCNT - FILLER COUNT (# OF NULLS)
704          001043 015             .BYTE <CR>      ;DT.FCHAR - CHARACTER AFTER WHICH NULLS ARE O/P
705          001044 000200          .WORD 200       ;DT.REL - RELOCATION CONSTANT
706          001046 000200          .WORD 200       ;DT.ADDR - CURRENT ADDRESS
707          001050 000000          .WORD 0          ;DT.ESIZ - EXERCISER SIZE
708          001052 000000          .WORD 0          ;DT.SSIZ - SYSTEM SIZE IN PAR FORMAT
709          001054 000000          .WORD 0          ;DT.WBUF - ADDR OF WRITE BUFFER
710          001056 000000          .WORD 0          ;DT.WLLMT - WRITE BUFFER LOW LIMIT
711          001060 000000          .WORD 0          ;DT.WHLMT - WRITE BUFFER HIGH LIMIT
712          001062 000000          .WORD 0          ;DT.SWR - SOFTWARE SWITCH REG(FRONT PANEL MEANS NOTHING)
713          001064 000000          .WORD 0          ;DT.EXS - # OF EXERCISER SYS ERRORS
714          001066 000000          .WORD 0          ;DT.PFL - # OF POWER FAILS
715          001070 001116          .WORD PARTAB    ;DT.PTA - ADDRESS OF PARITY TABLE
716          001072 001160          .WORD SCTAB     ;DT.SCT - ADDRESS OF CLOCK TABLE
717          001074 000000          .WORD 0          ;DT.OFFSET - OFFSET INTO CURRENT 124K BANK
718          001076 001100          .WORD DTABLE+DT.PAS ;DT.SYP - SYSTEM END OF PASS COUNT POINTER
719          001100 000000          .WORD 0          ;DT.PAS - SYSTEM END OF PASS COUNT IF NOT UNDER APT
720          001102 001104          .WORD DTABLE+DT.AP ;DT.APK - KEEP ALIVE COUNTER POINTER
721          001104 000000          .WORD 0          ;DT.AP - KEEP ALIVE IF NOT UNDER APT
722          001106 000000          .WORD 0          ;DT.RCS - FAKE KB ADDRESS FOR APT USE
723          001110 000024          .WORD ^D20      ;DT.HMX - MAX # OF HARD ERRORS ALLOWED BY OPTION MODULE
724          001112 000050          .WORD ^D40      ;DT.SMX - MAX # OF SOFT ERRORS ALLOWED BY OPTION MODULE
725          001114 001604          .WORD ^D900     ;DT.MTIME - MAX END OF PASS TIME(15 MIN.) BY A MODULE

```



```
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
```

```
745          .SBTTL DATABASE - SYSTEM CLOCK TABLE -- SCTAB -- GLOBAL DEFINITIONS
746
747
748          ;+
749          ; SYSTEM CLOCK TABLE GLOBAL DEFINITIONS
750          ; -
751
752          .GLOBL SCTAB
753          .GLOBL CLOCK
754          .GLOBL CLOCKL
755          .GLOBL CLOCKP
756
757          ;+
758          ; GLOBALS REFERENCED BY THIS DATABASE
759          ; -
760
761          .GLOBL KW11L          ;POINTS TO THE HEADER ADDRESS OF KW11-L
762                                ;OPTION MODULE
763          .GLOBL KW11P          ;POINTS TO THE HEADER ADDRESS OF KW11-P
764                                ;OPTION MODULE
765          .GLOBL LCLEAR          ;INITIALIZE MODULE TIME TABLE IN KW11-L
766          .GLOBL PCLEAR          ;INITIALIZE MODULE TIME TABLE IN KW11-P
```

```
768 .SBTTL DATABASE - SYSTEM CLOCK TABLE AND OFFSET ADDRESSES
769
770
771
772 ;+
773 ; SYSTEM CLOCK TABLE ALLOCATION
774 ;-
775
776 SCTAB:
777 001160 000000 CLOCK: .WORD 0 ;SC.CLO
778 001162 000000G CLOCKL: .WORD KW11L ;SC.CKL
779 001164 000000G CLOCKP: .WORD KW11P ;SC.CKP
780 001166 000000 .WORD 0 ;SC.ADR
781 001170 000000 .WORD 0 ;SC.HLD
782 001172 000000 .WORD 0 ;SC.SCA
783 001174 000000G .WORD LCLEAR ;SC.ALC
784 001176 000000G .WORD PCLEAR ;SC.APC
```

783 .SBTTL DATABASE - DEVICE REGISTER GLOBAL DEFINITIONS  
787  
783  
783

790 .GLOBL RCSR  
791 .GLOBL RBUF  
792 .GLOBL XCSR  
793 .GLOBL XBUF  
794 .GLOBL LPCSR  
795 .GLOBL LPBUF  
793  
797  
798  
799

800

801

802

803

804

805

806

807

808

809

810

.SBTTL DATABASE - DEVICE REGISTER DEFINITIONS

811	001200	177560	RCSR: .WORD	177560	;DL-11 RECEIVER CONTROL AND STATUS REGISTER
812	001202	177562	RBUF: .WORD	177562	;DL-11 RECEIVER DATA BUFFER REGISTER
813	001204	177564	XCSR: .WORD	177564	;DL-11 TRANSMITTER CONTROL AND STATUS REGISTER
814	001206	177566	XBUF: .WORD	177566	;DL-11 TRANSMITTER DATA BUFFER REGISTER
815	001210	177514	LPCSR: .WORD	177514	;LP 11 CONTROL STATUS REGISTER
816	001212	177516	LPBUF: .WORD	177516	;LP 11 BUFFER REGISTER
817					

```
819          .SBTTL DATABASE - CPU RELATED REGISTERS GLOBAL DEFINITIONS
820
821
822
823
824          ;+
825          ;11/70
826          ;-
827
828          .GLOBL CPULAR
829          .GLOBL CPUHAR
830          .GLOBL CPUMER
831          .GLOBL CPUMNT
832          .GLOBL CPUCPE
833          .GLOBL CPULSZ
834
835          ;+
836          ; CACHE CONTROL REGISTER AND TEMPORARY STORAGE
837          ;-
838
839          .GLOBL CCNTRL
840          .GLOBL KONTRL
841
842
843          .SBTTL DATABASE - CPU RELATED REGISTER DEFINITIONS
844
845
846
847
848          ;+
849          ; 11/70
850          ;-
851
852 001214 177740 CPULAR: .WORD 177740 ;LOW ERROR ADDRESS REGISTER
853 001216 177742 CPUHAR: .WORD 177742 ;HIGH ERROR ADDRESS REGISTER
854 001220 177744 CPUMER: .WORD 177744 ;MEMORY SYSTEM ERROR REGISTER
855 001222 177750 CPUMNT: .WORD 177750 ;MAINTENANCE REGISTER
856 001224 177766 CPUCPE: .WORD 177766 ;CPU ERROR REGISTER
857 001226 177760 CPULSZ: .WORD 177760 ;LOW SIZE REGISTER
858
859          ;+
860          ; CACHE CONTROL REGISTER AND TEMPORARY STORAGE
861          ;-
862
863
864 001230 177746 CCNTRL: .WORD 177746 ;CACHE CONTROL REGISTER
865 001232 000000 KONTRL: .WORD 0 ;TEMPORARY STORAGE FOR CACHE OPERATIONS
```

867 .SBTTL DATABASE - MISCELLANEOUS STORAGE GLOBAL DEFINITIONS

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888 001234 000000

.GLOBL CTRL0F

.SBTTL DATABASE - MISCELLANEOUS STORAGE

CTRL0F: .WORD 0 ;ENABLE/DISABLE TERMINAL OUTPUT FLAG

```
890 .SBTTL KERNEL - MODULE HEADER
891
892
893 ;++
894 ; MODULE NAME:
895 ; KERNEL ROUTINE
896 ;
897 ; FUNCTIONAL DESCRIPTION:
898 ; THIS ROUTINE CONTROLS THE EXECUTION OF THE DEC/X-11 SYSTEM
899 ; EXERCISER, I.E., IT IS RESPONSIBLE FOR STARTING THE EXERCISER
900 ; WHEN FIRST LOADED INTO MEMORY, INITIALIZING THE ACTUAL RUNNING
901 ; OF IT, COORDINATING THE RECEIVING AND SERVICING OF THE REAL TIME
902 ; EVENTS AND PERFORMING SCHEDULING AND ACTIVATING FUNCTIONS OF THE
903 ; EXERCISER.
904 ; THIS IS ACCOMPLISHED WITH THE USE OF THE FOLLOWING ROUTINES:
905 ;
906 ; DX.STRT THIS ROUTINE PERFORMS THE DUTIES REQUIRED
907 ; TO GET THE EXERCISER GOING WHEN FIRST
908 ; LOADED INTO MEMORY
909 ; DX.RSTRT THIS ROUTINE PERFORMS THOSE FUNCTIONS NEC-
910 ; ESSARY TO GET THE EXERCISE RESTARTED
911 ; AFTER IT HAS BEEN RUNNING
912 ; DX.INI THIS ROUTINE IS RESPONSIBLE FOR INITIAL-
913 ; IZATING THE EXERCISER
914 ; DX.CAPTURE THIS ROUTINE COORDINATES THE RECEIVING
915 ; OF THE EXERCISER'S REALTIME EVENTS AND
916 ; THE DISPATCHING OF CONTROL TO THE AP-
917 ; PROPRIATE SERVICE ROUTINES
918 ; DX.DEQUEUE THIS ROUTINE PERFORMS THE DE-QUEUEING AND
919 ; ACTIVATING FUNCTIONS OF THE MONITOR
920
921 ; INPUTS:
922 ; NONE
923 ;
924 ; IMPLICIT INPUTS:
925 ; NONE
926 ;
927 ; OUTPUTS:
928 ; NONE
929 ;
930 ; IMPLICIT OUTPUTS:
931 ; NONE
932 ;
933 ; PATHOLOGICAL CONNECTIONS:
934 ; MANY
935 ;
936 ; SUBORDINATE ROUTINES CALLED:
937 ; 1.
938 ;
939 ; FUNCTIONAL SIDE EFFECTS:
940 ;
941 ; VERSION:
942 ; 0.0
943 ;
944 ; EDIT BY DATE REASON
945 ;--
```

```
947          .SBTTL  KERNEL - COMMON DEFINITIONS AND REFERENCES
948
949
950
951          .MCALL  STRUCT
952 001236      STRUCT
(1) 001236      .PRINT ;SPMAC: VERSION 1.1
953          000001      $LSTIN = 1
954          000001      $LSTTAG = 1
955
956
957
958
959          ;
960          ;*****
961          ;
962          ; REFERENCED BY OTHER MODULES
963          ;
964          .GLOBL  DX.CAP          ;CAPTURE EVENT ROUTINE ENTRY POINT
965          .GLOBL  DX.DEQ        ;DE-QUEUING ROUTINE ENTRY POINT
966          .GLOBL  DX.RSTR       ;RESTART EXERCISER ENTRY POINT
967          .GLOBL  DX.INI        ;INITIALIZATION ROUTINE
968          .GLOBL  DX.HDR        ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S HEADER ADDRESS
969          .GLOBL  DX.RET        ;MONITOR'S NEXT-TO-EXECUTE SLOT: MODULE'S RESUME ADDRESS
970          .GLOBL  DX.MON        ;MONITOR ID
971          .GLOBL  DX.R5
972          .GLOBL  DX.KFL        ;CR FLAG
973          .GLOBL  DX.SP
974          ;
975          ;*****
976          ;
977          ; GLOBAL REFERENCES
978          ;
979          .GLOBL  ST.EXT        ;SOFTWARE TRAP HANDLER EXIT ADDRESS
980          .GLOBL  HT.EXT        ;HARD ERROR TRAP HANDLER EXIT ADDRESS
981          .GLOBL  APTSLP       ;APT SIZING AND SLEEP ROUTINE
982          .GLOBL  RSTRCY       ;RESTART RECOVERY ROUTINE
983          .GLOBL  KBINI        ;KEYBOARD DRIVER ENTRY POINT
984          .GLOBL  CMDRST       ;OUTPUT KEYBOARD PROMPT ROUTINE
985          .GLOBL  PREVT        ;PROCESS EVENT MODULE ENTRY POINT
986          .GLOBL  PWRDOW       ;POWER DOWN MODULE ENTRY POINT
987          .GLOBL  MSGDEQ       ;MESSAGE DE-QUEUING MODULE ENTRY POINT
988          .GLOBL  MSGDHOOK     ;MESSAGE HOOK MODULE ENTRY POINT
989          .GLOBL  SAVREG       ;SAVE REGISTERS ROUTINE
990          .GLOBL  RESREG       ;RESTORE REGISTERS ROUTINE
991          .GLOBL  LPINT        ;LP11 INTERRUPT ROUTINE
992          .GLOBL  CQINI        ;CONTROL QUEUE INITIALIZATOR MODULE ENTRY POINT
993          .GLOBL  TQINI        ;TYPE QUEUE INITIALIZATOR MODULE ENTRY POINT
994          .GLOBL  CHKACT       ;CHECK FOR ACTIVE MODULES
995          .GLOBL  CHKEOP       ;CHECK FOR END OF PASS TIME
996          .GLOBL  CLREOP       ;CLEAR EOP INDICATORS
997          .GLOBL  ACTIV        ;ACTIVATE NEXT OPTION MODULE ROUTINE
998          .GLOBL  AC.TPTR      ;MODULE-TYPE-LIST POINTER
999          .GLOBL  AC.TYPE      ;MODULE-TYPE-LIST
1000          .GLOBL  AC.MPTR     ;MODULE-QUEUE-LIST POINTER
1001          .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 KPN ;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 001346 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
```

```

1059          .SBTTL  KERNEL - START UP ROUTINE
1060
1061
1062          002200          .=2200
1063
1064          ;+
1065          ; THIS ROUTINE IS RESPONSIBLE FOR SETTING UP THE DIFFERENT SOFTWARE COMPONENTS
1066          ; OF THE MONITOR WHEN THE EXERCISER IS FIRST LOADED INTO MEMORY OR RESTARTED AT LOC. 100
1067          ;-
1068
1069
1070
1071 002200          DX.STRT:
1072
1073          ;+
1074          ; STUFF THE POWERFAILURE VECTOR AND SET UP THE STACKS
1075          ;-
1076
1077 002200          LET PWRFL := #PWRDOWN
1078 (4) 002200 012767 000000G 175616          MOV      #PWRDOWN,PWRFL
1079 002206          LET SP := #SPVALUE
1080 (4) 002206 012706 002200          MOV      #SPVALUE,SP
1081 002212          LET R5 := #R5VALUE
1082 (4) 002212 012705 001740          MOV      #R5VALUE,R5
1083
1084          ;+
1085          ; IF UNDER APT, WE MAY OR MAY NOT HAVE A SYSTEM TTY.  FOR NOW, ASSUME
1086          ; WE DO AND STUFF THE TTY ADDRESS.  ALSO, CLEAR THE "NO APT TTY" BIT IN
1087          ; CONFIGURATION WORD 0.
1088          ;-
1089          LET RCSR := #177560
1090 (4) 002216 012767 177560 176754          MOV      #177560,RCSR
1091 002224          LET DTABLE+DT.CF0 := DTABLE+DT.CF0 CLR.BY #NOAPTY
1092 (6) 002224 042767 000002 176566          BIC      #NOAPTY,DTABLE+D
1093
1094          INLINE <RESET>
1095 (2) 002232 000005          RESET
1096
1097          ;+
1098          ; IF FIRST TIME THROUGH DO THE FOLLOWING
1099          ;-
1100
1101          IF DTABLE+DT.ESIZ EQ #0 THEN
1102 (6) 002234 005767 176610          TST      DTABLE+DT.ESIZ
1103 (9) 002240 001022          BNE      50000$
1104
1105          ;+
1106          ; STUFF THE EXIT ADDRESS FOR THE TWO TRAP HANDLERS: SOFTWARE
1107          ; AND HARDWARE
1108          ;-
1109
1110          LET ST.EXT := #DX.CAP
1111 (4) 002242 012767 003242 000000G          MOV      #DX.CAP,ST.EXT
1112 002250          LET HT.EXT := #DX.CAP
1113 (4) 002250 012767 003242 000000G          MOV      #DX.CAP,HT.EXT

```

```

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

```

```

CLR DTABLE+DT.ST0
CLR DTABLE+DT.ST1
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,SIZPOL
MOV (SP)+,R5
RESET
BR 50001$
50000$:
BIC #177400,DTABLE+D
CLR DTABLE+DT.ST1
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,RSTRCY
MOV (SP)+,R5
CMP DTABLE+DT.ADDR,#
BEQ 50002$
MOV R5,-(SP)
MOV #200,-(R5)
MOV #DTABLE,-(R5)
JSR PC,PRRLOC
MOV (SP)+,R5
50002$:
50001$:

```

```

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
(9) 002370 001407
1143          CALL APTS LP IN <#DTABLE>
(3) 002372 010546
(4) 002374 012745 001004
(3) 002400 004767 000000G
(3) 002404 012605
1144          ELSE
(4) 002406 000403
(3) 002410
1145          LET DTABLE+DT.CFO := DTABLE+DT.CFO CLR.BY #APTPRES
(6) 002410 042767 000200 176402
1146          ENDF
(4) 002416
1147
1148          ;+
1149          ; LOWER THE PRIORITY
1150          ; -
1151
1152 002416          LET -(SP) := #0
(4) 002416 005046
1153 002420          LET -(SP) := #7$
(4) 002420 012746 002426
1154 002424          INLINE <RTI>
(2) 002424 000002
1155 002426          INLINE <7$:>
(2) 002426
1156
1157          ;+
1158          ; OUTPUT MESSAGE TO IDENTIFY SELF
1159          ; -
1160
1161 002426          CALL MSGDHOOK IN <#DTABLE,#MSGPOP,#DX.TITLE,#2$>
(3) 002426 010546
(7) 002430 012745 002462
(6) 002434 012745 001252
(5) 002440 012745 000002
(4) 002444 012745 001004
(3) 002450 004767 000000G
(3) 002454 012605
1162          INLINE <1$:>
(2) 002456
1163          INLINE <NOP>
(2) 002456 000240
1164          INLINE <BR 1$>
(2) 002460 000776
1165          INLINE <2$:>
(2) 002462
1166
1167          ;+
    
```

```

TST @#44
BEQ 50003$
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,APTS LP
MOV (SP)+,R5
BR 50004$
50003$:
BIC #APTPRES,DTABLE+
50004$:
CLR -(SP)
MOV #7$,-(SP)
RTI
7$:
MOV R5,-(SP)
MOV #2$,-(R5)
MOV #DX.TITLE,-(R5)
MOV #MSGPOP,-(R5)
MOV #DTABLE,-(R5)
JSR PC,MSGDHOOK
MOV (SP)+,R5
1$:
NOP
BR 1$
2$:
    
```

```

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$:
1177
1178
1179 ; +
1180 ; CONVERT SIZE TO DECIMAL ASCII
1181 ; -
1182
1183 002504 CALL BDACNV IN <R0,#DX.SZK>
(3) 002504 010546 MOV R5,-(SP)
(5) 002506 012745 001371 MOV #DX.SZK,-(R5)
(4) 002512 010045 MOV R0,-(R5)
(3) 002514 004767 000000G JSR PC,BDACNV
(3) 002520 012605 MOV (SP)+,R5
1184
1185 ; +
1186 ; OUTPUT SYSTEM SIZE MESSAGE
1187 ; -
1188
1189 002522 CALL MSGDHOOK IN <#DTABLE,#MSGPOP,#DX.SIZ,#4$>
(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 #KTPRES,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 ENDF
(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 ENDF
(4) 002650 50010$:
1220 002650 INLINE <6$:>
(2) 002650 6$:
```

```

1222          .SBTTL  KERNEL - RESTART ROUTINE
1223
1224
1225          ;+
1226          ; THIS ROUTINE PERFORMS THOSE FUNCTIONS NEEDED TO GET THE MONITOR GOING AGAIN
1227          ; AFTER IT HAS BEEN RUNNING
1228          ; -
1229
1230 002650          DX.RSTRT: LET SP := #SPVALUE
1231  (4) 002650 012706 002200          MOV      #SPVALUE, SP
1232          LET DX.SP := SP
1233  (4) 002654 010667 176370          MOV      SP, DX.SP
1234          LET R5 := #R5VALUE
1235  (4) 002660 012705 001740          MOV      #R5VALUE, R5
1236          LET DX.R5 := R5
1237  (4) 002664 010567 176356          MOV      R5, DX.R5
1238
1239          ;+
1240          ; CLEAR THE ERROR WORD.
1241          ; -
1242
1243          LET DTABLE+DT.ERR := #0
1244  (4) 002670 005067 176130          CLR      DTABLE+DT.ERR
1245
1246          ;+
1247          ; RESET THE BACKGROUND POINTER
1248          ; -
1249
1250          LET BA.MPTR := DTABLE+DT.BLST
1251  (4) 002674 016767 176140 000000G  MOV      DTABLE+DT.BLST, B
1252
1253
1254          ;+
1255          ; GO DO THE NECESSARY MONITOR INITIALIZATION, AND HURRY BACK
1256          ; -
1257
1258          CALL DX.INI
1259  (3) 002702 004767 000112          JSR      PC, DX.INI
1260
1261
1262          ;+
1263          ; IF THIS IS INITIAL START-UP AND WE ARE IN AUTOMATIC MODE (XXDP OR ACT-11),
1264          ; THEN MOVE THEXXDP MONITOR (EVEN IF IT DOESN'T EXIST, SINCE WE DON'T KNOW
1265          ; WHETHER WE'RE UNDER XXDP OR ACT-11) TO JUST ABOVE THE
1266          ; EXERCISER. THEN FAKE A "RUN" COMMAND TO GET STARTED.
1267          ; -
1268
1269          IF #AUTOST NOTSETIN DTABLE+DT.ST1 THEN
1270  (6) 002706 032767 020000 176102          BIT      #AUTOST, DTABLE+D
1271  (9) 002714 001031          BNE      50011$
1272
1273          IF #AUTO SETIN DTABLE+DT.CFO THEN
1274  (6) 002716 032767 000010 176074          BIT      #AUTO, DTABLE+DT.
1275  (9) 002724 001406          BEQ      50012$
1276
1277          CALL DPSTRT IN <#DTABLE>
1278  (3) 002726 010546          MOV      R5, -(SP)
1279  (4) 002730 012745 001004          MOV      #DTABLE, -(R5)

```



```

(3) 002734 004767 000000G
(3) 002740 012605
1265 002742          ENDIF
(4) 002742          50012$:
1266
1267          ;+
1268          ; IF THIS IS INITIAL START-UP WE ARE UNDER APT - GET THIS THING RUNNING RIGHT AWAY !
1269          ; -
1270
1271 002742          IF #APTPRES SETIN DTABLE+DT.CFO THEN
(6) 002742 032767 000200 176050          BIT          #APTPRES,DTABLE+
(9) 002750 001410          BEQ          50013$
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          ; OUTPUT THE KEYBOARD PROMPT AND LET THE OPERATOR KNOW THAT THE
1279          ; SYSTEM IS NOW AWAITING KEYBOARD INPUT
1280          ; -
1281
1282          CALL CMDRST IN <#DTABLE>
1283 003000          MOV          R5,-(SP)
(3) 003000 010546          MOV          #DTABLE,-(R5)
(4) 003002 012745 001004          JSR          PC,CMDRST
(3) 003006 004767 000000G          MOV          (SP)+,R5
(3) 003012 012605
1284 003014          INLINE <JMP          @#DX.DEQ>
(2) 003014 000137 003634          JMP          @#DX.DEQ

```

```

1286          .SBTTL  KERNEL - INITIALIZATION ROUTINE
1287
1288
1289          ;+
1290          ; THIS ROUTINE PERFORMS THE INITIALIZATION FUNCTIONS THAT ARE REQUIRED
1291          ; TO GET THE DEC/X-11 MONITOR GOING. ONCE THE INITIALIZATION IS COMPLETE
1292          ; AND THE MONITOR IS RUNNING, CONTROL IS RETURNED TO THE CALLER
1293          ;-
1294
1295 003020          ROUTINE DX.INI
1296  (2) 003020          DX.INI:
1297 003020          CALL SAVREG
1298  (3) 003020 004767 000000G          JSR      PC,SAVREG
1299          ;+
1300          ; INITIALIZE THE QUEUE STRUCTURES - BOTH THE CONTROL QUEUE AND THE
1301          ; TYPE QUEUE
1302          ;-
1303
1304 003024          CALL CQINI
1305  (3) 003024 004767 000000G          JSR      PC,CQINI
1306 003030          CALL TQINI
1307  (3) 003030 004767 000000G          JSR      PC,TQINI
1308          ;+
1309          ; INITIALIZE THE ERROR LOGGING STUFF
1310          ;-
1311 003034          IF #PDP70!PDP60 SETIN DTABLE+DT.CFO THEN
1312  (6) 003034 032767 014000 175756          BIT      #PDP70!PDP60,DTA
1313  (9) 003042 001402          BEQ      50002$
1314          CALL IERRLO
1315  (3) 003044 004767 000000G          JSR      PC,IERRLO
1316 003050          ENDIF
1317  (4) 003050          50002$:
1318          ;+
1319          ; INITIALIZE THE OPTION MODULE SERVICING MECHANISM, WHICH CONSISTS OF RESETTING
1320          ; THE POINTERS FOR THE MODULE-TYPE-LIST AND THE MODULE-QUEUE-LIST, CLEARING THE
1321          ; RELOCATION-MODE FLAGS, RESETTING THE ASSOCIATED BITS IN THE STATUS INDICATOR WORD
1322          ; ALSO, CLEAR THE SBKSEL BIT.
1323          ;-
1324 003050          LET AC.TPTR := #AC.TYPE
1325  (4) 003050 012767 000000G 000000G          MOV      #AC.TYPE,AC.TPTR
1326 003056          LET AC.MPTR := #AC.MODQ
1327  (4) 003056 012767 000000G 000000G          MOV      #AC.MODQ,AC.MPTR
1328 003064          LET DTABLE+DT.ST1 := DTABLE+DT.ST1 SET.BY #CKTIM
1329  (6) 003064 052767 100000 175724          BIS      #CKTIM,DTABLE+DT
1330 003072          LET DTABLE+DT.ST0 := DTABLE+DT.ST0 CLR.BY #MODEXH!MODHOLD!BKMODE!TMP10
1331  (6) 003072 042767 046002 175714          BIC      #MODEXH!MODHOLD!
1332 003100          LET DTABLE+DT.ST1 := DTABLE+DT.ST1 CLR.BY #SBKSEL
1333  (6) 003100 042767 010000 175710          BIC      #SBKSEL,DTABLE+D
1334          ;+

```

```
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) := 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
1363          ;+
1367          ; IF IT IS NOT YET TIME TO RELOCATE,
1363          ; 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
```

```

1381          .SBTTL  KERNEL - CAPTURE EVENT: DISPATCH TO SERVICE ROUTINE
1382
1383
1384
1385          ;+
1386          ; THIS ROUTINE FIELDS THE PHYSICAL EVENTS THAT ARE RECEIVED BY THE DECX/11
1387          ; MONITOR, I.E., SOFTWARE GENERATED TRAPS AND HARDWARE ERROR TRAPS.
1388          ; THE TYPE OF EVENT IS DETERMINED AND THE CORRESPONDING EVENT
1389          ; CODE IS ENTERED INTO THE MONITOR'S DATA TABLE. THE LINKAGE INFORMATION
1390          ; ON THE STACK IS THEN REMOVED AND ALSO ENTERED INTO THE DATA TABLE. THEN,
1391          ; BASED ON THE TYPE OF EVENT (DIRECT-RETURN OR DEFERRED-RETURN), THE MODULE'S
1392          ; REGISTERS MAY HAVE TO BE SAVED IN THE MODULE'S HEADER. CONTROL IS THEN
1393          ; PASSED TO THE APPROPRIATE SERVICE ROUTINE.
1394          ;-
1395          003242  DX.CAP:
1396
1397          ;+
1398          ; IF THIS EVENT ORIGINATED WITH A BACKGROUND MODULE, RESET THE MONITOR'S BACKGROUND MODE
1399          ; INDICATOR, THUS ALLOWING THE EVENT TO BE SERVICED IN IT'S ENTIRETY
1400          ;-
1401
1402          003242  IF #BKMODE SETIN DTABLE+DT.STO THEN
1403          (6) 003242 032767 040000 175544          BIT      #BKMODE,DTABLE+D
1404          (9) 003250 001403          BEQ      50010$
1405          003252          LET DTABLE+DT.STO := DTABLE+DT.STO CLR.BY #BKMODE
1406          (6) 003252 042767 040000 175534          BIC      #BKMODE,DTABLE+D
1407          003260          ENDIF
1408          (4) 003260          50010$:
1409
1410          ;+
1411          ; SAVE THE INFORMATION FOR THE NEW EVENT
1412          ;-
1413          003260  POP DTABLE+DT.EVNT          MOV      (SP)+,DTABLE+DT.
1414          (2) 003260 012667 175520
1415          003264  POP DTABLE+DT.PC          MCV      (SP)+,DTABLE+DT.
1416          (2) 003264 012667 175516
1417          003270  POP DTABLE+DT.PSW          MOV      (SP)+,DTABLE+DT.
1418          (2) 003270 012667 175514
1419          003274  LET DTABLE+DT.SP := SP          MOV      SP,DTABLE+DT.SP
1420          (4) 003274 010667 175512
1421
1422          ;+
1423          ; DETERMINE IF THE EVENT ORIGINATED FROM THE MONITOR, IF SO DON'T BOTHER
1424          ; SAVING THE REGISTERS AND SWITCHING STACKS
1425          ;-
1426          003300  IF DTABLE+DT.EVNT GE #EVNTHD THEN
1427          (6) 003300 026727 175500 000200          CMP      DTABLE+DT.EVNT,#
1428          (9) 003306 002405          BLT      50011$
1429          003310          IF DTABLE+DT.SP LOS #SPVALUE THEN
1430          (6) 003310 026727 175476 002200          CMP      DTABLE+DT.SP,#SP
1431          (9) 003316 101001          BHI      50012$
1432          003320          INLINE <BR 1$>
1433          (2) 003320 000423          BR 1$

```

```

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) := 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  
1459 003416  
(4) 003416  
1460  
1461  
1462  
1463  
1464  
1465 003416  
(3) 003416 010546  
(4) 003420 012745 001004  
(3) 003424 004767 000000G  
(3) 003430 012605

ENDIF  
  
;+  
; NOW GO SERVICE THE EVENT  
;-  
  
CALL PREVT IN <#DTABLE>

BIC #DEFRTN,DTABLE+D  
  
50014\$:  
  
MOV R5,-(SP)  
MOV #DTABLE,-(R5)  
JSR PC,PREVT  
MOV (SP)+,R5

```

1467          .SBTTL  KERNEL - CAPTURE EVENT: RETURN FROM SERVICING EVENT
1468
1469
1470
1471          ;+
1472          ; WAS THE EVENT AN ERROR, OR DID THE SERVICING OF THE EVENT RESULT
1473          ; IN AN ERROR? IF SO PASS CONTROL TO THE ERROR RECOVERY HANDLER AND
1474          ; DON'T EXPECT TO GET CONTROL BACK
1475          ;-
1476
1477          IF DTABLE+DT.ERR NE #0 THEN
(6) 003432 005767 175366          TST      DTABLE+DT.ERR
(9) 003436 001406          BEQ      50015$
1478          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
1479          ENDIF
(4) 003454          50015$:
1480
1481          ;+
1482          ; IF THE EVENT ORIGINATED WITH A BACKGROUND MODULE, DETERMINE IF DEFERRED
1483          ; RETURN. IF SO, LOAD THE HEADER ADDRESS AND RETURN ADDRESS INTO THE BACKGROUND
1484          ; QUEUE, AND SET THE "BK" SUSPEND BIT.
1485          ; OTHERWISE, THE MODULE IS WORKING ITS WAY THROUGH THE MONITOR
1486          ; QUEUES, SO SET THE BKDEF BIT IN THE BK STATUS WORD.
1487          ; IN EITHER CASE, SET THE DEFERRED RETURN BIT IN THE STATUS INDICATOR.
1488          ;-
1489
1490          LET R0 := DX.HDR
(4) 003454 016700 175560          MOV      DX.HDR,R0
1491          IF #BIT04 SETIN STAT(R0) THEN
(6) 003460 032760 000020 000026          BIT      #BIT04,STAT(R0)
(9) 003466 001426          BEQ      50016$
1492          LET STAT(R0) := STAT(R0) CLR.BY #ACTBIT
(6) 003470 042760 004000 000026          BIC      #ACTBIT,STAT(R0)
1493          IF #DEFRTN NOTSETIN DTABLE+DT.STO THEN
(6) 003476 032767 000400 175310          BIT      #DEFRTN,DTABLE+D
(9) 003504 001011          BNE      50017$
1494          LET BA.HDR := R0
(4) 003506 010067 000000G          MOV      R0,BA.HDR
1495          LET BA.RET := DTABLE+DT.PC
(4) 003512 016767 175270 000000G          MOV      DTABLE+DT.PC,BA.
1496          LET BA.STATUS := BA.STATUS SET.BY #SUSPND
(6) 003520 052767 000001 000000G          BIS      #SUSPND,BA.STATU
1497          ELSE
(4) 003526 000403          BR      50020$
(3) 003530          50017$:
1498          LET BA.STAT := BA.STAT SET.BY #BKDEF
(6) 003530 052767 000002 000000G          BIS      #BKDEF,BA.STAT
1499          ENDIF
(4) 003536          50020$:
1500          LET DTABLE+DT.STO := DTABLE+DT.STO SET.BY #DEFRTN
(6) 003536 052767 000400 175250          BIS      #DEFRTN,DTABLE+D
1501          ENDIF
003544

```





```

1537 .SBTTL KERNEL - DE-QUEUE NEXT MODULE IN CONTROL QUEUE ROUTINE
1538
1539
1540 ;+ SCHEDULER
1541 ; THIS ROUTINE IS ENTERED ONLY WHEN NO OTHER MONITOR MODULES OR OPTION
1542 ; MODULES ARE EXECUTING. ITS MAIN FUNCTIONS ARE TO DE-QUEUE THE NEXT
1543 ; MESSAGE TO BE PRINTED, ACTIVATE THE NEXT MODULE IN THE MODULE-QUEUE-
1544 ; LIST, AND TO DE-QUEUE THE NEXT MODULE IN THE CONTROL QUEUE AND ALLOW
1545 ; IT TO BEGIN EXECUTION. IF THERE ARE NO MODULES WAITING IN THE MODULE-
1546 ; QUEUE-LIST OR IN THE CONTROL QUEUE, THIS ROUTINE SIMPLY KEEPS CONTROL
1547 ; BY SITTING IN A CLOSED LOOP.
1548 ;--
1549
1550 003634 DX.DEQ: LET DX.QFLG := #0
1551 (4) 003634 005067 175404 CLR DX.QFLG
1552 ; REPEAT
1553 ; INCREMENT THE APT "KEEP-ALIVE" COUNTER.
1554 ;--
1555
1556
1557 003640 LET @DTABLE+DT.APK := @DTABLE+DT.APK + #1
1558 (6) 003640 005277 175236 INC @DTABLE+DT.APK
1559 ;+
1560 ; CHECK THE MESSAGE QUEUE FIRST
1561 ;--
1562
1563 003644 CALL MSGDEQ IN <#DTABLE>
1564 (3) 003644 010546 MOV R5,-(SP)
1565 (4) 003646 012745 001004 MOV #DTABLE,-(R5)
1566 (3) 003652 004767 000000G JSR PC,MSGDEQ
1567 (3) 003656 012605 MOV (SP)+,R5
1568
1569 ;+
1570 ; IF THE KB FLAG IS SET - WE HAD A CR - SO GO SERVICE
1571 ;--
1572
1573 003660 IF DX.KFL EQ #1 THEN
1574 (6) 003660 026727 175352 000001 CMP DX.KFL,#1
1575 (9) 003666 001010 BNE 50022$
1576
1577 003670 CALL CMDPRC IN <#DTABLE>
1578 (3) 003670 010546 MOV R5,-(SP)
1579 (4) 003672 012745 001004 MOV #DTABLE,-(R5)
1580 (3) 003676 004767 000000G JSR PC,CMDPRC
1581 (3) 003702 012605 MOV (SP)+,R5
1582
1583 003704 LET DX.KFL := #0
1584 (4) 003704 005067 175326 CLR DX.KFL
1585
1586 003710 ENDIF
1587 (4) 003710 50022$:
1588
1589 ;+
1590 ; IF WE ARE EXECUTING IN RUN MODE, GO THRU THE SCHEDULING ALGORITHM
1591 ;--

```

```

1578
1579 003710          IF #RUNMODE NOTSETIN DTABLE+DT.STO THEN
(6) 003710 032767 100000 175076
(9) 003716 001002
1580 003720          INLINE <JMP 6$>
(2) 003720 000167 000614
1581 003724          ENDIF
(4) 003724
1582
1583
1584
1585
1586
1587 003724          IF #RELTIME SETIN DTABLE+DT.STO THEN
(6) 003724 032767 010000 175062
(9) 003732 001415
1588 003734          CALL CHKACT IN <#DTABLE>
(3) 003734 010546
(4) 003736 012745 001004
(3) 003742 004767 000000G
(3) 003746 012605
1589 003750          IF.NO.ERROR THEN
(6) 003750 103406
1590 003752          CALL RELCTL IN <#DTABLE>
(3) 003752 010546
(4) 003754 012745 001004
(3) 003760 004767 000000G
(3) 003764 012605
1591 003766          ENDIF
(4) 003766
1592 003766          ENDIF
(4) 003766
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604 003766          IF #MEMPAS SETIN DTABLE+DT.ST1 THEN
(6) 003766 032767 040000 175022
(9) 003774 001434
1605 003776          CALL CHKACT IN <#DTABLE>
(3) 003776 010546
(4) 004000 012745 001004
(3) 004004 004767 000000G
(3) 004010 012605
1606 004012          IF.NO.ERROR THEN
(6) 004012 103425
1607 004014          IF #AUTO SETIN DTABLE+DT.CFO THEN
(6) 004014 032767 000010 174776
(9) 004022 001406

```

```

BIT #RUNMODE,DTABLE+
BNE 50023$
JMP 6$
50023$:
;+
; START BY LOOKING AT THE RELOCATION STUFF. IF IT'S TIME TO RELOCATE, GO SEE
; IF ALL MODULE ACTIVE BITS ARE CLEAR. IF SO, RELOCATE.
;-
BIT #RELTIME,DTABLE+
BEQ 50024$
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,CHKACT
MOV (SP)+,R5
BCS 50025$
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,RELCTL
MOV (SP)+,R5
50025$:
50024$:
;+
; IF THE LAST SYSTEM EOP WAS IN LOWEST MEMORY (MEMPAS=1), IT IS
; TIME TO CHECK FOR A COUPLE OF THINGS.
; (1) IF WE ARE RUNNING UNDER XXDP OR ACT-11 (AUTO=1), THEN IT IS TIME
; TO RETURN THE THE XXDP/ACT-11 MONITOR.
; (2) IF THERE ARE SBK MODULES THAT ARE SELECTED, IT IS TIME TO
; STOP ALL MODULES AND THEN RESTART THEM SO THE
; SBK MODULES WILL RESTART.
;-
BIT #MEMPAS,DTABLE+D
BEQ 50026$
MOV R5,-(SP)
MOV #DTABLE,-(R5)
JSR PC,CHKACT
MOV (SP)+,R5
BCS 50027$
BIT #AUTO,DTABLE+DT.
BEQ 50030$

```

```

1608 004024                CALL DPEOP IN <#DTABLE>
(3) 004024 010546
(4) 004026 012745 001004                MOV      R5,-(SP)
(3) 004032 004767 000000G                MOV      #DTABLE,-(R5)
(3) 004036 012605                JSR      PC,DPEOP
1609 004040                MOV      (SP)+,R5
(4) 004040                ENDIF
1610 004040                IF #SBKSEL SETIN DTABLE+DT.ST1 THEN 50030$:
(6) 004040 032767 010000 174750                BIT      #SBKSEL,DTABLE+D
(9) 004046 001407                BEQ      50031$
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                ENDIF 50031$:
1615 004066                ENDIF 50027$:
(4) 004066                ENDIF 50026$:
1616 004066                ENDIF
(4) 004066
1617
1618
1619
1620 ;+
1621 ; IF ITS NOT RELOCATION TIME AND NOT TIME FOR XXDP RETURN, THEN
1622 ; IF WE'RE NOT IN A 'HOLD' STATE WAITING ON A SBK OR NBK MODULE, SEE IF WE CAN
1623 ; GET SOMETHING GOING
1624 ;-
1625 004066                IF #RELTIME NOTSETIN DTABLE+DT.ST0 AND #MEMPAS NOTSETIN DTABLE+DT.ST1 THEN
(6) 004066 032767 010000 174720                BIT      #RELTIME,DTABLE+
(9) 004074 001022                BNE      50032$
(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                ENDIF 50033$:
1627 004126                CALL ACTIV IN <#DTABLE>
(3) 004126 010546                MOV      R5,-(SP)
(4) 004130 012745 001004                MOV      #DTABLE,-(R5)
(3) 004134 004767 000000G                JSR      PC,ACTIV
(3) 004140 012605                MOV      (SP)+,R5
1628 004142                ENDIF
(4) 004142                ENDIF 50034$:
1629 004142                ENDIF 50032$:
(4) 004142
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) 004155 012567 175060          MOV      (R5)+,DX.RET
1336 004162                        IF.NO.ERROR THEN
(6) 004162 103451                        BCS     50035$
1537 004164                        LET R0 := DX.HDR
(4) 004164 016700 175050          MOV      DX.HDR,R0
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$:
1652
1653                                ;+
1654                                ; DETERMINE IF IT'S A BACKGROUND MODULE, IF SO, ENTER IT INTO
1655                                ; THE BACKGROUND QUEUE AND CONTINUE DE-QUEUEING FROM THE CONTROL
1656                                ; QUEUE
1657                                ; -
1658
1659 004236                        IF #BIT04 SETIN STAT(R0) THEN
(6) 004236 032760 000020 000026    BIT     #BIT04,STAT(R0)
(9) 004244 001415          BEQ     50041$
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.STAT
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$:
1666                                     ;+
1667                                     ; THE MODULE IS ELIGIBLE TO RUN, SO SET THE DE-QUEUEING FLAG ACCORDINGLY
1668                                     ; -
1669
1670 004300                                LET DX.QFLG := #-1
(4) 004300 012767 177777 174736                                MOV      #-1,DX.QFLG
1671 004306                                ENDIF
(4) 004306                                50035$:
1672
1673                                     ;+
1674                                     ; IF THE CONTROL QUEUE IS EMPTY GO SEE IF THERE IS A MODULE IN THE BACKGROUND QUEUE
1375                                     ; THAT WANTS TO RUN (SELECTED AND NOT DROPPED).
1676                                     ; IF SO, SET THE DE-QUEUEING FLAG.
1377                                     ; -
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$
1380 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 ; "RELMODE" 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.STO THEN
(6) 004456 032767 020000 174330 BIT #RELMODE,DTABLE+
(9) 004464 001404 BEQ 50051$
1708 004466 LET DTABLE+DT.STO := DTABLE+DT.STO SET.BY #RELMODE
(6) 004466 052767 010000 174320 BIS #RELMODE,DTABLE+
1709 004474 ELSE
(4) 004474 000421 BR 50052$
(3) 004476 50051$:
1710 004476 CALL CLREOP IN <#DTABLE>
(3) 004476 010546 MOV R5,-(SP)
(4) 004500 012745 001004 MOV #DTABLE,-(R5)
(3) 004504 004767 000000G JSR PC,CLREOP
(3) 004510 012605 MOV (SP)+,R5
1711 004512 IF #AUTO SETIN DTABLE+DT.CFO OR #SBKSEL SETIN DTABLE+DT.ST1 THEN
(6) 004512 032767 000010 174300 BIT #AUTO,DTABLE+DT.
(8) 004520 001004 BNE 50053$
(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$
    
```

1720 004546  
(2) 004546 000167 177066  
1721 004552  
(2) 004552

INLINE <JMP 3\$>

INLINE <4\$:>

JMP 3\$

4\$:



```

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
1733 (4) 004552 010567 174470          MOV      R5,DX.R5
1733 004556          LET DX.SP := SP
1734 (4) 004556 010667 174466          MOV      SP,DX.SP
1734 004562          LET R0 := DX.HDR
1735 (4) 004562 016700 174452          MOV      DX.HDR,R0
1735 004566          LET R0 := R0 + #SVR6
1736 (6) 004566 062700 000076          ADD      #SVR6,R0
1736 004572          LET SP := (R0)
1737 (4) 004572 011006          MOV      (R0),SP
1737 004574          LET R5 := -(R0)
1738 (4) 004574 014005          MOV      -(R0),R5
1738 004576          LET R4 := -(R0)
1739 (4) 004576 014004          MOV      -(R0),R4
1739 004600          LET R3 := -(R0)
1740 (4) 004600 014003          MOV      -(R0),R3
1740 004602          LET R2 := -(R0)
1741 (4) 004602 014002          MOV      -(R0),R2
1741 004604          LET R1 := -(R0)
1742 (4) 004604 014001          MOV      -(R0),R1
1742 004606          LET R0 := -(R0)
1743 (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
1757 (2) 004610 016746 174424          MOV      DX.HDR,-(SP)
1757 004614          LET (SP) := (SP) + #STAT
1758 (6) 004614 062716 000026          ADD      #STAT,(SP)
1758 004620          IF #BIT04 SETIN @(SP) THEN
1759 (6) 004620 032776 000020 000000          BIT      #BIT04,@(SP)
1759 (9) 004626 001415          BEQ      50055$
1759 004630          LET (SP) := @(SP)
1760 (4) 004630 017616 000000          MOV      @(SP),(SP)
1760 004634          INLINE <BIC #177020,(SP)>
1761 (2) 004634 042716 177020          BIC      #177020,(SP)
1761

```

```

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
1769          (2) 004640 012746 000340          MOV      #PRI7,-(SP)
1770          004644          PUSH #5$
1771          (2) 004644 012746 004652          MOV      #5$,-(SP)
1772          004650          RTI
1773
1774          004652          INLINE <5$:>
1775          (2) 004652
1776          004652          LET DTABLE+DT.STO := DTABLE+DT.STO SET.BY #BKMODE
1777          (6) 004652 052767 040000 174134          BIS      #BKMODE,DTABLE+D
1778          004660          ELSE
1779          (4) 004660 000401          BR       50055$
1780          (3) 004662          50055$:
1781          004662          LET (SP) := #0
1782          (4) 004662 005016          CLR      (SP)
1783          004664          ENDIF
1784          (4) 004664          50056$:
1785          004664          PUSH DX.RET
1786          (2) 004664 016746 174352          MOV      DX.RET,-(SP)
1787
1788          ;+
1789          ; DO IT
1790          ;-
1791
1792          004670          RTI
1793
1794

```

```

1786
1787
1788
1789
1790
1791 004672
(2) 004672
1792
1793
1794
1795
1796
1797 004672
(6) 004672 032767 000020 174120
(9) 004700 001012
1798
1799 004702
(3) 004702 010546
(5) 004704 016745 174530
(4) 004710 012745 001004
(3) 004714 004767 000000G
(3) 004720 012605
1800
1801 004722
(3) 004722 004767 000172
1802 004726
(4) 004726
1803
1804
1805
1806
1807
1808 004726
(6) 004726 032767 000400 174064
(9) 004734 001412
1809 004736
(3) 004736 010546
(5) 004740 016745 174474
(4) 004744 012745 001004
(3) 004750 004767 000000G
(3) 004754 012605
1810 004756
(3) 004756 004767 000136
1811 004762
(4) 004762
1812
1813
1814
1815
1816
1817 004762
(6) 004762 032767 000100 174030
(8) 004770 001004
(6) 004772 032767 002000 174020
(9) 005000 001412
(6) 005002

```

```

;+
; THIS ROUTINE PERFORMS THE TURNING ON OF THE APPROPRIATE
; CPU OPTIONS.
;-
ROUTINE DX.TURNON
DX.TURNON:

;+
; CALL ROTON IF NOT SCRAWNY MONITOR
; AND OUTPUT APPROPRIATE MESSAGE
;-
IF #NCPUOP NOTSETIN DTABLE+DT.CFO THEN
    BIT    #NCPUOP,DTABLE+D
    BNE    50002$

    MOV    R5,-(SP)
    MOV    DX.FAK,-(R5)
    MOV    #DTABLE,-(R5)
    JSR    PC,KROTON
    MOV    (SP)+,R5

    JSR    PC,DX.TU1

ENDIF
50002$:

;+
; TURN ON KT IF AVAILABLE AND OUTPUT MESSAGE
;-
IF #KTPRES SETIN DTABLE+DT.CFO THEN
    BIT    #KTPRES,DTABLE+D
    BEQ    50003$

    MOV    R5,-(SP)
    MOV    DX.FAK,-(R5)
    MOV    #DTABLE,-(R5)
    JSR    PC,KKTON
    MOV    (SP)+,R5

    JSR    PC,DX.TU1

ENDIF
50003$:

;+
; TURN ON "PARITY" OR "ECC" MEMORY IF AVAILABLE, OUTPUT MESSAGE
;-
IF #ECCMEM SETIN DTABLE+DT.CFO OR #PARPRES SETIN DTABLE+DT.CFO THEN
    BIT    #ECCMEM,DTABLE+D
    BNE    50004$
    BIT    #PARPRES,DTABLE+
    BEQ    50005$
50004$:
50005$:

```

```

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

```

```

MOV R5,-(SP)
MOV DX.FAK,-(R5)
MOV #DTABLE,-(R5)
JSR PC,KPON
MOV (SP)+,R5
JSR PC,DX.TU1
50005$:
BIT #CAPRES,DTABLE+D
BEQ 50006$
MOV R5,-(SP)
MOV DX.FAK,-(R5)
MOV #DTABLE,-(R5)
JSR PC,KCON
MOV (SP)+,R5
JSR PC,DX.TU1
50006$:
BIT #ADDR22,DTABLE+D
BEQ 50007$
MOV R5,-(SP)
MOV DX.FAK,-(R5)
MOV #DTABLE,-(R5)
JSR PC,KMON
MOV (SP)+,R5
JSR PC,DX.TU1
50007$:
50000$:
50001$:
RTS PC

```

```

1846
1847 005120          ROUTINE DX.TU1
(2) 005120
1848
1849
1850
1851
1852
1853
1854
1855 005120          CALL MSGDHOOK IN <#DTABLE,#MSGPOP,DTABLE+DT.KBRSP,#2$>
(3) 005120 010546
(7) 005122 012745 005154
(6) 005126 016745 173674
(5) 005132 012745 000002
(4) 005136 012745 001004
(3) 005142 004767 000000G
(3) 005146 012605
1856 005150          INLINE <1$:>
(2) 005150
1857 005150          INLINE <NOP>
(2) 005150 000240
1858 005152          INLINE <BR 1$>
(2) 005152 000776
1959 005154          INLINE <2$:>
(2) 005154
1860 005154          LET DTABLE+DT.KBRSP := #0
(4) 005154 005067 173646
1861 005160          ENDRTN
(3) 005160
(3) 005160
(2) 005160 000207
1862
1863          002200          .END    DX.STRT

```

DX.TU1:

```

;+
; RETRIEVE MESSAGE FROM DT.KBRSP WORD IN DTABLE AND OUTPUT
; MESSAGE AND WAIT FOR IT TO FINISH OUTPUTTING BEFORE RETURNING
; CONTROL
;-

```

```

MOV    R5,-(SP)
MOV    #2$,-(R5)
MOV    DTABLE+DT.KBRSP,
MOV    #MSGPOP,-(R5)
MOV    #DTABLE,-(R5)
JSR    PC,MSGDHOOK
MOV    (SP)+,R5

```

1\$:

NOP

BR 1\$

2\$:

CLR DTABLE+DT.KBRSP

50000\$:

50001\$:

RTS PC

ACSR = 000102	BKMOD = 000020	DT.EXS= 000060	ENBEO= 010000	KIPDR6= 172314
ACTSIT= 004000	BKMODE= 040000	DT.FCH= 000037	ENBNUL= 000001	KIPDR7= 172316
ACTIV = ***** G	BKSLSH= 000134	DT.FCN= 000036	ENDLST= 000000	KKTON = ***** G
AC.MOD= ***** G	CAPRES= 000004	DT.HMX= 000104	EOPBIT= 000001	KMON = ***** G
AC.MPT= ***** G	CAST = ***** G	DT.KBE= 000024	ERREC = ***** G	KONTRL 001232 G
AC.TPT= ***** G	CASAT = 000004	DT.KBP= 000026	ERRTYP= 000106	KPON = ***** G
AC.TYP= ***** G	CCNTRL 001230 G	DT.KBR= 000022	EVNTBE= 000200	KROTON= ***** G
ADDR22= 001000	CDERCT= 000146	DT.KBU= 000030	EVNTHD= 000200	KRUN = ***** G
ADR = 000006	CDWDCT= 000144	DT.MLS= 000032	EVNTKT= 000203	KTERRO= 00040
APTFER= 000004	CHKACT= ***** G	DT.MT1= 000110	EVNTP= 000202	KTPRES= 000400
APTPRE= 000200	CHKEOP= ***** G	DT.OFF= 000070	EVNTR= 000201	KTSET = ***** G
APTSLP= ***** G	CKTIM = 100000	DT.PAS= 000074	FATERR= 100000	KTSTAT= 000020
APTVEC 000044	CLKPRE= 000001	DT.PC = 000002	HRDCNT= 000044	KXTND= 040000
AP.PB = ***** G	CLGCK 001160 G	DT.PFL= 000062	HRDPAS= 000050	KW11L = ***** G
ASB = 000106	CLOCKL 001162 G	DT.PSW= 000004	HTBUSS= ***** G	KW11P = ***** G
ASSEMB= 000010	CLOCKP 001164 G	DT.PTA= 000064	HTREIN= ***** G	LCLEAR= ***** G
ASTAT = 000104	CLREOP= ***** G	DT.RCS= 000102	HT.EXT= ***** G	LF = 000012
AUTO = 000010	CMDPRC= ***** G	DT.REL= 000040	ICONT = 000036	LPBUF 001212 G
AUTOST= 020000	CMDRST= ***** G	DT.SCT= 000066	ICOUNT= 000040	LPCSR 001210 G
AWAS = 000110	CONFIG= 000056	DT.SMX= 000106	IDNUM = 000122	LPINT = ***** G
BACTIV= ***** G	CPUCPE 001224 G	DT.SP = 000006	IE = 000100	LPSTAT= 000001
BA.HDR= ***** G	CPUHAR 001216 G	DT.SSI= 000046	IERRLO= ***** G	MAPSTA= 000200
BA.MPT= ***** G	CPULAR 001214 G	DT.ST0= 000010	INDPAR= 000040	MD.BSY= ***** G
BA.RET= ***** G	CPULSZ 001226 G	DT.ST1= 000012	INHDRP= 040000	MED = 076600
BA.STA= ***** G	CPUMER 001220 G	DT.SWR= 000056	INHEPR= 020000	MEMGMT= ***** G
BDACNV= ***** G	CPUMNT 001222 G	DT.SYP= 000072	INHREL= 001000	MEMPAR= ***** G
BIT0 = 000001	CQINI = ***** G	DT.WBU= 000050	INHRR= 000400	MEMPAS= 040000
BIT00 = 000001	CQOVF = 000001	DT.WHL= 000054	INIT = 000030	MODEXH= 004000
BIT01 = 000002	CR = 000015	DT.WLL= 000052	INTR = 000120	MODHOL= 002000
BIT02 = 000004	CSRA = 000100	DVID1 = 000014	IOMOD = 100000	MODESEL= 001000
BIT03 = 000010	CSRC = 000102	DX.CAP 003242 G	IOMODP= 102000	MSGCKD= 000010
BIT04 = 000020	CTRLC = 000003	DX.DEQ 003634 G	IOMODR= 112000	MSGCKS= 000011
BIT05 = 000040	CTRL0 = 000017	DX.FAK 001440	IOMODX= 110000	MSGDEQ= ***** G
BIT06 = 000100	CTRLOF 001234 G	DX.FK1 001442	IOTINS 000020	MSGDER= 000005
BIT07 = 000200	CTRLU = 000025	DX.HDR 001240 G	JACK = 035060	MSGDHO= ***** G
BIT08 = 000400	DCEVNT= 000011	DX.INI 003020 G	KBINI = ***** G	MSGDRP= 000017
BIT09 = 001000	DEFRTN= 000400	DX.KFL 001236 G	KBINT = ***** G	MSGECH= 177777
BIT1 = 000002	DEQCQ = ***** G	DX.MED 001402	KBVEC 000060	MSGEOP= 000013
BIT10 = 002000	DIAGMC= 000000	DX.MON 001346 G	KCON = ***** G	MSCHDR= 000004
BIT11 = 004000	DPEOP = ***** G	DX.QFL 001244	KIPARO= 172340	MSGHNG= 000022
BIT12 = 010000	DPSTRT= ***** G	DX.RET 001242 G	KIPAR1= 172342	MSGHRD= 000007
BIT13 = 020000	DROPMD= 100000	DX.RST 002650 G	KIPAR2= 172344	MSGMAP= 000021
BIT14 = 040000	DSEVNT= 000014	DX.RS 001246 G	KIPAR3= 172346	MSGNUL= 177775
BIT15 = 100000	DTABLE 001004 G	DX.SIZ 001353	KIPAR4= 172350	MSGPOP= 000002
BIT2 = 000004	DT.ADD= 000042	DX.SP 001250 G	KIPAR5= 172352	MSGPRM= 177776
BIT3 = 000010	DT.AP = 000100	DX.STR 002200	KIPAR6= 172354	MSGRES= 000001
BIT4 = 000020	DT.APK= 000076	DX.SZK 001371	KIPAR7= 172356	MSGSFT= 000006
BIT5 = 000040	DT.BLS= 000034	DX.TIT 001252	KIPDR0= 172300	MSGSKE= 000003
BIT6 = 000100	DT.CFO= 000014	DX.TUR 004672	KIPDR1= 172302	MSGSMB= 000015
BIT7 = 000200	DT.CF1= 000016	DX.TU1 005120	KIPDR2= 172304	MSGSMH= 000014
BIT8 = 000400	DT.ERR= 000020	ECCMEM= 000100	KIPDR3= 172306	MSGSMS= 000016
BIT9 = 001000	DT.ESI= 000044	ECCSTA= 000010	KIPDR4= 172310	MSGSTD= 000000
SKDEF = 000002	DT.EVN= 000000	EMTINS 000030	KIPDR5= 172312	MSGSYS= 000012

MSGVEC= 000020	RDWHMI= 000022	SR3 = 172516	WBUFEA= 000136	\$ISK3 = 000001
NSKMOD= 001000	RELCTI= ***** G	START = 000200	WBUFPA= 000134	\$LOCTA= 177777
NCPUDP= 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	\$NSKO = 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	SVRO = 000062	XBUF 001206 G	\$NSK3 = 000110
PARSTA= 000100	RES2 = 000060	SVR1 = 000064	XCSR 001204 G	\$SAVLE= 177777
PARTAB 001116 G	RICHR= 031060	SVR2 = 000066	XFLAG = 000005	\$SSKO = 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	XBNLE= 177777	\$TEMP = 000300
PDPLSI= 020000	RSTRT = 000112	SVR6 = 000076	\$ENDAD= ***** G	\$TSKO = 050007
PDP30 = 004000	RUBOUT= 000177	SYSCNT= 000052	\$ERFLG= 000400	\$TSK1 = 050050
PDP70 = 010000	RUNMOD= 100000	SYSERR= 000100	\$F\$AND= 000310	\$TSK2 = 050052
PREVT = ***** G	RSVALU= 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\$DD = 000340	\$DST = 000000
PRI6 = 000300	SBKSEL= 010000	TTINT = ***** G	\$F\$FAL= 000405	\$ELOC= 000402
PRI7 = 000340	SCTAB 001160 G	TTVEC 000064	\$F\$G00= 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\$L00= 000200	\$L0C = 005070
PR5 = 000240	SC.CKL= 000002	UIPAR3= 177646	\$F\$NAM= 000160	\$L0CN= 000000
PR6 = 000300	SC.CKP= 000004	UIPAR4= 177650	\$F\$NO = 000403	\$REG = 177777
PR7 = 000340	SC.CLO= 000000	UIPAR5= 177652	\$F\$OR = 000320	\$RETN= 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
RANNU= 000054	SOFCNT= 000042	UIPDR2= 177604	\$F\$TRU= 000404	\$TGS1= 000000
RBUF 001202 G	SOPPAS= 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	\$ISKO = 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)

3	COMMON EQUATE MODULE
532	APTMOD (COMMON DEFINITIONS & REFERENCES)
535	000000' .PRINT ;SPMAC: VERSION 1.1
633	APTSLP APT SLEEP AND CONFIGURE ROUTINE
688	APTSLP ROUTINE
786	APTSER (APT SOFT ERROR HANDLER)
839	APTSER CODE
873	APTDDED PUT OPTION MODULE NAME INTO MAILBOX
922	APTDDED ROUTINE
954	APTHER (APT HARD ERROR HANDLER)
1005	APTHER CODE
1043	APTSEL - SELECT OPTION MODULES FROM MAP
1090	APTSEL ROUTINE
1140	APTKIL - KILL APT BY HALTING
1187	APTKIL ROUTINE



```
508 .TITLE APTMOD (APT MODULE PACKAGE)
509 .IDENT /V0.0/
510 ;++
511 ; MODULE PACKAGE NAME:
512 ;     APTMOD
513 ;
514 ; FUNCTIONAL DESCRIPTION:
515 ;
516 ;     THIS MODULE PACKAGE CONTAINS SEVERAL APT ROUTINES:
517 ;         1. APTSLP     APT SLEEP AND CONFIGURATION ROUTINE
518 ;         2. APTSER     APT SOFT ERROR HANDLER
519 ;         3. APTDED     STUFF APT MAILBOX WITH FAILING MODULE NAME
520 ;         4. APThER     APT HARD ERROR ROUTINE
521 ;         5. APTSEL     SELECT-DESELECT ROUTINE
522 ;         6. APTKIL     STUFF CODE TO MAILBOX IF MONITOR ERROR
523 ;                       (I.E. SYSTEM ERROR) AND HALTS
524 ;
525 ; VERSION:
526 ;     0.0
527 ;
528 ; EDIT             DATE             BY             REASON
529 ;--
530
```

```

532          .SBTTL APTMOD (COMMON DEFINITIONS & REFERENCES)
533
534          .MCALL STRUCT
535 000000'   STRUCT
536 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
537          000001 $LSTIN=1
538          000001 $LSTTAG=1
539
540          ;*****
541          ;
542          ; REFERENCED BY OTHER MODULES
543          ;
544          .GLOBL APTSLP          ;APT SLEEP ROUTINE
545          .GLOBL APTSER          ;SOFT ERROR HANDLER
546          .GLOBL APThER          ;HARD ERROR HANDLER
547          .GLOBL APTKIL          ;APT HALT ROUTINE
548          .GLOBL APTDED          ;STUFF FAILING MODULE NAME
549          .GLOBL APTSEL          ;APT SELECT-DESELECT MODULE ROUTINE
550          .GLOBL AP.PB           ;APT PARAMETER BLOCK
551          ;*****
552          ; GLOBAL REFERENCES
553          ;
554          .GLOBL SAVREG          ;
555          .GLOBL RESREG          ;
556          .GLOBL RSTRCY          ;RESET RECOVERY ROUTINE
557          .GLOBL KSUM           ;'SUM' COMMAND ROUTINE
558          .GLOBL MD.COD         ;MSG CODE
559          .GLOBL MSGDEQ         ;MESSAGE DEQUER
560          .GLOBL RCSR           ;KB CSR
561
562          ;
563          ;*****
564          ;
565          ; LOCAL STORAGE
566          ;
567          ;
568          ;***** APT PARAMETER BLOCK *****
569          ;
570          ;
571          ;
572          ;
573 000000'   AP.PB:
574 000000' 000000 AP.P00: 0          ;PTABLE WORD 0 - BITS 16 & 17 OF MAILBOX ADDR
575 000002' 000014' AP.P02: #AP.MB1      ;LOWER 16 BITS OF MAILBOX ADDRESS
576 000004' 001604 AP.P04: 900.        ;RUN TIME (IN SECONDS) OF LONGEST TEST
577 000006' 000170 AP.P06: 120.        ;RUN TIME (IN SECONDS) OF FIRST PASS
578          ;INCLUDES THE DELAY REQUIRED BY DEVICES TO
579          ;RECOVER FROM POWER FAIL CAUSED BY APT TO
580          ;START UP DEC-X11 EXERCISER.
581 000010' 000000 AP.P10: 0          ;ADDITIONAL RUN TIME (IN SECONDS) OF FIRST
582          ;PASS WHEN A SECOND DEVICE IS ADDED.
583 000012' 000052 AP.P12: <AP.END-AP.MB1>/2 ;LENGTH (IN WORDS) OF E-TABLE + MAILBOX
584
585
586

```

```
587  
588 ;***** APT MAILBOX *****  
589 ;  
590 000014' 000000 AP.MB1: 0 ;MAILBOX WORD 1 - FATAL ERROR INDICATOR  
591 000016' 020040 AP.MB2: 20040 ;MODULE NAME'S FIRST TWO CHARACTERS  
592 000020' 020040 AP.MB3: 20040 ;MODULE NAME'S THIRD & FIFTH CHARACTERS  
593 000022' 000000 AP.MB4: 0 ;PASS COUNT  
594 000024' 000000 AP.MB5: 0 ;TO BE INCREMENTED EVERY 10 SECONDS  
595 000026' 000000 0  
596 000030' 000000 0  
597 000032' 000000 0  
598  
599 ;***** APT E-TABLE *****  
600 ;  
601 000034' 001 AP.ET1: .BYTE 1 ;E-TABLE - BYTE 1 - SOFT ENVIRONMENT  
602 ;DEFAULT TO APT MODE  
603 000035' 200 .BYTE 200 ;BYTE 2 - DEFAULT TO UUT TTY AND DON'T SIZE  
604  
605 000036' 000000 AP.SR1: 0 ;APT SW 1  
606 000040' 000037 AP.SR2: ^D<31> ;HARD ERROR LIMIT (1) AND SOFT ERROR LIMIT (31.)  
607 000042' 000000 AP.CPU: 0 ;CPU OPTIONS,15-11=TYPE,10=RTC,9=FPP,8=MM  
608 000044' 000000 100000 AP.MM0: 0,100000 ;MEMORY MAP, 100000=16K  
609 000050' 000000 000000 AP.MM1: 0,0 ;ONE BYTE FOR TYPE PLUS THREE BYTES  
610 000054' 000000 000000 AP.MM2: 0,0 ;FOR MAX ADDR OF THAT TYPE  
611 000060' 000000 000000 AP.MM3: 0,0 ;DECX  
612 000064' 000000 000000 0,0 ;DON'T CARE  
613 000070' 000000 000000 0,0 ;ABOUT THESE  
614 000074' 000000 000000 0,0 ;LOCATIONS  
615  
616 000100' 000020 AP.MAP: .BLKW 16. ;APT MODULE MAP  
617 ;ONE BYTE REQD FOR EACH MODULE  
618 000140'  
619 AP.END:  
620  
621 ;***** GENERAL LOCAL STORAGE *****  
622 ;  
623 000140' 047515 AP.MC0: .ASCII /MO/ ;MONITOR ERROR - STUFF "MO" INTO MAILBOX  
624 000142' 050503 AP.MC1: .ASCII /CQ/ ;CONTROL QUEUE OVERFLOW  
625 000144' 050524 .ASCII /TQ/ ;TYPE QUEUE OVERFLOW  
626 000146' 000000 .WORD 0  
627 000150' 042520 .ASCII /PE/ ;PARITY ERRORS  
628 000152' 042515 .ASCII /ME/ ;MEMORY ERROR - A LOCATION IS BAD  
629 000154' 052113 .ASCII /KT/ ;KT TRAP  
630 000156' 042523 .ASCII /SE/ ;TRAP THROUGH LOC. 4 OR 10  
631 .EVEN  
632 000160' 000162' AP.FAK: .WORD AP.CR ;POINTER TO NEXT WORD  
633 000162' 000015 AP.CR: .WORD CR ;A FAKE CARRIAGE RETURN FOR 'SUM' COMMAND  
634 ;  
635  
636  
637
```

```
539 .SBTTL APTSLP APT SLEEP AND CONFIGURE ROUTINE
540 .IDENT /V0.0/
541
542 ;++
543 ; MODULE NAME:
544 ; APTSLP
545 ;
546 ; FUNCTIONAL DESCRIPTION:
547 ; THIS ROUTINE WILL DETERMINE FROM THE APT ENVIRONMENT WORD 1
548 ; IF TRULY UNDER APT. IF SO A CHECK IS MADE ON AVAILABILITY OF
549 ; U.U.T. TERMINAL. IF NONE, THE NOAPTY BIT IS SET IN CONFIG WORD
550 ; 0. IF UNDER APT, A 15 SECOND SLEEP ROUTINE IS ENTERED TO
551 ; RECOVER FROM POWER LOGIC YANK.
552 ;
553 ; INPUTS:
554 ; DTABLE
555 ;
556 ; IMPLICIT INPUTS:
557 ; NONE
558 ;
559 ; OUTPUTS:
560 ; NONE
561 ;
562 ; IMPLICIT OUTPUTS:
563 ; DT.CFO
564 ; DT.APK
565 ; DT.SYP
566 ;
567 ; PATHOLOGICAL CONNECTIONS:
568 ; NONE
569 ;
570 ; SUBORDINATE ROUTINES CALLED:
571 ; SAVREG
572 ; RESREG
573 ;
574 ; FUNCTIONAL SIDE EFFECTS:
575 ; NONE
576 ;
577 ; CALLING SEQUENCE:
578 ; CALL APTSLP IN <DT>
579 ; WHERE DT = ADDRESS OF DTABLE
580 ;
581 ; VERSION:
582 ; 0.0
583 ;
584 ; EDIT DATE BY REASON
585 ;--
586
```

```
688 .SBTTL APTS LP ROUTINE
689
690 000164' ROUTINE APTS LP <DTADR>
(2) 000164' APTS LP:
691
692
693 ;+
694 ; SAVE REGISTERS, GET DTABLE ADDR.
695 ; -
696
697 000164' CALL SAVREG
(3) 000164' 004767 000000G JSR PC, SAVREG
698 000170' LET R0 := DTADR(R5)
(4) 000170' 016500 000000 MOV DTADR(R5), R0
699
700 ;+
701 ; GET ENVIRONMENT WORD 1 TO R3
702 ; -
703 000174' LET R3 := #AP.ET1
(4) 000174' 012703 000034' MOV #AP.ET1, R3
704
705 ;+
706 ; ZERO FATAL BIT INDICATOR IN CASE IT WAS SET
707 ; -
708
709 000200' LET AP.MB1 := #0
(4) 000200' 005067 177610 CLR AP.MB1
710
711 ;+
712 ; IF BIT0 SET IN ENVIRONMENT WORD 1, SET APTPRES BIT IN DT.CF0, SET UP KEEP ALIVE
713 ; AND SYSTEM PASS COUNT
714 ; -
715 000204' IF #BIT00 SETIN (R3) THEN
(6) 000204' 032713 000001 BIT #BIT00, (R3)
(9) 000210' 001440 BEQ 50002$
716 000212' LET DT.CF0(R0) := DT.CF0(R0) SET.BY #APTPRES
(6) 000212' 052760 000200 000014 BIS #APTPRES, DT.CF0(
717
718
719 ;+
720 ; *****
721 ; !!!!!!! NOTE: KEEP ALIVE COUNTER FOLLOWS - !!!!!!!
722 ; !!!!!!! REMOVE SEMI-COLON IF YOU EVER WANT !!!!!!!
723 ; !!!!!!! A DEC/X11 KEEP-ALIVE COUNTER. IF !!!!!!!
724 ; !!!!!!! YOU DO, IT WILL BE UPDATED !!!!!!!
725 ; !!!!!!! VERY OFTEN (MANY TIMES A SECOND) !!!!!!!
726 ; !!!!!!! IN DECX . !!!!!!!
727 ; *****
728 ; -
729
730 ; LET DT.APK(R0) := #AP.MB5
731
732
733
734 000220' LET DT.SYP(R0) := #AP.MB4
(4) 000220' 012760 000022' 000072 MOV #AP.MB4, DT.SYP(R
```

```

735
736
737 ;+
738 ; IF BIT 13 SETIN ENV WORD 1, SET NO APT TTY BIT (NOAPTY)
739 ; IN CONFIG WORD 0 (DT.CFO) AND FAKE OUT THE KB CSR.
740 ;-
740 000226' IF #BIT13 SETIN (R3) THEN
(6) 000226' 032713 020000 BIT #BIT13,(R3)
(9) 000232' 001410 BEQ 50003$
741 000234' LET DT.CFO(R0) := DT.CFO(R0) SET.BY #NOAPTY
(6) 000234' 052760 000002 000014 BIS #NOAPTY,DT.CFO(R)
742 000242' LET RCSR := R0 + #DT.RCS
(4) 000242' 010067 000000G MOV R0,RCSR
(6) 000246' 062767 000102 000000G ADD #DT.RCS,RCSR
743 000254' ENDIF
(4) 000254' 50003$:
744
745 ;+
746 ; INIT COUNT FOR SLEEP LOOP
747 ;-
748
749 000254' LET R1 := #25
(4) 000254' 012701 000025 MOV #25,R1
750 000260' LET R2 := #-1
(4) 000260' 012702 177777 MOV #-1,R2
751
752 ;+
753 ; GO TO SLEEP FOR A WHILE TO RECOVER FROM POWER FAIL LINE YANK
754 ;-
755 000264' WHILE R1 NE #0 DO
(4) 000264' 50004$:
(6) 000264' 005701 TST R1
(9) 000266' 001410 BEQ 50005$
756 000270' WHILE R2 NE #0 DO
(4) 000270' 50006$:
(6) 000270' 005702 TST R2
(9) 000272' 001402 BEQ 50007$
757 000274' LET R2 := R2 - #1
(6) 000274' 005302 DEC R2
758 000276' ENDDO
(4) 000276' 000774 BR 50006$
(3) 000300' 50007$:
759 000300' LET R2 := #-1
(4) 000300' 012702 177777 MOV #-1,R2
760 000304' LET R1 := R1 - #1
(6) 000304' 005301 DEC R1
761 000306' ENDDO
(4) 000306' 000766 BR 50004$
(3) 000310' 50005$:
762
763 000310' ELSE
(4) 000310' 000403 BR 50010$
(3) 000312' 50002$:
764
765
766 ;+
767 ; NOT APT - CLEAR APT PRESENT BIT IN DT.CFO

```

```
768 ;-
769
770 000312' LET DT.CFG(R0) := DT.CF0(R0) CLR.BY #APTPRES
(6) 000312' 042760 000200 000014
771 BIC #APTPRES,DT.CF0(
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
```

```
786 .SBTTL APTSER (APT SOFT ERROR HANDLER)
787
788 ;++
789 ; MODULE NAME:
790 ; APTSER
791 ;
792 ; FUNCTIONAL DESCRIPTION:
793 ; THIS MODULE IS CALLED WHEN THE EXERCISER IS
794 ; DETERMINED TO BE RUNNING UNDER APT ENVIRONMENT, TO HANDLE
795 ; A SOFT ERROR.
796 ;
797 ; THE ERROR COUNT IS COMPARED WITH THE LIMIT SET IN E-TABLE.
798 ; IF THE COUNT IS LESS THAN THE LIMIT, CONTROL GOES BACK TO
799 ; THE CALLER.
800 ;
801 ; IF THE COUNT EQUALS THE LIMIT THE RUN IS OVER (CALL APTDED).
802 ;
803 ;
804 ; INPUTS:
805 ; DTABLE ADDRESS
806 ;
807 ; IMPLICIT INPUTS:
808 ; 1. DT.PC
809 ;
810 ; OUTPUTS:
811 ; NONE
812 ;
813 ; IMPLICIT OUTPUTS:
814 ; NONE
815 ;
816 ; PATHOLOGICAL CONNECTIONS:
817 ; NONE
818 ;
819 ; SUBORDINATE ROUTINES CALLED:
820 ; 1. APTDED
821 ; 2. SAVREG
822 ; 3. RESREG
823 ;
824 ; FUNCTIONAL SIDE EFFECTS:
825 ; NONE
826 ;
827 ; CALLING SEQUENCE:
828 ; CALL APTSER IN <DTA>
829 ;
830 ; WHERE DTA = DATA TABLE ADDRESS
831 ;
832 ; VERSION:
833 ; 0.0
834 ;
835 ; EDIT DATE BY REASON
836 ;--
837
```



```

839          .SBTTL  APTSER CODE
840
841 000326'    ROUTINE APTSER <DTA>
(2) 000326'
842
843          ;+
844          ; SAVE REGS; GET DTABLE ADDR, MODULE'S HEADER ADDRESS,
845          ; AND SOFT ERROR LIMIT FROM E-TABLE.
846          ; -
847
848 000326'    CALL SAVREG
(3) 000326' 004767 000000G          JSR      PC, SAVREG
849
850 000332'    LET R0 := DTA(R5)
(4) 000332' 016500 000000          MOV      DTA(R5), R0
851 000336'    LET R1 := @DT.PC(R0)
(4) 000336' 017001 000002          MOV      @DT.PC(R0), R1
852 000342'    LET R2 := AP.SR2 CLR.BY #177740
(4) 000342' 016702 177472          MOV      AP.SR2, R2
(6) 000346' 042702 177740          BIC      #177740, R2
853
854          ;+
855          ; IF COUNT IS EQUAL TO THE LIMIT, WE ARE DONE - CALL APTDED
856          ; -
857
858 000352'    IF SOFCNT(R1) HIS R2 THEN
(6) 000352' 026102 000042          CMP      SOFCNT(R1), R2
(9) 000356' 103406          BLO      50002$
859          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
860
861          ENDIF
(4) 000374'          50002$:
862
863          ;+
864          ; RESTORE REGS AND RETURN
865          ; -
866
867 000374'    CALL RESREG          ;
(3) 000374' 004767 000000G          JSR      PC, RESREG
868
869          ENDRTN
(3) 000400'          50000$:
(3) 000400'          50001$:
(2) 000400' 000207          RTS      PC
870
871

```

```
873 .SBTTL APTDED PUT OPTION MODULE NAME INTO MAILBOX
874 .IDENT /V0.0/
875
876 ;++
877 ; MODULE NAME:
878 ; APTDED
879 ;
880 ; FUNCTIONAL DESCRIPTION:
881 ; THIS ROUTINE STUFFS THE 1ST,2ND,3RD AND FIFTH CHARACTERS
882 ; OF A FAILING OPTION MODULE NAME INTO THE 2ND AND 3RD WORDS
883 ; OF THE APT MAILBOX. IT RESETS AND RECOVERS. SET THE APT AND
884 ; FATAL ERROR BITS IN DTABLE'S ERROR WORD.
885 ;
886 ; INPUTS:
887 ; DTABLE ADDRESS
888 ; MODULE HEADER ADDRESS
889 ;
890 ; IMPLICIT INPUTS:
891 ; NONE
892 ;
893 ; OUTPUTS:
894 ; NONE
895 ;
896 ; IMPLICIT OUTPUTS:
897 ; DT.ERR
898 ;
899 ; PATHOLOGICAL CONNECTIONS:
900 ; NONE
901 ;
902 ; SUBORDINATE ROUTINES CALLED:
903 ; SAVREG
904 ; RESREG
905 ; RSTRCY
906 ;
907 ; FUNCTIONAL DESCRIPTION:
908 ; NONE
909 ;
910 ; CALLING SEQUENCE:
911 ; CALL APTDED IN <DT,MH> WHERE:
912 ; DT=DTABLE ADDRESS
913 ; MH=MODULE HEADER ADDRESS
914 ;
915 ; VERSION:
916 ; 0.0
917 ;
918 ; EDIT DATE BY REASON
919 ;--
920
```

```

922          .SBTTL  APTDED ROUTINE
923
924 000402'  ROUTINE APTDED <TB,MD>
(2) 000402'
925
926          ;+
927          ; SAVE REGISTERS,GET DTABLE ADDR., AND MODULE HEADER ADDR.
928          ; -
929 000402'  CALL SAVREG
(3) 000402' 004767 000000G          JSR      PC, SAVREG
930 000406'  LET R0 := TB(R5)
(4) 000406' 016500 000000          MOV      TB(R5),R0
931 000412'  LET R1 := MD(R5)
(4) 000412' 016501 000002          MOV      MD(R5),R1
932          ;+
933          ; STUFF APT MAILBOX WITH MODULE ASCII NAME,RESET AND RECOVER
934          ; UPDATE ERROR WORD IN DTABLE, AND
935          ; SET APT FATAL BIT IN MB1 AND
936          ; RETURN TO THE CALLER.
937          ; -
938
939 000416'  LET AP.MB2 := (R1)+
(4) 000416' 012167 177374          MOV      (R1)+,AP.MB2
940 000422'  LET R3 := #AP.MB3
(4) 000422' 012703 000020'          MOV      #AP.MB3,R3
941 000426'  LET (R3)+ :B= (R1)+
(4) 000426' 112123          MOV      (R1)+,(R3)+
942 000430'  INLINE <TSTB (R1)+>
(2) 000430' 105721          TSTB   (R1)+
943 000432'  LET (R3) :B= (R1)
(4) 000432' 111113          MOV      (R1),(R3)
944
945          INLINE <RESET>
(2) 000434' 000005          RESET
946 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
947
948 000450'  LET DT.ERR(R0) := DT.ERR(R0) SET.BY #APTFER!FATERR
(6) 000450' 052760 100004 000020          BIS      #APTFER!FATERR,D
949 000456'  LET AP.MB1 := AP.MB1 + #1
(6) 000456' 005267 177332          INC      AP.MB1
950 000462'  CALL RESREG
(3) 000462' 004767 000000G          JSR      PC,RESREG
951 000466'  ENDRTN
(3) 000466'
(3) 000466'
(2) 000466' 000207
952

```

50000\$:  
 50001\$:

RTS PC

```
954 .SBTTL APThER (APT HARD ERROR HANDLER)
955
956 ;++
957 ; MODULE NAME:
958 ;     APThER
959 ;
960 ; FUNCTIONAL DESCRIPTION:
961 ;     THIS MODULE IS CALLED WHEN THE EXERCISER IS
962 ;     DETERMINED TO BE RUNNING UNDER APT ENVIRONMENT, TO HANDLE
963 ;     A HARD ERROR.
964 ;
965 ;     IF THE ERROR COUNT IS LESS THAN THE LIMIT SET IN E-TABLE,
966 ;     CONTROL GOES BACK TO THE CALLER.
967 ;
968 ;     IF THE COUNT EQUALS THE LIMIT THE RUN IS OVER (CALL APTDED).
969 ;
970 ;
971 ; INPUTS:
972 ;     DTABLE ADDRESS
973 ;
974 ; IMPLICIT INPUTS:
975 ;     DT.PC
976 ;
977 ; OUTPUTS:
978 ;     NONE
979 ;
980 ; IMPLICIT OUTPUTS:
981 ;     NONE
982 ;
983 ; PATHOLOGICAL CONNECTIONS:
984 ;     NONE
985 ;
986 ; SUBORDINATE ROUTINES CALLED:
987 ;     1. APTDED
988 ;     2. SAVREG
989 ;     3. RESREG
990 ;
991 ; FUNCTIONAL SIDE EFFECTS:
992 ;     NONE
993 ;
994 ; CALLING SEQUENCE:
995 ;     CALL APThER IN <DTA>
996 ;
997 ;     WHERE DTA = DATA TABLE ADDRESS
998 ;
999 ; VERSION:
1000 ;     0.0
1001 ;
1002 ;     EDIT             DATE             BY             REASON
1003 ;--
1004
```

```

1006          .SBTTL  APATHER CODE
1007
1008
1009 000470'          ROUTINE APATHER <DTA>
1010          (2) 000470'          APATHER:
1011
1012          ;+
1013          ; SAVE REGS; GET DTABLE ADDR, MODULE'S HEADER ADDRESS,
1014          ; AND HARD ERROR LIMIT FROM E-TABLE.
1015          ; -
1016 000470'          CALL SAVREG
1017          (3) 000470' 004767 000000G          JSR      PC, SAVREG
1018 000474'          LET R0 := DTA(R5)
1019          (4) 000474' 016500 000000          MOV      DTA(R5), R0
1020 000500'          LET R1 := @DT.PC(R0)
1021          (4) 000500' 017001 000002          MOV      @DT.PC(R0), R1
1022 000504'          LET R2 := SWAP AP.SR2
1023          (6) 000504' 016702 177330          MOV      AP.SR2, R2
1024          (6) 000510' 000302          SWAB     R2
1025 000512'          LET R2 := R2 CLR.BY #177740
1026          (6) 000512' 042702 177740          BIC      #177740, R2
1027
1028          ;+
1029          ; IF COUNT IS EQUAL TO THE LIMIT, THEN WE ARE DONE
1030          ; -
1031 000516'          IF HRDCNT(R1) HIS R2 THEN
1032          (6) 000516' 026102 000044          CMP      HRDCNT(R1), R2
1033          (9) 000522' 103406          BLO     50002$
1034
1035          CALL APTDED IN <R0, R1>
1036          (3) 000524' 010546          MOV      R5, -(SP)
1037          (5) 000526' 010145          MOV      R1, -(R5)
1038          (4) 000530' 010045          MOV      R0, -(R5)
1039          (3) 000532' 004767 177644          JSR      PC, APTDED
1040          (3) 000536' 012605          MOV      (SP)+, R5
1041
1042          ENDIF
1043          (4) 000540'          50002$:
1044
1045          ;+
1046          ; RESTORE REGS AND RETURN
1047          ; -
1048 000540'          CALL RESREG
1049          (3) 000540' 004767 000000G          JSR      PC, RESREG
1050
1051          ENDRTN
1052          (3) 000544'          50000$:
1053          (3) 000544'          50001$:
1054          (2) 000544' 000207          RTS     PC
1055
1056

```

```
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
```

```

1090          .SBTTL  APTSEL ROUTINE
1091
1092 000546'          ROUTINE APTSEL <DTBL>
(2) 000546'
1093
1094          ;+
1095          ; SAVE REGISTERS,DTABLE AND GET MAP ADDRESS
1096          ; -
1097
1098 000546'          CALL SAVREG
(3) 000546' 004767 000000G          JSR      PC,SAVREG
1099 000552'          LET R0 := DTBL(R5)
(4) 000552' 016500 000000          MOV      DTBL(R5),R0
1100 000556'          LET R2 := DT.MLST(R0)
(4) 000556' 016002 000032          MOV      DT.MLST(R0),R2
1101
1102 000562'          LET R3 := #AP.MAP
(4) 000562' 012703 000100'        MOV      #AP.MAP,R3
1103
1104          ;+
1105          ; WHILE NOT AT END OF MODULE LIST, GET MODULE HEADER ADDR.; IF BYTE
1106          ; IN APT MAP = 0, DESELECT MODULE; IF NOT SELECT MODULE AND SET
1107          ; UP DEVICE COUNT
1108          ; -
1109
1110 000566'          WHILE (R2) NE #0 DD
(4) 000566'
(6) 000566' 005712          TST      (R2)
(9) 000570' 001416          BEQ      50003$
1111          LET R1 := (R2)+
(4) 000572' 012201          MOV      (R2)+,R1
1112          IFB (R3) EQ #0 THEN
(6) 000574' 105713          TSTB   (R3)
(9) 000576' 001004          BNE      50004$
1113
1114          ;+
1115          ; IF ZERO IN APT TABLE, DESELECT THE MODULE
1116          ; -
1117
1118          LET STAT(R1) := STAT(R1) CLR.BY #BIT14
(6) 000600' 042761 040000 000026          BIC      #BIT14,STAT(R1)
1119          ELSE
(4) 000606' 000405          BR       50005$
(3) 000610'
1120          50004$:
1121          LET STAT(R1) := STAT(R1) SET.BY #BIT14
(6) 000610' 052761 040000 000026          BIS      #BIT14,STAT(R1)
1122          LET DVID1(R1) :B= (R3)
(4) 000616' 111361 000014          MOVB   (R3),DVID1(R1)
1123          ENDIF
(4) 000622'          50005$:
1124
1125          ;+
1126          ; ADVANCE TO NEXT IN MAP
1127          ; -
1128

```

APTMOD (APT MODULE PACKAGE)  
APTMOD.MAC 14-SEP-78 12:22

MACY11 30A(1052) 20-SEP-78 17:28 PAGE 19-15  
APTSEL ROUTINE

SEQ 0081

```
1129 000622'          INLINE <TSTB   (R3)+>
(2) 000622' 105723          TSTB   (R3)+
1130 000624'          ENDDO          BR   50002$
(4) 000624' 000750          50003$:
(3) 000626'
1131
1132
1133          ;+
1134          ; RESTORE REGISTERS
1135          ;-
1136 000626'          CALL RESREG
(3) 000626' 004767 000000G          JSR   PC,RESREG
1137 000632'          ENDRTN
(3) 000632'          50000$:
(3) 000632'          50001$:
(2) 000632' 000207          RTS   PC
1138
```



APTMOD (APT MODULE PACKAGE)  
APTMOD.MAC 14-SEP-78 12:22

MACY11 30A(1052) 20-SEP-78 17:23 PAGE 19-16  
APTKIL - KILL APT BY HALTING

SEQ 0082

```
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
```

```

1187          .SBTTL  APTKIL ROUTINE
1188
1189 000634'          ROUTINE APTKIL <TAB>
(2) 000634'
1190
1191          ;+
1192          ; SAVE REGISTERS AND DTABLE ADDR.
1193          ; -
1194
1195 000634'          CALL SAVREG
(3) 000634' 004767 000000G          JSR      PC, SAVREG
1196 000640'          LET R0 := TAB(R5)
(4) 000640' 016500 000000          MOV      TAB(R5), R0
1197
1198          ;+
1199          ; COPY DT.ERR INTO R1 AND MASK OUT FATAL ERROR BITS
1200          ; -
1201
1202 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
1203
1204          ;+
1205          ; IF COPY OF DT.ERR IS NON ZERO, MOVE 'MO' TO WORD 2.
1206          ; SHIFT DT.ERR COPY UNTIL ZERO; ADVANCE R2 TO POINT TO PROPER
1207          ; ADDRESS OF 2 LETTER MSG
1208          ; -
1209
1210 000654'          IF R1 NE #0 THEN
(6) 000654' 005701          TST      R1
(9) 000656' 001416          BEQ      50002$
1211          LET AP.MB2 := AP.MC0
(4) 000660' 016767 177254 177130          MOV      AP.MC0, AP.MB2
1212          LET R2 := #AP.MC1
(4) 000666' 012702 000142'          MOV      #AP.MC1, R2
1213          LET R1 := R1 SHIFT -1
(7) 000672' 005201          ASR      R1
1214          WHILE R1 NE #0 DO
(4) 000674'          50003$:
(6) 000674' 005701          TST      R1
(9) 000676' 001404          BEQ      50004$
1215          LET R1 := R1 SHIFT -1
(7) 000700' 006201          ASR      R1
1216          LET R2 := R2 + #2
(6) 000702' 062702 000002          ADD      #2, R2
1217          ENDDO
(4) 000706' 000772          BR      50004$
(3) 000710'
1218
1219
1220          ;+
1221          ; STUFF 2 LETTER CODE TO MB3
1222          ; -
1223
1224          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	BIT08 = 000400	DT.FCH= 000037	INIT = 000030	MSGSFT= 000006
ACTBIT= 004000	BIT09 = 001000	DT.FCN= 000036	INTR = 000120	MSGSKE= 000003
ADDR22= 001000	BIT1 = 000002	DT.HMX= 000104	IOMOD = 100000	MSGSMB= 000015
ADR = 000006	BIT10 = 002000	DT.KBE= 000024	IOMODP= 102000	MSGSMH= 000014
APTOED 000402RG	BIT11 = 004000	DT.KBP= 000026	IOMODR= 112000	MSGSMS= 000016
APTFER= 000004	BIT12 = 010000	DT.KBR= 000022	IOMODX= 110000	MSGSTD= 000000
APThER 000470RG	BIT13 = 020000	DT.KBU= 000030	JACK = 035060	MSGSYS= 000012
APT.KIL 000634RG	BIT14 = 040000	DT.MLS= 000032	KIPAR0= 172340	MSGVEC= 000020
APTPRE= 000200	BIT15 = 100000	DT.MTI= 000110	KIPAR1= 172342	N3KMOD= 001000
APTSEL 000546RC	BIT2 = 000004	DT.OFF= 000070	KIPAR2= 172344	NCPUP= 000020
APTSEr 000326RG	BIT3 = 000010	DT.PAS= 000074	KIPAR3= 172346	NDPTY= 000002
APTSLP 000164RG	BIT4 = 000020	DT.PC = 000002	KIPAR4= 172350	NULL = 000000
AP.CPU 000042R	BIT5 = 000040	DT.PFL= 000062	KIPAR5= 172352	OWEN = 024020
AP.CR 000162R	BIT6 = 000100	DT.PSW= 000004	KIPAR6= 172354	PAERR = 000010
AP.END 000140R	BIT7 = 000200	DT.PTA= 000064	KIPAR7= 172356	PARPRE= 002000
AP.ET1 000034R	BIT8 = 000400	DT.RCS= 000102	KIPDR0= 172300	PARSTA= 000100
AP.FAK 000160R	BIT9 = 001000	DT.REL= 000040	KIPDR1= 172302	PASCNT= 000034
AP.MAP 000100R	BKDEF = 000002	DT.SCT= 000066	KIPDR2= 172304	PDPLSI= 020000
AP.MB1 000014R	BKMOD = 000020	DT.SMX= 000106	KIPDR3= 172306	PDP60 = 004000
AP.MB2 000016R	BKMODE= 040000	DT.SP = 000006	KIPDR4= 172310	PDP70 = 010000
AP.MB3 000020R	BKSLSH= 000134	DT.SSI= 000046	KIPDR5= 172312	PRI0 = 000000
AP.MB4 000022R	CAPRES= 000004	DT.ST0= 000010	KIPDR6= 172314	PRI1 = 000040
AP.MB5 000024R	CASTAT= 000004	DT.ST1= 000012	KIPDR7= 172316	PRI4 = 000200
AP.MCC 000140R	CDERCT= 000146	DT.SWR= 000056	KSUM = ***** G	PRI5 = 000240
AP.MC1 000142R	CDWDCT= 000144	DT.SYP= 000072	KTERRO= 000040	PRI6 = 000300
AP.MM0 000044R	CKTIM = 100000	DT.WBU= 000050	KTPRES= 000400	PRI7 = 000340
AP.MM1 000050R	CLKPRE= 000001	DT.WHL= 000054	KTSTAT= 000020	PR0 = 000000
AP.MM2 000054R	CONFIG= 000056	DT.WLL= 000052	KTXTND= 040000	PR4 = 000200
AP.MM3 000060R	CQDVF = 000001	DVID1 = 000014	LF = 000012	PR5 = 000240
AP.PB 000000RG	CR = 000015	ECCMEM= 000100	LPSTAT= 000001	PR6 = 000300
AP.P00 000000R	CSRA = 000100	ECCSTA= 000010	MAPSTA= 000200	PR7 = 000340
AP.P02 000002R	CSRC = 000102	ENBEOp= 010000	MD = 000002	PS = 177776
AP.P04 000004R	CTRLC = 000003	ENBNUL= 000001	MD.COD= ***** G	PSW = 177776
AP.P06 000006R	CTRL0 = 000017	ENDLST= 000000	MED = 076600	RANNUM= 000054
AP.P10 000010R	CTRLU = 000025	EOPBIT= 000001	MEMPAS= 040000	RBUFEA= 000130
AP.P12 000012R	DCEVNT= 000011	ERRTYP= 000106	MODEXH= 004000	RBUFPA= 000126
AP.SR1 000036R	DEFRTN= 000400	EVNTBE= 000200	MODHOL= 002000	RBUFSA= 000132
AP.SR2 000040R	DIAGMC= 000000	EVNTHD= 000200	MODSEL= 001000	RBUFVA= 000124
ASB = 000106	DROPMO= 100000	EVNTKT= 000203	MSGCKD= 000010	RCSR = ***** G
ASSEMB= 000010	DSEVNT= 000014	EVNTPE= 000202	MSGCKS= 000011	RDSERV= 000101
ASTAT = 000104	DTA = 000000	EVNTRE= 000201	MSGDEQ= ***** G	RDWHMI= 000022
AUTO = 000010	DTADR = 000000	FATERR= 100000	MSGDER= 000005	RELERR= 000020
AUTOST= 020000	DTBL = 000000	HRDCNT= 000044	MSGDRP= 000017	RELMOD= 020000
AWAS = 000110	DT.ADD= 000042	HRDPAS= 000050	MSGECH= 177777	RELTIM= 010000
BIT0 = 000001	DT.AP = 000100	ICONT = 000036	MSGEOP= 000013	RESREG= ***** G
BIT00 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGHDR= 000004	RES1 = 000056
BIT01 = 000002	DT.BLS= 000034	IDNUM = 000122	MSGHNG= 000022	RES2 = 000060
BIT02 = 000004	DT.CFC= 000014	IE = 000100	MSGHRD= 000007	RICHAR= 031060
BIT03 = 000010	DT.CF1= 000016	INDPAR= 000040	MSGMAP= 000021	RPTDAT= 002000
BIT04 = 000020	DT.ERR= 000020	INHDRP= 040000	MSGNUL= 177775	RSTRCY= ***** G
BIT05 = 000040	DT.ESI= 000044	INHEPR= 020000	MSGPOP= 000002	RSTRT = 000112
BIT06 = 000100	DT.EVN= 000000	INHREL= 001000	MSGPRM= 177776	RUBOUT= 000177
BIT07 = 000200	DT.EXS= 000060	INHRR= 000400	MSGRES= 000001	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	TMPID = 000002	XFLAG = 000005	\$ISKO = 000001	\$\$LGC = 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	\$\$REU= 000000
SOFCNT= 000042	UIPAR2= 177644	\$ERFLG= 000400	\$LSTTA= 000001	\$\$RTN1= 050000
SOPAS= 000046	UIPAR3= 177646	\$F\$AND= 000310	\$NESTL= 177777	\$\$RTN2= 050001
SPACE = 000040	UIPAR4= 177650	\$F\$BAD= 000401	\$NSKO = 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	\$\$AVLE= 177777	\$\$TO = 000000
SR2 = 177576	UIPDR1= 177602	\$F\$FAL= 000405	\$\$SSKO = 050004	\$\$TAG= 050000
SR3 = 172516	UIPDR2= 177604	\$F\$G00= 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\$L00= 000200	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:29  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0087

3 COMMON EQUATE MODULE  
538 BUSERR PROCESS BUS ERROR TRAPS AND RESERVED INS. TRAPS  
605 COMMON DEFINITIONS AND REFERENCES FOR BUSERR  
608 000000' .PRINT ;SPMAC: VERSION 1.1  
685 BUSERR ROUTINE  
941 MODCHK GET OPTION MODULE HEADER ADDRESS AND OFFSET  
958 MODCHK ROUTINE

```
508
509
510 .TITLE BESRV -BUSS ERROR ROUTINE
511 .IDENT /0.0/
512
513 ;++
514 ; MODULE PACKAGE NAME:
515 ;     BESRV
516 ;
517 ; FUNCTIONAL DESCRIPTION:
518 ;     IT CONSISTS OF THE FOLLOWING MODULES:
519 ;         1. BUSERR - PROCESS THE PC,PSW,SP & VECTOR AT THE TIME
520 ;             OF THE TRAP TO 4 OR 10, AND
521 ;         2. MODCHK - INCREMENT THE # OF SYS ERRORS BY AN OPTION
522 ;             MODULE AND RETURN THE MODULE NAME AND OFFSET PC.
523 ;             IT WILL ALSO DROP THE OPTION MODULE IF TOO MANY SYS
524 ;             ERRORS HAVE BEEN CAUSED BY IT.
525 ;     NOTE: IT IS ASSUMED THE PROCESSOR IS AT PRIORITY 7
526 ;           WHEN THIS ROUTINE IS ENTERED.
527 ;
528 ; VERSION:
529 ;     0.0
530 ;
531 ;     EDIT             DATE             BY             REASON
532 ;--
533
534
535
536
```

```
538 .SBTTL BUSERR PROCESS BUS ERROR TRAPS AND RESERVED INS. TRAPS
539 .IDENT /V0.0/
540
541 ;++
542 ; MODULE NAME:
543 ;     BUSERR
544 ;
545 ; FUNCTIONAL DESCRIPTION:
546 ;
547 ;     THIS ROUTINE WILL PROCESS A BUS ERROR TRAP AND A RESERVED
548 ;     INSTRUCTION TRAP. IT WILL INCREMENT
549 ;     THE EXERCISER SYSTEM ERROR COUNT AND SET A SYSTEM
550 ;     ERROR BIT IN THE ERROR WORD. IT WILL FORM AND QUEUE
551 ;     A SYSTEM ERROR MESSAGE CONTAINING THE PC, PSW, AND SP AT
552 ;     THE TIME OF THE SYSTEM ERROR. IF THE SYSTEM ERROR WAS
553 ;     CAUSED BY AN OPTION MODULE - THE OPTION MODULE NAME
554 ;     AND OFFSET ADDRESS WHERE SYSTEM ERROR OCCURRED WILL ALSO
555 ;     BE OUTPUTTED.
556 ;
557 ; INPUTS:
558 ;     ADDRESS OF DATA TABLE
559 ;
560 ; IMPLICIT INPUTS:
561 ;     1. DT.CFO
562 ;     2. DT.EVNT
563 ;     3. DT.MLST
564 ;     4. DT.PSW
565 ;     5. DT.SP
566 ;     6. DT.PC
567 ;     7. DT.ESIZ
568 ;
569 ; OUTPUTS:
570 ;     NONE
571 ;
572 ; IMPLICIT OUTPUTS:
573 ;     1. DT.ERR
574 ;     2. DT.EXS
575 ;
576 ; PATHOLOGICAL CONNECTIONS:
577 ;     NONE
578 ;
579 ; SUBORDINATE ROUTINES CALLED:
580 ;     1. RSTRCY     RECOVER FROM RESET
581 ;     2. ERRLOG     ERROR LOGGING
582 ;     3. MSGDHOOK   HOOK A MESSAGE
583 ;     4. BOA16      16 BIT BINARY TO OCTAL ASCII CONVERSION ROUTINE
584 ;     5. BDACNV     BINARY TO DECIMAL ASCII CONVERSION ROUTINE
585 ;     6. GPA        GET PHYSICAL ADDRESS ROUTINE
586 ;     7. SAVREG     SAVE REGISTERS
587 ;     8. RESREG     RESTORE REGISTERS
588 ;     9. BOAC       BINARY TO OCTAL ASCII CONVERSION
589 ;
590 ; FUNCTIONAL SIDE EFFECTS:
591 ;     NONE
592 ;
593 ; CALLING SEQUENCE:
```



BESRV -BUGS ERROR ROUTINE  
BESRV.MAC 14-SEP-78 11:16

MACY11 30A(1052) 20-SEP-78 17:29 PAGE 19-2  
BUSERR PROCESS BUS ERRDR TRAPS AND RESERVED INS. TRAPS

SEQ 0090

```
594 ; CALL BUSERR IN <DTADR>
595 ;
596 ; WHERE DTADR = ADDRESS OF DATA TABLE
597 ;
598 ; VERSION:
599 ; 0.0
600 ;
601 ; EDIT DATE BY REASON
602 ;--
603
```

```
605          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR BUSERR
606          ;
607          .MCALL STRUCT
608 000000'   STRUCT
(1) 000000'   .PRINT ;SPMAC: VERSION 1.1
609          000001   $LSTIN=1
610          000001   $LSTTAG=1
611
612
613          ;
614          ;*****
615          ;
616          ; REFERENCED BY OTHER MODULES
617          ;
618          .GLOBL BUSERR          ;ROUTINE'S ENTRY ADDRESS
619          .GLOBL BE.MCNT        ;# OF SYS ERRORS BY MONITOR
620          ;
621          ;*****
622          ;
623          ; GLOBAL REFERENCES
624          ;
625          .GLOBL MSGDHOOK          ; TT INIT ROUTINE
626          .GLOBL SAVREG          ;SAVE REGISTERS
627          .GLOBL RESREG          ;RESTORE REGISTERS
628          .GLOBL GPA            ;GET PHYSICAL ADDRESS
629          .GLOBL BOA16          ;ROUTINE TO CONVERT BINARY TO OCTAL ASCII
630          .GLOBL BOAC           ;CONVERT 18 OR 22 BIT PHYS ADDR TO OCTAL ASCII
631          .GLOBL BDACNV         ;BINARY TO DECIMAL ASCII CONVERSION RTNE
632          .GLOBL ERRLOG         ;ERROR LOGGING
633          .GLOBL DRPMOD         ;DROP A MODULE
634          .GLOBL RSTRCY         ;RECOVER CPU REGS AFTER RESET
635          ;
636          ;*****
637          ;
638          ; LOCAL STORAGE
639          ;
640 000000' 000000   BE.VA: 0          ;VIRTUAL ADDRESS
641 000002' 000000   BE.PA: 0          ;LOWER 16 BITS OF PHYSICAL ADDRESS
642 000004' 000000   BE.EA: 0          ;EXTENDED ADDRESS WORD
643 000006' 000000   BE.VPC: .WORD 0   ;STORE PC
644 000010' 000000   BE.VEC: .WORD 0   ;STORE EITHER A 4 OR 10 (THE VECTOR LOC.)
645 000012' 000000   BE.OFFPC: .WORD 0 ;STORE OFFSET PC HERE
646 000014' 000000   BE.PSW: .WORD 0   ;STORE PSW
647 000016' 000000   BE.STK: .WORD 0   ;STORE STACK POINTER
648 000020' 000012   BE.MX2: .WORD ^D<10> ;MAX # OF ALLOWABLE SYS ERRORS BY THE MONITOR
649 000022' 000000   BE.MCNT: .WORD 0   ;LOC TO HOLD NUMBER OF SYS ERRORS BY MONITOR
650 000024' 000036   BE.MX3: .WORD ^D<30> ;MAX # OF ALLOWABLE SYS ERRORS BY EXERCISER
651 000026' 000004   BE.MAX: .WORD ^D<4>  ;MAX NO. OF SYS ERRORS ALLOWED BY AN OPTION MODULE
652
653
654          ;+
655          ; THE ORDER OF THE FOLLOWING LABELS SHOULD NOT BE CHANGED
656          ; BECAUSE THEY ARE PART OF THE SYSTEM ERROR MESSAGE
657          ;-
658
659 000030' 022445 025052 020040 BE.SEMSG: .ASCII /%%** SYSTEM ERROR **%/
```



```

685          .SBTTL  BUSERR ROUTINE
686
687 000244'  ROUTINE BUSERR <TABADR>
688
689
690          ;+
691          ; SAVE REGISTERS AND GET DTABLE
692          ; -
693 000244'  CALL SAVREG
694 000250' 004767 000000G          JSR      PC, SAVREG
695
696
697          LET R0 := TABADR(R5)          MOV      TABADR(R5), R0
698
699          ;+
700          ; GET EVENT CODE AND STORE VECTOR ADDRESS 4 OR 10 IN BE.VEC
701          ; 4 = BUS ERROR AND 10 = RESERVED INSTRUCTION
702          ; -
703 000254'  IF DT.EVNT(R0) EQ #EVNTBE THEN
704 000254' 026027 000000 000200          CMP      DT.EVNT(R0), #EVN
705 000262' 001004          BNE      50002$
706          LET BE.VEC := #4          MOV      #4, BE.VEC
707
708          ELSE
709          BR      50003$
710          LET BE.VEC := #10          MOV      #10, BE.VEC
711          ENDIF          MOV      #10, BE.VEC
712
713          ;+
714          ; SAVE PC, PSW, AND STACK POINTER AT TIME OF TRAP
715          ; -
716 000302'  LET BE.VPC := DT.PC(R0)          MOV      DT.PC(R0), BE.VPC
717 000310' 016067 000002 177476          LET BE.PSW := DT.PSW(R0)          MOV      DT.PSW(R0), BE.PS
718 000316' 016067 000004 177476          LET BE.STK := DT.SP(R0)          MOV      DT.SP(R0), BE.STK
719
720          ;+
721          ; SET SYS ERROR OCCURRED BIT IN ERROR WORD
722          ; -
723 000324'  LET DT.ERR(R0) := DT.ERR(R0) SET.BY #SYSERR          BIS      #SYSERR, DT.ERR(R
724
725          ;+
726          ; INCREMENT EXERCISER SYS ERROR COUNT
727          ; -

```

```
727
728 000332' LET DT.EXS(R0) := DT.EXS(R0) + #1
(6) 000332' 005260 000060 INC DT.EXS(R0)
729
730
731 ;+
732 ; IF EXERCISER SYS ERROR COUNT >= MAX,ALSO SET FATAL ERROR BIT IN DT.ERR
733 ; -
734
735 000336' IF DT.EXS(R0) HIS BE.MX3 THEN
(6) 000336' 026067 000060 177460 CMP DT.EXS(R0),BE.MX
(9) 000344' 103403 BLO 50004$
736 000346' LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6) 000346' 052760 100000 000020 BIS #FATERR,DT.ERR(R
737 000354' ENDIF
(4) 000354' 50004$:
738
739
740
741 ;+
742 ; GET OPTION MODULE LIST POINTER. IF THE PC IS GREATER THAN
743 ; STARTING ADDRESS OF MODULES,
744 ; BUT LESS THAN THE HIGHEST ADDRESS OF EXERCISER,
745 ; DETERMINE WHICH MODULE CAUSED THE
746 ; ERROR BY CALLING MODCHK. R1 WILL CONTAIN STARTING ADDRESS
747 ; OF OFFENDING MODULE WHEN OPERATION COMPLETE.
748 ; BE.OFFPC WILL CONTAIN OFFSET PC INTO A MODULE.
749 ; IF NOT A MODULE
750 ; INCREMENT # OF SYS ERRORS CAUSED BY MONITOR AND IF TOO MANY,SET FATAL
751 ; ERROR BIT IN ERROR WORD, AND SET BE.OFFPC TO 0.
752 ; -
753
754 000354' LET R1 := DT.MLST(R0)
(4) 000354' 016001 000032 MOV DT.MLST(R0),R1
755
756 ;+
757 ; GET THE LOWEST ADDRESS IN THE MODULE LIST
758 ; -
759 000360' LET R4 := (R1)
(4) 000360' 011104 MOV (R1),R4
760 000362' WHILE (R1) NE #0 DO
(4) 000362' 50005$:
(6) 000362' 005711 TST (R1)
(9) 000364' 001405 BEQ 50006$
761 000366' IF R4 HI (R1) THEN
(6) 000366' 020411 CMP R4,(R1)
(9) 000370' 101401 BLOS 50007$
762 000372' LET R4 := (R1)
(4) 000372' 011104 MOV (R1),R4
763 000374' ENDIF
(4) 000374' 50007$:
764 000374' INLINE <TST (R1)+>
(2) 000374' 005721 TST (R1)+
765 000376' ENDDO
(4) 000376' 000771 BR 50005$
(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)+ := (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

```

```

799
800      ;+
801      ; IF TOO MANY SYS ERRORS CAUSED BY MONITOR SET FATAL BIT IN ERROR WORD
802      ; -
803      000476'          IF BE.MCNT HIS BE.MX2 THEN
(6) 000476' 026767 177320 177314          CMP      BE.MCNT, BE.MX2
(9) 000504' 103403          BLO      50014$
804      000506'          LET DT.ERR(R0) := DT.ERR(R0) SET.BY #FATERR
(6) 000506' 052760 100000 000020          BIS      #FATERR, DT.ERR(R
805      000514'          ENDIF
(4) 000514'          50014$:
806      000514'          ENDIF
(4) 000514'          50013$:
807
808
809
810
811
812
813
814
815      ;+
816      ; CONVERT VECTOR TO ASCII
817      ; -
818
819      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
820
821
822
823      ;+
824      ; CONVERT VPC TO ASCII
825      ; -
826
827      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
828
829
830
831      ;+
832      ; CONVERT PHYSICAL ADDRESS TO ASCII
833      ; -
834
835      000554'          LET BE.VA := BE.VPC
(4) 000554' 016767 177226 177216          MOV      BE.VPC, BE.VA
836      000562'          CALL GPA IN <R0, #BE.VA>
(3) 000562' 010546          MOV      R5, -(SP)
(5) 000564' 012745 000000'          MOV      #BE.VA, -(R5)

```

(4) 000570' 010045  
(3) 000572' 004767 000000G  
(3) 000576' 012605  
837 000600'  
(3) 000600' 010546  
(7) 000602' 012745 000160'  
(6) 000606' 016745 177172  
(5) 000612' 016745 177164  
(4) 000616' 010045  
(3) 000620' 004767 000000G  
(3) 000624' 012605

CALL BOAC IN <R0,BE.PA,BE.EA,#BE.APPC>

MOV R0,-(R5)  
JSR PC,GPA  
MOV (SP)+,R5

MOV R5,-(SP)  
MOV #BE.APPC,-(R5)  
MOV BE.EA,-(R5)  
MOV BE.PA,-(R5)  
MOV R0,-(R5)  
JSR PC,BOAC  
MOV (SP)+,R5

833  
839 ;+  
840 ; CONVERT PSW TO ASCII  
841 ;-

842  
843 000626'  
(3) 000626' 010546  
(5) 000630' 012745 000172'  
(4) 000634' 016745 177154  
(3) 000640' 004767 000000G  
(3) 000644' 012605

CALL BOA16 IN <BE.PSW,#BE.APSW>

MOV R5,-(SP)  
MOV #BE.APSW,-(R5)  
MOV BE.PSW,-(R5)  
JSR PC,BOA16  
MOV (SP)+,R5

844  
845  
846  
847 ;+  
848 ; CONVERT STACK TO ASCII  
849 ;-

850  
851 000646'  
(3) 000646' 010546  
(5) 000650' 012745 000202'  
(4) 000654' 016745 177136  
(3) 000660' 004767 000000G  
(3) 000664' 012605

CALL BOA16 IN <BE.STK,#BE.ASTK>

MOV R5,-(SP)  
MOV #BE.ASTK,-(R5)  
MOV BE.STK,-(R5)  
JSR PC,BOA16  
MOV (SP)+,R5

852  
853 ;+  
854 ; CONVERT SYS ERROR COUNT TO ASCII  
855 ;-

856  
857 000666'  
(3) 000666' 010546  
(5) 000670' 012745 000212'  
(4) 000674' 016045 000060  
(3) 000700' 004767 000000G  
(3) 000704' 012605

CALL BDACNV IN <DT.EXS(R0),#BE.AEXS>

MOV R5,-(SP)  
MOV #BE.AEXS,-(R5)  
MOV DT.EXS(R0),-(R5)  
JSR PC,BDACNV  
MOV (SP)+,R5

858  
859  
860 ;+  
861 ; RESET THE WORLD TO STOP ANY INTERRUPTS  
862 ;-

863  
864 000706' 000005  
865  
866  
867  
RESET



```
868
869
870 ;+
871 ; RECOVER THE BAD THINGS THAT THE RESET DID
872 ; AND GET THE PRIORITY BACK DOWN TO ZERO SO WE CAN TYPE
873 ; -
874
875
376 000710' CALL RSTRCY IN <R0>
(3) 000710' 010546
(4) 000712' 010045
(3) 000714' 004767 000000G
(3) 000720' 012605
877 000722' LET -(SP) := #0
(4) 000722' 005046
378 000724' LET -(SP) := #456$
(4) 000724' 012746 000732'
379
880 000730' INLINE <RTI>
(2) 000730' 000002
881 000732' INLINE <456$:>
(2) 000732'
882
883
884 ;+
885 ; QUEUE THE SYSTEM ERROR MESSAGE (STANDARD SYSTEM ERROR)
886 ; -
887
888 000732' CALL MSGDHOOK IN <R0,#MSGPOP,#BE.SEMSG,#2$>
(3) 000732' 010546
(7) 000734' 012745 000764'
(6) 000740' 012745 000030'
(5) 000744' 012745 000002
(4) 000750' 010045
(3) 000752' 004767 000000G
(3) 000756' 012605
889
890 000760' INLINE <1$: NOP>
(2) 000760' 000240
891 000762' INLINE <BR 1$>
(2) 000762' 000776
892
893
894 ;+
895 ; IF ERROR WAS CAUSED BY OPTION MODULE, ALSO QUEUE THE OPTION
896 ; MODULE NAME AND APC
897 ; -
898
899 2$: IF BE.OFPC NE #0 THEN
(6) 000764' 005767 177022
(9) 000770' 001415
900 000772' CALL MSGDHOOK IN <R0,#MSGPOP,#BE.ATMSG,#4$>
(3) 000772' 010546
(7) 000774' 012745 001024'
(6) 001000' 012745 000222'
(5) 001004' 012745 000002
```

MOV	R5,-(SP)
MOV	R0,-(R5)
JSR	PC,RSTRCY
MOV	(SP)+,R5
CLR	-(SP)
MOV	#456\$,-(SP)
RTI	
456\$:	
MOV	R5,-(SP)
MOV	#2\$,-(R5)
MOV	#BE.SEMSG,-(R5)
MOV	#MSGPOP,-(R5)
MOV	R0,-(R5)
JSR	PC,MSGDHOOK
MOV	(SP)+,R5
1\$:	NOP
BR 1\$	
TST	BE.OFPC
BEQ	50015\$
MOV	R5,-(SP)
MOV	#4\$,-(R5)
MOV	#BE.ATMSG,-(R5)
MOV	#MSGPOP,-(R5)

```
(4) 001010' 010045                                MOV    R0,-(R5)
(3) 001012' 004767 000000G                          JSR    PC,MSGDHQOK
(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
```

BESRV -BUSS ERROR ROUTINE  
BESRV.MAC 14-SEP-78 11:16

MACY11 30A(1052) 20-SEP-78 17:29 PAGE 19-12  
BUSERR ROUTINE

SEQ 0100

```
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'
(3) 001120'          50000$:
(2) 001120' 000207          50001$:
          RTS      PC
```

DESRV -BUSS ERROR ROUTINE  
BESRV.MAC 14-SEP-78 11:16

MACY11 30A(1052) 20-SEP-78 17:29 PAGE 19-13  
MODCHK GET OPTION MODULE HEADER ADDRESS AND OFFSET

SEQ 0101

```
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
```

```
958          .SBTTL  MODCHK ROUTINE
959
960 001122'    ROUTINE MODCHK
(2) 001122'
961
962          ;+
963          ; GET ADDRESS OF OPTION MODULE LIST,PUT IN R2
964          ; GET ADDRESS AT WHICH SYS ERROR OCCURRED INTO R3
965          ; -
966
967 001122'    LET R2 := DT.MLST(R0)
(4) 001122' 016002 000032          MOV      DT.MLST(R0),R2
968 001126'    LET R3 := DT.PC(R0)
(4) 001126' 016003 000002          MOV      DT.PC(R0),R3
969
970          ;+
971          ; INITIALIZATION COMPLETE
972          ; -
973
974
975          ;+
976          ; FIND THE SMALLEST ADDRESS THAT DOES NOT EXCEED THE
977          ; ADDRESS OF SYS ERROR
978          ; -
979
980
981 001132'    LET R1 := #0
(4) 001132' 005001          CLR      R1
982 001134'    WHILE (R2) NE #0 DO
(4) 001134'
(6) 001134' 005712          50002$: TST      (R2)
(9) 001136' 001407          BEQ      50003$
983 001140'    IF (R2) LOS R3 THEN
(6) 001140' 021203          CMP      (R2),R3
(9) 001142' 101003          BHI      50004$
984 001144'    IF (R2) HI R1 THEN
(6) 001144' 021201          CMP      (R2),R1
(9) 001146' 101401          BLOS     50005$
985 001150'    LET R1 := (R2)
(4) 001150' 011201          MOV      (R2),R1
986 001152'    ENDIF
(4) 001152'
987 001152'    ENDIF
(4) 001152'          50005$:
988 001152'    INLINE <TST      (R2)+>
(2) 001152' 005722          TST      (R2)+
989 001154'    ENDDO
(4) 001154' 000767          BR      50002$
(3) 001156'          50003$:
990
991
992
993          ;+
994          ; CALCULATE THE OFFSET ADDRESS
995          ; -
996
```

BESRV -BUSS ERROR ROUTINE  
BESRV.MAC 14-SEP-78 11:16

MACY11 30A(1052) 20-SEP-78 17:29 PAGE 19-15  
MODCHK ROUTINE

SEQ 0103

```

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          ;+
1001          ; ADD ONE TO OPTION MODULE'S SYS ERROR COUNT
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	BIT2 = 000004	DT.MTI= 000110	JACK = 035060	NBKMOD= 001000
ACTBIT= 004000	BIT3 = 000010	DT.OFF= 000070	KIPAR0= 172340	NCPUDP= 000020
ADDR22= 001000	BIT4 = 000020	DT.PAS= 000074	KIPAR1= 172342	NDAPTY= 000002
ADR = 000006	BIT5 = 000040	DT.PC = 000002	KIPAR2= 172344	NULL = 000000
APTFER= 000004	BIT6 = 000100	DT.PFL= 000062	KIPAR3= 172346	DWEN = 024020
APTPRE= 000200	BIT7 = 000200	DT.PSW= 000004	KIPAR4= 172350	PAERR = 000010
ASB = 000106	BIT8 = 000400	DT.PTA= 000064	KIPAR5= 172352	PARPRE= 002000
ASSEMB= 000010	BIT9 = 001000	DT.RCS= 000102	KIPAR6= 172354	PARSTA= 000100
ASTAT = 000104	BKDEF = 000002	DT.REL= 000040	KIPAR7= 172356	PASCNT= 000034
AUTO = 000010	BKMOD = 000020	DT.SCT= 000066	KIPDR0= 172300	PDPLSI= 020000
AUTOST= 020000	BKMODE= 040000	DT.SMX= 000106	KIPDR1= 172302	PDP60 = 004000
AWAS = 000110	BKSLSH= 000134	DT.SP = 000005	KIPDR2= 172304	PDP70 = 010000
BDACNV= ***** G	BOAC = ***** G	DT.SSI= 000046	KIPDR3= 172306	PRI0 = 000000
BE.AEX 000212R	BOA16 = ***** G	DT.ST0= 000010	KIPDR4= 172310	PRI1 = 000040
BE.AMO 000226R	BUSERR 000244RG	DT.ST1= 000012	KIPDR5= 172312	PRI4 = 000200
BE.ADF 000234R	CAPRES= 000004	DT.SWR= 000056	KIPDR6= 172314	PRI5 = 000240
BE.APP 000160R	CASTAT= 000004	DT.SYP= 000072	KIPDR7= 172316	PRI6 = 000300
BE.APS 000172R	CDCERCT= 000146	DT.WBU= 000050	KTPRES= 000040	PRI7 = 000340
BE.AST 000202R	CDWDCT= 000144	DT.WHL= 000054	KTSTAT= 000020	PRC = 000000
BE.ATM 000222R	CKTIM = 100000	DT.WLL= 000052	KTXTND= 040000	PR4 = 000200
BE.AVE 000140R	CLKPRE= 000001	DVID1 = 000014	LF = 000012	PR5 = 000240
BE.AVP 000150R	CONFIG= 000056	ECCMEM= 000100	LPSTAT= 000001	PR6 = 000300
BE.EA 000004R	CQDVF = 000001	ECCSTA= 000010	MAPSTA= 000200	PR7 = 000340
BE.MAX 000026R	CR = 000015	ENBEOP= 010000	MED = 076600	PS = 177776
BE.MCN 000022RG	CSRA = 000100	ENBNUL= 000001	MEMPAS= 040000	PSW = 177776
BE.MX2 000020R	CSRC = 000102	ENDLST= 000000	MODCHK 001122R	RANNUM= 000054
BE.MX3 000024R	CTRLC = 000003	EOPBIT= 000001	MODEXH= 004000	RBUFEA= 000130
BE.DFP 000012R	CTRL0 = 000017	ERRLOG= ***** G	MODHOL= 002000	RBUFPA= 000126
BE.PA 000002R	CTRLU = 000025	ERRTYP= 000106	MODSEL= 001000	RBUFSZ= 000132
BE.PSW 000014R	DCEVNT= 000011	EVNTBE= 000200	MSGCKD= 000010	RBUFVA= 000124
BE.SEM 000030R	DEFRTN= 000400	EVNTHD= 000200	MSGCKS= 000011	RDSEVR= 000101
BE.STK 000016R	DIAGMC= 000000	EVNTKT= 000203	MSGDER= 000005	RDWHMI= 000022
BE.VA 000006R	DROPMD= 100000	EVNTPE= 000202	MSGDHO= ***** G	RELERR= 000020
BE.VEC 000010R	DRPMOD= ***** G	EVNTRE= 000201	MSGDRP= 000017	RELMOD= 020000
BE.VPC 000006R	DSEVNT= 000014	FATERR= 100000	MSGECH= 177777	RELTIM= 010000
BIT0 = 000001	DT.ADD= 000042	GPA = ***** G	MSGGEP= 000013	RESREG= ***** G
BIT00 = 000001	DT.AP = 000100	HRDCNT= 000044	MSGHDR= 000004	RES1 = 000056
BIT01 = 000002	DT.APK= 000076	HRDPAS= 000050	MSGHNG= 000022	RES2 = 000060
BIT02 = 000004	DT.BLS= 000034	ICONT = 000036	MSGHRD= 000007	RICHAR= 031060
BIT03 = 000010	DT.CFO= 000014	ICOUNT= 000040	MSGMAP= 000021	RPTDAT= 002000
BIT04 = 000020	DT.CF1= 000016	IDNUM = 000122	MSGNUL= 177775	RSTRCY= ***** G
BIT05 = 000040	DT.ERR= 000020	IE = 000100	MSGPOP= 000002	RSTRT = 000112
BIT06 = 000100	DT.ESI= 000044	INDPAR= 000040	MSGPRM= 177776	RUBOUT= 000177
BIT07 = 000200	DT.EVN= 000000	INHDRP= 040000	MSGRES= 000001	RUNMOD= 100000
BIT08 = 000400	DT.EXS= 000060	INHREL= 001000	MSGSFT= 000006	R5VALU= 001740
BIT09 = 001000	DT.FCH= 000037	INHRRR= 000400	MSGSKS= 000003	SAM = 075464
BIT1 = 000002	DT.FCN= 000036	INIT = 000030	MSGSMB= 000015	SAVREG= ***** G
BIT10 = 002000	DT.HMX= 000104	INTR = 000120	MSGSMH= 000014	SBADR = 000102
BIT11 = 004000	DT.KBE= 000024	IOMOD = 100000	MSGSMS= 000016	SBKMOD= 000000
BIT12 = 010000	DT.KBP= 000026	IOMODP= 102000	MSGSTD= 000000	SBKSEL= 010000
BIT13 = 020000	DT.KBR= 000022	IOMODR= 112000	MSGSYS= 000012	SC.ADR= 000006
BIT14 = 040000	DT.KBU= 000030	IOMODX= 110000	MSGVEC= 000020	SC.ALC= 000014
BIT15 = 100000	DT.MLS= 000032			SC.APC= 000016

SC.CKL= 000002	SVR6 = 000076	WBUFPA= 000134	\$F\$OR = 000320	\$TSK0 = 050002
SC.CKP= 000004	SYSCNT= 000052	WBUFQR= 000140	\$F\$RTI= 000350	\$TSK1 = 050003
SC.CLO= 000000	SYSERR= 000100	WBUFQZ= 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	\$SARGC= 000000
SENDLS= 177777	TQOVF = 000002	WTINRE= 000352	\$F\$TRU= 000404	\$S\$BYTE= 000403
SQFCNT= 000042	UIPAR0= 177640	WTWHMI= 000222	\$F\$UNT= 000130	\$S\$CASE= 000000
SQFPAS= 000046	UIPAR1= 177642	XFLAG = 000005	\$F\$WHI= 000120	\$S\$DST = 000000
SPACE = 000040	UIPAR2= 177644	XOFF = 000023	\$F\$YES= 000402	\$S\$ELOC= 000402
SPOINT= 000032	UIPAR3= 177646	XON = 000021	\$IFLEV= 177777	\$S\$ERFL= 000000
SPVALU= 002200	UIPAR4= 177650	\$BGNLE= 177777	\$ISK0 = 000001	\$S\$FLAG= 000001
SR0 = 177572	UIPAR5= 177652	\$ERFLG= 000400	\$ISK1 = 000001	\$S\$FROM= 000000
SR1 = 177574	UIPAR6= 177654	\$F\$AND= 000310	\$LOCTA= 177777	\$S\$LOC = 001146R
SR2 = 177576	UIPAR7= 177656	\$F\$BAD= 000401	\$LSTIN= 000001	\$S\$LOCN= 000000
SR3 = 172516	UIPDR0= 177600	\$F\$BLA= 000170	\$LSTTA= 000001	\$S\$REG = 177777
STAT = 000026	UIPDR1= 177602	\$F\$CAS= 000150	\$NESTL= 177777	\$S\$RETU= 000000
STATBI= C64757	UIPDR2= 177604	\$F\$DEC= 000220	\$NSK0 = 000300	\$S\$RTN1= 050000
STAT1 = 000027	UIPDR3= 177606	\$F\$DO = 000340	\$NSK1 = 000120	\$S\$RTN2= 050001
SUSPND= 000001	UIPDR4= 177610	\$F\$FAL= 000405	\$NSK2 = 000110	\$S\$SRC = 000000
SVR0 = 000062	UIPDR5= 177612	\$F\$GDO= 000400	\$NSK3 = 000110	\$S\$TGSV= 000000
SVR1 = 000064	UIPDR6= 177614	\$F\$IF = 000110	\$SAVLE= 177777	\$S\$TGS1= 000000
SVR2 = 000066	UIPDR7= 177616	\$F\$INC= 000210	\$SSK0 = 050003	\$S\$TGS2= 000000
SVR3 = 000070	WASADR= 000104	\$F\$L00= 000200	\$TAGLE= 177777	\$S\$TO = 000000
SVR4 = 000072	WBSTAT= 000040	\$F\$NAM= 000160	\$TAGNU= 050006	\$S\$TAG= 050000
SVR5 = 000074	WBUFEA= 000136	\$F\$ND = 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 17:30  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0106

3 COMMON EQUATE MODULE  
531 COMMON DEFINITIONS AND REFERENCES  
534 000000' .PRINT ;SPMAC: VERSION 1.1  
599 IERRLO INITIALIZE 11/60 ERROR LOG  
642 INITIALIZE 11/60 ERROR LOG ROUTINE  
675 ERRLOG 11/60 ERROR LOGGING ROUTINE  
733 ERRLOG 11/60 ERROR LOG ROUTINE

```
508 .TITLE BLOG60 BINDER FOR 11/60 ERROR LOG ROUTINES
509
510 .IDENT /V0.0/
511
512 ;++
513 ; MODULE PACKAGE NAME:
514 ;     BLOG60
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ;     THIS IS A BINDER TO KEEP 11/60 ERROR LOG MODULES PACKAGED
518 ;     TOGETHER.
519 ;     CONTAINS THE FOLLOWING MODULES:
520 ;         IERRLO - INITIALIZE 11/60 ERROR LOG
521 ;         ERRLOG  - 11/60 ERROR LOG
522 ;
523 ; VERSION:
524 ;     0.0
525 ;
526 ;     EDIT           DATE           BY           REASON
527 ;--
```

```

529
530
531      .SBTTL COMMON DEFINITIONS AND REFERENCES
532
533      .MCALL STRUCT
534      STRUCT
535      (1) 000000' .PRINT ;SPMAC: VERSION 1.1
536      000001' $LSTIN = 1
537      000001' $LSTTAG = 1
538
539      ;*****
540      ; REFERENCED BY OTHER MODULES
541      ;
542      .GLOBL IERRLO ;MODULE ENTRY POINT
543      .GLOBL ERRLOG ;MODULE ENTRY POINT
544
545      ;*****
546      ; GLOBAL REFERENCES
547      ;
548      .GLOBL SAVREG ;SAVE REGISTER ROUTINE
549      .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION (16 BIT)
550      .GLOBL MSGDHOOK ;MESSAGE HOOK ROUTINE
551      .GLOBL RESREG ;RESTORE REGISTERS
552
553      ;*****
554      ; LOCAL EQUATES
555      ;
556      L6.OFF = 100 ;OFFSET INTO ERROR LOG REGISTERS
557      L6.ENB = 100001 ;ENABLE AND FREEZE ON FIRST ERROR
558      L6.STO = ^D<8> ;SIZE OF LOG BUFFER
559
560      ;*****
561      ; LOCAL STORAGE
562      ;
563      L6.BUF: .BLKW L6.STO ;LOG BUFFER FOR 11/60 ERROR LOG
564      L6.TAB: .WORD L6.JAM ;TABLE POINTING TO LOCAL STORAGE FOR
565      .WORD L6.SRV ;MESSAGE
566      .WORD L6.PBA
567      .WORD L6.CUA
568      .WORD L6.FLG
569      .WORD L6.WHA
570      .WORD L6.CDA
571      .WORD L6.CTA
572
573      ;*****
574      ; LOCAL MESSAGES
575      ;
576      L6.MSG:
577      000040' 030445 027461 030066 .ASCII '%11/60 ERROR LOG%'
578      000046' 042440 051122 051117
579      000054' 046040 043517 045
580      000061' 112 046501 020057
581      L6.JAM: .ASCII 'JAM/ '
582      .BLKB ^D<6>
583      .ASCII ' SRV/ '
584      000074' 020040 051440 053122
585      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>
583	000171'	040	020040	044127		.ASCII	' WHAMI/ '
	000176'	046501	027511	040			
589	000203'	000006			L6.WHA:	.BLKB	^D<3>
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	

```
597
598
599      .SBTTL IERRLO INITIALIZE 11/60 ERROR LOG
600      .IDENT /V0.0/
601
602      ;++
603      ; MODULE NAME:
604      ;     IERRLO
605      ;
606      ; FUNCTIONAL DESCRIPTION:
607      ;     THIS MODULE WILL INITIALIZE THE 11/60 ERROR LOG TO THE ENABLE
608      ;     AND FREEZE ON FIRST ERROR STATES.
609      ;
610      ; INPUTS:
611      ;     NONE
612      ;
613      ; IMPLICIT INPUTS:
614      ;     NONE
615      ;
616      ; OUTPUTS:
617      ;     NONE
618      ;
619      ; IMPLICIT OUTPUTS:
620      ;     NONE
621      ;
622      ; PATHOLOGICAL CONNECTIONS:
623      ;     NONE
624      ;
625      ; SUBORDINATE ROUTINES CALLED:
626      ;     NONE
627      ;
628      ; FUNCTIONAL SIDE EFFECTS:
629      ;     NONE
630      ;
631      ; CALLING SEQUENCE:
632      ;     CALL IERRLO
633      ;
634      ; VERSION:
635      ;     0.0
636      ;
637      ;     EDIT             DATE             BY             REASON
638      ;--
```



```
674  
675 .SBTTL ERRLOG 11/60 ERROR LOGGING ROUTINE  
676 .IDENT /V0.0/  
677  
678 ;++  
679 ; MODULE NAME:  
680 ; ERRLOG  
681 ;  
682 ; FUNCTIONAL DESCRIPTION:  
683 ; THIS MODULE WILL READ THE 11/60 PROCESSOR ERROR LOG SCRATCHPAD AND  
684 ; TRANSFER THE ERROR LOG TO PHYSICAL CORE AND THEN GENERATE A  
685 ; MESSAGE THAT WILL OUTPUT 8 OF THE SCRATCHPAD REGISTERS (100-107).  
686 ; THESE REGISTERS ARE IDENTIFIED AS FOLLOWS:  
687 ; JAM - JAM REGISTER STATUS  
688 ; SRV - SERVICE REGISTER OF STATUS  
689 ; PBA - PHYSICAL BUS ADDRESS REGISTER (BIT 17,16)  
690 ; CUA - MICROPROGRAM ADDRESS  
691 ; FLG/INT - FLAG REQUEST REGISTER OF STATUS/LAST VECTOR SERVICED  
692 ; WHAMI - VARIETY OF PROCESSOR OPTION STATUS BITS, PROCESSOR  
693 ; OPTION STATUS  
694 ; CDATA - CACHE DATA  
695 ; CTAG/CPU- CACHE TAG DATA/HIT REGISTER  
696 ;  
697 ; INPUTS:  
698 ; DTABLE  
699 ;  
700 ; IMPLICIT INPUTS:  
701 ; NONE  
702 ;  
703 ; OUTPUTS:  
704 ; NONE  
705 ;  
706 ; IMPLICIT OUTPUTS:  
707 ; NONE  
708 ;  
709 ; PATHOLOGICAL CONNECTIONS:  
710 ; NONE  
711 ;  
712 ; SUBORDINATE ROUTINES CALLED:  
713 ; SAVREG - SAVE REGISTERS  
714 ; BOA16 - BINARY TO OCTAL ASCII CONVERSION (16 BITS)  
715 ; MSGDHOOK - MESSAGE HOOK ROUTINE  
716 ; RESREG - RESTORE REGISTERS  
717 ;  
718 ; FUNCTIONAL SIDE EFFECTS:  
719 ; NONE  
720 ;  
721 ; CALLING SEQUENCE:  
722 ; CALL ERRLOG IN <A>  
723 ; A -- ADDRESS OF DTABLE  
724 ;  
725 ; VERSION:  
726 ; 0.0  
727 ;  
728 ; EDIT DATE BY REASON  
729 ;--
```

```

731
732
733      .SBTTL ERRLOG 11/60 ERROR LOG ROUTINE
734
735 000300'      ROUTINE ERRLOG <DTABLE>
(2) 000300'
733
737
738      ;+
739      ; SAVE REGISTERS AND GET ADDRESS OF LOG BUFFER ALONG WITH INITIALIZING
740      ; LOG COMMAND AND ADDRESS TO ERROR LOG REGISTER 100
741      ; -
742
743 000300'      CALL SAVREG
(3) 000300' 004767 000000G      JSR      PC, SAVREG
744 000304'      LET R1 := #L6.BUF
(4) 000304' 012701 000000'      MOV      #L6.BUF, R1
745 000310'      LET R2 := #L6.OFF
(4) 000310' 012702 000100      MOV      #L6.OFF, R2
746
747      ;+
748      ; NOW READ ERROR LOG WORD AND LOAD IT INTO LOG BUFFER UNTIL
749      ; ERROR LOG IS SAVED.
750      ; -
751
752 000314'      WHILE R2 LT #L6.OFF+#L6.STO DO
(4) 000314'
(6) 000314' 020227 000110      50002$:  CMP      R2, #L6.OFF+#L6.S
(9) 000320' 002007      BGE      50003$
753
754 000322'      LET 1$ := R2
(4) 000322' 010267 000002      MOV      R2, 1$
755 000326'      INLINE <MED>
(2) 000326' 076600      MED
756 000330'      INLINE <1$: .WORD 0>
(2) 000330' 000000      1$:      .WORD 0
757
758      ;+
759      ; LOAD R0 (DATA FROM LOG) INTO LOG BUFFER AND POINT LOG BUFFER
760      ; TO NEXT ENTRY AND R2 TO NEXT COMMAND AND ADDRESS
761      ; -
762
763 000332'      LET (R1)+ := R0
(4) 000332' 010021      MOV      R0, (R1)+
764 000334'      LET R2 := R2 + #1
(6) 000334' 005202      INC      R2
765
766 000336'      ENDDO
(4) 000336' 000766      BR      50002$
(3) 000340'      50003$:
767
768      ;+
769      ; POINT TO "FP.LOG JAM" ENTRY IN LOG BUFFER AND POINT TO INDEX TABLE, CONVERT TO OCTAL
770      ; ASCII AND LOAD MESSAGE ALSO DO THE SAME TO FOLLOWING 7 ENTRIES IN LOG BUFFER.
771      ; -
772

```



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.STD DO		
(4)	000352'			50004\$:	
(6)	000352'	020227 000010		CMP	R2,#L6.STD
(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		000001'	.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	CTRLQ = 000017	ERRLOG 000300RG	L6.ENB= 100001	PRI4 = 000200
ADR = 000006	CTRLU = 000025	ERRTYP= 000105	L6.FLG 000163R	PRI5 = 000240
APTFER= 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	DROPMQ= 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
AUTGST= 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.CFO= 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	ROSERV= 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	INHRRR= 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	ICMODR= 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	MSGPDP= 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.STO= 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	NDAPTY= 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	SQFPAS= 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	SRO = 177572
CSRA = 000100	ENBNUL= 000001	L6.CDA 000223R	PDP70 = 010000	SR1 = 177574

SR2 = 177576	UIPAR4= 177650	WTWHMI= 000222	\$F\$RTN= 000300	\$S\$ARGC= 000002
SR3 = 172516	UIPAR5= 177652	XFLAG = 000005	\$F\$SEL= 000140	\$S\$BYTE= 000403
STAT = 000026	UIPAR6= 177654	XOFF = 000023	\$F\$THE= 000330	\$S\$CASE= 000000
STATBI= 064757	UIPAR7= 177656	XON = 000021	\$F\$STRU= 000404	\$S\$DST = 000000
STAT1 = 000027	UIPDR0= 177600	\$BGNLE= 177777	\$F\$UNT= 000130	\$S\$ELOC= 000000
SUSPND= 000001	UIPDR1= 177602	\$ERFLG= 000400	\$F\$WHI= 000120	\$S\$ERFL= 000000
SVR0 = 000062	UIPDR2= 177604	\$F\$AND= 000310	\$F\$YES= 000402	\$S\$FLAG= 000340
SVR1 = 000064	UIPDR3= 177606	\$F\$BAD= 000401	\$IFLEV= 177777	\$S\$FROM= 000000
SVR2 = 000066	UIPDR4= 177610	\$F\$BLA= 000170	\$LOCTA= 177777	\$S\$LOC = 000356R
SVR3 = 000070	UIPDR5= 177612	\$F\$CAS= 000150	\$LSTIN= 000001	\$S\$LOCN= 000000
SVR4 = 000072	UIPDR6= 177614	\$F\$DEC= 000220	\$LSTTA= 000001	\$S\$REG = 177777
SVR5 = 000074	UIPDR7= 177616	\$F\$DO = 000340	\$NESTL= 177777	\$S\$RETU= 000000
SVR6 = 000076	WASADR= 000104	\$F\$FAL= 000405	\$NSKO = 000300	\$S\$RTN1= 050000
SYSCNT= 000052	WBSTAT= 000040	\$F\$GDD= 000400	\$NSK1 = 000120	\$S\$RTN2= 050001
YSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$SAVLE= 177777	\$S\$SRC = 000000
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$SSKO = 050005	\$S\$TGSV= 000000
TQOVF = 000002	WBUFQ= 000140	\$F\$LQ= 000200	\$TAGLE= 177777	\$S\$TGS1= 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$TAGNU= 050006	\$S\$TGS2= 000000
UIPAR1= 177642	WDFR = 000116	\$F\$NO = 000403	\$TEMP = 000300	\$S\$TO = 000000
UIPAR2= 177644	WDT0 = 000114	\$F\$OR = 000320	\$TSKO = 050004	\$S\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:31  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0117

3 COMMON EQUATE MODULE  
530 COMMON DEFINITIONS AND REFERENCES  
533 000000' .PRINT ;SPMAC: VERSION 1.1  
575 IERRLO INITIALIZE 11/70 REGISTERS AND ERROR LOG  
622 INITIALIZE 11/70 REGISTERS ROUTINE  
642 ERRLOG 11/70 ERROR LOGGING  
690 11/70 ERROR LOG ROUTINE

BLOG70 BINDER FOR ERRLOG MODULES  
BLOG70.MAC 06-SEP-78 15:53

MACY11 30A(1052) 20-SEP-78 17:31 PAGE 19  
COMMON EQUATE MODULE

SEQ 0118

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526

```
.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  
;--
```

528  
 529  
 530  
 531  
 532  
 533  
 534  
 535  
 536  
 537  
 538  
 539  
 540  
 541  
 542  
 543  
 544  
 545  
 546  
 547  
 548  
 549  
 550  
 551  
 552  
 553  
 554  
 555  
 556  
 557  
 558  
 559  
 560  
 561  
 562  
 563  
 564  
 565  
 566  
 567  
 568  
 569  
 570  
 571  
 572

```

000000'
(1) 000000'
000001
000001
000000'
030445 027461 030067
042440 051122 051117
046040 043517
000020' 046445 046505 051105
000026' 042522 027507 040
000033' 000006
000041' 040 020040 050103
000046' 042525 051122 043505
000054' 020057
000056' 000006
000064' 000045
000066' 042101 051104 020057
000074' 000010
000104' 000045
  
```

```

.SBTTL COMMON DEFINITIONS AND REFERENCES

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

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

;*****
; GLOBAL REFERENCES
;
.GLOBL BOA16 ;CONVERT TO OCTAL ASCII (16 BITS)
.GLOBL BOAC ;CONVERT TO OCTAL ASCII (22 BITS)
.GLOBL MSGDHQOK ;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
;
L7.MSG:
.ASCII '%11/70 ERROR LOG'

L7.MED: .BLKB ^D<6>
.ASCII ' CPUERREG/ '

L7.CED: .BLKB ^D<6>
.ASCII '/%/

L7.MS1: .ASCII 'ADDR/ '
L7.ADR: .BLKB ^D<8>
.ASCII '/%/
.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 FOLLOING 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
625          IERRLO:
626
627          ;+
628          ; INITIALIZE THE 11/70 REGISTERS; MEMORY SYSTEM ERROR, LOW ERROR ADDRESS,
629          ; HIGH ADDRESS, CONTROL, MAINTENANCE AND CPU ERROR
630          ;-
631 000106'  LET @CPUMER := @CPUMER
632 (4) 000106' 017777 000000G 000000G          MOV      @CPUMER,@CPUMER
633 000114'  LET @CPULAR := #0
634 (4) 000114' 005077 000000G          CLR      @CPULAR
635 000120'  LET @CPUHAR := #0
636 (4) 000120' 005077 000000G          CLR      @CPUHAR
637 000124'  LET @CCNTRL := KONTRL
638 (4) 000124' 016777 000000G 000000G          MOV      KONTRL,@CCNTRL
639 000132'  LET @CPUCPE := #0
640 (4) 000132' 005077 000000G          CLR      @CPUCPE
641 000136'  LET @CPUMNT := #0
642 (4) 000136' 005077 000000G          CLR      @CPUMNT
643
644          ENDRTN
645
646          50000$:
647          50001$:
648          RTS      PC
649
```



```
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(1777744) AND THE CPU ERROR REGISTER(1777766).
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 (1777740)
654 ;     AND THE HIGH ERROR ADDRESS REGISTER (1777742)
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 ;--
```

```
689
690
691
692 000144'
(2) 000144'
693
694
695
696
697
698
699 000144'
(2) 000144' 010045
(3) 000146' 010145
700
701 000150'
(4) 000150' 016500 000000
702 000154'
(4) 000154' 016001 000000
703
704
705
706
707
708
709 000160'
(3) 000160' 010546
(5) 000162' 012745 000033'
(4) 000166' 017745 000000G
(3) 000172' 004767 000000G
(3) 000176' 012605
710
711
712
713
714
715
716 000200'
(3) 000200' 010546
(5) 000202' 012745 000056'
(4) 000206' 017745 000000G
(3) 000212' 004767 000000G
(3) 000216' 012605
717
718
719
720
721
722 000220'
(3) 000220' 010546
(7) 000222' 012745 000252'
(6) 000226' 012745 000000'
(5) 000232' 012745 000002
(4) 000236' 010045
(3) 000240' 004767 000000G
(3) 000244' 012605

.SBTTL 11/70 ERROR LOG ROUTINE
ROUTINE ERRLOG <DTABLE>
ERRLOG:

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

PUSH R0,R1
MOV R0,-(SP)
MOV R1,-(SP)

LET R0 := DTABLE(R5)
MOV DTABLE(R5),R0
LET R1 := DT.EVNT(R0)
MOV DT.EVNT(R0),R1

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

CALL BOA16 IN <@CPUMER,#L7.MED>
MOV R5,-(SP)
MOV #L7.MED,-(R5)
MOV @CPUMER,-(R5)
JSR PC,BOA16
MOV (SP)+,R5

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

CALL BOA16 IN <@CPUCPE,#L7.CED>
MOV R5,-(SP)
MOV #L7.CED,-(R5)
MOV @CPUCPE,-(R5)
JSR PC,BOA16
MOV (SP)+,R5

;+
; OUTPUT MESSAGE
;-

CALL MSGDHOOK IN <R0,#MSGPOP,#L7.MSG,#2$>
MOV R5,-(SP)
MOV #2$,-(R5)
MOV #L7.MSG,-(R5)
MOV #MSGPOP,-(R5)
MOV R0,-(R5)
JSR PC,MSGDHOOK
MOV (SP)+,R5
```



ACSR = 000102	CPUHAR= ***** G	DT.WHL= 000054	KTERRO= 000040	PRI1 = 000040
ACTBIT= 004000	CPULAR= ***** G	DT.WLL= 000052	KTPRES= 000400	PRI4 = 000200
ADDR22= 001000	CPUMER= ***** G	DVID1 = 000014	KTSTAT= 000020	PRI5 = 000240
ADR = 000006	CPUMNT= ***** G	ECCMEM= 000100	KTXTND= 040000	PRI6 = 000300
APTFER= 000004	CQDVF = 000001	ECCSTA= 000010	LF = 000012	PRI7 = 000340
APTPRE= 000200	CR = 000015	ENBEOP= 010000	LPSTAT= 000001	PRO = 000000
ASB = 000106	CSRA = 000100	ENBNUL= 000001	L7.ADR 000074R	PR4 = 000200
ASSEMB= 000010	CSRC = 000102	ENDLST= 000000	L7.CED 000056R	PR5 = 000240
ASTAT = 000104	CTRLC = 000003	EOPBIT= 000001	L7.MED 000033R	PR6 = 000300
AUTO = 000010	CTRL0 = 000017	ERRLOG 000144RG	L7.MSG 000000R	PR7 = 000340
AUTJST= 020000	CTRLU = 000025	ERRTYP= 000106	L7.MS1 000066R	PS = 177776
AWAS = 000110	DCEVNT= 000011	EVNTBE= 000200	MAPSTA= 000200	PSW = 177776
BIT0 = 000001	DEFRTN= 000400	EVNTHD= 000200	MED = 076600	RANNUM= 000054
BIT00 = 000001	DIAGMC= 000000	EVNTKT= 000203	MEMPAS= 040000	RBUFEA= 000130
BIT01 = 000002	DROPMO= 100000	EVNTPE= 000202	MODEXH= 004000	RBUFPA= 000126
BIT02 = 000004	DSEVNT= 000014	EVNTRE= 000201	MODHOL= 002000	RBUFSZ= 000132
BIT03 = 000010	DTABLE= 000000	FATERR= 100000	MODSEL= 001000	RBUFVA= 000124
BIT04 = 000020	DT.ADD= 000042	HRDCNT= 000044	MSGCKD= 000010	RDSERV= 000101
BIT05 = 000040	DT.AP = 000100	HRDPAS= 000050	MSGCKS= 000011	RDWHMI= 000022
BIT06 = 000100	DT.APK= 000076	ICONT = 000036	MSGDER= 000005	RELERR= 000020
BIT07 = 000200	DT.BLS= 000034	ICOUNT= 000040	MSGDHO= ***** G	RELMOD= 020000
BIT08 = 000400	DT.CFO= 000014	IDNUM = 000122	MSGDRP= 000017	RELTIM= 010000
BIT09 = 001000	DT.CF1= 000016	IE = 000100	MSGECH= 177777	RES1 = 000056
BIT1 = 000002	DT.ERR= 000020	IERRLO 000106RG	MSGEDP= 000013	RES2 = 000060
BIT10 = 002000	DT.ESI= 000044	INDPAR= 000040	MSGHDR= 000004	RICHAR= 031060
BIT11 = 004000	DT.EVN= 000000	INHDRP= 040000	MSGHNG= 000022	RPTDAT= 002000
BIT12 = 010000	DT.EXS= 000060	INHPRP= 020000	MSGHRD= 000007	RSTRT = 000112
BIT13 = 020000	DT.FCH= 000037	INHREL= 001000	MSGMAP= 000021	RUBCUT= 000177
BIT14 = 040000	DT.FCN= 000036	INHRE= 000400	MSGNUL= 177775	RUNMOD= 100000
BIT15 = 100000	DT.HMX= 000104	INIT = 000030	MSGPOP= 000002	RVALU= 001740
BIT2 = 000004	DT.KBE= 000024	INTR = 000120	MSGPRM= 177776	SAM = 075464
BIT3 = 000010	DT.KBP= 000026	IOMOD = 100000	MSGRES= 000001	SBACR = 000102
BIT4 = 000020	DT.KBR= 000022	IOMODP= 102000	MSGSFT= 000006	SBKMOD= 000000
BIT5 = 000040	DT.KBU= 000030	IOMODR= 112000	MSGSKE= 000003	SBKSEL= 010000
BIT6 = 000100	DT.MLS= 000032	IOMODX= 110000	MSGSMB= 000015	SC.ADR= 000006
BIT7 = 000200	DT.MTI= 000110	JACK = 035060	MSGSMH= 000014	SC.ALC= 000014
BIT8 = 000400	DT.OFF= 000070	KIPAR0= 172340	MSGSMS= 000016	SC.APC= 000016
BIT9 = 001000	DT.PAS= 000074	KIPAR1= 172342	MSGSTD= 000000	SC.CKL= 000002
BKDEF = 000002	DT.PC = 000002	KIPAR2= 172344	MSGSYS= 000012	SC.CKP= 000004
BKMOD = 000020	DT.PFL= 000062	KIPAR3= 172346	MSGVEC= 000020	SC.CLD= 000000
BKMODE= 040000	DT.PSW= 000004	KIPAR4= 172350	NBKMOD= 001000	SC.HLD= 000010
BKSLSH= 000134	DT.PTA= 000064	KIPAR5= 172352	NCPUOP= 000020	SC.SCA= 000012
BOAC = ***** G	DT.RCS= 000102	KIPAR6= 172354	NOAPTY= 000002	SENDLS= 177777
BOA16 = ***** G	DT.REL= 000040	KIPAR7= 172356	NULL = 000000	SGFCNT= 000042
CAPRES= 000004	DT.SCT= 000066	KIPDR0= 172300	OWEN = 024020	SOPPAS= 000046
CASAT= 000004	DT.SMX= 000106	KIPDR1= 172302	PAERR = 000010	SPACE = 000040
CCNTRL= ***** G	DT.SP = 000006	KIPDR2= 172304	PARPRE= 002000	SPOINT= 000032
CDERCT= 000146	DT.SSI= 000046	KIPDR3= 172306	PARSTA= 000100	SPVALU= 002200
CDWDCT= 000144	DT.STO= 000010	KIPDR4= 172310	PASCNT= 000034	SRO = 177572
CKTIM = 100000	DT.ST1= 000012	KIPDR5= 172312	PDPLSI= 020000	SR1 = 177574
CLKPRE= 000001	DT.SWR= 000056	KIPDR6= 172314	PDP60 = 004000	SR2 = 177576
CONFIG= 000056	DT.SYP= 000072	KIPDR7= 172316	PDP70 = 010000	SR3 = 172516
CPUCPE= ***** G	DT.WBU= 000050	KONTRL= ***** G	PRI0 = 000000	STAT = 000026

STATBI= 064757	UIPAR7= 177656	XON = 000021	\$F\$TRU= 000404	\$\$ELOD= 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	\$ISKO = 000001	\$\$LCCN= 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
SYS CNT= 000052	WBSTAT= 000040	\$F\$G00= 000400	\$NSKO = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:31  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0127

3 COMMON EQUATE MODULE  
563 GPA (COMMON DEFINITICNS & REFERENCES)  
566 000000' .PRINT ;SPMAC: VERSION 1.1  
584 GPA (CODE)

```
508 .TITLE GPA (GET PHYSICAL ADDRESS)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     GPA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE DETERMINES THE 16-BIT, 18-BIT OR 22-BIT
517 ;     PHYSICAL ADDRESS FROM THE 16-BIT VIRTUAL ADDRESS GIVEN.
518 ;
519 ;     IF 18-BIT ADDRESS, THE TOP 2 BITS ARE ALLIGNED TO POSITIONS
520 ;     4 & 5 OF THE EA WORD. IF 22-BIT ADDRESS SYSTEM,
521 ;     THE TOP 6 BITS ARE RIGHT JUSTIFIED IN THE EA WORD.
522 ;
523 ; INPUTS:
524 ;     1. DTABLE
525 ;     2. TABLE ADDRESS CONTAINING VA,PA,EA WORDS
526 ;         WHERE VA = VIRTUAL ADDRESS
527 ;         PA = PHYSICAL ADDRESS
528 ;         EA = EXTENDED ADDRESS
529 ;
530 ; IMPLICIT INPUTS:
531 ;     1. DT.CFO
532 ;     2. DT.STO
533 ;
534 ; OUTPUTS:
535 ;     NONE
536 ;
537 ; IMPLICIT OUTPUTS:
538 ;     1. PHYSICAL ADDRESS WORD
539 ;     2. EXTENDED ADDRESS BITS WORD
540 ;
541 ; PATHOLOGICAL CONNECTIONS:
542 ;     NONE
543 ;
544 ; SUBORDINATE ROUTINES CALLED:
545 ;     SAVREG
546 ;     RESREG
547 ;
548 ; FUNCTIONAL SIDE EFFECTS:
549 ;     NONE
550 ;
551 ; CALLING SEQUENCE:
552 ;     CALL GPA IN <DT,TBL>
553 ;         WHERE DT = DTABLE ADDRESS
554 ;         TBL = TABLE ADDRESS (CONTAINS VA,PA,EA WORDS)
555 ;
556 ; VERSION:
557 ;     0.0
558 ;
559 ;     EDIT             DATE             BY             REASON
560 ;--
561
```

GPA (GET PHYSICAL ADDRESS)  
GPA.MAC 28-JUL-78 09:08

MACY11 30A(1052) 20-SEP-78 17:31 PAGE 19-1  
GPA (COMMON DEFINITIONS & REFERENCES)

SEQ 0129

```
563 .SBTTL GPA (COMMON DEFINITIONS & REFERENCES)
564
565 .MCALL STRUCT
566 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
567 000001 $LSTIN=1
568 000001 $LSTTAG=1
569
570 ;*****
571 ; REFERENCED BY OTHER MODULES
572 ;
573 .GLOBL GPA ;MODULE ENTRY POINT
574 ;
575 ;*****
576 ;
577 ; GLOBAL REFERENCES:
578 ;
579 .GLOBL SAVREG
580 .GLOBL RESREG
581 ;
582
```



```

584 .SBTTL GPA (CODE)
585
586 000000' ROUTINE GPA <DT,TBL>
(2) 000000'
587
588 ;+
589 ; SAVE REGS, GET DTABLE ADDRESS AND TABLE ADDRESS
590 ; -
591
592 000000' CALL SAVREG
(3) 000000' 004767 000000G JSR PC, SAVREG
593 000004' LET R0 := DT(R5) MOV DT(R5), R0
(4) 000004' 016500 000000
594 000010' LET R1 := TBL(R5) MOV TBL(R5), R1
(4) 000010' 016501 000002
595
596 ;+
597 ; FETCH VIRTUAL ADDRESS FROM FIRST WORD OF USER SUPPLIED TABLE
598 ; AND LOAD INTO THE PHYSICAL ADDRESS WORD (2ND WORD OF USER SUPPLIED
599 ; TABLE).
600 ; -
601
602 000014' LET (R1) := (R1)+
(4) 000014' 012111 MOV (R1)+, (R1)
603
604 ;+
605 ; IF KT IS NOT ON, CLEAR EA WORD AND RETURN
606 ; -
607
608 000016' IF #KTSTAT NOTSETIN DT.STO(R0) THEN
(6) 000016' 032760 000020 000010 BIT #KTSTAT, DT.STO(R
(9) 000024' 001003 BNE 50002$
609 000026' LET 2(R1) := #0
(4) 000026' 005061 000002 CLR 2(R1)
610
611 000032' ELSE
(4) 000032' 000454 BR 50003$
(3) 000034' 50002$:
612
613 ;+
614 ; KT IS ON. GENERATE 18 OR 22 BIT ADDRESS.
615 ; BRING PAGE BITS 13-15 OF PA IN PCS.1-3, ADD TO KIPARO
616 ; ADDRESS, AND GET THE CONTENTS OF THE FORMED KIPAR REG.
617 ; -
618
619 000034' LET (R1) := SWAP (R1)
(6) 000034' 000311 SWAB (R1)
620 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)
621 000046' LET (R1) := (R1) CLR.BY #177761
(6) 000046' 042711 177761 BIC #177761, (R1)
622 000052' LET (R1) := (R1) + #KIPARO
(6) 000052' 062711 172340 ADD #KIPARO, (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)          MOV      (R1),R2
(6) 000062' 011102          SWAB      R2
(6) 000064' 000302
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

```

GPA (GET PHYSICAL ADDRESS)  
GPA.MAC 28-JUL-78 09:08

MACY11 30A(1052) 20-SEP-78 17:31 PAGE 19-4  
GPA (CODE)

SEQ 0132

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

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ACTBIT= 004000	CTRLD = 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
APTFRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGDRP= 000017	RELTIM= 010000
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGECH= 177777	RESREG= ***** G
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGEOP= 000013	RES1 = 000056
ASTAT = 000104	DT = 000000	GPA 000000RG	MSGHDR= 000004	RES2 = 000060
AUTO = 000010	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AUTDST= 020000	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AWAS = 000110	DT.APK= 000076	ICQNT = 000036	MSGMAP= 000021	RSTRT = 000112
BIT0 = 000001	DT.BLS= 000034	ICQNT= 000040	MSGNUL= 177775	RUBOUT= 000177
BIT00 = 000001	DT.CFO= 000014	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT01 = 000002	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RVALU= 001740
BIT02 = 000004	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT03 = 000010	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SAVREG= ***** G
BIT04 = 000020	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	SBADR = 000102
BIT05 = 000040	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	SBKMOD= 000000
BIT06 = 000100	DT.FCH= 000037	INHRRE= 000400	MSGSMH= 000014	SBKSEL= 010000
BIT07 = 000200	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SC.ADR= 000006
BIT08 = 000400	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.ALC= 000014
BIT09 = 001000	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.APC= 000016
BIT1 = 000002	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.CKL= 000002
BIT10 = 002000	DT.KBR= 000022	IOMODR= 112000	NBKMOD= 001000	SC.CKP= 000004
BIT11 = 004000	DT.KBU= 000030	IOMODX= 110000	NCPUOP= 000020	SC.CLD= 000000
BIT12 = 010000	DT.MLS= 000032	JACK = 035060	NDAPTY= 000002	SC.HLD= 000010
BIT13 = 020000	DT.MTI= 000110	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT14 = 040000	DT.OFF= 000070	KIPAR1= 172342	OWEN = 024020	SENDLS= 177777
BIT15 = 100000	DT.PAS= 000074	KIPAR2= 172344	PAERR = 000010	SQFCNT= 000042
BIT2 = 000004	DT.PC = 000002	KIPAR3= 172346	PARPRE= 002000	SQFPAS= 000046
BIT3 = 000010	DT.PFL= 000062	KIPAR4= 172350	PARSTA= 000100	SPACE = 000040
BIT4 = 000020	DT.PSW= 000004	KIPAR5= 172352	PASCNT= 000034	SPOINT= 000032
BIT5 = 000040	DT.PTA= 000064	KIPAR6= 172354	PDPLSI= 020000	SPVALU= 002200
BIT6 = 000100	DT.RCS= 000102	KIPAR7= 172356	PDP60 = 004000	SR0 = 177572
BIT7 = 000200	DT.REL= 000040	KIPDR0= 172300	PDP70 = 010000	SR1 = 177574
BIT8 = 000400	DT.SCT= 000066	KIPDR1= 172302	PRI0 = 000000	SR2 = 177576
BIT9 = 001000	DT.SMX= 000106	KIPDR2= 172304	PRI1 = 000040	SR3 = 172516
BKDEF = 000002	DT.SP = 000006	KIPDR3= 172306	PRI4 = 000200	STAT = 000026
BKMOD = 000020	DT.SSI= 000046	KIPDR4= 172310	PRI5 = 000240	STATBI= 064757
BKMODE= 040000	DT.STO= 000010	KIPDR5= 172312	PRI6 = 000300	STAT1 = 000027
BKSLSH= 000134	DT.ST1= 000012	KIPDR6= 172314	PRI7 = 000340	SUSPND= 000001
CAPRES= 000004	DT.SWR= 000056	KIPDR7= 172316	PRO = 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	TBL = 000002
CSRC = 000102	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132	TMPIG = 000002

TQOVF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$FLAG= 000001
UIPAR0= 177640	WBUFPA= 000134	\$F\$G00= 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	\$\$LDCN= 000000
UIPAR3= 177646	WDFR = 000116	\$F\$L00= 000200	\$NSK0 = 000300	\$\$REG = 177777
UIPAR4= 177650	WDT0 = 000114	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$RETN= 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	\$TSKO = 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\$DD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:32  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0135

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES FOR KTERR  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
611 KTERR ROUTINE

```
508 .TITLE KTERR PROCESS MEMORY MANAGEMENT TRAPS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     KTERR
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS ROUTINE PROCESSES MEMORY MANAGEMENT TRAPS (THROUGH
517 ;     LOC 250). IT WILL FETCH THE CONTENTS OF THE STATUS
518 ;     REGISTERS, CONVERT TO ASCII, AND ENQUEUE A MESSAGE
519 ;     CONTAINING THE CONTENTS. THE KT ERROR BIT WILL BE SET
520 ;     IN THE DTABLE ERROR WORD.
521 ;
522 ; INPUTS:
523 ;     ADDRESS OF DATA TABLE
524 ;
525 ; IMPLICIT INPUTS:
526 ;     DT.STO
527 ;     DT.CFO
528 ;
529 ; OUTPUTS:
530 ;     NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ;     DT.ERR
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ;     NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ;     MSGDHOOK - PRINT A MESSAGE
540 ;     RSTRCY - RESET RECOVERY
541 ;
542 ;     ERRLOG
543 ;     BOA16
544 ;     SAVREG
545 ;     RESREG
546 ; FUNCTIONAL SIDE EFFECTS:
547 ;     NONE
548 ;
549 ; CALLING SEQUENCE:
550 ;     CALL KTERR IN <DTADR>
551 ;     WHERE DTADR = ADDRESS OF DATA TABLE
552 ;
553 ; VERSION:
554 ;     0.0
555 ;
556 ;     EDIT          DATE          BY          REASON
557 ;--
558
559
```

```
561 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KTERR
562
563 .MCALL STRUCT
564 000000' STRUCT
565 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
566
567 000001 $LSTIN=1
568 000001 $LSTTAG=1
569
570 ;
571 ;*****
572 ; REFERENCED BY OTHER MODULES:
573 ;
574 .GLOBL KTERR ;MODULE ENTRY POINT
575 ;
576 ;
577 ;*****
578 ;
579 ; GLOBAL REFERENCES:
580 ;
581 .GLOBL ERRLOG ;ERROR LOGGING
582 .GLOBL RSTRCY ;RESET RECOVERY
583 .GLOBL MSGDHOOK ;PRINT A MESSAGE
584 .GLOBL BOA16 ;OCTAL TO ASCII CONVERSION ROUTINE
585 .GLOBL SAVREG
586 .GLOBL RESREG
587 ;
588 ;*****
589 ;
590 ; LOCAL STORAGE:
591 ;
592
593 000000' 044445 046114 020056 KT.BADKT: .ASCIZ /%ILL. TRAP THRU KT VCT.%/
594 000006' 051124 050101 052040
595 000014' 051110 020125 052113
596 000022' 053040 052103 022456
597 000030' 000
598 000031' 045 025052 045440 KT.MG1: .ASCII /%** KT TRAP **%/
599 000036' 020124 051124 050101
600 000044' 025040 022452
601 000050' 051445 030122 020040 .ASCII /%SR0 SR2%/
602 000056' 020040 051123 022462
603 000064' 000006 KT.SS0MG: .BLKB 6
604 000072' 020040 .ASCII / /
605 000074' 000006 KT.SS2MG: .BLKB 6
606 000102' 000045 .ASCIZ /%/
607
608 000104' 051445 030522 020040 KT.MG2: .ASCII /%SR1 SR3%/
609 000112' 020040 051123 022463
610 000120' 000006 KT.SS1MG: .BLKB 6
611 000126' 020040 .ASCII / /
612 000130' 000006 KT.SS3MG: .BLKB 6
613 000136' 000045 .ASCIZ /%/
614
615 .EVEN
616 ;
617 ;
```



KTERR PROCESS MEMORY MANAGEMENT TRAPS MACY11 30A(1052) 20-SEP-78 17:32 PAGE 19-2  
KTERR.MAC 28-JUL-78 09:22 COMMON DEFINITIONS AND REFERENCES FOR KTERR

SEQ 0138

608  
609

\*\*\*\*\*

```

611          .SBTTL  KTERR ROUTINE
612
613 000140'          ROUTINE KTERR <DTADR>
(2) 000140'
614
615          ;+
616          ; INIT AND SAVE DTABLE
617          ; -
618
619 000140'          CALL SAVREG
(3) 000140' 004767 000000G          JSR      PC, SAVREG
620 000144'          LET RO := DTADR(R5)
(4) 000144' 016500 000000          MOV      DTADR(R5), R0
621
622
623
624          ;+
625          ; SET KT ERROR BIT IN ERROR WORD IN DTABLE
626          ; -
627
628 000150'          LET DT.ERR(RO) := DT.ERR(RO) SET.BY #KTERROR
(6) 000150' 052760 000040 000020          BIS      #KTERROR, DT.ERR(
629
630
631
632          ;+
633          ; IF KT ENABLED, SET FATAL BIT IN ERROR WORD
634          ; -
635
636 000156'          IF #KTSTAT SETIN DT.STO(RO) THEN
(6) 000156' 032760 000020 000010          BIT      #KTSTAT, DT.STO(R
(9) 000164' 001535
637
638          ;+
639          ; SET FATAL BIT, GET THE CONTENTS OF SR0, SR2, CONVERT
640          ; TO ASCII, AND ENQUEUE MESSAGE
641          ; -
642 000166'          LET DT.ERR(RO) := DT.ERR(RO) SET.BY #FATERR
(6) 000166' 052760 100000 000020          BIS      #FATERR, DT.ERR(R
643
644 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          MCV      @#SR0, -(R5)
(3) 000206' 004767 000000G          JSR      PC, BOA16
(3) 000212' 012605          MOV      (SP)+, R5
645 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
646
647          ;+
648          ; SEE IF EXTENDED KT
649          ; -

```

650							
651	000234'			IF #KTXND SETIN DT.CF0(R0) THEN			
(6)	000234'	032760	040000	000014		BIT	#KTXND,DT.CF0(R
(9)	000242'	001420				BEQ	50003\$
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 MSGDH00K 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,MSGDH00K
(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 #KTXND SETIN DT.CF0(R0) THEN
(6) 000362' 032760 040000 000014                                BIT      #KTXND,DT.CF0(R
(9) 000370' 001415                                BEQ      50004$
681 000372'                                CALL MSGDH00K 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,MSGDH00K
(3) 000416' 012605                                MOV      (SP)+,R5
682 000420'                                INLINE <6$:      NOP>                                6$:      NOP
(2) 000420' 000240                                INLINE <BR      6$>                                BR      6$
683 000422'
(2) 000422' 000776                                BR      6$
684
685 000424'                                INLINE <2$:>                                2$:
(2) 000424'                                ENDIF                                50004$:
686 000424'
(4) 000424'
687 ;+
688 ; DO ERROR LOGGING IF NECESSARY
689 ; -
690
691 000424'                                IF #PDP60 SETIN DT.CF0(R0) OR #PDP70 SETIN DT.CF0(R0) THEN
(6) 000424' 032760 004000 000014                                BIT      #PDP60,DT.CF0(R0
(8) 000432' 001004                                BNE      50005$
(6) 000434' 032760 010000 000014                                BIT      #PDP70,DT.CF0(R0
(9) 000442' 001405                                BEQ      50006$
(6) 000444'                                50005$:
692
693 000444'                                CALL ERRLOG IN <R0>
(3) 000444' 010546                                MOV      R5,-(SP)
(4) 000446' 010045                                MOV      R0,-(R5)
(3) 000450' 004767 000000G                                JSR      PC,ERRLOG
(3) 000454' 012605                                MOV      (SP)+,R5
694
695 000456'                                ENDIF                                50006$:
(4) 000456'
696
697 000456'                                ELSE
(4) 000456' 000427                                BR      50007$
(3) 000460'                                50002$:
698 ;+
699 ; RESET AND RESTORE, LOWER PRIORITY
700 ; -
701 000460'                                INLINE <RESET>                                RESET
(2) 000460' 000005                                CALL RSTRCY IN <R0>
702 000462'
(3) 000462' 010546                                MOV      R5,-(SP)

```

703					
704					
705	000474'		LET -(SP) := #0		
706	000476'	005046	LET -(SP) := #444\$		
707	000476'	012746	000504'	000504'	
708	000502'		INLINE <RTI>		
709	000502'	000002			
710	000504'		INLINE <444\$:>		
711					
712					
713			;+ ; NO, KT IS NOT ENABLED, SO ENQUEUE THE BAD MESSAGE		
714			;-		
715	000504'		CALL MSGDHOOK IN <R0,#MSGPOP,#KT.BADKT,#3\$>		
716	000504'	010546			
717	000506'	012745	000536'		
718	000512'	012745	000000'		
719	000516'	012745	000002		
720	000522'	010045			
721	000524'	004767	000000G		
722	000530'	012605			
723	000532'		INLINE <8\$: NOP>		
724	000532'	000240			
725	000534'		INLINE <BR 8\$>		
726	000534'	000776			
727	000536'		INLINE <3\$:>		
728	000536'				
729	000536'		ENDIF		
730	000536'				
731					
732					
733					
734					
735					
736					
737					
738					
739					
740					
741					
742					
743					
744					
745					
746					
747					
748					
749					
750					
751					
752					
753					
754					
755					
756					
757					
758					
759					
760					
761					
762					
763					
764					
765					
766					
767					
768					
769					
770					
771					
772					
773					
774					
775					
776					
777					
778					
779					
780					
781					
782					
783					
784					
785					
786					
787					
788					
789					
790					
791					
792					
793					
794					
795					
796					
797					
798					
799					
800					
801					
802					
803					
804					
805					
806					
807					
808					
809					
810					
811					
812					
813					
814					
815					
816					
817					
818					
819					
820					
821					
822					
823					
824					
825					
826					
827					
828					
829					
830					
831					
832					
833					
834					
835					
836					
837					
838					
839					
840					
841					
842					
843					
844					
845					
846					
847					
848					
849					
850					
851					
852					
853					
854					
855					
856					
857					
858					
859					
860					
861					
862					
863					
864					
865					
866					
867					
868					
869					
870					
871					
872					
873					
874					
875					
876					
877					
878					
879					
880					
881					
882					
883					
884					
885					
886					
887					
888					
889					
890					
891					
892					
893					
894					
895					
896					
897					
898					
899					
900					
901					
902					
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					
950					
951					
952					
953					
954					
955					
956					
957					
958					
959					
960					
961					
962					
963					
964					
965					
966					
967					
968					
969					
970					
971					
972					
973					
974					
975					
976					
977					
978					
979					
980					
981					
982					
983					
984					
985					
986					
987					
988					
989					
990					

ACSR = 000102	CSRC = 000102	ENDLST= 000000	KT.SS1 000120R	PR4 = 000200
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	KT.SS2 000074R	PR5 = 000240
ADDR22= 001000	CTRL0 = 000017	ERRLOG= ***** G	KT.SS3 000130R	PR6 = 000300
ADR = 000006	CTRLU = 000025	ERRTYP= 000105	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	DROPMD= 100000	EVNTPE= 000202	MEMPAS= 040000	RBUFEA= 000130
ASTAT = 000104	DSEVNT= 000014	EVNTRE= 000201	MODEXH= 004000	RBUFPA= 000126
AUTO = 000010	DTADR = 000000	FATERR= 100000	MODHDL= 002000	RBUFSZ= 000132
AUTOST= 020000	DT.ADD= 000042	HRDCNT= 000044	MCDSEL= 001000	RBUFVA= 000124
AWAS = 000110	DT.AP = 000100	HRDPAS= 000050	MSGCKD= 000010	RDSERV= 000101
BIT0 = 000001	DT.APK= 000076	ICONT = 000036	MSGCKS= 000011	ROWMI= 000022
BIT00 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGDER= 000005	RELERR= 000020
BIT01 = 000002	DT.CFO= 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	RVALU= 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	SCFCNT= 000042
BKSLSH= 000134	DT.STO= 000010	KIPDR5= 172312	PARSTA= 000100	SOFPAS= 000046
BOA16 = ***** G	DT.ST1= 000012	KIPDR6= 172314	PASCNT= 000034	SPACE = 000040
CAPRES= 000004	DT.SWR= 000056	KIPDR7= 172316	PDPLSI= 020000	SPCINT= 000032
CASTAT= 000004	DT.SYP= 000072	KTERR 000140RG	PDP60 = 004000	SPVALU= 002200
CDWDCT= 000144	DT.WBU= 000050	KTERR0= 000040	PDP70 = 010000	SRC = 177572
CKTIM = 100000	DT.WHL= 000054	KTPRES= 000400	PRI0 = 000000	SR1 = 177574
CLKPRE= 000001	DT.WLL= 000052	KTSTAT= 000020	PRI1 = 000040	SR2 = 177576
CONFIG= 000056	DVID1 = 000014	KTXTND= 040000	PRI4 = 000200	SR3 = 172516
CQOVF = 000001	ECCMEM= 000100	KT.BAD 000000R	PRI5 = 000240	STAT = 000026
CR = 000015	ECCSTA= 000010	KT.MG1 000031R	PRI6 = 000300	STATBI= 064757
CSRA = 000100	ENBEOP= 010000	KT.MG2 000104R	PRI7 = 000340	STAT1 = 000027
	ENBNUL= 000001	KT.SS0 000064R	PRO = 000000	SUSPND= 000001

SVR0 = 000062	UIPDR2= 177604	\$F\$AND= 000310	\$F\$YES= 000402	\$SELOC= 000402
SVR1 = 000064	UIPDR3= 177606	\$F\$BAD= 000401	\$IFLEV= 177777	\$SERFL= 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\$G00= 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\$L00= 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)

.MAIN. MACY11 30A(1052) 20-SEP-73 17:33  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0145

3 COMMON EQUATE MODULE  
353 COMMON DEFINITIONS AND REFERENCES  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
580 KTSET ROUTINE



```
508 .TITLE KTSET MAP MEMORY MANAGEMENT REGISTERS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     KTSET
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     ROUTINE WILL SET UP THE PDRS (USER AND KERNEL) TO THE FOLLOWING:
517 ;     READ/WRITE PROTECTION (NO SYSTEM TRAP/ABORT ACTION),
518 ;     UPWARD PAGE ADDRESSING,
519 ;     MAXIMUM BLOCK COUNT (4K).
520 ;     THE PARS WILL BE SET UP WITH PAR0 SET TO BANK0, PAR1-PAR5 SET UP TO
521 ;     DEFINED OFFSET, AND PAR7 TO THE I/O PAGE.
522 ;
523 ; INPUTS:
524 ;     DTABLE ADDRESS
525 ;
526 ; IMPLICIT INPUTS:
527 ;     DT.ADDR - CURRENT BASE OFFSET
528 ;
529 ; OUTPUTS:
530 ;     NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ;     NONE
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ;     NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ;     NONE
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ;     NONE
543 ;
544 ; CALLING SEQUENCE:
545 ;     CALL KTSET IN <A>
546 ;     A - ADDRESS OF DTABLE
547 ;
548 ; VERSION:
549 ;     0.0
550 ;
551 ;     EDIT          DATE          BY          REASON
552 ;--
553
```

```
555
556 .SBTTL COMMON DEFINITIONS AND REFERENCES
557
558 .MCALL STRUCT
559 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
560 000001 $LSTIN = 1
561 000001 $LSTTAG = 1
562
563 ;*****
564 ; REFERENCED BY OTHER MODULES
565 ;
566 .GLOBL KTSET ;MODULE ENTRY POINT
567
568 ;*****
569 ; GLOBAL REFERENCES
570 ;
571 .GLOBL SAVREG ;SAVE REGISTERS
572 .GLOBL RESREG ;RESTORE REGISTERS
573
574 ;*****
575 ; LOCAL EQUATES
576 ;
577 077406 KS.PDR = 77406 ;PDR VALUE
```

```

579
580      .SBTTL KTSET ROUTINE
581
582      ROUTINE KTSET <DTABLE>
583
584
585      ;+
586      ; SAVE REGISTERS, RETRIEVE ADDRESS OF DTABLE
587      ; INITIALIZE POINTER TO BANK0, SETUP R2 TO 77406 VALUE
588      ;-
589
590      CALL SAVREG
591      LET R0 := DTABLE(R5)
592      LET R2 := #KS.PDR
593
594      ;+
595      ; CALL KTSET0, PASS ADDRESS OF UIPDRO
596      ;-
597
598      LET R3 := #UIPDRO
599      CALL KTSET0
600
601      ;+
602      ; CALL KTSET0, PASS ADDRESS OF KIPDRO
603      ;-
604
605      LET R3 := #KIPDRO
606      CALL KTSET0
607
608      ;+
609      ; RESTORE REGISTERS AND RETURN
610      ;-
611      CALL RESREG
612      ENDRTN
613
614      ROUTINE KTSET0
615
616      ;+
617      ; LOAD APRO BY MAPPING TO BANK0, SET UP R4 TO POINT TO APR7
618      ;-
619
620      LET R4 := R3 + #16
621

```

582	000000'								
(2)	000000'								
590	000000'								
(3)	000000'	004767	000000G				JSR	PC, SAVREG	
591	000004'								
(4)	000004'	016500	000000				MOV	DTABLE(R5), R0	
592	000010'								
(4)	000010'	012702	077406				MOV	#KS.PDR, R2	
598	000014'								
(4)	000014'	012703	177600				MOV	#UIPDRO, R3	
599	000020'								
(3)	000020'	004767	000016				JSR	PC, KTSET0	
605	000024'								
(4)	000024'	012703	172300				MOV	#KIPDRO, R3	
606	000030'								
(3)	000030'	004767	000006				JSR	PC, KTSET0	
611	000034'								
(3)	000034'	004767	000000G				JSR	PC, RESREG	
612	000040'								
(3)	000040'								
(3)	000040'								
(2)	000040'	000207					RTS	PC	
614	000042'								
(2)	000042'								
620	000042'								
(4)	000042'	010304					MOV	R3, R4	

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

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	RBUFPA= 000126
ACTBIT= 004000	CTRLD = 000017	ERRTP= 000106	MODEXH= 004000	RBUFSZ= 000132
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MODHOL= 002000	RBUFVA= 000124
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MODSEL= 001000	RDSERV= 000101
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGCKD= 000010	RDWHMI= 000022
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGCKS= 000011	RELERR= 000020
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	RELMOD= 020000
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	RELTIM= 010000
ASTAT = 000104	DTABLE= 000000	HRDCNT= 000044	MSGECH= 177777	RESREG= ***** G
AUTO = 000010	DT.ADD= 000042	HRDPAS= 000050	MSGEDP= 000013	RES1 = 000056
AUTOST= 020000	DT.AP = 000100	ICONT = 000036	MSGHDR= 000004	RES2 = 000060
AWAS = 000110	DT.APK= 000076	ICOUNT= 000040	MSGHNG= 000022	RICHAR= 031060
BIT0 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGHRD= 000007	RPTDAT= 002000
BIT00 = 000001	DT.CFO= 000014	IE = 000100	MSGMAP= 000021	RSTRT = 000112
BIT01 = 000002	DT.CF1= 000016	INDPAR= 000040	MSGNUL= 177775	RUBCUT= 000177
BIT02 = 000004	DT.ERR= 000020	INHDRP= 040000	MSGPOP= 000002	RUNMOD= 100000
BIT03 = 000010	DT.ESI= 000044	INHEPR= 020000	MSGPRM= 177776	RVALU= 001740
BIT04 = 000020	DT.EVN= 000000	INHREL= 001000	MSGRES= 000001	SAM = 075464
BIT05 = 000040	DT.EXS= 000060	INHRE= 000400	MSGSFT= 000006	SAVREG= ***** G
BIT06 = 000100	DT.FCH= 000037	INIT = 000030	MSGSKE= 000003	SBADR = 000102
BIT07 = 000200	DT.FCN= 000036	INTR = 000120	MSGSMB= 000015	SBKMOD= 000000
BIT08 = 000400	DT.HMX= 000104	IOMOD = 100000	MSGSMH= 000014	SBKSEL= 010000
BIT09 = 001000	DT.KBE= 000024	IOMODP= 102000	MSGSMS= 000016	SC.ADR= 000006
BIT1 = 000002	DT.KBP= 000026	IOMODR= 112000	MSGSTD= 000000	SC.ALC= 000014
BIT10 = 002000	DT.KBR= 000022	IOMODX= 110000	MSGSYS= 000012	SC.APC= 000016
BIT11 = 004000	JACK = 000030	KIPAR0= 172340	MSGVEC= 000020	SC.CKL= 000002
BIT12 = 010000	DT.MLS= 000032	KIPAR1= 172342	NBKMOD= 001000	SC.CKP= 000004
BIT13 = 020000	DT.MTI= 000110	KIPAR2= 172344	NCPUOP= 000020	SC.CLO= 000000
BIT14 = 040000	DT.OFF= 000070	KIPAR3= 172346	NOAPTY= 000002	SC.HLD= 000010
BIT15 = 100000	DT.PAS= 000074	KIPAR4= 172348	NULL = 000000	SC.SCA= 000012
BIT2 = 000004	DT.PC = 000002	KIPAR5= 172350	OWEN = 024020	SENDLS= 177777
BIT3 = 000010	DT.PFL= 000062	KIPAR6= 172352	PAERR = 000010	SQFCNT= 000042
BIT4 = 000020	DT.PSW= 000004	KIPAR7= 172356	PARPRE= 002000	SQFPAS= 000046
BIT5 = 000040	DT.PTA= 000064	KIPDR0= 172300	PARSTA= 000100	SPACE = 000040
BIT6 = 000100	DT.RCS= 000102	KIPDR1= 172302	PASCNT= 000034	SPOINT= 000032
BIT7 = 000200	DT.REL= 000040	KIPDR2= 172304	PDPLSI= 020000	SPVALU= 002200
BIT8 = 000400	DT.SCT= 000066	KIPDR3= 172306	PDP60 = 004000	SR0 = 177572
BIT9 = 001000	DT.SMX= 000106	KIPDR4= 172310	PDP70 = 010000	SR1 = 177574
BKDEF = 000002	DT.SP = 000006	KIPDR5= 172312	PRI0 = 000000	SR2 = 177576
BKMOD = 000020	DT.SSI= 000046	KIPDR6= 172314	PRI1 = 000040	SR3 = 172516
BKMODE= 040000	DT.STO= 000010	KIPDR7= 172316	PRI4 = 000200	STAT = 000026
BKSLSH= 000134	DT.ST1= 000012	KS.PDR= 077406	PRI5 = 000240	STATBI= 064757
CAPRES= 000004	DT.SWR= 000056	KTERRO= 000040	PRI6 = 000300	STAT1 = 000027
CASTAT= 000004	DT.SYP= 000072	KTPRES= 000400	PRI7 = 000340	SUSFND= 000001
CDERCT= 000146	DT.WBU= 000050	KTSET 000000RG	PR0 = 000000	SVR0 = 000062
CDWDCT= 000144	DT.WHL= 000054	KTSET0 000042R	PR4 = 000200	SVR1 = 000064
CKTIM = 100000	DT.WLL= 000052	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CLKPRE= 000001	DVID1 = 000014	KTXTND= 040000	PR6 = 000300	SVR3 = 000070
CONFIG= 000056	ECCMEM= 000100	LF = 000012	PR7 = 000340	SVR4 = 000072
CQOVF = 000001	ECCSTA= 000010	LPSTAT= 000001	PS = 177776	SVR5 = 000074
CR = 000015	ENBEOP= 010000	MAPSTA= 000200	PSW = 177776	SVR6 = 000076
CSRA = 000100	ENBNUL= 000001	MED = 076600	RANNUM= 000054	SYSNT= 000052
CSRC = 000102	ENDLST= 000000		RBUFEA= 000130	SYSERR= 000100

TMP10 = 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	WBUFQ= 000140	\$F\$GDO= 000400	\$LSTTA= 000001	\$FROM= 000000
UIPAR3= 177646	WBUFSZ= 000142	\$F\$IF = 000110	\$NESTL= 177777	\$LOC = 000066R
UIPAR4= 177650	WDFR = 000116	\$F\$INC= 000210	\$NSKO = 000300	\$LOCN= 000000
UIPAR5= 177652	WDT0 = 000114	\$F\$LGO= 000200	\$NSK1 = 000120	\$REG = 177777
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$SAVLE= 177777	\$RETU= 000000
UIPAR7= 177656	WTWHMI= 000222	\$F\$NO = 000403	\$SSKO = 050003	\$RTN1= 050000
UIPDR0= 177600	XFLAG = 000005	\$F\$DR = 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	\$TSKO = 050002	\$TGS1= 000000
UIPDR4= 177610	\$ERFLG= 000400	\$F\$THE= 000330	\$TSK1 = 050003	\$TGS2= 000000
UIPDR5= 177612	\$F\$AND= 000310	\$F\$TRU= 000404	\$SARGC= 000000	\$TC = 000000
UIPDR6= 177614	\$F\$BAD= 000401	\$F\$UNT= 000130	\$S\$YTE= 000403	\$TAG= 050000
UIPDR7= 177616	\$F\$BLA= 000170	\$F\$WHI= 000120	\$S\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:33  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0152

3 COMMON EQUATE MODULE  
565 COMMON DEFINITIONS AND REFERENCES FOR PARERR  
568 000000' .PRINT ;SPMAC: VERSION 1.1  
621 PARERR ROUTINE

```
508 .TITLE PARERR PROCESS PARITY ERROR TRAPS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     PARERR
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE PROCESSES PARITY ERROR TRAPS. WHEN ENTERED
517 ;     IT WILL TEST FOR PARITY ENABLED. IF NOT ENABLED, AN APPROPRIATE
518 ;     MESSAGE IS ENQUEUED.
519 ;     A CHECK IS PERFORMED TO DETERMINE WHICH PARITY CSR
520 ;     HAS THE ERROR BIT SET. THE CSR AND ITS CONTENTS ARE ENQUEUED
521 ;     IN A PARITY MESSAGE(STANDARD).
522 ;
523 ; INPUTS:
524 ;     ADDRESS OF DATA TABLE
525 ;
526 ; IMPLICIT INPUTS:
527 ;     DT.STO
528 ;     DT.CFO
529 ;     DT.PTA
530 ;
531 ; OUTPUTS:
532 ;     NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ;     NONE
536 ;
537 ; PATHOLOGICAL CONNECTIONS:
538 ;     NONE
539 ;
540 ; SUBORDINATE ROUTINES CALLED:
541 ;     ENQTQ -     QUEUE A MESSAGE
542 ;     BOA16 -    BINARY TO OCTAL ASCII CONVERSION ROUTINE
543 ;     KTSET -    SETUP KT REGISTERS
544 ;     SAVREG
545 ;     RESREG
546 ;     MSGDHOOK
547 ;     ERRLOG
548 ;     RSTRCY
549 ;
550 ; FUNCTIONAL SIDE EFFECTS:
551 ;     NONE
552 ;
553 ; CALLING SEQUENCE:
554 ;     CALL PARERR IN <DTADR>
555 ;     WHERE DTADR = ADDRESS OF DATA TABLE
556 ;
557 ; VERSION:
558 ;     0.0
559 ;
560 ;     EDIT             DATE             BY             REASON
561 ;--
562
563
```



```
565 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR PARERR
566
567 .MCALL STRUCT
568 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
569
570 000001 $LSTIN=1
571 000001 $LSTTAG=1
572
573 ;
574 ; REFERENCED BY OTHER MODULES:
575 ;
576 .GLOBL PARERR ;MODULE ENTRY POINT
577 .GLOBL PA.CNT ;NUMBER OF PARITY ERRORS
578 ;
579 ;*****
580 ;
581 ; GLOBAL REFERENCES:
582 ;
583 .GLOBL MSGDHOOK ;ENQUEUE A MESSAGE
584 .GLOBL RSTRCY ;RESET RECOVERY ROUTINE
585 .GLOBL ERRLOG ;1160, 1170 ERROR LOGGING
586 .GLOBL CCNTRL ;CACHE CONTROL REGISTER
587 .GLOBL SAVREG ;SAVE REGISTERS
588 .GLOBL RESREG ;RESTORE REGISTERS
589 .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION ROUTINE
590 .GLOBL KTSET ;SETUP KT APRS'
591 ;
592 ;*****
593 ;
594 ;*****
595 ;
596 ; LOCAL STORAGE:
597 ;
598 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
599 000034' 000 PA.SP: .BYTE 0
600 000035' 040 051503 020122 .ASCII / CSR CONTENTS%/
000042' 020040 041440 047117
000050' 042524 052116 022523
601 000056' 000006 PA.CSR: .BLKB 6
602 000064' 020040 .ASCII / /
603 000066' 000006 PA.CSC: .BLKB 6
604 000074' 000045 .ASCIZ /%/
605
606 000076' 000000 PA.CNT: .WORD 0 ;NUMBER OF PARITY ERRORS
607 000100' 000010 PA.MAX: .WORD 10 ;MAX NUMBER OF ALLOWABLE PARITY ERRORS
608 .EVEN
609 ;*****
610 ; LOCAL EQUATES
611 ;
612 170370 PA.M36 = 170370 ;MAP REGISTER 36 ADDRESS
613 170372 PA.M37 = 170372 ;MAP REGISTER 36 UPPER 6 BITS
```

PARERR PROCESS PARITY ERROR TRAPS  
PARERR.MAC 09-SEP-78 07:43

MACY11 30A(1052) 20-SEP-78 17:33 PAGE 19-2  
COMMON DEFINITIONS AND REFERENCES FOR PARERR

SEQ 0155

614  
615  
616  
617  
618  
619

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

```

621          .SBTTL  PARERR ROUTINE
622
623 000102'  ROUTINE PARERR <DTABLE>
(2) 000102'
624
625          ;+
626          ; INITIALIZE AND SAVE DTABLE ADDRESS
627          ; -
628
629 000102'  CALL SAVREG
(3) 000102' 004767 000000G          JSR    PC, SAVREG
630 000106'  LET R0 := DTABLE(R5)          MOV    DTABLE(R5), R0
(4) 000106' 016500 000000
631
632          ;+
633          ; SET ERROR BIT IN DTABLE'S ERROR WORD
634          ; -
635
636 000112'  LET DT.ERR(R0) := DT.ERR(R0) SET.BY #PAERR
(6) 000112' 052760 000010 000020          BIS    #PAERR, DT.ERR(R0)
637
638
639
640          ;+
641          ; RESET, RECOVER FROM RESET, AND LOWER PRIORITY
642          ; -
643 000120'  INLINE <RESET>
(2) 000120' 000005          RESET
644 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
645 000134'  LET -(SP) := #0
(4) 000134' 005046          CLR    -(SP)
646 000136'  LET -(SP) := #443$
(4) 000136' 012746 000144'          MOV    #443$, -(SP)
647 000142'  INLINE <RTI>
(2) 000142' 000002          RTI
648
649          INLINE <443$:>
(2) 000144'          443$:
650
651          ;+
652          ; IS PARITY OR ECC ENABLED?
653          ; -
654
655 000144'  IF #PARSTAT SETIN DT.STO(R0) OR #ECCSTAT SETIN DT.STO(R0) THEN
(6) 000144' 032760 000100 000010          BIT    #PARSTAT, DT.STO(
(8) 000152' 001004          BNE    50002$
(6) 000154' 032760 000010 000010          BIT    #ECCSTAT, DT.STO(
(9) 000162' 001552          BEQ    50003$
(6) 000164'          50002$:
656
657
658

```

```

659
660
661      ;+
662      ; IF 11/60 OR 11/70, CALL ERRLOG
663      ; -
664      000164'      IF #PDP70 SETIN DT.CFO(R0) OR #PDP60 SETIN DT.CFO(R0) THEN
665      (6) 000164' 032760 010000 000014      BIT      #PDP70,DT.CFO(R0)
666      (8) 000172' 001004      BNE      50004$
667      (6) 000174' 032760 004000 000014      BIT      #PDP60,DT.CFO(R0)
668      (9) 000202' 001405      BEQ      50005$
669      (6) 000204'      50004$:
670      000204'      CALL ERRLOG IN <R0>
671      (3) 000204' 010546      MOV      R5,-(SP)
672      (4) 000206' 010045      MOV      R0,-(R5)
673      (3) 000210' 004767 000000G      JSR      PC,ERRLOG
674      (3) 000214' 012605      MOV      (SP)+,R5
675      000216'      ENDIF
676      (4) 000216'      50005$:
677
678      ;+
679      ; GET PTABLE ADDRESS
680      ; -
681      000216'      LET R1 := DT.PTA(R0)
682      (4) 000216' 016001 000064      MOV      DT.PTA(R0),R1
683
684      ;+
685      ; IF INDIRECT CSRS THEN SAVE APPROPRIATE REGISTERS (CACHE, KT AND MAP)
686      ; -
687      000222'      IF #INDPAR SETIN DT.CFO(R0) THEN
688      (6) 000222' 032760 000040 000014      BIT      #INDPAR,DT.CFO(R0)
689      (9) 000230' 001443      BEQ      50006$
690      000232'      LET -(SP) := @CCNTRL
691      (4) 000232' 017746 000000G      MOV      @CCNTRL,-(SP)
692      000236'      LET -(SP) := @#SR0
693      (4) 000236' 013746 177572      MOV      @#SR0,-(SP)
694      000242'      LET -(SP) := @#SR3
695      (4) 000242' 013746 172516      MOV      @#SR3,-(SP)
696      000246'      LET -(SP) := @#PA.M36
697      (4) 000246' 013746 170370      MOV      @#PA.M36,-(SP)
698      000252'      LET -(SP) := @#PA.M37
699      (4) 000252' 013746 170372      MOV      @#PA.M37,-(SP)
700      000256'      LET -(SP) := @#KIPAR6
701      (4) 000256' 013746 172354      MOV      @#KIPAR6,-(SP)
702
703      ;+
704      ; NOW SETUP APPROPRIATE REGISTERS TO GO AFTER INDIRECT CSRS
705      ; -
706      000262'      LET @CCNTRL := @CCNTRL SET.BY #14
707      (6) 000262' 052777 000014 000000G      BIS      #14,@CCNTRL
708      000270'      CALL KTSET IN <R0>
709      (3) 000270' 010546      MOV      R5,-(SP)
710      (4) 000272' 010045      MOV      R0,-(R5)
711      (3) 000274' 004767 000000G      JSR      PC,KTSET

```

```

(3) 000300' 012605
692 000302'
(4) 000302' 012737 177400 172354
693 000310'
(4) 000310' 012737 160000 170370
694 000316'
(4) 000316' 012737 000077 170372
695 000324'
(6) 000324' 052737 000001 177572
696 000332'
(6) 000332' 052737 000060 172516
697 000340'
(4) 000340'
698
699
700
701
702
703
704 000340'
(4) 000340'
(6) 000340' 005711
(9) 000342' 001462
705 000344'
(6) 000344' 032771 100000 000000
(8) 000352' 001004
(6) 000354' 032771 000020 000000
(9) 000362' 001451
(6) 000364'
706 000364'
(4) 000364' 012103
707 000366'
(3) 000366' 010546
(5) 000370' 012745 000056'
(4) 000374' 010345
(3) 000376' 004767 000000G
(3) 000402' 012605
708 000404'
(3) 000404' 010546
(5) 000406' 012745 000066'
(4) 000412' 011345
(3) 000414' 004767 000000G
(3) 000420' 012605
709 000422'
(4) 000422' 112767 000040 177404
710 000430'
(3) 000430' 010546
(7) 000432' 012745 000462'
(6) 000436' 012745 000000'
(5) 000442' 012745 000002
(4) 000446' 010045
(3) 000450' 004767 000000G
(3) 000454' 012605
711 000456'
(2) 000456' 000240
712 000460'

      LET @#KIPAR6 := #177400
      LET @#PA.M36 := #160000
      LET @#PA.M37 := #77
      LET @#SR0 := @#SR0 SET.BY #BIT00
      LET @#SR3 := @#SR3 SET.BY #60
      ENDIF
      50006$:
      ;+
      ; SEARCH PARITY TABLE UNTIL ERROR IS FOUND AND O/P MESSAGE...
      ;-
      WHILE (R1) NE #0 DO
      50007$:
      TST (R1)
      BEQ 50010$
      IF #BIT15 SETIN @(R1) OR #BIT04 SETIN @(R1) THEN
      BIT #BIT15,@(R1)
      BNE 50011$
      BIT #BIT04,@(R1)
      BEQ 50012$
      50011$:
      LET R3 := (R1)+
      MOV (R1)+,R3
      CALL BOA16 IN <R3,#PA.CSR>
      MOV R5,-(SP)
      MOV #PA.CSR,-(R5)
      MOV R3,-(R5)
      JSR PC,BOA16
      MOV (SP)+,R5
      CALL BOA16 IN <(R3),#PA.CSC>
      MOV R5,-(SP)
      MOV #PA.CSC,-(R5)
      MOV (R3)-,(R5)
      JSR PC,BOA16
      MOV (SP)+,R5
      LET PA.SP :B= #SPACE
      MOVB #SPACE,PA.SP
      CALL MSGDHOOK IN <R0,#MSGPOP,#PA.MSG,#20$>
      MOV R5,-(SP)
      MOV #20$,-(R5)
      MOV #PA.MSG,-(R5)
      MOV #MSGPOP,-(R5)
      MOV R0,-(R5)
      JSR PC,MSGDHOOK
      MOV (SP)+,R5
      INLINE <60$: NOP>
      60$: NOP
      INLINE <BR 60$>

```

```
(2) 000460' 000776  
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) 000468' 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 ;+  
726 ; NO PARITY OR ECC ENABLED SO ISSUE ERROR MSG -  
727 ; OR NO ERROR WAS FOUND IN THE PARITY CSRS.  
728 ; DO NOT PRINT THE MSG IF THIS IS AN 11/70 BECAUSE  
729 ; IT DOES NOT HAVE PARITY CSRS. IT MAY OR MAY NOT  
730 ; HAVE ECC CSRS IN THE TABLE  
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 MSGDH00K 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,MSGDH00K  
(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$:
```

```
743
744 ;+
745 ; IF INDIRECT CSRS THEN RESTORE CACHE, KT AND MAP REGISTERS
746 ;-
747
748 000556'          INLINE <100$:>
(2) 000556'
749 000556'          IF #INDPAR SETIN DT.CFO(R0) THEN
(6) 000556' 032760 000040 000014          BIT      #INDPAR,DT.CFO(R
(9) 000564' 001414          BEQ      50015$
750 000566'          LET @#KIPAR6 := (SP)+
(4) 000566' 012637 172354          MOV      (SP)+,@#KIPAR6
751 000572'          LET @#PA.M37 := (SP)+
(4) 000572' 012637 170372          MOV      (SP)+,@#PA.M37
752 000576'          LET @#PA.M36 := (SP)+
(4) 000576' 012637 170370          MOV      (SP)+,@#PA.M36
753 000602'          LET @#SR3 := (SP)+
(4) 000602' 012637 172516          MOV      (SP)+,@#SR3
754 000606'          LET @#SR0 := (SP)+
(4) 000606' 012637 177572          MOV      (SP)+,@#SR0
755 000612'          LET @CCNTRL := (SP)+
(4) 000612' 012677 000000G          MOV      (SP)+,@CCNTRL
756
757 000616'          ENDIF
(4) 000616'          50015$:
758 ;+
759 ; RESTORE REGISTERS AND RETURN
760 ;-
761
762 000616'          CALL RESREG
(3) 000616' 004767 000000G          JSR      PC,RESREG
763
764
765 000622'          ENDRTN
(3) 000622'
(3) 000622'
(2) 000622' 000207          50000$:
766          000001          .END          50001$:
          RTS      PC
```

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	PRO = 000000
ADR = 000006	CTRLD = 000017	ERRLOG= ***** G	MODHOL= 002000	PR4 = 000200
APTFER= 000004	CTRLU = 000025	ERRTYP= 000105	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	DROPMD= 100000	EVNTPE= 000202	MSGDHO= ***** G	PSW = 177776
AUTO = 000010	DSEVNT= 000014	EVNTRE= 000201	MSGDRP= 000017	RANNUM= 000054
AUTJST= 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	ICGNT = 000036	MSGHNG= 000022	RBUFVA= 000124
BIT01 = 000002	DT.BLS= 000034	ICGUNT= 000040	MSGHRD= 000007	RDSERV= 000101
BIT02 = 000004	DT.CFO= 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	INHRE= 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.KER= 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	RSVALU= 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= 000052	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	SENDSL= 177777
CAPRES= 000004	DT.ST1= 000012	KIPDR6= 172314	PA.M36= 170370	SDFCNT= 000042
CASTAT= 000004	DT.SWR= 000056	KIPDR7= 172316	PA.M37= 170372	SQFPAS= 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	SRO = 177572
CLKPRE= 000001	DVID1 = 000014	KTXTND= 040000	PRI0 = 000000	SR1 = 177574
CONFIG= 000056	ECCMEM= 000100	LF = 000012	PRI1 = 000040	SR2 = 177576
CQCWF = 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 = 000034	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\$G00= 000400	\$NSK0 = 000300	\$\$LGC = 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\$L00= 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
TMPID = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:34  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0163

3 COMMON EQUATE MODULE  
583 PCDATA (COMMON DEFINITIONS AND REFERENCES)  
586 000000' .PRINT ;SPMAC: VERSION 1.1  
634 PCDATA (CODE)  
1012 TOP16 SUBROUTINE  
1052 TOP12 SUBROUTINE

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\$GDO= 000400	\$LSTIN= 000001	\$\$FROM= 000000
UIPAR1= 177642	WBUFRQ= 000140	\$F\$IF = 000110	\$LSTTA= 000001	\$\$LOC = 000172R
UIPAR2= 177644	WBUFSS= 000142	\$F\$INC= 000210	\$NESTL= 177777	\$\$LOCN= 000000
UIPAR3= 177646	WDFR = 000116	\$F\$L00= 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)

```
508 .TITLE PCDATA (PROCESS CDATA & DATCK TRAP CALLS)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PCDATA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS IS A TWO ENTRY MODULE (PCDATA & PDATCK BEING ENTRY POINTS).
517 ; THIS MODULE IS CALLED AS A RESULT OF CDATA OR DATCK TRAP CALL
518 ; EXECUTED BY AN OPTION MODULE, TO CHECK A BLOCK OF DATA
519 ; TRANSFERRED BY A DEVICE.
520 ;
521 ; IF THERE ARE NO ERRORS IN THE BLOCK, CONTROL RETURNS IMME-
522 ; DIATELY TO THE INSTRUCTION FOLLOWING THE TRAP CALL IN THE MODULE.
523 ;
524 ; IF THERE ARE ERRORS, HOWEVER, AN ERROR MESSAGE IS ENQUEUED (WITH
525 ; TRAP ADDRESS FILLED AS THE MODULE'S RETURN ADDRESS), WORD
526 ; COUNT AND ERROR COUNT ARE SAVED IN THE MODULE HEADER.
527 ;
528 ; IF 'PRINT ALL DATA ERRORS' SWITCH IS SET, ALL ERRORS ARE
529 ; ENQUEUED, OTHERWISE ONLY THE FIRST THREE ERRORS ARE ENQUEUED AND
530 ; THE REST ARE LOGGED. THE SOFT ERROR COUNT IS INCREMENTED EVERY
531 ; TIME AN ERROR OCCURS AND AT THE END OF THE BLOCK CHECK, A SUMMARY MESSAGE
532 ; IS PRINTED FOR CDATA CALLS. SOFT
533 ; ERROR WORD AND SOFT ERRORS PER PASS WORD ARE INCREMENTED IN THE MODULE'S
534 ; HEADER.
535 ;
536 ; IF THE TRAP CALL IS DATCK, THE TOTAL NUMBER OF ERRORS IS PASSED
537 ; TO A LOCATION SPECIFIED IN THE TRAP CALL.
538 ;
539 ;
540 ; INPUTS:
541 ; 1. DTABLE ADDRESS
542 ;
543 ; IMPLICIT INPUTS:
544 ; 1. DT.PC
545 ; 2. DT.STO
546 ; 3. DT.SWR
547 ; 4. DT.EVNT
548 ;
549 ; OUTPUTS:
550 ; NONE
551 ;
552 ; IMPLICIT OUTPUTS:
553 ; 1. DT.STO
554 ; 2. DT.PC
555 ;
556 ; PATHOLOGICAL CONNECTIONS:
557 ; NONE
558 ;
559 ; SUBORDINATE ROUTINES CALLED:
560 ; 1. ENQTQ ;ENQUEUES AN ERROR MESSAGE
561 ; 2. UNIPA ;MAP22 ROUTINE
562 ; 3. SAVREG ;
563 ; 4. RESREG ;
```

```
564 ; 5. GETPSW ;GETS PS WORD
565 ; 6. APTSER ;APT SOFT ERROR ROUTINE
566 ; 7. DRPMOD ;DROP A MODULE
567 ;
568 ; FUNCTIONAL SIDE EFFECTS:
569 ; NONE
570 ;
571 ; CALLING SEQUENCE:
572 ; CALL PCDATA IN <AA>
573 ; CALL PDATCK IN <AA>
574 ;
575 ; WHERE AA = DTABLE ADDRESS
576 ;
577 ; VERSION:
578 ; 0.0
579 ;
580 ; EDIT DATE BY REASON
581 ;--
```

```
583          .SBTTL PCDATA (COMMON DEFINITIONS AND REFERENCES)
584
585          .MCALL STRUCT
586 000000'    STRUCT
587 (1) 000000'    .PRINT ;SPMAC: VERSION 1.1
588          000001    $LSTIN=1
589          000001    $LSTTAG=1
590
591          ;*****
592          ; REFERENCED BY OTHER MODULES
593          ;
594          .GLOBL PCDATA          ;MODULE ENTRY POINT
595          .GLOBL PDATCK          ;MODULE ENTRY POINT
596          ;*****
597          ; GLOBAL REFERENCES
598          ;
599          .GLOBL ENQTQ           ;ENQUE A MESSAGE ROUTINE
600          .GLOBL UNIPA           ;MAP22 ROUTINE
601          .GLOBL DRPMOD          ;DROP A MODULE
602          .GLOBL SAVREG
603          .GLOBL RESREG
604          .GLOBL GETPSW          ;GET PS WORD ROUTINE
605          .GLOBL APTSER
606
607          ;*****
608          ; LOCAL STORAGE
609          ;
610          ; NOTE: DO NOT DISTURB THE ORDER OF THE NEXT 10 WORDS AS THEY FORM
611          ;       A TABLE.
612
613 000000' 000000    PC.RPA: 0          ;RBUFPA WORD
614 000002' 000000    PC.REA: 0          ;RBUF EA WORD
615 000004' 000000    PC.RLO: 0          ;RBUF LOWER 16 BITS
616 000006' 000000    PC.RHI: 0          ;RBUF UPPER 2 BITS/6 BITS
617 000010' 000000    PC.RTP: 0          ;RBUF TOP 12 BITS/16 BITS
618 000012' 000000    PC.WPA: 0          ;WBUFPA WORD
619 000014' 000000    PC.WEA: 0          ;WBUF EA WORD
620 000016' 000000    PC.WLO: 0          ;WBUF LOWER 16 BITS
621 000020' 000000    PC.WHI: 0          ;WBUF UPPER 2 BITS/6 BITS
622 000022' 000000    PC.WTP: 0          ;WBUF TOP 12 BITS/16 BITS
623
624 000024' 000000    PC.RSZ: 0          ;READ BUFFER SIZE
625 000026' 000000    PC.TPC: 0          ;PC OF CALL
626 000030' 000000    PC.TMP: 0          ;TEMP STORAGE
627 000032' 000000    PC.DCF: 0          ;DATCK FLAG
628          .WORD 0          ;USER MODE STACK
629          .WORD 0
630          .WORD 0
631          .WORD 0
632
```

```

634 .SBTTL PCDATA (CODE)
635
636 000044' ROUTINE PCDATA <DT>
(2) 000044'
637 PCDATA:
638 000044' LET PC.DCF := #0
(4) 000044' 005067 177762 CLR PC.DCF
639 000050' INLINE <BR COM>
(2) 000050' 000403 BR COM
640
641 000052' ROUTINE PDATA <DT>
(2) 000052' PDATA:
642
643 000052' LET PC.DCF := #1
(4) 000052' 012767 000001 177752 MOV #1,PC.DCF
644
645 ;+
646 ; SAVE REGISTERS
647 ;-
648
649 000060' INLINE <COM:>
(2) 000060' COM:
650 000060' CALL SAVREG
(3) 000060' 004767 000000G JSR PC,SAVREG
651
652 ;+
653 ; SET R0 TO DTABLE ADDRESS, R1 TO HEADER ADDRESS AND R2 TO TABLE ADDRESS
654 ; POINTING TO RBUF PA,EA & SIZE. R3 AND R4 WILL BE POINTERS TO READ AND
655 ; WRITE BUFFERS RESPECTIVELY. R2 ALSO SERVES AS A WHILE-DO LOOP CONTROL.
656 ;-
657
658 000064' LET R0 := DT(R5)
(4) 000064' 016500 000000 MOV DT(R5),R0
659 000070' LET R1 := @DT.PC(R0)
(4) 000070' 017001 000002 MOV @DT.PC(R0),R1
660
661 ;+
662 ; DEVELOP RETURN ADDRESS FOR MESSAGE
663 ;-
664 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
665
666 ;+
667 ; GET ADDRESS OF RBUF,PA,EA
668 ;-
669
670 000110' LET R2 := PC.TPC + #4
(4) 000110' 016702 177712 MOV PC.TPC,R2
(6) 000114' 062702 000004 ADD #4,R2
671
672 ;+
673 ; R3 GETS PA WORD AND SAVE ALSO IN PC.RPA
674 ;-
675
676 000120' LET R2 := (R2)

```

```

(4) 000120' 011202                                MOV      (R2),R2
677 000122'                                LET R3 := (R2)+
(4) 000122' 012203                                MOV      (R2)+,R3
678 000124'                                LET PC.RPA := R3
(4) 000124' 010367 177650                          MOV      R3,PC.RPA
679
680 ;+
681 ; SAVE EA BITS AND SIZE
682 ; -
683
684 000130'                                LET PC.REA := (R2)+
(4) 000130' 012267 177646                          MOV      (R2)+,PC.REA
685 000134'                                LET PC.RSZ := (R2)
(4) 000134' 011267 177664                          MOV      (R2),PC.RSZ
686 ;
687 ;+
688 ; GET WBUFPA INTO R4
689 ; -
690
691 000140'                                LET R4 := WBUFPA(R1)
(4) 000140' 016104 000134                          MOV      WBUFPA(R1),R4
692
693 ;+
694 ; FINALLY GET WBUFEA BITS INTO LOCAL STORAGE
695 ; -
696
697 000144'                                LET PC.WPA := R4
(4) 000144' 010467 177642                          MOV      R4,PC.WPA
698 000150'                                LET PC.WEA := WBUFEA(R1)
(4) 000150' 016167 000136 177636                  MOV      WBUFEA(R1),PC.WE
699
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'
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 ; -
  
```

50002\$:



```

718
719 000200'          LET R2 := R2 CLR.BY #BIT15
(6) 000200' 042702 100000
720 000204'          LET R2 := R2 + #1          ;UPDATE WORD COUNT
(6) 000204' 005202
721
722                ;+
723                ; UPDATE READ BUFFER POINTER AND EA BITS
724                ; -
725
726 000206'          LET R2 := R2 SHIFT 1          ;FORM OFFSET
(7) 000206' 006302
727 000210'          LET R3 := R3 + R2          ;NEW RBUF PTR
(6) 000210' 060203
728 000212'          LET PC.RPA := R3
(4) 000212' 010367 177562
729 000216'          IFCOND CS THEN
(6) 000216' 103003
730 000220'          LET PC.REA := PC.REA + #20
(6) 000220' 062767 000020 177554
731 000226'          ENDIF
(4) 000226'
732
733                ;+
734                ; UPDATE WRITE BUFFER POINTER AND IT'S EA BITS
735                ; -
736
737 000226'          LET R4 := R4 + R2          ;NEW WBUFF PTR
(6) 000226' 060204
738 000230'          IFCOND CS THEN
(6) 000230' 103003
739 000232'          LET PC.WEA := PC.WEA + #20
(6) 000232' 062767 000020 177554
740 000240'          ENDIF
(4) 000240'
741
742                ;+
743                ; RESTORE WORD COUNT
744                ; -
745
746 000240'          LET R2 := R2 SHIFT -1
(7) 000240' 006202
747 000242'          ENDIF
(4) 000242'
748
749                ;+
750                ; NOW LOAD LOCAL STORAGE WITH WRITE BUFFER PA IN R4
751                ; -
752 000242'          LET PC.WPA := R4
(4) 000242' 010467 177544
753
754                ;+
755                ; IF KT IS ON THEN DETERMINE PHYSICAL ADDRESS
756                ; IN PAR FORMAT.....
757                ; -
758

```

50004\$:

50005\$:

50003\$:

BIC #BIT15,R2  
 INC R2  
 ASL R2  
 ADD R2,R3  
 MOV R3,PC.RPA  
 BCC 50004\$  
 ADD #20,PC.REA  
 ADD R2,R4  
 BCC 50005\$  
 ADD #20,PC.WEA  
 ASR R2  
 MOV R4,PC.WPA

```

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 000346' ELSE
(4) 000346' 000414 BR 50010$
(3) 000350'
771 000350' CALL TOP12 IN <#PC.RPA> 50007$:
(3) 000350' 010546 MOV R5,-(SP)
(4) 000352' 012745 000000' MOV #PC.RPA,-(R5)
(3) 000356' 004767 000752 JSR PC, TOP12
(3) 000362' 012605 MOV (SP)+,R5
772 000364' CALL TOP12 IN <#PC.WPA>
(3) 000364' 010546 MOV R5,-(SP)
(4) 000366' 012745 000012' MOV #PC.WPA,-(R5)
(3) 000372' 004767 000736 JSR PC, TOP12
(3) 000376' 012605 MOV (SP)+,R5
773 000400' ENDIF
(4) 000400' 50010$:
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

```

782	000414'	016737	177370	177654		MOV	PC.RTP,@#UIPAR6
783	000422'				LET @#UIPAR7 := #177600		
784	000422'	012737	177600	177656		MOV	#177600,@#UIPAR7
785					;++		
786					; SET R3 TO SELECT PAR6 AND R4 TO SELECT PAR5 AND ALSO		
787					; LOAD R3 AND R4 WITH LOWER 6 BITS OF RBUF AND WBUF ADDRESSES		
788					;-		
789	000430'				LET R3 := R2 CLR.BY #177700		
790	000430'	042703	177700			BIC	#177700,R3
791	000434'				LET R3 := R3 SET.BY #140000		
792	000434'	052703	140000			BIS	#140000,R3
793	000440'				LET R4 := R4 CLR.BY #177700		
794	000440'	042704	177700			BIC	#177700,R4
795	000444'				LET R4 := R4 SET.BY #120000		
796	000444'	052704	120000			BIS	#120000,R4
797					;++		
798					; GET THE PSW AND GO TO USER MODE		
799					;-		
800	000450'				CALL GETPSW OUT <PC.TMP>		
801	000450'	162705	000002			SUB	#1*2,R5
802	000454'	004767	000000G			JSR	PC,GETPSW
803	000460'	012567	177344			MOV	(R5)+,PC.TMP
804	000464'				LET PC.TMP := PC.TMP SET.BY #140000		
805	000464'	052767	140000	177336		BIS	#140000,PC.TMP
806	000472'				PUSH PC.TMP,#33\$		
807	000472'	016746	177332			MOV	PC.TMP,-(SP)
808	000476'	012746	000504'			MOV	#33\$,-(SP)
809	000502'				INLINE <RTI>		
810	000502'	000002				RTI	
811	000504'				INLINE <33\$:>		
812	000504'					33\$:	
813					;++		
814					; SET UP USER MODE STACK		
815					;-		
816	000504'				LET SP := #PCDATA		
817	000504'	012706	000044'			MOV	#PCDATA,SP
818	000510'				ENDIF		
819	000510'					50006\$:	
820	000510'				INLINE <1\$:>		
821	000510'					1\$:	
822					;++		
823					; NOW WE ARE READY TO COMPARE DATA WORDS		
824					;-		
825	000510'				WHILE R2 LT PC.RSZ DO		
826	000510'					50011\$:	
827	000510'	020267	177310			CMP	R2,PC.RSZ
828	000514'	002113				BGE	50012\$
829	000516'				LET R2 := R2 + #1		

```

(6) 000516' 005202
818 000520'          IF (R3) NE (R4) THEN          INC      R2
(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$:
827 000550'          LET PC.WPA := (R3)
(4) 000550' 011367 177236          MOV      (R3),PC.WPA
828 000554'          LET PC.WEA := (R4)
(4) 000554' 011467 177234          MOV      (R4),PC.WEA
829 000560'          IF #KTSTAT SETIN DT.ST0(R0) THEN
(6) 000560' 032760 000020 000010          BIT      #KTSTAT,DT.ST0(R0)
(9) 000566' 001406          BEQ      50016$
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          MCV      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

```



```

880
881
882 ;+
883 ; NOW YOU GO TO KERNEL MODE, FOR SURE. CHECK IF THERE WERE ANY ERRORS.
884 ; IF NONE, UPDATE DT.PC FOR PROPER RETURN AND CLEAR THE DEFERRED SERVICE
885 ; BIT IN DT.STO, AND THAT'S IT !!!
886 ;-
887 000744' IF #KTSTAT SETIN DT.STO(R0) THEN
(6) 000744' 032760 000020 000010 BIT #KTSTAT,DT.STO(R
(9) 000752' 001406 BEQ 50023$
888 000754' LET PC.TMP := PC.TMP CLR.BY #140000
(6) 000754' 042767 140000 177046 BIC #140000,PC.TMP
889
890 ;+
891 ; *****
892 ; NOTE !!!!! DIRECT REFERENCE TO PSW FOLLOWS !!!!! IN AN LSI
893 ; MACHINE WITH KT ( SHOULD THIS EVER EXIST ), DIRECT
894 ; ACCESS TO PSW MAY NOT BE ALLOWED !!! THE USE OF SOME TRAP MAY
895 ; BE NECESSARY TO SWITCH TO KERNEL !!!!!
896 ; *****
897 ;-
898 000762' LET @#PSW := PC.TMP
(4) 000762' 016737 177042 177776 MOV PC.TMP,@#PSW
899 000770' ENDIF
(4) 000770' 50023$:
900
901
902
903 ;+
904 ; IF ERRORS, UPDATE ERRORS COUNTS AND DO APT STUFF IF NECESSARY
905
906 000770' IF CDERCT(R1) NE #0 THEN
(6) 000770' 005761 000146 TST CDERCT(R1)
(9) 000774' 001447 BEQ 50024$
907 000776' LET SOFCNT(R1) := SOFCNT(R1) + #1
(6) 000776' 005261 000042 INC SOFCNT(R1)
908 001002' LET SOFPAS(R1) := SOFPAS(R1) + #1
(6) 001002' 005261 000046 INC SOFPAS(R1)
909
910 ;+
911 ; IF UNDER APT, CALL APT SOFT ERROR ROUTINE
912 ;-
913 001006' IF #APTPRES SETIN DT.CFO(R0) THEN
(6) 001006' 032760 000200 000014 BIT #APTPRES,DT.CFO(
(9) 001014' 001406 BEQ 50025$
914 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
915 001030' ELSE
(4) 001030' 000431 BR 50026$
(3) 001032' 50025$:
916
917 ;+
918 ; NOT APT.....SEE IF TOO MANY ERRORS

```

```

919          ; -
920
921
922          ; +
923          ; IF BIT 15 SET IN SWITCH REGISTER... OR IF TOO MANY SOFT ERRORS AND
924          ; BIT 14 SET IN SWITCH REGISTER... ALSO DUMP MODULE....
925          ; -
926
927          IF #BIT15 SET IN DT.SWR(R0) THEN
(6) 001032' 032760 100000 000056          BIT          #BIT15,DT.SWR(R0)
(9) 001040' 001407          BEQ          50027$
928          CALL DRPMOD IN <R0,R1>
(3) 001042' 010545          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
929          ELSE
(4) 001056' 000415          BR          50030$
(3) 001060'          50027$:
930          IF SOFCNT(R1) HI DT.SMX(R0) AND #BIT14 NOTSET IN DT.SWR(R0) THEN
(6) 001060' 026160 000042 000106          CMP          SOFCNT(R1),DT.SM
(9) 001066' 101412          BLOS         50031$
(6) 001070' 032760 040000 000056          BIT          #BIT14,DT.SWR(R0)
(9) 001076' 001003          BNE         50031$
931          CALL DRPMOD IN <R0,R1>
(3) 001100' 010545          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
932          ENDIF
(4) 001114'          50031$:
933          ENDIF
(4) 001114'          50030$:
934          ENDIF
(4) 001114'          50026$:
935
936          ENDIF
(4) 001114'          50024$:
937
938          ; +
939          ; UPDATE RETURN P.C.
940          ; -
941          LET DT.PC(R0) := DT.PC(R0) + #6
(6) 001114' 062760 000006 000002          ADD          #6,DT.PC(R0)
942
943          ; +
944          ; IF DATAACK, UPDATE AGAIN
945          ; -
946
947          IF PC.DCF NE #0 THEN
(6) 001122' 005767 176704          TST          PC.DCF
(9) 001126' 001403          BEQ          50032$
948          LET DT.PC(R0) := DT.PC(R0) + #2
(6) 001130' 062760 000002 000002          ADD          #2,DT.PC(R0)

```

```
949 001136'          ENDIF
(4) 001136'
950
951                ;+
952                ; MARK IT AS DEFFERRED RETURN
953                ; -
954
955 001136'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #DEFRTN
(6) 001136' 042760 000400 000010          BIC          #DEFRTN,DT.STO(R
956
957                ;+
958                ; IF THERE ARE SOME ERRORS, IF COUNT IS DONE, CHECK
959                ; IF THE TRAP CALL WAS A CDATA CALL. IF SO, ENQUE THE ERROR
960                ; SUMMARY MESSAGE AND CHANGE THE RETURN ADDRESS TO LOCATION
961                ; IN THE TRAP CALL. FOR DATCK TRAP, CLEAR THE DEFERRED SERVICE BIT
962                ; IN DT.STO AND LOAD ERROR COUNT INTO LOCATION IN THE TRAP CALL.
963                ; IN CASE YOU FORGOT R2 CONTAINED WORD COUNT SO FAR.
964                ; -
965
966 001144'          INLINE <3$:>
(2) 001144'
967 001144'          IF CDERCT(R1) NE #0 THEN          3$:
(6) 001144' 005761 000146          TST          CDERCT(R1)
(9) 001150' 001440          BEQ          50033$
968 001152'          IF R2 GE PC.RSZ THEN
(6) 001152' 020267 176646          CMP          R2,PC.RSZ
(9) 001156' 002435          BLT          50034$
969
970
971
972
973                ;+
974                ; UPDATE WORD COUNT AND SET R2 TO RETURN ADDRESS
975                ; -
976 001160'          LET CDWDCT(R1) := R2
(4) 001160' 010261 000144          MOV          R2,CDWDCT(R1)
977 001164'          LET R2 := PC.TPC + #6
(4) 001164' 016702 176636          MOV          PC.TPC,R2
(6) 001170' 062702 000006          ADD          #6,R2
978
979                ;+
980                ; IF CDATA, ENQUEUE SUMMARY MESSAGE
981                ; -
982 001174'          IF PC.DCF EQ #0 THEN
(6) 001174' 005767 176632          TST          PC.DCF
(9) 001200' 001014          BNE          50035$
983 001202'          CALL ENQTQ IN <R0,#MSGCKS,PC.TPC,R1,(R2)>
(3) 001202' 010546          MOV          R5,-(SP)
(8) 001204' 011245          MOV          (R2),-(R5)
(7) 001206' 010145          MOV          R1,-(R5)
(6) 001210' 016745 176612          MOV          PC.TPC,-(R5)
(5) 001214' 012745 000011          MOV          #MSGCKS,-(R5)
(4) 001220' 010045          MOV          R0,-(R5)
(3) 001222' 004767 000000G          JSR          PC,ENQTQ
(3) 001226' 012605          MOV          (SP)+,R5
984 001230'          ELSE
```



PCDATA.MAC 31-JUL-78 16:22

PCDATA (CODE)

```

(4) 001230' 000410
(3) 001232'
985
986
987 ;+
988 ; DATAACK.....R2 CONTAINS RETURN ADDRESS
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
(4) 001244' 042760 000400 000010 BIC #DEFRTN,DT.ST0(R)
997 001252' ENDIF
(4) 001252' ENDIF 50036$:
998 001252' ENDIF 50034$:
999 001252' ENDIF 50033$:
(4) 001252'
1000
1001
1002
1003
1004 ;+
1005 ; CLEAN UP AND GOODBYE
1006 ; -
1007
1008 001252' CALL RESREG
(3) 001252' 004767 000000G JSR PC,RESREG
1009 001256' ENDRTN
(3) 001256' 50000$:
(3) 001256' 50001$:
(2) 001256' 000207 RTS PC
1010

```

```

1012          .SBTTL TOP16 SUBROUTINE
1013
1014 001260'          ROUTINE TOP16 <AA>
1015          (2) 001260'
1016
1017          ;+
1018          ; THIS ROUTINE COMBINES THE 6 EA BITS WITH THE TOP 10 BITS OF
1019          ; THE PA WORD AND RETURNS THE RESULT IN PC.RTP/PC.WTP WORD.
1020          ; AA IS THE TABLE ADDRESS AND THE TABLE IS OF THE FORM:
1021          ;
1022          ;
1023          ;
1024          ;-
1025 001260'          CALL SAVREG
1026          (3) 001260' 004767 000000G
1027
1028          ;+
1029          ; SET R0 TO TABLE ADDRESS, R2 GETS 16-BIT PA AND R1 GETS 6-BIT
1030          ; EA. SHIFT PA 6 PLACES TO THE RIGHT AND BRING EA BITS IN POSITION
1031          ; 15-10. ADD EA AND PA TO GET TOP 16-BIT WORD. RETURN THE
1032          ; RESULT.
1033          ;-
1034 001264'          LET R0 := AA(R5)
1035          (4) 001264' 016500 000000
1036          001270'          LET R2 := (R0)+
1037          (4) 001270' 012002
1038          001272'          LET R1 := (R0)+
1039          (4) 001272' 012001
1040          001274'          LET R2 := R2 SHIFT -6
1041          (7) 001274' 006202
1042          (7) 001276' 006202
1043          (7) 001300' 006202
1044          (7) 001302' 006202
1045          (7) 001304' 006202
1046          (7) 001306' 006202
1047          001310'          LET R2 := R2 CLR.BY #176000
1048          (6) 001310' 042702 176000
1049          001314'          LET R1 := SWAP R1
1050          (6) 001314' 000301
1051          001316'          LET R1 := R1 SHIFT 2
1052          (7) 001316' 006301
1053          (7) 001320' 006301
1054          001322'          LET (R0) := R2 + R1
1055          (4) 001322' 010210
1056          (6) 001324' 060110
1057
1058          ;+
1059          ; RESTORE REGISTERS
1060          ;-
1061          CALL RESREG
1062          (3) 001326' 004767 000000G
1063
1064          ENDRTN
1065          001332'

```

TOP16:

JSR PC, SAVREG

MOV AA(R5), R0

MOV (R0)+, R2

MOV (R0)+, R1

ASR R2

ASR R2

ASR R2

ASR R2

ASR R2

ASR R2

BIC #176000, R2

SWAB R1

ASL R1

ASL R1

MOV R2, (R0)

ADD R1, (R0)

JSR PC, RESREG

(3) 001332'  
(3) 001332'  
(2) 001332' 000207  
1050

50000\$:  
50001\$:

RTS PC

```

1052          .SBTTL TOP12 SUBROUTINE
1053
1054 001334'    ROUTINE TOP12 <BB>
(2) 001334'
1055
1056          ;+
1057          ; THIS ROUTINE COMBINES THE 2 EA BITS WITH THE TOP 10 BITS OF
1058          ; THE PA WORD AND RETURNS THE RESULT INTO LOCATION PC.RTP/PC.WTP.
1059          ; BB IS THE TABLE ADDRESS AND THE TABLE IS OF THE FORM:
1060          ;
1061          ;           PA
1062          ;           EA
1063          ;           LO
1064          ;           HI
1065          ;           (           ) - RESULT GOES HERE
1066          ;-
1067 001334'    CALL SAVREG
(3) 001334' 004767 000000G
1068
1069          ;+
1070          ; SET R0 TO TABLE ADDRESS. R2 GETS PA AND R1 GETS EA.
1071          ; BRING EA BITS IN POSITION 9-10 AND SHIFT PA 6 PLACES
1072          ; TO THE RIGHT. ADD EA AND PA WORDS TO GET TOP 12 BIT WORD.
1073          ; RETURN THE RESULT.
1074          ;-
1075
1076 001340'    LET R0 := BB(R5)
(4) 001340' 016500 000000
1077 001344'    LET R2 := (R0)+
(4) 001344' 012002
1078 001346'    LET R1 := (R0)+
(4) 001346' 012001
1079 001350'    LET R1 := R1 SHIFT 6
(7) 001350' 006301
(7) 001352' 006301
(7) 001354' 006301
(7) 001356' 006301
(7) 001360' 006301
(7) 001362' 006301
1080 001364'    LET R2 := R2 SHIFT -6
(7) 001364' 006202
(7) 001366' 006202
(7) 001370' 006202
(7) 001372' 006202
(7) 001374' 006202
(7) 001376' 006202
1081 001400'    LET R2 := R2 CLR.BY #176000
(6) 001400' 042702 176000
1082 001404'    LET 4(R0) := R2 + R1
(4) 001404' 010260 000004
(6) 001410' 060160 000004
1083
1084 001414'    CALL RESREG
(3) 001414' 004767 000000G
1085
1086 001420'    ENDRTN
  
```

TOP12:

JSR PC, SAVREG  
 MOV BB(R5), R0  
 MOV (R0)+, R2  
 MOV (R0)+, R1  
 ASL R1  
 ASL R1  
 ASL R1  
 ASL R1  
 ASL R1  
 ASL R1  
 ASR R2  
 ASR R2  
 ASR R2  
 ASR R2  
 ASR R2  
 ASR R2  
 BIC #176000, R2  
 MOV R2, 4(R0)  
 ADD R1, 4(R0)  
 JSR PC, RESREG

PCDATA (PROCESS CDATE & DATCK TRAP CALLS) MACY11 30A(1052) 20-SEP-78 17:34 PAGE 19-17  
PCDATA.MAC 31-JUL-78 16:22 TOP12 SUBROUTINE

SEQ 0181

(3) 001420'  
(3) 001420'  
(2) 001420' 000207  
1087  
1083 000001 .END

50000\$:  
50001\$:  
RTS PC

AA = 000000	CQDVF = 000001	ECCMEM= 000100	LF = 000012	PC.WTP 000022R
ACSR = 000102	CR = 000015	ECCSTA= 000010	LPSTAT= 000001	PDATCK 000052RG
ACTBIT= 004000	CSRA = 000100	ENBEOB= 010000	MAPSTA= 000200	PDPLSI= 020000
ADDR22= 001000	CSRC = 000102	ENBNUL= 000001	MED = 076600	PDP60 = 004000
ADR = 000006	CTRLC = 000003	ENDLST= 000000	MEMPAS= 040000	PDP70 = 010000
APTFER= 000004	CTRL0 = 000017	ENQTQ = ***** G	MODEXH= 004000	PRI0 = 000000
APTPRE= 000200	CTRLU = 000025	EOPBIT= 000001	MODHOL= 002000	PRI1 = 000040
APTSER= ***** G	DCEVNT= 000011	ERRTYP= 000106	MODSEL= 001000	PRI4 = 000200
ASB = 000106	DEFRTN= 000400	EVNTBE= 000200	MSGCKD= 000010	PRI5 = 000240
ASSEMB= 000010	DIAGMC= 000000	EVNTHD= 000200	MSGCKS= 000011	PRI6 = 000300
ASTAT = 000104	DROPMO= 100000	EVNTKT= 000203	MSGDER= 000005	PRI7 = 000340
AUTO = 000010	DRPMOD= ***** G	EVNTPE= 000202	MSGDRP= 000017	PR0 = 000000
AUTJST= 020000	DSEVNT= 000014	EVNTRE= 000201	MSGGECH= 177777	PR4 = 000200
AWAS = 000110	DT = 000000	FATERR= 100000	MSGGEP= 000013	PR5 = 000240
BB = 000000	DT.ADD= 000042	GETPSW= ***** G	MSGHDR= 000004	PR6 = 000300
BIT0 = 000001	DT.AP = 000100	HRDCNT= 000044	MSGHNG= 000022	PR7 = 000340
BIT00 = 000001	DT.APK= 000076	HRDPAS= 000050	MSGHRD= 000007	PS = 177776
BIT01 = 000002	DT.BLS= 000034	ICONT = 000036	MSGMAP= 000021	PSW = 177776
BIT02 = 000004	DT.CFO= 000014	ICOUNT= 000040	MSGNUL= 177775	RANNUM= 000054
BIT03 = 000010	DT.CF1= 000016	IDNUM = 000122	MSGPOP= 000002	RBUFEA= 000130
BIT04 = 000020	DT.ERR= 000020	IE = 000100	MSGPRM= 177776	RBUFPA= 000126
BIT05 = 000040	DT.ESI= 000044	INDPAR= 000040	MSGRES= 000001	RBUFsz= 000132
BIT06 = 000100	DT.EVN= 000000	INHDRP= 040000	MSGsFT= 000006	RBUFVA= 000124
BIT07 = 000200	DT.EXS= 000060	INHPR= 020000	MSGsKE= 000003	RDSERV= 000101
BIT08 = 000400	DT.FCH= 000037	INHREL= 001000	MSGsMB= 000015	RDWHMI= 000022
BIT09 = 001000	DT.FCN= 000036	INHRE= 000400	MSGsMH= 000014	RELERR= 000020
BIT1 = 000002	DT.HMX= 000104	INIT = 000030	MSGsMS= 000016	RELMOD= 020000
BIT10 = 002000	DT.KBE= 000024	INTR = 000120	MSGSTD= 000000	RELTIM= 010000
BIT11 = 004000	DT.KBP= 000026	IOMOD = 100000	MSGSYS= 000012	RESREG= ***** G
BIT12 = 010000	DT.KBR= 000022	IOMODP= 102000	MSGVEC= 000020	RES1 = 000056
BIT13 = 020000	DT.KBU= 000030	IOMODR= 112000	NBKMOD= 001000	RES2 = 000060
BIT14 = 040000	DT.MLS= 000032	IOMODX= 110300	NCPUOP= 000020	RICHAR= 031060
BIT15 = 100000	DT.MTI= 000110	JACK = 035060	NOAPTY= 000002	RPTDAT= 002000
BIT2 = 000004	DT.OFF= 000070	KIPAR0= 172340	NULL = 000000	RSTRT = 000112
BIT3 = 000010	DT.PAS= 000074	KIPAR1= 172342	OWEN = 024020	RUBOUT= 000177
BIT4 = 000020	DT.PC = 000002	KIPAR2= 172344	PAERR = 000010	RUNMOD= 100000
BIT5 = 000040	DT.PFL= 000062	KIPAR3= 172346	PARPRE= 002000	RSVALU= 001740
BIT6 = 000100	DT.PSW= 000004	KIPAR4= 172350	PARSTA= 000100	SAM = 075464
BIT7 = 000200	DT.PTA= 000064	KIPAR5= 172352	PASCNT= 000034	SAVREG= ***** G
BIT8 = 000400	DT.RCS= 000102	KIPAR6= 172354	PCDATA 000044RG	SBADR = 000102
BIT9 = 001000	DT.REL= 000040	KIPAR7= 172356	PC.DCF 000032R	SBKMOD= 000000
BKDEF = 000002	DT.SCT= 000066	KIPDR0= 172300	PC.REA 000002R	SBKSEL= 010000
BKMOD = 000020	DT.SMX= 000106	KIPDR1= 172302	PC.RHI 000006R	SC.ADR= 000006
BKMODE= 040000	DT.SP = 000006	KIPDR2= 172304	PC.RLO 00004R	SC.ALC= 000014
BKSLSH= 000134	DT.SSI= 000046	KIPDR3= 172306	PC.RPA 000000R	SC.APC= 000016
CAPRES= 000004	DT.STO= 000010	KIPDR4= 172310	PC.RSZ 000024R	SC.CKL= 000002
CASTAT= 000004	DT.ST1= 000012	KIPDR5= 172312	PC.RTP 000010R	SC.CKP= 000004
CDERCT= 000146	DT.SWR= 000056	KIPDR6= 172314	PC.TMP 000030R	SC.CLO= 000000
CDWDCT= 000144	DT.SYP= 000072	KIPDR7= 172316	PC.TPC 000026R	SC.HLD= 000010
CKTIM = 100000	DT.WBU= 000050	KTERRO= 000040	PC.WEA 000014R	SC.SCA= 000012
CLKPRE= 000001	DT.WHL= 000054	KTPRES= 000400	PC.WHI 000020R	SENDLS= 177777
COM 000060R	DT.WLL= 000052	KTSTAT= 000020	PC.WLO 000016R	SQFCNT= 000042
CONFIG= 000056	DVID1 = 000014	KTXTND= 040000	PC.WPA 000012R	SQFPAS= 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	\$SARGC= 000002
SR1 = 177574	UIPAR5= 177652	\$BGNLE= 177777	\$ISK0 = 000001	\$S\$BYTE= 000403
SR2 = 177576	UIPAR6= 177654	\$ERFLG= 000400	\$ISK1 = 000001	\$S\$CASE= 000000
SR3 = 172516	UIPAR7= 177656	\$F\$AND= 000310	\$ISK2 = 000001	\$S\$DST = 000000
STAT = 000026	UIPDR0= 177600	\$F\$BAD= 000401	\$ISK3 = 000001	\$S\$ELOC= 000402
STATBI= 064757	UIPDR1= 177602	\$F\$BLA= 000170	\$LOCTA= 177777	\$S\$ERFL= 000000
STAT1 = 000027	UIPDR2= 177604	\$F\$CAS= 000150	\$LSTIN= 000001	\$S\$FLAG= 000001
SUSPND= 000001	UIPDR3= 177606	\$F\$DEC= 000220	\$LSTTA= 000001	\$S\$FROM= 000000
SVR0 = 000062	UIPDR4= 177610	\$F\$DD = 000340	\$NESTL= 000000	\$S\$LOC = 001200R
SVR1 = 000064	UIPDR5= 177612	\$F\$FAL= 000405	\$NSK0 = 000300	\$S\$LCCN= 000000
SVR2 = 000066	UIPDR6= 177614	\$F\$GOD= 000400	\$NSK1 = 000300	\$S\$REG = 177777
SVR3 = 000070	UIPDR7= 177616	\$F\$IF = 000110	\$NSK2 = 000110	\$S\$RETU= 000000
SVR4 = 000072	UNIPA = ***** G	\$F\$INC= 000210	\$NSK3 = 000110	\$S\$RTN1= 050000
SVR5 = 000074	WASADR= 000104	\$F\$LOU= 000200	\$NSK4 = 000110	\$S\$RTN2= 050001
SVR6 = 000076	WBSTAT= 000040	\$F\$NAM= 000160	\$NSK5 = 000110	\$S\$SRC = 000000
SYSCNT= 000052	WBUFEA= 000136	\$F\$NO = 000403	\$SAVLE= 177777	\$S\$TGSV= 000000
YSERR= 000100	WBUFPA= 000134	\$F\$OR = 000320	\$SSK0 = 050012	\$S\$TGS1= 000000
TMPIO = 000002	WBUFRQ= 000140	\$F\$RTI= 000350	\$TAGLE= 177777	\$S\$TGS2= 000000
TOP12 001334R	WBUFSZ= 000142	\$F\$RTN= 000300	\$TAGNU= 050002	\$S\$TO = 000000
TOP16 001260R	WDFR = 000116	\$F\$SEL= 000140	\$TEMP = 000300	\$S\$TAG= 050000
TQOVF = 000002	WDT0 = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:35  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0184

3 COMMON EQUATE MODULE  
555 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
591 PCTRLC ROUTINE



.MAIN. MACY11 30A(1052) 20-SEP-78 17:40  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0264

3 COMMON EQUATE MODULE  
558 COMMON DEFINITIONS AND REFERENCES  
563 000000' .PRINT ;SPMAC: VERSION 1.1  
601 ACTIVATE BACKGROUND MODULE ROUTINE

```
503 .TITLE PCTRLC - PROCESS CTRL C
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     PCTRLC
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CONTROLS THE PROCESSING OF A CTRL C COMMAND FROM
517 ;     THE OPERATOR.
518 ;
519 ; INPUTS:
520 ;     DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ;     DT.ADDR, DT.STO, DT.CFO
524 ;
525 ; OUTPUTS:
526 ;     NONE
527 ;
528 ; IMPLICIT OUTPUTS:
529 ;     DT.STO
530 ;
531 ; PATOLOGICAL CONNECTIONS:
532 ;     NONE
533 ;
534 ; SUBORDINATE MODULES CALLED:
535 ;     MSGDHOOK                ;HOOK MESSAGE TO PROPER DRIVER
536 ;     PRRLOC                  ;PROCESS RELOCATION
537 ;     WBFLIM                  ;DETERMINE WRITE BUFFER LIMITS
538 ;     UNIMAP                  ;LOAD UNIBUS MAP
539 ;     RSTRCY                  ;RECOVER FROM RESET
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ;     NONE
543 ;
544 ; CALLING SEQUENCE:
545 ;     CALL PCTRLC IN <A>
546 ;                     A=DATA TABLE ADDRESS
547 ;
548 ; VERSION:
549 ;     0.0
550 ;
551 ;     EDIT                BY                DATE                REASON
552 ;
553 ;--
```

```
555 .SBTTL COMMON DEFINITIONS AND REFERENCES
556
557 .MCALL STRUCT
558 000000' STRUCT
559 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
559 000001 $LSTIN=1
560 000001 $LSTTAG=1
561
562 ;
563 ;*****
564 ;
565 ; REFERENCED BY OTHER MODULES
566 ;
567 .GLOBL PCTRLC ;MODULE ENTRY POINT
568 ;
569 ;*****
570 ;
571 ; GLOBAL REFERENCES
572 ;
573 .GLOBL MSGDHOOK ;HOOK MESSAGE TO PROPER DRIVER
574 .GLOBL PRRLOC ;PROCESS RELOCATION
575 .GLOBL WBFLIM ;DETERMINE WRITE BUFFER LIMITS
576 .GLOBL UNIMAP ;LOAD UNIBUS MAP
577 .GLOBL RSTRCY ;RECOVER FROM RESET
578 .GLOBL DX.RSTRT ;MONITOR RESTART ADDRESS
579 .GLOBL XCSR ;KEYBOARD PRINTER CSR
580 ;
581 ;*****
582 ;
583 ; LOCAL STORAGE
584 ;
585 000000' 041536 000045 PC.MSG: .ASCIZ /~C%/
586 .EVEN
587 ;
588 ;*****
589 ;
```

```

591          .SBTTL PCTRLC ROUTINE
592
593          ROUTINE PCTRLC <TABL>
594 000004'          PCTRLC:
(2) 000004'
595
596
597          ;+
598          ; SAVE REGISTERS
599          ; -
600
601 000004'          PUSH R0
(2) 000004' 010046          MOV      R0,-(SP)
602
603
604          ;+
605          ; SET R0 TO THE START OF THE DATA TABLE
606          ; -
607
608 000006'          LET R0 := TABL(R5)
(4) 000006' 016500 000000          MOV      TABL(R5),R0
609
610
611          ;+
612          ; RESET THE UNIBUS
613          ; -
614
615 000012'          INLINE <RESET>
(2) 000012' 000005          RESET
616
617
618          ;+
619          ; CLEAR THE PSW. (LOWER THE PRIORITY)
620          ; -
621
622 000014'          LET -(SP) := #0
(4) 000014' 005046          CLR      -(SP)
623 000016'          LET -(SP) := #1$
(4) 000016' 012746 000024'          MOV      #1$,-(SP)
624 000022'          INLINE <RTI>
(2) 000022' 000002          RTI
625
626 000024'          INLINE <1$:>
(2) 000024'          1$:
627
628
629          ;+
630          ; RESTORE THE PROCESSOR HARDWARE TO THE STATE IT WAS IN BEFORE THE RESET.
631          ; -
632
633 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
634

```

```
635
636          ;+
637          ; OUTPUT "C".
638          ; -
639
640 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
641 000064'          INLINE <2$: NOP>
(2) 000064' 000240          2$: NOP
642 000066'          INLINE <BR 2$>
(2) 000066' 000776          BR 2$
643
644 000070'          INLINE <3$:>
(2) 000070'          3$:
645
646
647          ;+
648          ; IF THE EXERCISER IS NOT IN LOWEST MEMORY MOVE IT THERE, THEN RESET THE WRITE
649          ; BUFFER LIMITS, AND RELOAD THE UNIBUS MAP IF IT EXISTS.
650          ; -
651
652 000070'          IF DT.ADDR(R0) NE #200 THEN
(6) 000070' 026027 000042 000200          CMP          DT.ADDR(R0),#200
(9) 000076' 00142E          BEQ          50002$
653 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
654 000116'          CALL WBFLIM IN <R0>
(3) 000116' 010546          MOV          R5,-(SP)
(4) 000120' 010045          MOV          R0,-(R5)
(3) 000122' 004767 000000G          JSR          PC,WBFLIM
(3) 000126' 012605          MOV          (SP)+,R5
655 000130'          IF #ADDR22 SETIN DT.CF0(R0) THEN
(6) 000130' 032760 001000 000014          BIT          #ADDR22,DT.CF0(R
(9) 000136' 001405          BEQ          50003$
656 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
657 000152'          ENDIF
(4) 000152'          50003$:
658 000152'          ENDIF
(4) 000152'          50002$:
659
660          ;+
661
```

```
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!RELTME  
(6) 000152' 042760 110000 000010 BIC #RUNMODE!RELTME  
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'  
(3) 000166' 50000S:  
(2) 000166' 000207 50001S:  
683 000001 .END RTS PC
```

ACSR = 000102	CTRLOC = 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.CFO= 000014	IDNUM = 000122	MSGNUL= 177775	RPTDAT= 002000
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPDP= 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	RJNMOD= 100000
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSKE= 000003	RVALU= 001740
BIT05 = 000040	DT.FCH= 000037	INHRR= 000400	MSGSMB= 000015	SAM = 075464
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMH= 000014	SADR = 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	SDFCNT= 000042
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PASCNT= 000034	SDFPAS= 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	SRC = 177572
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PDP70 = 010000	SR1 = 177574
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI0 = 000000	SR2 = 177576
BKMOD = 000020	DT.STO= 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	KTERR0= 000040	PRI6 = 000300	STAT1 = 000027
CASSTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PRI7 = 000340	SUSPND= 000001
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PRRLOC= ***** G	SVRO = 000062
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PRO = 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	\$\$LCC = 000136R
UIPAR1= 177642	WBUFPA= 000134	\$F\$G00= 000400	\$LSTTA= 000001	\$\$LCCN= 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\$L00= 000200	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR5= 177652	WDT0 = 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	\$\$TC = 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)



.MAIN. MACY11 3CA(1052) 20-SEP-78 17:36  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0192

3 COMMON EQUATE MODULE  
575 PFAIL (COMMON DEFINITIONS AND REFERENCES)  
578 '000000' .PRINT ;SPMAC: VERSION 1.1  
606 PFAIL (CODE)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:41  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0272

3 COMMON EQUATE MODULE  
563 COMMON DEFINITIONS AND REFERENCES  
566 000000' .PRINT ;SPMAC: VERSION 1.1  
501 BADMEM ROUTINE

```
508 .TITLE PFAIL (PROCESS POWER FAIL/POWER UP)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PFAIL
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE CONTAINS THE FOLLOWING TWO ROUTINES:
517 ;
518 ; 1. PWRDOWN ;HANDLES POWER DOWN SEQUENCE.
519 ; 2. PWRUP ;HANDLES POWER UP SEQUENCE.
520 ;
521 ;
522 ; ON POWER FAIL, IF THE SYSTEM CLOCK IS AVAILABLE IT IS TURNED
523 ; OFF, POWER-UP VECTOR IS SET UP, AND THE CPU HALTS.
524 ;
525 ; ON POWER UP, THE FOLLOWING IS DONE:
526 ;
527 ; 1. SET UP POWER-DOWN VECTOR.
528 ; 2. REINITIALIZE STACKS (R5 AND R6).
529 ; 3. INCREMENT POWER FAIL COUNT IN DTABLE.
530 ; 4. INITIALIZE APARS IF REQUIRED.
531 ; 5. TURN KT ON IF IT WAS ON BEFORE POWER FAIL.
532 ; 6. TURN SYSTEM CLOCK ON IF AVAILABLE.
533 ; 7. PRINT POWER FAIL MESSAGE.
534 ; 8. JUMP TO BOX00 (DX.RSTR).
535 ;
536 ; INPUTS:
537 ; NONE
538 ;
539 ; IMPLICIT INPUTS:
540 ; 1 .DT.STO ;STATUS WORD #0
541 ; 2. DT.CFO ;CONFIGURATION WORD #0
542 ; 3. DT.PFL ;POWER FAIL COUNT WORD
543 ;
544 ; OUTPUTS:
545 ; NONE
546 ;
547 ; IMPLICIT OUTPUTS:
548 ; 1. DT.STO
549 ; 2. DT.PFL
550 ;
551 ; PATHOLOGICAL CONNECTIONS:
552 ; 1. DTABLE ADDRESS
553 ;
554 ; SUBORDINATE ROUTINES CALLED:
555 ; 1. CLKON ;CLOCK ON ROUTINE
556 ; 2. CLKOFF ;CLOCK OFF ROUTINE
557 ; 3. MSGDHOOK ;OUTPUTS A MESSAGE
558 ; 4. KTSET ;STUFFS THE KERNEL & USER MODE APARS
559 ; 5. RSTRCY ;RESET RECOVERY ROUTINE
560 ;
561 ; FUNCTIONAL SIDE EFFECTS:
562 ; NONE
563 ;
```

PFAIL (PROCESS POWER FAIL/POWER UP)  
PFAIL.MAC 22-AUG-78 08:29

MACY11 30A(1052) 20-SEP-78 17:36 PAGE 19-1  
COMMON EQUATE MODULE

SEQ 0194

```
564 ; CALLING SEQUENCE:  
565 ; THE MODULES ARE ENTERED DIRECTLY AS A RESULT OF POWER FAIL/  
566 ; POWER UP.  
567 ;  
568 ; VERSION:  
569 ; 0.0  
570 ;  
571 ; EDIT DATE BY REASON  
572 ;--  
573
```

PFAIL (PROCESS POWER FAIL/POWER UP)  
PFAIL.MAC 22-AUG-78 08:29

MACY11 30A(1052) 20-SEP-78 17:36 PAGE 19-2  
PFAIL (COMMON DEFINITIONS AND REFERENCES)

SEQ 0195

```
575 .SBTTL PFAIL (COMMON DEFINITIONS AND REFERENCES)
576
577 .MCALL STRUCT
578 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
579
580 000001 $LSTIN=1
581 000001 $LSTTAG=1
582
583 ;*****
584 ; GLOBAL REFERENCES
585 ;
586 .GLOBL PWRFL ;POWER FAIL VECTOR - LOC. 24
587 .GLOBL PWRDOWN ;POWER FAIL ENTRY POINT
588 .GLOBL PWRUP ;POWER UP ENTRY POINT
589 .GLOBL RSTRCY ;RESET RECOVERY ROUTINE
590 .GLOBL KTSET ;STUFF APARS
591 .GLOBL CLKON ;TURN SYSTEM CLOCK ON
592 .GLOBL CLKOFF ;TURN SYSTEM CLOCK OFF
593 .GLOBL MSGDHOOK ;TYPE A MESSAGE ROUTINE
594 .GLOBL DX.RSTRT ;JUMP TO BOX00
595 .GLOBL DTABLE ;DATA TABLE
596
597 ;*****
598 ; LOCAL STORAGE
599 ;
600 000000' 000000G PF.EXT: #DX.RSTRT ;RESTART ROUTINE
601 000002' 050045 053517 051105 PF.MSG: .ASCIZ /%POWER FAIL OCCURRED%/
000010' 043040 044501 020114
000016' 041517 052503 051122
000024' 042105 000045
602
603 .EVEN
604
```

```
606 .SBTTL PFAIL (CODE)
607
608 ;+
609 ; POWER DOWN ROUTINE
610 ;-
611
612 000030' INLINE <PWRDOWN:>
(2) 000030' PWRDOWN:
613
614 ;+
615 ; SET R0 TO DTABLE ADDRESS. SET UP POWER-UP VECTOR,
616 ; AND TURN OFF SYSTEM CLOCK IF AVAILABLE. THEN HALT.
617 ;-
618
619 000030' LET SP := #SPVALUE
(4) 000030' 012706 002200 MOV #SPVALUE,SP
620 000034' LET R5 := #R5VALUE
(4) 000034' 012705 001740 MOV #R5VALUE,R5
621 000040' LET R0 := #DTABLE
(4) 000040' 012700 000000G MOV #DTABLE,R0
622 000044' LET PWRFL := #PWRUP
(4) 000044' 012767 000076' 000000G MOV #PWRUP,PWRFL
623 000052' IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 000052' 032760 000001 000014 BIT #CLKPRES,DT.CF0(
(9) 000060' 001405 BEQ 50000$
624 000062' CALL CLKOFF IN <R0>
(3) 000062' 010546 MOV R5,-(SP)
(4) 000064' 010045 MOV R0,-(R5)
(3) 000066' 004767 000000G JSR PC,CLKOFF
(3) 000072' 012605 MOV (SP)+,R5
625 000074' ENDIF
(4) 000074' 50000$:
626 000074' INLINE <HALT>
(2) 000074' 000000 HALT
627
```

```

629
630          ;+
631          ; POWER UP ROUTINE
632          ; -
633
634 000076'          INLINE <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
(4) 000076' 012767 000030' 000000G          MOV          #PWRDOWN,PWRFL
643 C00104'          INLINE <RESET>
(2) 000104' 000005          RESET
644 C00106'          LET SP := #SPVALUE
(4) 000106' 012706 002200          MOV          #SPVALUE,SP
645 000112'          LET R5 := #R5VALUE
(4) 000112' 012705 001740          MOV          #R5VALUE,R5
646 000116'          LET R0 := #DTABLE
(4) 000116' 012700 000000G          MOV          #DTABLE,R0
647 000122'          LET DT.PFL(R0) := DT.PFL(R0) + #1
(6) 000122' 005260 000062          INC          DT.PFL(R0)
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
(6) 000126' 032760 000400 000014          BIT          #KTPRES,DT.CF0(R
(9) 000134' 001412          BEQ          50001$
654 000136'          CALL KTSET IN <R0>
(3) 000136' 010546          MOV          R5,-(SP)
(4) 000140' 010045          MCV          R0,-(R5)
(3) 000142' 004767 000000G          JSR          PC,KTSET
(3) 000146' 012605          MOV          (SP)+,R5
655 000150'          CALL RSTRCY IN <R0>
(3) 000150' 010546          MOV          R5,-(SP)
(4) 000152' 010045          MOV          R0,-(R5)
(3) 000154' 004767 000000G          JSR          PC,RSTRCY
(3) 000160' 012605          MOV          (SP)+,R5
656 000162'          ENDIF
(4) 000162'          50001$:
657
658          ;+
659          ; IF SYSTEM CLOCK AVAILABLE, TURN IT ON
660          ; -
661
662 000162'          IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 000162' 032760 000001 000014          BIT          #CLKPRES,DT.CF0(
(9) 000170' 001405          BEQ          50002$
663 000172'          CALL CLKON IN <R0>
(3) 000172' 010546          MOV          R5,-(SP)

```

(4) 000174' 010045  
(3) 000176' 004767 000000G  
(3) 000202' 012605  
664 000204'  
(4) 000204'  
665  
666  
667  
668  
669  
670 000204'  
(4) 000204' 012702 000012  
671 000210'  
(4) 000210'  
(6) 000210' 005702  
(9) 000212' 001410  
672 000214'  
(4) 000214' 012701 177777  
673 000220'  
(4) 000220'  
(6) 000220' 005701  
(9) 000222' 001402  
674 000224'  
(6) 000224' 005301  
675 000226'  
(4) 000226' 000774  
(3) 000230'  
676 000230'  
(6) 000230' 005302  
677 000232'  
(4) 000232' 000766  
(3) 000234'  
678  
679  
680  
681  
682  
683 000234'  
(4) 000234' 005046  
684 000236'  
(4) 000236' 012746 000244'  
685 000242'  
(2) 000242' 000002  
686  
687  
688  
689  
690  
691 000244'  
(2) 000244'  
692 000244'  
(3) 000244' 010546  
(7) 000246' 012745 000276'  
(6) 000252' 012745 000002'  
(5) 000256' 012745 000002  
(4) 000262' 010045

ENDIF

;  
; DO A TIMING LOOP TO LET EVERYBODY SETTLE DOWN  
;-

LET R2 := #^D<10>

WHILE R2 NE #0 DO

LET R1 := #-1

WHILE R1 NE #0 DO

LET R1 := R1 - #1

ENDDO

LET R2 := R2 - #1

ENDDO

;  
; NOW LOWER PROIRITY TO 0  
;-

LET -(SP) := #0

LET -(SP) := #1\$

INLINE <RTI>

;  
; PRINT THE POWER FAIL MESSAGE AND JUMP TO DX.RSTRT.  
;-

INLINE <1\$:>

CALL MSGDHOOK IN <R0,#MSGPOP,#PF.MSG,#3\$>

MOV R0,-(R5)  
JSR PC,CLKON  
MOV (SP)+,R5  
  
50002\$:  
  
MOV #^D<10>,R2  
  
50003\$:  
TST R2  
BEQ 50004\$  
MOV #-1,R1  
  
50005\$:  
TST R1  
BEQ 50006\$  
DEC R1  
BR 50005\$  
  
50006\$:  
DEC R2  
BR 50003\$  
  
50004\$:  
  
1\$:  
MOV R5,-(SP)  
MOV #3\$,-(R5)  
MOV #PF.MSG,-(R5)  
MOV #MSGPOP,-(R5)  
MOV R0,-(R5)



PFAIL (PROCESS POWER FAIL/POWER UP)  
PFAIL.MAC 22-AUG-78 08:29

MACY11 30A(1052) 20-SEP-78 17:36 PAGE 19-6  
PFAIL (CODE)

SEQ 0199

(3) 000264' 004767 000000G  
(3) 000270' 012605  
693 000272'  
(2) 000272'  
694 000272'  
(2) 000272' 000240  
695 000274'  
(2) 000274' 000776  
696 000276'  
(2) 000276'  
697 000276'  
(2) 000276' 000177 177476  
698  
699 000001

INLINE <2\$:>  
INLINE <NOP>  
INLINE <BR 2\$>  
INLINE <3\$:>  
INLINE <JMP @PF.EXT>  
.END

JSR PC,MSGDHOOK  
MOV (SP)+,R5  
2\$:  
NOP  
BR 2\$  
3\$:  
JMP @PF.EXT

ACSR = 000102	CSRA = 000100	ENBEOP= 010000	MED = 076600	PS = 177776
ACTBIT= 004000	CSRC = 000102	ENBNUL= 000001	MEMPAS= 040000	PSW = 177776
ADDR22= 001000	CTRLC = 000003	ENDLST= 000000	MODEXH= 004000	PWRDOW 000030RG
ADR = 000006	CTRLO = 000017	EOPBIT= 000001	MODHOL= 002000	PWRFL = ***** G
APTFER= 000004	CTRLU = 000025	ERRTYP= 000106	MODSEL= 001000	PWRUP 000076RG
APTPRE= 000200	DCEVNT= 000011	EVNTBE= 000200	MSGCKD= 000010	RANNUM= 000054
ASB = 000106	DEFRTN= 000400	EVNTHD= 000200	MSGCKS= 000011	RBUFEA= 000130
ASSEMB= 000010	DIAGMC= 000000	EVNTKT= 000203	MSGDER= 000005	RBUFPA= 000126
ASTAT = 000104	DROPMO= 100000	EVNTPE= 000202	MSGDHO= ***** G	RBUFSZ= 000132
AUTO = 000010	DSEVNT= 000014	EVNTRE= 000201	MSGDRP= 000017	RBUFVA= 000124
AUTOST= 020000	DTABLE= ***** G	FATERR= 100000	MSGECH= 177777	RDSERV= 000101
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGEOP= 000013	RDWHMI= 000022
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGHDR= 000004	RELERR= 000020
BIT00 = 000001	DT.APK= 000076	ICONT = 000036	MSGHNG= 000022	RELMOD= 020000
BIT01 = 000002	DT.BLS= 000034	ICOUNT= 000040	MSGHRD= 000007	RELTIM= 010000
BIT02 = 000004	DT.CFO= 000014	IDNUM = 000122	MSGMAP= 000021	RES1 = 000056
BIT03 = 000010	DT.CF1= 000016	IE = 000100	MSGNUL= 177775	RES2 = 000060
BIT04 = 000020	DT.ERR= 000020	INDPAR= 000040	MSGPOP= 000002	RICHAR= 031060
BIT05 = 000040	DT.ESI= 000044	INHDRP= 040000	MSGPRM= 177776	RPTDAT= 002000
BIT06 = 000100	DT.EVN= 000000	INHEPR= 020000	MSGRES= 000001	RSTRCY= ***** G
BIT07 = 000200	DT.EXS= 000060	INHREL= 001000	MSGSFT= 000006	RSTRT = 000112
BIT08 = 000400	DT.FCH= 000037	INHRR= 000400	MSGSK= 000003	RUBOUT= 000177
BIT09 = 001000	DT.FCN= 000036	INIT = 000030	MSGSMB= 000015	RUNMCD= 100000
BIT1 = 000002	DT.HMX= 000104	INTR = 000120	MSGSMH= 000014	R5VALU= 001740
BIT10 = 002000	DT.KBE= 000024	IOMOD = 100000	MSGSMS= 000016	SAM = 075464
BIT11 = 004000	DT.KBP= 000026	IOMODP= 102000	MSGSTD= 000000	SBADR = 000102
BIT12 = 010000	DT.KBR= 000022	IOMODR= 112000	MSGSYS= 000012	SBKMOD= 000000
BIT13 = 020000	DT.KBU= 000030	IOMODX= 110000	MSGVEC= 000020	SBKSEL= 010000
BIT14 = 040000	DT.MLS= 000032	JACK = 035060	NBKMOD= 001000	SC.ADR= 000006
BIT15 = 100000	DT.MTI= 000110	KIPAR0= 172340	NCPUOP= 000020	SC.ALC= 000014
BIT2 = 000004	DT.OFF= 000070	KIPAR1= 172342	NDAPTY= 000002	SC.APC= 000016
BIT3 = 000010	DT.PAS= 000074	KIPAR2= 172344	NULL = 000000	SC.CKL= 000002
BIT4 = 000020	DT.PC = 000002	KIPAR3= 172346	OWEN = 024020	SC.CKP= 000004
BIT5 = 000040	DT.PFL= 000062	KIPAR4= 172350	PAERR = 000010	SC.CLO= 000000
BIT6 = 000100	DT.PSW= 000004	KIPAR5= 172352	PARPRE= 002000	SC.HLD= 000010
BIT7 = 000200	DT.PTA= 000064	KIPAR6= 172354	PARSTA= 000100	SC.SCA= 000012
BIT8 = 000400	DT.RCS= 000102	KIPAR7= 172356	PASCNT= 000034	SENDLS= 177777
BIT9 = 001000	DT.REL= 000040	KIPDR0= 172300	PDPLSI= 020000	SDFCNT= 000042
BKDEF = 000002	DT.SCT= 000066	KIPDR1= 172302	PDP60 = 004000	SDFPAS= 000046
BKMOD = 000020	DT.SMX= 000106	KIPDR2= 172304	PDP70 = 010000	SPACE = 000040
BKMODE= 040000	DT.SP = 000006	KIPDR3= 172306	PF.EXT 000000R	SPOINT= 000032
BKSLSH= 000134	DT.SSI= 000046	KIPDR4= 172310	PF.MSG 000002R	SPVALU= 002200
CAPRES= 000004	DT.ST0= 000010	KIPDR5= 172312	PRI0 = 000000	SRO = 177572
CASTAT= 000004	DT.ST1= 000012	KIPDR6= 172314	PRI1 = 000040	SR1 = 177574
CDERCT= 000146	DT.SWR= 000056	KIPDR7= 172316	PRI4 = 000200	SR2 = 177576
CDWDCT= 000144	DT.SYP= 000072	KTERRO= 000040	PRI5 = 000240	SR3 = 172516
CKTIM = 100000	DT.WBU= 000050	KTPRES= 000400	PRI6 = 000300	STAT = 000026
CLKOFF= ***** G	DT.WHL= 000054	KTSET = ***** G	PRI7 = 000340	STATBI= 064757
CLKON = ***** G	DT.WLL= 000052	KTSTAT= 000020	PRO = 000000	STAT1 = 000027
CLKPRE= 000001	DVID1 = 000014	KTXTND= 040000	PR4 = 000200	SUSPND= 000001
CONFIG= 000056	DX.RST= ***** G	LF = 000012	PR5 = 000240	SVR0 = 000062
CQCVF = 000001	ECCMEM= 000100	LPSTAT= 000001	PR6 = 000300	SVR1 = 000064
CR = 000015	ECCSTA= 000010	MAPSTA= 000200	PR7 = 000340	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
SVR5 = 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\$G00= 000400	\$NESTL= 177777	\$\$LDC = 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\$L00= 000200	\$\$AVLE= 177777	\$\$RETV= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$\$SK0 = 050004	\$\$RTN1= 000000
UIPAR3= 177646	WDT0 = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:36  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0202

3 COMMON EQUATE MODULE  
560 COMMON DEFINITIONS AND REFERENCES  
563 000000' .PRINT ;SPMAC: VERSION 1.1  
604 PRRLOC ROUTINE

```

594          .SBTTL BDACNV ROUTINE
595
596 000012'  ROUTINE BDACNV <NUM,ADR>
(2) 000012'
597
598          ;+
599          ; SAVE REGISTERS.
600          ; -
601
602 000012'  CALL SAVREG
(3) 000012' 004767 000000G
603
604
605          ;+
606          ; SET R0 TO THE START OF THE STORAGE AREA
607          ; OF THE RESULT.
608          ; -
609
610 000016'  LET R0 := ADR(R5)
(4) 000016' 016500 000002
611
612
613          ;+
614          ; SET R1 TO THE ADDRESS OF THE POWERS OF TEN TABLE.
615          ; -
616
617 000022'  LET R1 := #BD.TEN
(4) 000022' 012701 000000'
618
619
620          ;+
621          ; SET R4 TO COUNT 5 CHARACTERS.
622          ; -
623
624 000026'  LET R4 := R0 + #5
(4) 000026' 010004
(6) 000030' 062704 000005
625
626
627          ;+
628          ; PERFORM THE CONVERSION.
629          ; 1. GET A POWER OF TEN FROM THE TABLE.
630          ; 2. SUBTRACT THIS VALUE FROM THE NUMBER WE ARE
631          ; CONVERTING AS MANY TIMES AS POSSIBLE, KEEPING A COUNT
632          ; OF THE NO. OF TIMES WE SUBTRACTED.
633          ; 3. CONVERT THIS COUNT TO ASCII AND STORE IT IN THE
634          ; STORAGE TABLE.
635          ; 4. GET THE NEXT POWER OF TEN AND REPEAT.
636          ; -
637
638 000034'  WHILE R0 LO R4 DO
(4) 000034'
(6) 000034' 020004
(9) 000036' 103016
639
640 000040'  LET R3 := #0
    
```

BDACNV:

JSR PC,SAVREG

MOV ADR(R5),R0

MOV #BD.TEN,R1

MOV R0,R4  
 ADD #5,R4

50002\$:

CMP R0,R4  
 BHS 50003\$

```
508 .TITLE PRRLOC - PROCESS RELOCATION
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRRLOC
514 ;
515 ; FUNCTIONAL DESCRIPTION
516 ; CONTROLS THE RELOCATING OF THE RUNTIME EXERCISER FROM
517 ; ITS CURRENT LOCATION IN MEMORY TO A NEW LOCATION.
518 ;
519 ; INPUTS:
520 ; DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ; DT.ADDR, DT.ERR, DT.STO, DT.SWR
524 ;
525 ; OUTPUTS:
526 ; NONE
527 ;
528 ; IMPLICIT OUTPUTS:
529 ; DT.ADDR
530 ;
531 ; PATHOLOGICAL CONNECTIONS:
532 ; NONE
533 ;
534 ; SUBORDINATE ROUTINES CALLED:
535 ; NEWBA ;ESTABLISH NEW BASE ADDRESS
536 ; RELOC ;RELOCATE CODE
537 ; MSGDHOOK ;HOOK MSG TO PROPER OUTPUT DRIVER
538 ; BOAC ;BINARY TO OCTAL ASCII CONVERSION
539 ; SAVREG ;SAVE REGISTERS
540 ; RESREG ;RESTORE REGISTERS
541 ;
542 ; FUNCTIONAL SIDE EFFECTS:
543 ; QUEUEING MECHANISMS ARE INACTIVE DURING RELOCATION
544 ; PROCESSING.
545 ;
546 ; CALLING SEQUENCE:
547 ; CALL PRRLOC IN <A,B>
548 ; A=DATA TABLE ADDRESS
549 ; B=NEW BASE OFFSET (IF SET TO #0, NEWBA WILL CALCULATE
550 ; A NEW BASE ADDRESS.)
551 ;
552 ; VERSION:
553 ; 0.0
554 ;
555 ; EDIT BY DATE REASON
556 ;
557 ;--
558
```

```
560 .SBTTL COMMON DEFINITIONS AND REFERENCES
561
562 .MCALL STRUCT
563 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
564 000001 $LSTTAG=1
565 000001 $LSTIN=1
566
567
568 ;
569 ;*****
570 ;
571 ; REFERENCED BY OTHER MODULES
572 ;
573 .GLOBL PRRLOC ;MODULE ENTRY POINT
574 .GLOBL PR.RL1 ;1ST RELOCATION FLAG
575 ;
576 ;*****
577 ;
578 ; GLOBAL REFERENCES
579 ;
580 .GLOBL MSGDHOOK ;HOOK MSG TO PROPER OUTPUT DRIVER
581 .GLOBL NEWBA ;ROUTINE TO FIND NEW BASE OFFSET
582 .GLOBL RELOC ;ROUTINE TO MOVE CODE
583 .GLOBL BOAC ;BINARY TO OCTAL ASCII CONVERSION
584 .GLOBL SAVREG ;SAVE REGISTERS
585 .GLOBL RESREG ;RESTORE REGISTERS
586 ;
587 ;*****
588 ;
589 ; LOCAL STORAGE
590 ;
591 000000' 000000 PR.RL1: .WORD 0 ;FIRST RELOCATION FLAG - PATHOLOGICALLY
592 ; SET AT INITIALIZATION
593 000002' 000002 PR.TBL: .BLKW 2 ;TABLE FOR BOAC ROUTINE
594 000006' 051045 046105 041517 PR.MSG: .ASCII /%RELOCATED TO / ;RELOCATION MESSAGE
000014' 052101 042105 052040
000022' 020117
595 000024' 000010 PR.NBR: .BLKB 10 ;STORAGE FOR ASCII BASE ADDRESS
596 000034' 000045 .ASCIZ %%
597 000036' 051045 046105 041517 PR.MS1: .ASCIZ /%RELOCATION ABORTED%/ ;RELOCATION ABORTED MESSAGE
000044' 052101 047511 020116
000052' 041101 051117 042524
000060' 022504 000
598 000064'
599 .EVEN
600 ;
601 ;*****
602 ;
```

```
604 .SBTTL PRRLOC ROUTINE
605
606
607 000064' ROUTINE PRRLOC <TABL,BASE>
(2) 000064' PRRLOC:
608
609
610 ;+
611 ; SAVE REGISTERS
612 ; -
613
614 000064' CALL SAVREG
(3) 000064' 004767 000000G JSR PC, SAVREG
615
616
617 ;+
618 ; SET R0 TO START OF DATA TABLE
619 ; -
620
621 000070' LET R0 := TABL(R5)
(4) 000070' 016500 000000 MOV TABL(R5), R0
622
623
624 ;+
625 ; GET THE CURRENT BASE ADDRESS OFFSET
626 ; -
627
628 000074' LET R3 := DT.ADDR(R0)
(4) 000074' 016003 000042 MOV DT.ADDR(R0), R3
629
630
631 ;+
632 ; CALL NEWBA TO GENERATE A NEW BASE OFFSET. (THIS MUST BE DONE, EVEN IF A NEW
633 ; BASE WAS PASSED TO THIS MODULE, TO ASSURE THAT NEWBA IS PROPERLY
634 ; INITIALIZED.) AFTER CALLING NEWBA, IF A NEW BASE ADDRESS WAS PASSED
635 ; TO THE MODULE, USE IT INSTEAD OF THE ONE NEWBA CALCULATED.
636 ; -
637
638 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
639 000126' IF BASE(R5) NE #0 THEN
(6) 000126' 005765 000002 TST BASE(R5)
(9) 000132' 001402 BEQ 50002$
640 000134' LET R4 := BASE(R5)
(4) 000134' 016504 000002 MOV BASE(R5),R4
641 000140' ENDIF
(4) 000140' 50002$:
642
643
```



```
644          ;+
645          ; CALL THE ROUTINE WHICH WILL MOVE THE CODE
646          ; -
647
648 000140'    CALL RELOC IN <R0,R3,R4,PR.RL1>
(3) 000140' 010546
(7) 000142' 016745 177632
(6) 000146' 010445
(5) 000150' 010345
(4) 000152' 010045
(3) 000154' 004767 000000G
(3) 000160' 012605
649
650
651
652          ;+
653          ; IF THERE WAS A RELOCATION ERROR, THE EXERCISER WAS RETURNED
654          ; TO ITS STARTING POINT, SO THE NEW BASE EQUALS THE OLD.
655          ; -
656
657 000162'    IF #RELERR SETIN DT.ERR(R0) THEN
(6) 000162' 032760 000020 000020
(9) 000170' 001401
658          LET R4 := R3
(4) 000172' 010304
659          ENDF
(4) 000174'
660
661
662          ;+
663          ; IF THE RELOCATION ERROR BIT IS NOT SET
664          ; IN THE ERROR WORD OF THE DATA TABLE, WE HAD A SUCCESSFUL
665          ; RELOCATION, SO FORM AND OUTPUT A MESSAGE, UNLESS INHIBITED BY THE
666          ; SOFTWARE SWITCH REGISTER. OTHERWISE, OUTPUT AN ERROR MESSAGE.
667          ; -
668
669 000174'    IF #RELERR NOTSETIN DT.ERR(R0) THEN
(6) 000174' 032760 000020 000020
(9) 000202' 001061
670          IF #INHREL NOTSETIN DT.SWR(R0) THEN
(6) 000204' 032760 001000 000056
(9) 000212' 001054
671
672
673          ;+
674          ; CONVERT THE NEW BASE OFFSET TO A PHYSICAL ADDRESS AND PUT IT
675          ; IN THE PROPER FORMAT FOR BOAC TO USE.
676          ; -
677
678          LET R1 := R4 SHIFT #+6
(4) 000214' 010401
(7) 000216' 006301
(7) 000220' 006301
(7) 000222' 006301
(7) 000224' 006301
(7) 000226' 006301
MOV R5,-(SP)
MOV PR.RL1,-(R5)
MOV R4,-(R5)
MOV R3,-(R5)
MOV R0,-(R5)
JSR PC,RELOC
MOV (SP)+,R5
BIT #RELERR,DT.ERR(R
BEQ 50003$
MOV R3,R4
50003$:
BIT #RELERR,DT.ERR(R
BNE 50004$
BIT #INHREL,DT.SWR(R
BNE 50005$
MOV R4,R1
ASL R1
ASL R1
ASL R1
ASL R1
ASL R1
```

```
(7) 000230' 006301
679 000232'
(6) 000232' 010402
(6) 000234' 000302
680
681
682 ;+
683 ; IF THE UNIBUS MAP IS ENABLED, SET UP FOR 22-BIT ADDRESS CALCULATION,
684 ; OTHERWISE SET UP FOR 18-BIT CALCULATION.
685 ;-
686 000236' IF #MAPSTAT SETIN DT.STO(R0) THEN
(6) 000236' 032760 000200 000010 BIT #MAPSTAT,DT.STO(
(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$: NOP>
(2) 000340' 000240 1$: NOP
697 000342' INLINE< BR 1$>
(2) 000342' 000776 BR 1$
698
699 000344' INLINE <2$:>
(2) 000344' 2$:
700 000344' ENDIF
(4) 000344' 50005$:
701
702 ;+
```

```
703 ; OUTPUT AN ERROR MESSAGE.  
704 ;-  
705  
706 ELSE  
707  
708 CALL MSGDHOOK IN <R0,#MSGPOP,#PR.MS1,#10$>  
709  
710 INLN<9$: NOP>  
711 INLN <BR 9$>  
712 INLN <10$:>  
713  
714 ENDIF  
715  
716  
717 ;+  
718 ; CLEAR THE FIRST RELOCATION FLAG  
719 ;-  
720  
721 LET PR.RL1 := #0  
722  
723  
724 ;+  
725 ; RESTORE REGISTERS  
726 ;-  
727  
728 CALL RESREG  
729  
730  
731 ENDRTN  
732  
733 .END
```

50004\$: BR 50010\$  
50004\$: MOV R5,-(SP)  
50004\$: MOV #10\$,-(R5)  
50004\$: MOV #PR.MS1,-(R5)  
50004\$: MOV #MSGPOP,-(R5)  
50004\$: MOV R0,-(R5)  
50004\$: JSR PC,MSGDHOOK  
50004\$: MOV (SP)+,R5  
9\$: NOP  
9\$: BR 9\$  
10\$:  
50010\$:  
CLR PR.RL1  
50000\$:  
50001\$: RTS PC

ACSR = 000102	CSRA = 000100	ENDLST= 000000	MODHOL= 002000	PR6 = 000300
ACTBIT= 004000	CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	PR7 = 000340
ADDR22= 001000	CTRLC = 000003	ERRTYP= 000106	MSGCKD= 000010	PS = 177776
ADR = 000006	CTRLD = 000017	EVNTBE= 000200	MSGCKS= 000011	PSW = 177776
APTFER= 000004	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RANNUM= 000054
APTPRE= 000200	DCEVNT= 000011	EVNTKT= 000203	MSGDHO= ***** G	RBUFEA= 000130
ASB = 000106	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RBUFPA= 000126
ASSEMB= 000010	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RBUFSZ= 000132
ASTAT = 000104	DRGPMO= 100000	FATERR= 100000	MSGEOP= 000013	RBUFVA= 000124
AUTO = 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHDR= 000004	RDSERV= 000101
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHNG= 000022	RDWHMI= 000022
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGHRD= 000007	RELERR= 000020
BASE = 000002	DT.APK= 000076	ICOUNT= 000040	MSGMAP= 000021	RELMOD= 020000
BIT0 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGNUL= 177775	RELOC = ***** G
BIT00 = 000001	DT.CFO= 000014	IE = 000100	MSGPOP= 000002	RELTIM= 010000
BIT01 = 000002	DT.CF1= 000016	INDPAR= 000040	MSGPRM= 177776	RESREG= ***** G
BIT02 = 000004	DT.ERR= 000020	INHDRP= 040000	MSGRES= 000001	RES1 = 000056
BIT03 = 000010	DT.ESI= 000044	INHPRP= 020000	MSGSFT= 000006	RES2 = 000060
BIT04 = 000020	DT.EVN= 000000	INHREL= 001000	MSGSKE= 000003	RICHAR= 031060
BIT05 = 000040	DT.EXS= 000060	INHRE= 000400	MSGSMB= 000015	RPTDAT= 002000
BIT06 = 000100	DT.FCH= 000037	INIT = 000030	MSGSMH= 000014	RSTRT = 000112
BIT07 = 000200	DT.FCN= 000036	INTR = 000120	MSGSMS= 000016	RUBOUT= 000177
BIT08 = 000400	DT.HMX= 000104	IOMOD = 100000	MSGSTD= 000000	RUNMOD= 100000
BIT09 = 001000	DT.KBE= 000024	IOMODP= 102000	MSGSYS= 000012	RVALU= 001740
BIT1 = 000002	DT.KBP= 000026	IOMODR= 112000	MSGVEC= 000020	SAM = 075464
BIT10 = 002000	DT.KBR= 000022	IOMODX= 110000	NBKMOD= 001000	SAVREG= ***** G
BIT11 = 004000	DT.KBU= 000030	JACK = 035060	NCPUDP= 000020	SBADR = 000102
BIT12 = 010000	DT.MLS= 000032	KIPAR0= 172340	NEWBA = ***** G	SBKMOD= 000000
BIT13 = 020000	DT.MTI= 000110	KIPAR1= 172342	NOAPTY= 000002	SBKSEL= 010000
BIT14 = 040000	DT.OFF= 000070	KIPAR2= 172344	NULL = 000000	SC.ADR= 000006
BIT15 = 100000	DT.PAS= 000074	KIPAR3= 172346	GWEN = 024020	SC.ALC= 000014
BIT2 = 000004	DT.PC = 000002	KIPAR4= 172350	PAERR = 000010	SC.APC= 000016
BIT3 = 000010	DT.PFL= 000062	KIPAR5= 172352	PARPRE= 002000	SC.CKL= 000002
BIT4 = 000020	DT.PSW= 000004	KIPAR6= 172354	PARSTA= 000100	SC.CKP= 000004
BIT5 = 000040	DT.PTA= 000064	KIPAR7= 172356	PASCNT= 000034	SC.CLO= 000000
BIT6 = 000100	DT.RCS= 000102	KIPDR0= 172300	PDPLSI= 020000	SC.HLD= 000010
BIT7 = 000200	DT.REL= 000040	KIPDR1= 172302	PDP60 = 004000	SC.SCA= 000012
BIT8 = 000400	DT.SCT= 000066	KIPDR2= 172304	PDP70 = 010000	SENCLS= 177777
BIT9 = 001000	DT.SMX= 000106	KIPDR3= 172306	PRI0 = 000000	SOFCNT= 000042
BKDEF = 000002	DT.SP = 000006	KIPDR4= 172310	PRI1 = 000040	SOFPAS= 000046
BKMOD = 000020	DT.SSI= 000046	KIPDR5= 172312	PRI4 = 000200	SPACE = 000040
BKMODE= 040000	DT.STO= 000010	KIPDR6= 172314	PRI5 = 000240	SPOINT= 000032
BKSLSH= 000134	DT.ST1= 000012	KIPDR7= 172316	PRI6 = 000300	SPVALU= 002200
BOAC = ***** G	DT.SWR= 000056	KTERRD= 000040	PRI7 = 000340	SRO = 177572
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PRRLOC 000064RG	SR1 = 177574
CASSTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR.MSG 000006R	SR2 = 177576
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR.MS1 000036R	SR3 = 172516
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PR.NBR 000024R	STAT = 000026
CKTIM = 100000	DVIDI = 000014	LPSTAT= 000001	PR.NR1 000000RG	STATBI= 064757
CLKPRE= 000001	ECCMEM= 000100	MAPSTA= 000200	PR.TBL 000002R	STAT1 = 000027
CONFIG= 000056	ECCSTA= 000010	MED = 076600	PR0 = 000000	SUSPND= 000001
CQOVF = 000001	ENBEOP= 010000	MEMPAS= 040000	PR4 = 000200	SVRO = 000062
CR = 000015	ENBNUL= 000001	MODEXH= 004000	PR5 = 000240	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
SVR3 = 000076	UIPDR7= 177616	\$F\$DO = 000340	\$LOCTA= 177777	\$\$FLAG= 000001
SYSCNT= 000052	WASADR= 000104	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$FROM= 000000
YSERR= 000100	WBSTAT= 000040	\$F\$G00= 000400	\$LSTTA= 000001	\$\$LOC = 000244R
TABL = 000000	WBUFEA= 000136	\$F\$IF = 000110	\$NESTL= 177777	\$\$LOCN= 000000
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSK0 = 000300	\$\$REG = 177777
TQOVF = 000002	WBUFQ = 000140	\$F\$LOO= 000200	\$NSK1 = 000110	\$\$REU= 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	\$\$AVLE= 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	XDN = 000021	\$F\$TRU= 000404	\$TSK1 = 050005	\$\$STAG= 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  
CCRE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:37  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0211

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES  
565 000000' .PRINT ;SPMAC: VERSION 1.1  
605 RELOC ROUTINE  
780 ROUTINE RL.UP - MOVE CODE UP  
838 ROUTINE RL.DN - MOVE CODE DOWN  
904 ROUTINE RL.GOBACK - RETURN CODE TO ORIGINAL LOCATION

```
632          .SBTTL BADVEC ROUTINE
633
634          ;+
635          ; EXECUTE EMT INSTRUCTION TO INVOKE ROUTINE TO DETERMINE A BAD VECTOR
636          ; THE EMT WILL ALSO PUSH THE PSW ONTO THE STACK SO IT
637          ; CAN BE READ. BADVC1 IS FOR VECTORS LESS THAN 400,
638          ; BADVEC IS FOR VECTORS 400 AND UP.
639          ;-
640
641          IROUTINE BADVC1
642          (2) 000014' 104000
643          (3) 000016'
644          (3) 000016'
645          (2) 000016' 000002
646
647          IROUTINE BADVEC
648          (2) 000020' 104000
649          (3) 000022'
650          (3) 000022'
651          (2) 000022' 000002
```

BADVC1:  
EMT  
50000\$:  
50001\$: RTI

BADVEC:  
EMT  
50000\$:  
50001\$: RTI

```
508 .TITLE RELOC - RELOCATE CODE
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; RELOC
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE RELOCATES THE MOVABLE PORTION OF THE RUNTIME
517 ; EXERCISER FROM ITS CURRENT LOCATION IN MEMORY TO A NEW AREA.
518 ; IF A BAD MEMORY LOCATION IS ENCOUNTERED DURING THE MOVE,
519 ; THE EXERCISER IS RETURNED TO THE CURRENT LOCATION AND
520 ; AN ERROR INDICATOR IS SET.
521 ;
522 ; INPUTS:
523 ; DATA TABLE ADDRESS, OLD ADDRESS OFFSET, NEW ADDRESS OFFSET,
524 ; FIRST RELOCATION FLAG
525 ;
526 ; IMPLICIT INPUTS:
527 ; DT.ADDR,DT.ESIZ,DT.ERR
528 ;
529 ; OUTPUTS:
530 ; NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ; DT.ERR
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ; NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ; SAVREG
540 ; RESREG
541 ; BADMEM ;REPORT BAD MEMORY TRANSFER
542 ; GPA ;GET PHYSICAL ADDRESS
543 ;
544 ; FUNCTIONAL SIDE EFFECTS:
545 ; CURRENT COPY OF CODE MAY BE OVERLAYED
546 ;
547 ; CALLING SEQUENCE:
548 ; CALL RELOC IN <A,B,C,D>
549 ; A=ADDRESS OF DATA TABLE
550 ; B=OLD ADDRESS OFFSET
551 ; C=NEW ADDRESS OFFSET
552 ; D=FIRST RELOCATION FLAG
553 ;
554 ; VERSION:
555 ; 0.0
556 ;
557 ; EDIT BY DATE REASON
558 ;
559 ;--
```



```

561          .SBTTL COMMON DEFINITIONS AND REFERENCES
562
563
564          .MCALL STRUCT
565 00C000'   STRUCT
566 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
567          000001 $LSTIN=1
568          000001 $LSTTAG=1
569
570          ;
571          ;*****
572          ;
573          ; REFERENCED BY OTHER MODULES
574          ;
575          .GLOBL RELOC          ;MODULE ENTRY POINT
576          ;
577          ;*****
578          ;
579          ; GLOBAL REFERENCES
580          ;
581          .GLOBL GPA          ;ROUTINE TO GET A PHYSICAL ADDRESS
582          .GLOBL SAVREG
583          .GLOBL RESREG
584          .GLOBL BADMEM      ;REPORT BAD MEMORY TRANSFER
585          ;
586          ;*****
587          ;
588          ;
589          ; LOCAL STORAGE
590          ;
591 000000' 000000 RL.REG: .WORD 0          ;REGISTER VALUE NEEDED TO POINT
592          ; TO TOP OF EXERCISER
593 000002' 000000 RL.PAR: .WORD 0          ;PAR VALUE NEEDED TO POINT
594          ; TO TOP OF EXERCISER
595 000004' 000000 RL.ERRCHK: .WORD 0      ;ERROR-SERVICING FLAG. WHEN SET,
596          ; SERVICE MOVE ERRORS. WHEN NOT SET,
597          ; IGNORE ERRORS. SET AT BEGINNING
598          ; OF ROUTINE, CLEARED AFTER 1ST ERROR
599 000006' 000000 RL.RVAL: .WORD 0      ;TEMPORARY STORAGE FOR RL.DN SUBROUTINE
600 000010' 00C003 RL.TB1: .BLKW 3          ;STORAGE FOR GPA ROUTINE
601 000016' 000000 RL.DBD: .WORD 0          ;BAD DATA OF FAILING LOCATION
602 000020' 000000 RL.DGD: .WORD 0          ;GOOD DATA OF FAILING LOCATION
603

```

```

605          .SBTTL RELOC ROUTINE
606
607
608
609 000022'          ROUTINE RELOC <TABL,OLDPAR,NEWPAR,RL1>
610 (2) 000022'
611
612          ;+
613          ; SAVE REGISTERS
614          ; -
615
616 000022'          CALL SAVREG
617 (3) 000022' 004767 000000G          JSR          PC, SAVREG
618
619          ;+
620          ; SET R0 TO THE START OF THE DATA TABLE
621          ; -
622
623 000026'          LET R0 := TABL(R5)
624 (4) 000026' 016500 000000          MOV          TABL(R5),R0
625
626          ;+
627          ; RAISE THE PROCESSOR'S PRIORITY TO 7.
628          ; -
629
630 000032'          PUSH #PRI7
631 (2) 000032' 012746 000340          MOV          #PRI7,-(SP)
632 000036'          PUSH #10$
633 (2) 000036' 012746 000044'          MOV          #10$,-(SP)
634 000042'          INLINE <RTI>
635 (2) 000042' 000002          RTI
636
637 000044'          INLINE <10$:>
638 (2) 000044'          10$:
639          ;+
640          ; IS THIS THE FIRST RELOCATION?
641          ; -
642
643 000044'          IF RL1(R5) EQ #1 THEN
644 (6) 000044' 026527 000006 000001          CMP          RL1(R5),#1
645 (9) 000052' 001031          BNE          50002$
646
647          ;+
648          ; YES. FIND THE REGISTER AND PAR VALUES USED AS OFFSETS TO POINT TO
649          ; THE TOP OF THE EXERCISER. THE REGISTER VALUE (RL.REG) IS BITS 0-12 OF THE EXERCISER
650          ; SIZE (DT.ESIZ). THE PAR VALUE (RL.PAR) IS 200 MULTIPLIED BY ONE LESS
651          ; THAN THE VALUE IN BITS 13-15 OF DT.ESIZ.
652          ; -
653
654 000054'          LET R3 := DT.ESIZ(R0)
655 (4) 000054' 016003 000044          MOV          DT.ESIZ(R0),R3
656 000060'          LET RL.REG := R3 CLR.BY #160000
    
```

651	(4)	000060'	010367	177714			MOV	R3,RL.REG
652	(6)	000064'	042767	160000	177706		BIC	#160000,RL.REG
653		000072'				LET R3 := SWAP R3		
654	(6)	000072'	000303				SWAB	R3
655		000074'				LET R3 := R3 SHIFT #-5		
656	(7)	000074'	006203				ASR	R3
657	(7)	000076'	006203				ASR	R3
658	(7)	000100'	006203				ASR	R3
659	(7)	000102'	006203				ASR	R3
660	(7)	000104'	006203				ASR	R3
661		000106'				LET R3 := R3 CLR.BY #177770		
662	(6)	000106'	042703	177770			BIC	#177770,R3
663	655	000112'				LET RL.PAR := #0		
664	(4)	000112'	005067	177664			CLR	RL.PAR
665	656	000116'				LET R3 := R3 - #1		
666	(6)	000116'	005303				DEC	R3
667	657	000120'				WHILE R3 GT #0 DO		
668	(4)	000120'					50003\$:	
669	(6)	000120'	005703				TST	R3
670	(9)	000122'	003405				BLE	50004\$
671	658	000124'				LET RL.PAR := RL.PAR + #200		
672	(6)	000124'	062767	000200	177650		ADD	#200,RL.PAR
673	659	000132'				LET R3 := R3 - #1		
674	(6)	000132'	005303				DEC	R3
675	660	000134'				ENDDO		
676	(4)	000134'	000771				BR	50003\$
677	(3)	000136'					50004\$:	
678	661	000136'				ENDIF		
679	(4)	000136'					50002\$:	
680	662							
681	663							
682	664							
683	665							
684	666							
685	667							
686	668	000136'				LET RL.ERRCHK := #1		
687	(4)	000136'	012767	000001	177640		MOV	#1,RL.ERRCHK
688	669							
689	670							
690	671							
691	672							
692	673							
693	674							
694	675	000144'				IF NEWPAR(R5) HI OLDPAR(R5) THEN		
695	(6)	000144'	026565	000004	000002		CMP	NEWPAR(R5),OLDPA
696	(9)	000152'	101427				BLOS	50005\$
697	676							
698	677							
699	678							
700	679							
701	680							
702	681							
703	682							
704	683							
705	684							
706								
707								
708								
709								
710								
711								
712								
713								
714								
715								
716								
717								
718								
719								
720								
721								
722								
723								
724								
725								
726								
727								
728								
729								
730								
731								
732								
733								
734								
735								
736								
737								
738								
739								
740								
741								
742								
743								
744								
745								
746								
747								
748								
749								
750								
751								
752								
753								
754								
755								
756								
757								
758								
759								
760								
761								
762								
763								
764								
765								
766								
767								
768								
769								
770								
771								
772								
773								
774								
775								
776								
777								
778								
779								
780								
781								
782								
783								
784								
785								
786								
787								
788								
789								
790								
791								
792								
793								
794								
795								
796								
797								
798								
799								
800								
801								
802								
803								
804								
805								
806								
807								
808								
809								
810								
811								
812								
813								
814								
815								
816								
817								
818								
819								
820								
821								
822								
823								
824								
825								
826								
827								
828								
829								
830								
831								
832								
833								
834								
835								
836								
837								
838								
839								
840								
841								
842								
843								
844								
845								
846								
847								
848								
849								
850								
851								
852								

```

685          ; -
686
687          LET R1 := RL.REG + #20002
(4) 000154' 016701 177620          MOV      RL.REG,R1
(6) 000160' 062701 020002          ADD      #20002,R1
688
689          LET @#KIPAR1 := OLDPAR(R5) + RL.PAR
(4) 000164' 016537 000002 172342          MOV      OLDPAR(R5),@#KIP
(6) 000172' 066737 177604 172342          ADD      RL.PAR,@#KIPAR1
690
691
692          ; +
693          ; SET THE NEW ADDRESS POINTER TO THE TOP ADDRESS OF THE NEW AREA+2
694          ; -
695
696          LET R2 := RL.REG + #40002
(4) C00200' 016702 177574          MOV      RL.REG,R2
(6) 000204' 062702 040002          ADD      #40002,R2
697          LET @#KIPAR2 := NEWPAR(R5) + RL.PAR
(4) 000210' 016537 000004 172344          MOV      NEWPAR(R5),@#KIP
(6) C00216' 066737 177560 172344          ADD      RL.PAR,@#KIPAR2
698          CALL RL.UP
(3) 000224' 004767 000146          JSR      PC,RL.UP
699
700
701          ELSE
(4) 000230' 000414          BR      50005$
(3) 000232'
702
703
704          ; +
705          ; MOVING DOWN
706          ; -
707
708          ; +
709          ; SET THE OLD ADDRESS POINTER TO THE BOTTOM OF THE OLD AREA
710          ; -
711
712          LET R1 := #20000
(4) 000232' 012701 020000          MOV      #20000,R1
713          LET @#KIPAR1 := OLDPAR(R5)
(4) 000236' 016537 000002 172342          MOV      OLDPAR(R5),@#KIP
714
715
716          ; +
717          ; SET THE NEW ADDRESS POINTER TO THE BOTTOM OF THE NEW AREA
718          ; -
719
720          LET R2 := #40000
(4) 000244' 012702 040000          MOV      #40000,R2
721          LET @#KIPAR2 := NEWPAR(R5)
(4) 000250' 016537 000004 172344          MOV      NEWPAR(R5),@#KIP
722          CALL RL.DN
(3) 000256' 004767 000224          JSR      PC,RL.DN
723          ENDF
(4) 000262'          BR      50006$

```

```

724
725
726
727
728      ;+
729      ; NOW RELOAD THE MEMORY MANAGEMENT PAR'S SO THE EXERCISER
730      ; CALL RUN OUT OF THE NEW AREA. PAR0 IS NOT CHANGED SINCE IT POINTS
731      ; TO THE LOWEST 4K OF THE EXERCISER, WHICH ALWAYS STAYS IN LOWEST MEMORY.
732      ;-
733      000262'      LET R1 := #KIPAR1
(4) 000262' 012701 172342      MOV      #KIPAR1,R1
734      000266'      LET R2 := NEWPAR(R5)
(4) 000266' 016502 000004      MOV      NEWPAR(R5),R2
735      000272'      LET R3 := #6
(4) 000272' 012703 000006      MOV      #6,R3
736      000276'      REPEAT
(3) 000276'      50007$:
737      000276'      LET R3 := R3 - #1
(6) 000276' 005303      DEC      R3
738      000300'      LET (R1)+ := R2
(4) 000300' 010221      MOV      R2,(R1)+
739      000302'      LET R2 := R2 + #200
(6) 000302' 062702 000200      ADD      #200,R2
740      000306'      UNTIL R3 EQ #0
(3) 000306' 005703      TST      R3
(6) 000310' 001372      BNE      50007$
741
742
743      ;+
744      ; UPDATE THE CURRENT BASE ADDRESS.
745      ;-
746
747      000312'      LET DT.ADDR(R0) := NEWPAR(R5)
(4) 000312' 016560 000004 000042      MOV      NEWPAR(R5),DT.AD
748
749
750      ;+
751      ; LOWER THE PROCESSOR'S PRIORITY.
752      ;-
753
754      000320'      PUSH #PRIO
(2) 000320' 012746 000000      MOV      #PRIO,-(SP)
755      000324'      PUSH #20$
(2) 000324' 012746 000332'      MOV      #20$,-(SP)
756      000330'      INLINE <RTI>
(2) 000330' 000002      RTI
757
758      000332'      INLINE <20$:>
(2) 000332'      20$:
759
760
761      ;+
762      ; IF THE RELOCATION WAS UNSUCCESSFUL BECAUSE OF A BAD MEMORY
763      ; LOCATION, OUTPUT A MESSAGE INDICATING THAT LOCATION.
764      ;-
765

```

766	000332'		IF #RELERR SETIN DT.ERR(R0) THEN		
(6)	000332'	032750	000020	000020	BIT #RELERR,DT.ERR(R
(9)	000340'	001413			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					

```

780          .SBTTL  ROUTINE RL.UP - MOVE CODE UP
781
782          ;+
783          ; THIS SUBROUTINE MOVES CODE TO AN AREA HIGHER IN MEMORY THAN THE CURRENT
784          ; AREA.  THE MOVE BEGINS WITH THE HIGHEST ADDRESS TO BE RELOCATED AND WORKS
785          ; DOWNWARD TO THE LOWEST ADDRESS.  EACH TIME THE PAR'S ARE ADJUSTED TO THE
786          ; NEXT LOWEST 4K AREA, THE REGISTERS ARE SET TO THE TOP OF THE 4K AREA.
787          ; -
788
789          000376'  ROUTINE RL.UP
(2)          000376'
790
791
792          000376'  WHILE @#KIPAR1 HIS OLDPAR(R5) DO
(4)          000376'
(6)          000376' 023765 172342 000002
(9)          000404' 103437
793          000406'  WHILE R1 GT #20000 DO
(4)          000406'
(6)          000406' 020127 020000
(9)          000412' 003421
794
795
796          ;+
797          ; DECREMENT THE POINTERS AND MOVE A WORD
798          ; -
799
800          000414'  LET -(R2) := -(R1)
(4)          000414' 014142
801
802
803
804          ;+
805          ; COMPARE THE CONTENTS OF THE OLD AND NEW ADDRESSES.  IF THEY MATCH, KEEP
806          ; GOING.  IF THEY DON'T MATCH, THEN SEE IF THE ERROR-SERVICING FLAG IS SET.  IF
807          ; IT IS, CALL THE ERROR ROUTINE AND THEN MOVE THE CODE BACK TO
808          ; WHERE IT WAS AND RETURN TO THE CALLER.  IF THE
809          ; ERROR-SERVICING FLAG IS NOT SET, THEN WE HAVE ALREADY DETECTED AN ERROR
810          ; AND DO NOT WISH TO CHECK FOR ANY MORE, SO JUST CONTINUE MOVING CODE.
811          ; -
812          000416'  IF (R1) NE (R2) THEN
(6)          000416' 021112
(9)          000420' 001415
813          000422'  IF RL.ERRCHK EQ #1 THEN
(6)          000422' 026727 177356 000001
(9)          000430' 001011
814          000432'  CALL RL.GOBACK
(3)          000432' 004767 000210
815          000436'  LET R1 := R1 + #2
(6)          000436' 062701 000002
816          000442'  LET R2 := R2 + #2
(6)          000442' 062702 000002
817          000446'  CALL RL.DN
(3)          000446' 004767 000034
818          000452'  RETURN
(4)          000452' 000414
  
```

```

319 000454'                               ENDIF
(4) 000454'                               50007$:
820 000454'                               ENDIF
(4) 000454'                               50006$:
821 000454'                               ENDDO
(4) 000454' 000754                       BR      50004$
(3) 000456'                               50005$:
822
823
824                                     ;+
825                                     ; POSITION THE POINTERS TO THE TOP OF THE NEXT LOWER 4K PIECE.
826                                     ;-
827
828 000456'                               LET R1 := #40000
(4) 000456' 012701 040000                 MOV     #40000,R1
829 000462'                               LET R2 := #60000
(4) 000462' 012702 060000                 MOV     #60000,R2
830 000466'                               LET @#KIPAR1 := @#KIPAR1 - #200
(6) 000466' 162737 000200 172342        SUB     #200,@#KIPAR1
831 000474'                               LET @#KIPAR2 := @#KIPAR2 - #200
(6) 000474' 162737 000200 172344        SUB     #200,@#KIPAR2
832 000502'                               ENDDO
(4) 000502' 000735                       BR      50002$
(3) 000504'                               50003$:
833 000504'                               ENDRTN
(3) 000504'                               50000$:
(3) 000504'                               50001$:
(2) 000504' 000207                       RTS     PC
834
835
836

```



```

838 .SBTTL ROUTINE RL.DN - MOVE CODE DOWN
839
840 ;+
841 ; THIS SUBROUTINE MOVES CODE TO AN AREA LOWER IN MEMORY THAN THE CURRENT
842 ; AREA. THE MOVE BEGINS WITH THE LOWEST ADDRESS TO BE RELOCATED AND WORKS
843 ; UPWARD TO THE HIGHEST ADDRESS. EACH TIME THE PAR'S ARE ADJUSTED TO POINT TO
844 ; THE NEXT HIGHER 4K AREA OF MEMORY, THE REGISTERS ARE SET TO POINT TO
845 ; THE BOTTOM OF THE 4K AREA. AND ARE INCREMENTED UNTIL THEY POINT TO THE
846 ; TOP ADDRESS OF THAT 4K AREA. WHEN THE HIGHEST 4K SEGMENT OF THE EXERCISER IS
847 ; REACHED, HOWEVER, THE REGISTERS MUST BE INCREMENTED ONLY TO THE HIGHEST EXERCISER
848 ; ADDRESS AND NOT TO THE TOP OF THE 4K AREA.
849 ; -
850
851 ROUTINE RL.DN
852 000506'
(2) 000506'
853
854
855 000506' ROUTINE RL.DN
(4) 000506' 016503 000002
(6) 000512' 066703 177264
856 000516' WHILE @#KIPAR1 LOS R3 DO
(4) 000516'
(6) 000516' 023703 172342
(9) 000522' 101050
857 000524' IF @#KIPAR1 LD R3 THEN
(6) 000524' 023703 172342
(9) 000530' 103004
858 000532' LET RL.RVAL := #37776
(4) 000532' 012767 037776 177246
859 000540' ELSE
(4) 000540' 000406
(3) 000542'
860 000542' LET RL.RVAL := RL.REG + #20000
(4) 000542' 016767 177232 177236
(6) 000550' 062767 020000 177230
861 000556' ENDIF
(4) 000556'
862 000556' WHILE R1 LOS RL.RVAL DO
(4) 000556'
(6) 000556' 020167 177224
(9) 000562' 101015
863
864
865 ;+
866 ; MOVE A WORD
867 ; -
868
869 000564' LET (R2) := (R1)
(4) 000564' 011112
870
871
872 ;+
873 ; COMPARE THE CONTENTS OF THE OLD AND NEW ADDRESSES AND THEN INCREMENT
874 ; THE POINTERS. IF THE CONTENTS MATCH, KEEP GOING. IF THEY DON'T MATCH,
875 ; SEE IF THE ERROR-SERVICING FLAG IS SET. IF IT IS,

```

RL.DN:

```

MOV OLDPAR(R5),R3
ADD RL.PAR,R3
50002$:
CMP @#KIPAR1,R3
BHI 50003$
CMP @#KIPAR1,R3
BHIS 50004$
MOV #37776,RL.RVAL
BR 50005$
50004$:
MOV RL.REG,RL.RVAL
ADD #20000,RL.RVAL
50005$:
50006$:
CMP R1,RL.RVAL
BHI 50007$

```

MOV (R1),(R2)

```

876 ; CALL THE ERROR ROUTINE, THEN MOVE THE CODE BACK TO WHERE IT WAS
877 ; AND RETURN TO THE CALLER. IF THE ERROR-SERVICING FLAG IS NOT
878 ; SET, THEN WE HAVE ALREADY ENCOUNTERED AN ERROR AND DO NOT WISH TO
879 ; CHECK FOR ANY MORE, SO JUST CONTINUE MOVING CODE.
880 ;-
881
882 000566' IF (R1)+ NE (R2)+ THEN
(6) 000566' 022122 CMP (R1)+,(R2)+
(9) 000570' 001411 BEQ 50010$
883 000572' IF RL.ERRCHK EQ #1 THEN
(6) 000572' 026727 177206 000001 CMP RL.ERRCHK,#1
(9) 000600' 001005 BNE 50011$
884 000602' CALL RL.GOBACK
(3) 000602' 004767 000040 JSR PC,RL.GOBACK
885 000606' CALL RL.UP
(3) 000606' 004767 177564 JSR PC,RL.UP
886 000612' RETURN
(4) 000612' 000414 BR 50000$
887 000614' ENDIF
(4) 000614' 50011$:
888 000614' ENDIF
(4) 000614' 50010$:
889 000614' ENDDO
(4) 000614' 000760 BR 50006$
(3) 000616' 50007$:
890
891
892 ;+
893 ; POSITION THE POINTERS TO THE BOTTOM OF THE NEXT 4K SEGMENT.
894 ;-
895
896 000616' LET R1 := #20000
(4) 000616' 012701 020000 MOV #20000,R1
897 000622' LET R2 := #40000
(4) 000622' 012702 040000 MOV #40000,R2
898 000626' LET @#KIPAR1 := @#KIPAR1 + #200
(6) 000626' 062737 000200 172342 ADD #200,@#KIPAR1
899 000634' LET @#KIPAR2 := @#KIPAR2 + #200
(6) 000634' 062737 000200 172344 ADD #200,@#KIPAR2
900 000642' ENDDO
(4) 000642' 000725 BR 50002$
(3) 000644' 50003$:
901 000644' ENDRTN
(3) 000644' 50000$:
(3) 000644' 50001$:
(2) 000644' 000207 RTS PC
902

```

```

904          .SBTTL  ROUTINE RL.GOBACK - RETURN CODE TO ORIGINAL LOCATION
905
906
907          ;+
908          ; THIS SUBROUTINE IS CALLED UPON THE DETECTION OF A RELOCATION ERROR.
909          ; IT PERFORMS THE FOLLOWING FUNCTIONS:
910          ; (1) SAVES THE ADDRESS OF THE FAILING LOCATION, ALONG WITH ITS
911          ;     ACTUAL AND SHOULD-BE CONTENTS.
912          ; (2) CLEARS THE ERROR-SERVICING FLAG.
913          ; (3) SETS THE RELOCATION ERROR BIT OF THE ERROR WORD IN THE
914          ;     DATA TABLE.
915          ; (4) REVERSES THE POINTERS.  I.E., MOVES THE CONTENTS OF THE OLD ADDRESS
916          ;     POINTER INTO THE NEW ADDRESS POINTER, AND VICE VERSA, SO WE
917          ;     CAN MOVE THE CODE BACK.
918          ; (5) REVERSES THE VALUES IN OLDPAR AND NEWPAR.  THIS ALSO IS
919          ;     NECESSARY FOR MOVING THE CODE BACK.
920          ;-
921
922
923          000646'          ROUTINE RL.GOBACK
924          (2) 000646'
925
926          000646'          LET RL.TB1 := R2
927          (4) 000646' 010267 177136          MOV          R2,RL.TB1
928          000652'          LET RL.DBD := (R2)
929          (4) 000652' 011267 177140          MOV          (R2),RL.DBD
930          000656'          LET RL.DGD := (R1)
931          (4) 000656' 011167 177136          MOV          (R1),RL.DGD
932          000662'          CALL GPA IN <R0,#RL.TB1>
933          (3) 000662' 010546          MOV          R5,-(SP)
934          (5) 000664' 012745 000010'          MOV          #RL.TB1,-(R5)
935          (4) 000670' 010045          MOV          R0,-(R5)
936          (3) 000672' 004767 000000G          JSR          PC,GPA
937          (3) 000676' 012605          MOV          (SP)+,R5
938
939          000700'          LET RL.ERRCHK := #0
940          (4) 000700' 005067 177100          CLR          RL.ERRCHK
941          000704'          LET DT.ERR(R0) := DT.ERR(R0) SET.BY #RELERR
942          (6) 000704' 052760 000020 000020          BIS          #RELERR,DT.ERR(R
943
944          000712'          LET R4 := @KIPAR1
945          (4) 000712' 013704 172342          MOV          @KIPAR1,R4
946          000716'          LET @KIPAR1 := @KIPAR2
947          (4) 000716' 013737 172344 172342          MOV          @KIPAR2,@KIPAR
948          000724'          LET @KIPAR2 := R4
949          (4) 000724' 010437 172344          MOV          R4,@KIPAR2
950
951          000730'          LET R4 := NEWPAR(R5)
952          (4) 000730' 016504 000004          MOV          NEWPAR(R5),R4
953          000734'          LET NEWPAR(R5) := OLDPAR(R5)
954          (4) 000734' 016565 000002 000004          MOV          OLDPAR(R5),NEWPA
955          000742'          LET OLDPAR(R5) := R4
956          (4) 000742' 010465 000002          MOV          R4,OLDPAR(R5)
957          000746'          ENDRTN
  
```

RELOC - RELOCATE CODE MACY11 30A(1052) 20-SEP-78 17:37 PAGE 19-12  
RELOC.MAC 28-JUL-78 09:09 ROUTINE RL.GOBACK - RETURN CODE TO ORIGINAL LOCATION

SEQ 0224

(3) 000746'  
(3) 000746'  
(2) 000746' 000207  
943 000001

.END

50000\$:  
50001\$:

RTS PC

ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFPA= 000126
ACTBIT= 004000	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RBUFZS= 000132
ADDR22= 001000	CTRLD = 000017	EVNTBE= 000200	MSGCKD= 000010	RBUFVA= 000124
ADR = 000006	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RDSERV= 000101
APTFER= 000004	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RDWHMI= 000022
APTPRE= 000200	DEFRTN= 000400	EVNTP= 000202	MSGDRP= 000017	RELERR= 000020
ASB = 000106	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RELMOD= 020000
ASSEMB= 000010	DROPMO= 100000	FATERR= 100000	MSGEOP= 000013	RELCC = 000022RG
ASTAT = 000104	DSEVNT= 000014	GPA = ***** G	MSGHDR= 000004	RELTIM= 010000
AUTO = 000010	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RESREG= ***** G
AUTOST= 020000	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RES1 = 000056
AWAS = 000110	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RES2 = 000060
BADMEM= ***** G	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RICHAR= 031060
BIT0 = 000001	DT.CFO= 000014	IDNUM = 000122	MSGPOP= 000002	RL.DBD 000016R
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RL.DGD 000020R
BIT01 = 000002	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	RL.DN 000050R
BIT02 = 000004	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	RL.ERR 000004R
BIT03 = 000010	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	RL.GDB 000046R
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	RL.PAR 000002R
BIT05 = 000040	DT.FCH= 000037	INHRE= 000400	MSGSMH= 000014	RL.REG 000001R
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	RL.RVA 000006R
BIT07 = 000200	DT.HMX= 000104	LNTR = 000120	MSGSTD= 000000	RL.TB1 000010R
BIT08 = 000400	DT.KBE= 000024	ICMOD = 100000	MSGSYS= 000012	RL.UP 000376R
BIT09 = 001000	DT.KBP= 000026	ICMODP= 102000	MSGVEC= 000020	RL1 = 000006
BIT1 = 000002	DT.KBR= 000022	ICMODR= 112000	NBKM0D= 001000	RPTDAT= 002000
BIT10 = 002000	DT.KBU= 000030	ICMODX= 110000	NCPUOP= 000020	RSTRT = 000112
BIT11 = 004000	DT.MLS= 000032	JACK = 035060	NEWPAR= 000004	RUBOUT= 000177
BIT12 = 010000	DT.MTI= 000110	KIPAR0= 172340	NOAPTY= 000002	RUNMOD= 100000
BIT13 = 020000	DT.OFF= 000070	KIPAR1= 172342	NULL = 000000	RVALU= 001740
BIT14 = 040000	DT.PAS= 000074	KIPAR2= 172344	OLDPAR= 000002	SAM = 075464
BIT15 = 100000	DT.PC = 000002	KIPAR3= 172346	OWEN = 024020	SAVREG= ***** G
BIT2 = 000004	DT.PFL= 000062	KIPAR4= 172350	PAERR = 000010	SBADR = 000102
BIT3 = 000010	DT.PSW= 000004	KIPAR5= 172352	PARPRE= 002000	SBKMOD= 000000
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PARSTA= 000100	SBKSEL= 010000
BIT5 = 000040	DT.RCS= 000102	KIPAR7= 172356	PASCNT= 000034	SC.ADR= 000006
BIT6 = 000100	DT.REL= 000040	KIPDR0= 172300	PDPLSI= 020000	SC.ALC= 000014
BIT7 = 000200	DT.SCT= 000066	KIPDR1= 172302	PDP60 = 004000	SC.APC= 000016
BIT8 = 000400	DT.SMX= 000106	KIPDR2= 172304	PDP70 = 010000	SC.CKL= 000002
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PRI0 = 000000	SC.CKP= 000004
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI1 = 000040	SC.CLO= 000000
BKMOD = 000020	DT.STO= 000010	KIPDR5= 172312	PRI4 = 000200	SC.HLD= 000010
BKMODE= 040000	DT.ST1= 000012	KIPDR6= 172314	PRI5 = 000240	SC.SCA= 000012
BKSLSH= 000134	DT.SWR= 000056	KIPDR7= 172316	PRI6 = 000300	SENDLS= 177777
CAPRES= 000004	DT.SYP= 000072	KTERR0= 000040	PRI7 = 000340	SQFCNT= 000042
CASSTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PRO = 000000	SQFPAS= 000046
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR4 = 000200	SPACE = 000040
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR5 = 000240	SPOINT= 000032
CKTIM = 100000	DVID1 = 000014	LF = 000012	PR6 = 000300	SPVALU= 002200
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PR7 = 000340	SRO = 177572
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	PS = 177776	SR1 = 177574
CQCVF = 000001	ENBEOP= 010000	MED = 076600	PSW = 177776	SR2 = 177576
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RANNUM= 000054	SR3 = 172516
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFEA= 000130	STAT = 000026

STATBI= 064757	UIPDRO= 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	\$LQCTA= 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	\$NSKO = 000300	\$\$FROM= 000000
SVR4 = 000072	UIPDR7= 177616	\$F\$GOD= 000400	\$NSK1 = 000120	\$\$LDC = 000600R
SVR5 = 000074	WASADR= 000104	\$F\$IF = 000110	\$NSK2 = 000120	\$\$LDCN= 000000
SVR6 = 000076	WBSTAT= 000040	\$F\$INC= 000210	\$NSK3 = 000110	\$\$REG = 177777
SYSCNT= 000052	WBUFEA= 000136	\$F\$LQD= 000200	\$NSK4 = 000110	\$\$RETU= 000000
SYSERR= 000100	WBUFPA= 000134	\$F\$NAM= 000160	\$\$SAVLE= 177777	\$\$RTN1= 050000
TABL = 000000	WBUFRQ= 000140	\$F\$NO = 000403	\$\$SSKO = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:38  
EQUATE.MAC .13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0227

3 COMMON EQUATE MODULE  
560 COMMON DEFINITIONS & REFERENCES  
563 000000' .PRINT ;SPMAC: VERSION 1.1  
597 RPIRQ (CODE)

```
508 .TITLE RPIRQ (PIRQ SERVICE ROUTINE)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; RPIRQ
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE SERVICES THE PIRQ TRAP CALL (WHICH IS AN IOT).
517 ; IT ENQUEUES HEADER ADDR. AND RETURN PC OF THE MODULE
518 ; INTO THE CONTROL QUEUE, UPDATES THE SYSTEM STACK AND RETURNS TO
519 ; THE INTERRUPTED ROUTINE. TO ENSURE THAT THE ENQUEUEING ROUTINE
520 ; IS NOT INTERRUPTED, THIS MODULE IS HANDLED AT PRIORITY 7.
521 ; IF A BACKGROUND MODULE IS THE INTERRUPTED MODULE, IT WILL
522 ; BE TAKEN OFF THE AIR.
523 ;
524 ; INPUTS:
525 ; NONE
526 ;
527 ; IMPLICIT INPUTS:
528 ; NONE
529 ;
530 ; OUTPUTS:
531 ; NONE
532 ;
533 ; IMPLICIT OUTPUTS:
534 ; NONE
535 ;
536 ; PATHOLOGICAL CONNECTIONS:
537 ; DATA TABLE ADDRESS
538 ; BACKGROUND HEADER STORAGE
539 ;
540 ; SUBORDINATE ROUTINES CALLED:
541 ; ENQCQ ;PUT ITEMS ON THE CONTROL QUEUE.
542 ; ERREC ;ERROR RECOVERY ROUTINE
543 ; RSTRCY ;RESTORE STATUS ROUTINE
544 ;
545 ; FUNCTIONAL SIDE EFFECTS:
546 ; NONE
547 ;
548 ; CALLING SEQUENCE:
549 ; THIS ROUTINE IS ENTERED BY EXECUTING A 'PIRQ' INSTRUCTION.
550 ;
551 ; VERSION:
552 ; 0.0
553 ;
554 ; EDIT DATE BY REASON
555 ;-----
556 ;
557 ;--
558
```



```
560 .SBTTL COMMON DEFINITIONS & REFERENCES
561
562 .MCALL STRUCT
563 000000' STRUCT
564 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
565 000001 $LSTIN=1
566 000001 $LSTTAG=1
567 ;*****
568 ;
569 ; REFERENCED BY OTHER MODULES:
570 ;
571 .GLOBL RPIRQ ;MODULE ENTRY POINT
572 ;*****
573 ; GLOBAL REFERENCES
574 ;
575 .GLOBL ENQCQ ;ROUTINE TO ENQUE ITEMS ON CONTROL QUEUE
576 .GLOBL ERREC ;ERROR RECOVERY ROUTINE
577 .GLOBL RSTRCY ;RESTORE STATUS ROUTINE
578
579 .GLOBL DTABLE ;SYSTEM DATA TABLE
580 .GLOBL DX.R5 ;SYSTEM LINK REGISTER
581 .GLOBL DX.SP ;SYSTEM STACK
582 .GLOBL DX.DEQ ;RET ADDR IN BOX00
583
584 .GLOBL BA.HDR ;BKMOD HEADER ADDRESS
585 .GLOBL BA.RET ;BKMOD RETURN ADDRESS
586 .GLOBL BA.STAT ;BKMOD STATUS WORD
587
588 ; LOCAL STORAGE
589 ;
590 000000' 000000G RP.EXT: DX.DEQ ;GOTO ADDRESS IF INTERRUPTED MODULE IS A BK
591 000005 .REPT ^D<5> ;R5 STORAGE
592 .WORD 0
593 .ENDR
594 000014' RP.R5S: ;R5 STACK ALLOCATION FOR THIS MODULE
595
```

```

597      .SBTTL RPIRQ (CODE)
598
599      ;*****
600      ; PIRQ SERVICE ROUTINE STARTS HERE
601      ;
602
603      000014'      INLINE <RPIRQ:>
604      (2) 000014'
605
606      ;+
607      ; SAVE R1 AND LOAD IT WITH MODULE'S BEGIN ADDRESS
608      ; -
609      000014'      PUSH R1
610      (2) 000014' 010146      MOV      R1,-(SP)
611      000016'      LET R1 := 2(SP)
612      (4) 000016' 016601 000002      MOV      2(SP),R1
613
614      ;+
615      ; SAVE R5 AND ALLOCATE LOCAL R5 STACK
616      ; -
617      000022'      PUSH R5
618      (2) 000022' 010546      MOV      R5,-(SP)
619      000024'      LET R5 := #RP.R5S
620      (4) 000024' 012705 000014'      MOV      #RP.R5S,R5
621
622      ;+
623      ; ENQUEUE THE MODULE INTO THE CONTROL QUEUE
624      ; -
625      000030'      CALL ENQCQ IN <#DTABLE,(R1),2(R1)>
626      (3) 000030' 010546      MOV      R5,-(SP)
627      (6) 000032' 016145 000002      MOV      2(R1),-(R5)
628      (5) 000036' 011145      MOV      (R1),-(R5)
629      (4) 000040' 012745 000000G      MOV      #DTABLE,-(R5)
630      (3) 000044' 004767 000000G      JSR      PC,ENQCQ
631      (3) 000050' 012605      MOV      (SP)+,R5
632
633      ;+
634      ; IF NO ERROR RESTORE R5 STACK
635      ; -
636      000052'      IF.NO.ERROR THEN
637      (6) 000052' 103500      ECS      50000$
638      000054'      POP R5
639      (2) 000054' 012605      MOV      (SP)+,R5
640
641      ;+
642      ; NOW GET ADDRESS OF DTABLE AND DETERMINE IF BACKGROUND
643      ; MODULE WAS INTERRUPTED.
644      ; -
645      000056'      LET R1 := #DTABLE
646      (4) 000056' 012701 000000G      MOV      #DTABLE,R1
647      000062'      IF #BKMODE SETIN DT.ST0(R1) THEN

```

```

(6) 000062' 032761 040000 000010
(9) 000070' 001464
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
347 000072' IF #BIT07!BIT06!BIT05 NOTSETIN 10(SP) THEN
(6) 000072' 032766 000340 000010 BIT #BIT07!BIT06!BIT
(9) 000100' 001060 BEQ 50001$
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' 01004G 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 ;+
668 ; NOW TAKE MODULE OFF THE AIR BY CLEARING ACTBIT.
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.STO.
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.STO(R1) := DT.STO(R1) CLR.BY #BKMODE
(6) 000212' 042761 040000 000010          BIC      #BKMODE,DT.STO(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 ;+
730 ; WAS NOT BACKGROUND MODULE THEN RESTORE R1 AND ADJUST THE STACK
731 ; AND RETURN TO INTERRUPTED MODULE.
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$:
738
739 ;+
740 ; ERROR, CONTROL QUEUE OVERFLOW DO A RESET AND SET UP R5 AND R6 STACKS.
741 ; -
742
743 000254'                                INLINE <RESET>
(2) 000254' 000005                                RESET
744 000256'                                LET R5 := DX.R5
(4) 000256' 016705 000000G                        MOV      DX.R5,R5
745 000262'                                LET SP := DX.SP
(4) 000262' 016706 000000G                        MOV      DX.SP,SP
746 000268'                                CALL RSTRCY IN <#DTABLE>
(3) 000266' 010546                                MOV      R5,-(SP)
(4) 000270' 012745 000000G                        MOV      #DTABLE,-(R5)
(3) 000274' 004767 000000G                        JSR      PC,RSTRCY
(3) 000300' 012605                                MOV      (SP)+,R5
747
748 ;+
749 ; NOW CALL ERROR RECOVERY ROUTINE
750 ; -
751
752 000302'                                CALL ERREC IN <#DTABLE>
(3) 000302' 010546                                MOV      R5,-(SP)
(4) 000304' 012745 000000G                        MOV      #DTABLE,-(R5)
(3) 000310' 004767 000000G                        JSR      PC,ERREC
(3) 000314' 012605                                MOV      (SP)+,R5
753 000316'                                ENDIF
(4) 000316'                                50003$:
754 000001                                .END

```

ACSR = 000102	CR = 000015	DX.SP = ***** G	KTXTND= 040000	PR7 = 000340
ACTBIT= 004000	CSRA = 000100	ECCMEM= 000100	LF = 000012	PS = 177776
ADDR22= 001000	CSRC = 000102	ECCSTA= 000010	LPSTAT= 000001	PSW = 177776
ADR = 000006	CTRLC = 000003	ENBEOP= 010000	MAPSTA= 000200	RANNUM= 000054
APTFER= 000004	CTRLO = 000017	ENBNUL= 000001	MED = 076600	RBUFEA= 000130
APTPRE= 000200	CTRLU = 000025	ENDLST= 000000	MEMPAS= 040000	RBUFPA= 000126
ASB = 000106	DCEVNT= 000011	ENQCQ = ***** G	MODEXH= 004000	RBUFSZ= 000132
ASSEMB= 000010	DEFRTN= 000400	EOPBIT= 000001	MODHOL= 002000	REUFVA= 000124
ASTAT = 000104	DIAGMC= 000000	ERREC = ***** G	MODSEL= 001000	RDSERV= 000101
AUTO = 000010	DROPMO= 100000	ERRTYP= 000106	MSGCKD= 000010	RDWHMI= 000022
AUTOJST= 020000	DSEVNT= 000014	EVNTBE= 000200	MSGCKS= 000011	RELERR= 000020
AWAS = 000110	DTABLE= ***** G	EVNTHD= 000200	MSGDER= 000005	RELMOD= 020000
BA.HDR= ***** G	DT.ADD= 000042	EVNTKT= 000203	MSGDRP= 000017	RELTIM= 010000
BA.RET= ***** G	DT.AP = 000100	EVNTPE= 000202	MSGECH= 177777	RES1 = 000056
BA.STA= ***** G	DT.APK= 000076	EVNTRE= 000201	MSGEDP= 000013	RES2 = 000060
BIT0 = 000001	DT.BLS= 000034	FATERR= 100000	MSGHDR= 000004	RICHAR= 031060
BIT00 = 000001	DT.CFO= 000014	HRDCNT= 000044	MSGHNG= 000022	RPIRQ 000014RG
BIT01 = 000002	DT.CF1= 000016	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
BIT02 = 000004	DT.ERR= 000020	ICONT = 000036	MSGMAP= 000021	RP.EXT 000000R
BIT03 = 000010	DT.ESI= 000044	ICOUNT= 000040	MSGNUL= 177775	RP.R5S 000014R
BIT04 = 000020	DT.EVN= 000000	IDNUM = 000122	MSGPDP= 000002	RSTRCY= ***** G
BIT05 = 000040	DT.EXS= 000060	IE = 000100	MSGPRM= 177776	RSTRT = 000112
BIT06 = 000100	DT.FCH= 000037	INDPAR= 000040	MSGRES= 000001	RUBOUT= 000177
BIT07 = 000200	DT.FCN= 000036	INHDRP= 040000	MSGSFT= 000006	RUNMOD= 100000
BIT08 = 000400	DT.HMX= 000104	INHEPR= 020000	MSGSKE= 000003	RVALU= 001740
BIT09 = 001000	DT.KBE= 000024	INHREL= 001000	MSGSMB= 000015	SAM = 075464
BIT1 = 000002	DT.KBP= 000026	INHRR= 000400	MSGSMH= 000014	SBADR = 000102
BIT10 = 002000	DT.KBR= 000022	INIT = 000030	MSGSMS= 000016	SBKMOD= 000000
BIT11 = 004000	DT.KBU= 000030	INTR = 000120	MSGSTD= 000000	SBKSEL= 010000
BIT12 = 010000	DT.MLS= 000032	IOMOD = 100000	MSGSYS= 000012	SC.ADR= 000006
BIT13 = 020000	DT.MTI= 000110	IOMODP= 102000	MSGVEC= 000020	SC.ALC= 000014
BIT14 = 040000	DT.OFF= 000070	IOMODR= 112000	NBKMOD= 001000	SC.APC= 000016
BIT15 = 100000	DT.PAS= 000074	IOMODX= 110000	NCPUOP= 000020	SC.CKL= 000002
BIT2 = 000004	DT.PC = 000002	JACK = 035060	NDAPTY= 000002	SC.CKP= 000004
BIT3 = 000010	DT.PFL= 000062	KIPAR0= 172340	NULL = 000000	SC.CLO= 000000
BIT4 = 000020	DT.PSW= 000004	KIPAR1= 172342	CWEN = 024020	SC.HLD= 000010
BIT5 = 000040	DT.PTA= 000064	KIPAR2= 172344	PAERR = 000010	SC.SCA= 000012
BIT6 = 000100	DT.RCS= 000102	KIPAR3= 172346	PARPRE= 002000	SENDLS= 177777
BIT7 = 000200	DT.REL= 000040	KIPAR4= 172350	PARSTA= 000100	SGFCNT= 000042
BIT8 = 000400	DT.SCT= 000066	KIPAR5= 172352	PASCNT= 000034	SOPPAS= 000046
BIT9 = 001000	DT.SMX= 000106	KIPAR6= 172354	PDPLSI= 020000	SPACE = 000040
BKDEF = 000002	DT.SP = 000006	KIPAR7= 172356	PDP60 = 004000	SPOINT= 000032
BKMOD = 000020	DT.SSI= 000046	KIPDR0= 172300	PDP70 = 010000	SPVALU= 002200
BKMODE= 040000	DT.STO= 000010	KIPDR1= 172302	PRI0 = 000000	SRO = 177572
BKSLSH= 000134	DT.ST1= 000012	KIPDR2= 172304	PRI1 = 000040	SR1 = 177574
CAPRES= 000004	DT.SWR= 000056	KIPDR3= 172306	PRI4 = 000200	SR2 = 177576
CASTAT= 000004	DT.SYP= 000072	KIPDR4= 172310	PRI5 = 000240	SR3 = 172516
CDERCT= 000146	DT.WBU= 000050	KIPDR5= 172312	PRI6 = 000300	STAT = 000026
CDWDCT= 000144	DT.WHL= 000054	KIPDR6= 172314	PRI7 = 000340	STATBI= 064757
CKTIM = 100000	DT.WLL= 000052	KIPDR7= 172316	PRO = 000000	STAT1 = 000027
CLKPRE= 000001	DVID1 = 000014	KTERRO= 000040	PR4 = 000200	SUSPND= 000001
CONFIG= 000056	DX.DEQ= ***** G	KTPRES= 000400	PR5 = 000240	SVRO = 000062
CQOVF = 000001	DX.R5 = ***** G	KTSTAT= 000020	PR6 = 000300	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	\$\$LGC = 000100R
SYSCNT= 000052	WBSTAT= 000040	\$F\$GOO= 000400	\$LSTTA= 000001	\$\$LGCN= 000000
SYSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$NESTL= 177777	\$\$REG = 177777
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSK0 = 000110	\$\$RETU= 000000
TQOVF = 000002	WBUFRQ= 000140	\$F\$LQO= 000200	\$NSK1 = 000110	\$\$RTN1= 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$RTN2= 000000
UIPAR1= 177642	WDFR = 000116	\$F\$NO = 000403	\$\$AVLE= 177777	\$\$SRC = 000000
UIPAR2= 177644	WDT0 = 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	\$\$TD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:38  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0236

3 COMMON EQUATE MODULE  
553 COMMON DEFINITIONS AND REFERENCES  
556 000000' .PRINT ;SPMAC: VERSION 1.1  
582 RSTRCY ROUTINE



```
508 .TITLE RSTRCY - RESET RECOVERY
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; RSTRCY
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; RESTORES PROCESSOR HARDWARE OPTIONS TO THE STATES THEY WERE
517 ; IN BEFORE THE "RESET" WAS ISSUED.
518 ;
519 ; INPUTS:
520 ; DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ; DT.STO
524 ; DT.CFO
525 ; DT.PTA
526 ;
527 ; OUTPUTS:
528 ; NONE
529 ;
530 ; IMPLICIT OUTPUTS:
531 ; NONE
532 ;
533 ; PATHOLOGICAL CONNECTIONS:
534 ; DT.KBRSP
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ; KPON - PARITY ON ROUTINE
538 ;
539 ; FUNCTIONAL SIDE EFFECTS:
540 ; NONE
541 ;
542 ; CALLING SEQUENCE:
543 ; CALL RSTRCY IN <A>
544 ; A=DATA TABLE ADDRESS
545 ;
546 ; VERSION:
547 ; 0.0
548 ;
549 ; EDIT BY DATE REASON
550 ;
551 ;--
```

```
553          .SBTTL COMMON DEFINITIONS AND REFERENCES
554
555          .MCALL STRUCT
556 000000'   STRUCT
557 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
558          000001 $LSTIN=1
559          000001 $LSTTAG=1
560          ;
561          ;*****
562          ;
563          ; REFERENCED BY OTHER MODULES
564          ;
565          .GLOBL RSTRCY          ;MODULE ENTRY POINT
566          ;
567          ;*****
568          ;
569          ; GLOBAL REFERENCES
570          ;
571          .GLOBL KONTRL          ;BUFFER TO CACHE REGISTER
572          .GLOBL CCNTRL          ;CACHE REGISTER
573          .GLOBL KPON           ;PARITY ON ROUTINE
574          ;
575          ;*****
576          ;
577          ; LOCAL STORAGE
578          ;
579 000000' 000002' RST.PTR: .WORD RST.BUF          ;FAKE BUFFER POINTER
580 000002' 000015 RST.BUF: .WORD <CR>          ;FAKE BUFFER
```

```

582          .SBTTL RSTRCY ROUTINE
583
584          ROUTINE RSTRCY <TABL>
585 000004'
(2) 000004'
586
587
588          ;+
589          ; SAVE REGISTERS
590          ; -
591
592 000004'          PUSH R0,R1
(2) 000004' 010046          MOV          R0,-(SP)
(3) 000006' 010146          MOV          R1,-(SP)
593
594
595          ;+
596          ; SET R0 TO THE START OF THE DATA TABLE
597          ; -
598
599 000010'          LET R0 := TABL(R5)
(4) 000010' 016500 000000          MOV          TABL(R5),R0
600
601
602          ;+
603          ; IF MEMORY MANAGEMENT WAS ENABLED, RE-ENABLE IT, INCLUDING
604          ; 22-BIT ADDRESSING IF IT EXISTS.
605          ; -
606
607 000014'          IF #KTSTAT SETIN DT.STO(R0) THEN
(6) 000014' 032760 000020 000010          BIT          #KTSTAT,DT.STO(R
(9) 000022' 001412          BEQ          50002$
608 000024'          LET @#SR0 := @#SR0 SET.BY #001001
(6) 000024' 052737 001001 177572          BIS          #001001,@#SR0
609 000032'          IF #ADDR22 SETIN DT.CFO(R0) THEN
(6) 000032' 032760 001000 000014          BIT          #ADDR22,DT.CFO(R
(9) 000040' 001403          BEQ          50003$
610 000042'          LET @#SR3 := @#SR3 SET.BY #BIT04
(6) 000042' 052737 000020 172516          BIS          #BIT04,@#SR3
611 000050'          ENDIF
(4) 000050'          50003$:
612 000050'          ENDIF
(4) 000050'          50002$:
613
614
615          ;+
616          ; IF THE UNIBUS MAP HAD BEEN ENABLED, RE-ENABLE IT.
617          ; -
618
619
620 000050'          IF #MAPSTAT SETIN DT.STO(R0) THEN
(6) 000050' 032760 000200 000010          BIT          #MAPSTAT,DT.STO(
(9) 000056' 001403          BEQ          50004$
621 000060'          LET @#SR3 := @#SR3 SET.BY #BIT05
(6) 000060' 052737 000040 172516          BIS          #BIT05,@#SR3
622 000066'          ENDIF

```

```

(4) 000066' 50004$:
623
624
625 ;+
626 ; IF PARITY OR ECC HAD BEEN ENABLED, RE-LOAD THE PARITY REGISTERS.
627 ; BY CALLING KPON ROUTINE AND PASSING THE ADDRESS
628 ; OF FAKE BUFFER. THEN CLEAR OUT KEYBOARD RESPONSE LOCATION IN
629 ; DTABLE TO ELIMINATE OUTPUTTING MESSAGE.
630 ;-
631
632 C00066' IF #PARSTAT SETIN DT.STO(R0) OR #ECCSTAT SETIN DT.STO(R0) THEN
(6) 000066' 032760 000100 000010 BIT #PARSTAT,DT.STO(
(8) 000074' 001004 BNE 50005$
(6) 000076' 032760 000010 000010 BIT #ECCSTAT,DT.STO(
(9) 000104' 001411 BEQ 50006$
(5) C00106' 50005$:
633 000106' CALL KPON IN <R0,#RST.PTR>
(3) C00106' 010546 MOV R5,-(SP)
(5) 000110' 012745 000000' MOV #RST.PTR,-(R5)
(4) 000114' 010045 MOV R0,-(R5)
(3) 000116' 004767 000000G JSR PC,KPON
(3) 000122' 012605 MOV (SP)+,R5
634 000124' LET DT.KBRSP(R0) := #0 CLR DT.KBRSP(R0)
(4) 000124' 005060 000022 ENDIF
635 000130' 50006$:
(4) 000130'
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.CFO(R0) THEN
(6) 000130' 032760 000004 000014 BIT #CAPRES,DT.CFO(R
(9) 000136' 001424 BEQ 50007$
645 000140' IF #CASTAT NOTSETIN DT.STO(R0) THEN
(6) 000140' 032760 000004 000010 BIT #CASTAT,DT.STO(R
(9) 000146' 001006 BNE 50010$
646 000150' LET KONTRL := KONTRL SET.BY #14 BIS #14,KONTRL
(6) 000150' 052767 000014 000000G MOV KONTRL,@CCNTRL
647 000156' LET @CCNTRL := KONTRL
(4) 000156' 016777 000000G 000000G
648 000164' ENDIF
(4) 000164' 50010$:
649 000164' IF #PARSTAT NOTSETIN DT.STO(R0) THEN
(6) 000164' 032760 000100 000010 BIT #PARSTAT,DT.STO(
(9) 000172' 001006 BNE 50011$
650 000174' LET KONTRL := KONTRL SET.BY #203 BIS #203,KONTRL
(6) 000174' 052767 000203 000000G
651 000202' LET @CCNTRL := KONTRL MOV KONTRL,@CCNTRL
(4) 000202' 016777 000000G 000000G
652 000210' ENDIF
(4) 000210' 50011$:
653 000210' ENDIF

```

```
(4) 000210'
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'
(3) 000214'
(2) 000214' 000207
663 000001 .END
```

50007\$:

50000\$:  
50001\$:

RTS PC

ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODEXH= 004000	RBUFSZ= 000132
ACTBIT= 004000	CTRLC = 000003	ERRTYP= 000106	MODHOL= 002000	RBUFVA= 000124
ADDR22= 001000	CTRLQ = 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	DROPMD= 100000	FATERR= 100000	MSGECH= 177777	RES1 = 000056
ASTAT = 000104	DSEVNT= 000014	HRDCNT= 000044	MSGGEP= 000013	RES2 = 000060
AUTO = 000010	DT.ADD= 000042	HRDPAS= 000050	MSGHDR= 000004	RICHAR= 031060
AUTDST= 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.CFO= 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	RUBCNT= 000177
BIT03 = 000010	DT.ESI= 000044	INHEPR= 020000	MSGRES= 000001	RUNMOD= 100000
BIT04 = 000020	DT.EVN= 000000	INHREL= 001000	MSGSFT= 000006	RVALU= 001740
BIT05 = 000040	DT.EXS= 000060	INHRR= 000400	MSGSK= 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	NOPTY= 000002	SC.CLO= 000000
BIT14 = 040000	DT.OFF= 000070	KIPAR2= 172344	NULL = 000000	SC.HLD= 000010
BIT15 = 100000	DT.PAS= 000074	KIPAR3= 172346	QWEN = 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	SPINT= 000032
BIT7 = 000200	DT.REL= 000040	KIPDR1= 172302	PDP60 = 004000	SPVALU= 002200
BIT8 = 000400	DT.SCT= 000066	KIPDR2= 172304	PDP70 = 010000	SRO = 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	STATI = 000027
CASSTAT= 000004	DT.SYP= 000072	KPON = ***** G	PRO = 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
CQDVF = 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

.MAIN. MACY11 30A(1052) 20-SEP-78 17:39  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0244

3 COMMON EQUATE MODULE  
566 COMMON DEFINITIONS AND REFERENCES  
570 000000' .PRINT ;SPMAC: VERSION 1.1  
630 ACTIVATE ROUTINE  
763 SET UP FOR THE NEXT PASS THRU MODULE LIST

```
508 .TITLE ACTIV - ACTIVATE NEXT OPTION MODULE
509 .IDENT /V0.0/
510
511
512 ;++
513 ; MODULE NAME:
514 ;     ACTIV
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ;     THIS ROUTINE FETCHES FROM THE MODULE QUEUE LIST SUPPLIED BY THE
518 ;     CONFIGURATOR LINKER PROGRAM THE ADDRESS OF THE NEXT OPTION MODULE
519 ;     TO BE ACTIVATED, AND ACTIVATES IT BY MAKING AN ENTRY FOR IT IN
520 ;     THE CONTROL QUEUE
521 ;
522 ; INPUTS:
523 ;     1. ADDRESS OF DATA TABLE
524 ;
525 ; IMPLICIT INPUTS:
526 ;     1. DT.STO
527 ;     2. DT.ST1
528 ;
529 ; OUTPUTS:
530 ;     NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ;     1. DT.STO
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ;     NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ;     1. ENQCQ             MAKE ENTRY IN CONTROL QUEUE
540 ;     2. SAVREG
541 ;     3. RESREG
542 ;
543 ; FUNCTIONAL SIDE EFFECTS:
544 ;     1. THIS ROUTINE IS RESPONSIBLE FOR SETTING THE FOLLOWING BITS IN THE
545 ;     SYSTEM'S STATUS INDICATOR WORD:
546 ;         MODHOLD         SET TO INDICATE THAT A SBK OR NBK MODULE
547 ;                         IS PRESENTLY RUNNING
548 ;         MODEXH          SET TO INDICATE THAT ALL OF THE EXERCISER'S
549 ;                         OPTION MODULES HAVE BEEN ACTIVATED
550 ;         TMPPIO          SET IF BKMOD IS DOING A ONE
551 ;                         ONE ITERATION PASS, BKMOD WILL TAKE
552 ;                         AN APPEARANCE OF AN IOMOD.
553 ;
554 ; CALLING SEQUENCE:
555 ;     CALL ACTIV IN <DTABLE>
556 ;
557 ;         DTABLE = ADDRESS OF DATA TABLE
558 ;
559 ; VERSION:
560 ;     0.0
561 ;
562 ;     EDIT             BY             DATE             REASON
563 ;--
```



ACTIV - ACTIVATE NEXT OPTION MODULE  
ACTIV.MAC 13-SEP-78 16:25

MACY11 30A(1052) 20-SEP-78 17:39 PAGE 19-1  
COMMON EQUATE MODULE

SEQ 0246

564

```

566
567
568
569
570 000000'
(1) 000000'
571          000001
572          000001
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600          001777
601          060757
602          177777
603
604
605
606 000000' 000000
607 000002'
608 000002'
609          000050
610
611
612
613
614 000122' 000000
615
616 000124' 000000
617 000126' 000000
618 000130' 001000
619 000132' 000020
620 000134' 100000

.SBTTL COMMON DEFINITIONS AND REFERENCES

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

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

AC.TPTR: .WORD SBKMOD ;MODULE-TYPE-LIST POINTER (ASSEMBLED TO SBKMOD)
AC.TYPE: .WORD SBKMOD ;SPECIAL MODULE TYPE
        .WORD NBKMOD ;NON-BACKGROUND
        .WORD BKMCD ;BACKGROUND
        .WORD IOMOD ;I/O MODULE - INTERRUPT DRIVEN

```

ACTIV - ACTIVATE NEXT OPTION MODULE  
ACTIV.MAC 13-SEP-78 16:25

MACY11 30A(1052) 20-SEP-78 17:39 PAGE 20-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 0248

521 000136' 110000  
522 000140' 102000  
523 000142' 112000  
524 000144' 177777  
625  
526  
627  
628

.WORD IOMODX ;I/O MODULE - NPR DRIVEN  
.WORD ICMODP ;I/O MODULE - RUNS ONLY ON 32K BOUNDARIES  
.WORD IOMODR ;I/O MODULE - RUNS ONLY AT LOWEST MEMORY BANK  
.WORD SENDLST ;SPECIAL END-OF-LIST INDICATOR  
;  
;\*\*\*\*\*  
;

```

630 .SBTTL ACTIVATE ROUTINE
631
632     000146'          ROUTINE ACTIV <DTABLE>
(2)    000146'
633
634
635     ;+
636     ; SAVE REGISTERS AND PUT THE DATA TABLE ADDRESS IN R0.
637     ; -
638     000146'          CALL SAVREG
(3)    000146' 004767 000000G          JSR      PC,SAVREG
639     000152'          LET R0 := DTABLE(R5)
(4)    000152' 016500 000000          MOV      DTABLE(R5),R0
640
641     ;+
642     ; IF WE'VE BEEN THROUGH THE TYPE LIST ALREADY, RESET THE POINTERS TO
643     ; IOMOD AND TO START OF MODQ LIST, THEN CLEAR THE MODSEL AND MODEXH
644     ; BITS INSTATUS WORD 0.
645     ; -
646     000156'          IF #MODEXH SETIN DT.STO(R0) THEN
(6)    000156' 032760 004000 000010          BIT      #MODEXH,DT.STO(R
(9)    000164' 001414          BEQ      50002$
647     000166'          LET AC.TPTR := #AC.TYPE + #6
(4)    000166' 012767 000126' 177730          MOV      #AC.TYPE,AC.TPTR
(6)    000174' 062767 000006 177722          ADD      #6,AC.TPTR
648     000202'          LET AC.MPTR := #AC.MODQ
(4)    000202' 012767 000002' 177570          MOV      #AC.MODQ,AC.MPTR
649     000210'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #MODSEL!MODEXH
(6)    000210' 042760 005000 000010          BIC      #MODSEL!MODEXH,D
650     000216'          ENDIF
(4)    000216'          50002$:
651
652     ;+
653     ; SEARCH UNTIL THE MODULE-TYPE-LIST IS EXHAUSTED
654     ; -
655
656     000216'          WHILE @AC.TPTR NE #SENDLST DO
(4)    000216'          50003$:
(6)    000216' 027727 177702 177777          CMP      @AC.TPTR,#SENDLS
(9)    000224' 001563          BEQ      50004$
657
658     ;+
659     ; SAVE THE POINTER IN R3.
660     ; -
661
662     000226'          LET R3 := @AC.TPTR
(4)    000226' 017703 177672          MOV      @AC.TPTR,R3
663
664     ;+
665     ; SEARCH THE MODULE-QUEUE-LIST FOR A MODULE OF THE CURRENT TYPE
666     ; -
667
668     000232'          WHILE @AC.MPTR NE #0 DO
(4)    000232'          50005$:
(6)    000232' 005777 177542          TST      @AC.MPTR
(9)    000236' 001515          BEQ      50006$

```

```

669 000240'
(4) 000240' 017701 177534
670 000244'
(4) 000244' 010104
671
672
673
674
675
676 000246'
(4) 000246' 016101 000026
677 000252'
(4) 000252' 010102
(6) 000254' 042702 060757
678
679
680
681
682
683
684 000260'
(6) 000260' 020203
(9) 000262' 001077
685 000264'
(6) 000264' 032701 040000
(9) 000270' 001474
(6) 000272' 032701 024000
(9) 000276' 001071
686
687
688
689
690
691 000300'
(4) 000300' 017701 177474
692
693
694
695
696
697
698
699 000304'
(6) 000304' 005764 000034
(9) 000310' 001403
700 000312'
(4) 000312' 016104 000112
701 000316'
(4) 000316' 000402
(3) 000320'
702 000320'
(4) 000320' 016104 000030
703 000324'
(4) 000324'
704
705

```

```

          LET R1 := @AC.MPTR
                                          MOV      @AC.MPTR,R1
          LET R4 := R1
                                          MOV      R1,R4
;+
; CLEAR OUT ALL BUT THE MODULE TYPE BITS IN THE MODULE'S STATUS WORD.
;-
          LET R1 := STAT(R1)
                                          MOV      STAT(R1),R1
          LET R2 := R1 CLR.BY #STATBITS
                                          MOV      R1,R2
                                          BIC      #STATBITS,R2
;+
; IF ONE IS FOUND MAKE SURE IT IS SELECTED AND THAT IT HAS NOT
; BEEN DROPPED AND IS NOT ACTIVE.
;-
          IF R2 EQ R3 THEN
                                          CMP      R2,R3
                                          BNE     50007$
          IF #BIT14 SETIN R1 AND #BIT13!ACTBIT NOTSETIN R1 THEN
                                          BIT      #BIT14,R1
                                          BEQ     50010$
                                          BIT      #BIT13!ACTBIT,R1
                                          BNE     50010$
;+
; SAVE THE MODULE LIST POINTER.
;-
          LET R1 := @AC.MPTR
                                          MOV      @AC.MPTR,R1
;+
; IF THE OPTION MODULE'S PASS COUNT IS NOT 0, THEN SAVE ITS RESTART ADDRESS.
; OTHERWISE, SAVE THE START ADDRESS AS THIS IS THE FIRST PASS.
;-
          IF PASCNT(R4) NE #0 THEN
                                          TST     PASCNT(R4)
                                          BEQ     50011$
          LET R4 := RSTRT(R1)
                                          MOV     RSTRT(R1),R4
          ELSE
                                          BR      50012$
          LET R4 := INIT(R1)
                                          MOV     INIT(R1),R4
          ENDIF
                                          50012$:
;+

```

```
706 ; BEFORE WE EN-QUEUE THE MODULE MAKE SURE ITS STACK WILL BE SETUP PROPERLY
707 ; THEN EN-QUEUE THE MODULE IN THE CONTROL QUEUE
708 ;--
709
710 000324' LET SVR6(R1) := SPOINT(R1)
(4) 000324' 016161 000032 000076 MOV SPOINT(R1),SVR6(
711 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
712
713 ;+
714 ; MARK THE OPTION MODULE AS BEING ACTIVE
715 ;--
716
717 000350' LET STAT(R1) := STAT(R1) SET.BY #ACTBIT
(6) 000350' 052761 004000 000026 BIS #ACTBIT,STAT(R1)
718
719 ;+
720 ; IF THIS MODULE IS OF THE SBK OR NBK TYPE SET THE MODULE-HOLD-FLAG
721 ; WHICH WILL ALLOW THE MODULE TO RUN BY ITSELF
722 ;--
723
724 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$:
725 000400' LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODHOLD
(6) 000400' 052760 002000 000010 BIS #MODHOLD,DT.ST0(
726 000406' ENDIF
(4) 000406' 50014$:
727 000406' IF R2 EQ #SBKMOD THEN
(6) 000406' 020227 000000 CMP R2,#SBKMOD
(9) 000412' 001003 BNE 50015$
728 000414' LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
(6) 000414' 052760 010000 000012 BIS #SBKSEL,DT.ST1(R
729 000422' ENDIF
(4) 000422' 50015$:
730
731 ;+
732 ; IF MODULE IS A BKMOD THEN IT IS NECESSARY TO DO A ONE ITERATION
733 ; PASS. THIS IS ACCOMPLISHED BY SETTING THE TMPID BIT AND CHANGING THE
734 ; BKMOD STAT TO THAT OF AN IOMOD. PRENDIT WILL CHANGE THE MODE
735 ; BACK AT END-OF-PASS TIME.
736 ;--
737
738 000422' IF R2 EQ #BKMOD THEN
(6) 000422' 020227 000020 CMP R2,#BKMOD
(9) 000426' 001011 BNE 50016$
```

ACTIV - ACTIVATE NEXT OPTION MODULE  
ACTIV.MAC 13-SEP-78 16:25

MACY11 30A(1052) 20-SEP-78 17:39 PAGE 21-3  
ACTIVATE ROUTINE

SEQ 0252

```
739 000430'          LET DT.ST0(R0) := DT.ST0(R0) SET.BY #TMPID
(6) 000430' 052760 000002 000010          BIS      #TMPID,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$:
743
744
745          ;+
746          ; ADVANCE THE MODULE-QUEUE-LIST POINTER, THEN RESTORE THE
747          ; REGISTERS AND DO A RETURN
748          ; -
749 000452'          LET AC.MPTR := AC.MPTR + #2
(6) 000452' 062767 000002 177320          ADD      #2,AC.MPTR
750 000460'          INLINE <BR      1$>
(2) 000460' 000450          BR      1$
751 000462'          ENDIF
(4) 000462'          50010$:
752
753          ;+
754          ; THE MODULE IS NOT OF THE CURRENT TYPE OR IF IT IS, IT IS
755          ; PRESENTLY DESELECTED OR DROPPED, SO UPDATE THE MODULE-QUEUE-LIST
756          ; POINTER AND KEEP SEARCHING
757          ; -
758
759          ENDIF
(4) 000462'          50007$:
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

763
764
765
766
767
768 ;+
769 ; THE MODULE-QUEUE-LIST HAS BEEN EXHAUSTED SO SET THE POINTER BACK TO THE
770 ; BEGINNING OF IT AND UPDATE THE MODULE-TYPE-LIST POINTER
771 ;--
772 000472' LET AC.MPTR := #AC.MODQ
(4) 000472' 012767 000002' 177300 MOV #AC.MODQ,AC.MPTR
773 000500' LET AC.TPTR := AC.TPTR + #2
(6) 000500' 062767 000002 177416 ADD #2,AC.TPTR
774
775 ;+
776 ; IF THE NEW CURRENT TYPE IS THAT OF IOMODP MAKE SURE THE IOMODP
777 ; MODULES CAN RUN AT THIS ADDRESS
778 ;--
779
780 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$
781 000536' LET AC.TPTR := AC.TPTR + #2
(6) 000536' 062767 000002 177360 ADD #2,AC.TPTR
782 000544' ENDDIF
(4) 000544' 50017$:
783
784 ;+
785 ; IF ITS AN IOMODR TYPE MAKE SURE IT CAN RUN AT THIS ADDRESS
786 ;--
787
788 000544' IF @AC.TPTR EQ #IOMODR AND DT.ADDR(R0) NE #200 THEN
(6) 000544' 027727 177354 112000 CMP @AC.TPTR,#IOMODR
(9) 000552' 001007 BNE 50020$
(6) 000554' 026027 000042 000200 CMP DT.ADDR(R0),#200
(9) 000562' 001403 BEQ 50020$
789 000564' LET AC.TPTR := AC.TPTR + #2
(6) 000564' 062767 000002 177332 ADD #2,AC.TPTR
790 000572' ENDDIF
(4) 000572' 50020$:
791 000572' ENDDC
(4) 000572' 000611 BR 50003$
(3) 000574' 50004$:
792
793 ;+
794 ; THE MODULE-TYPE-LIST HAS BEEN EXHAUSTED, SO SET THE MODULE-QUEUE-LIST
795 ; EXHAUSTED INDICATOR WHICH SAYS THAT ALL OF THE OPTION MODULES HAVE
796 ; BEEN ACTIVATED
797 ;--
798
799 000574' LET DT.STO(R0) := DT.STO(R0) SET.BY #MODEXH
(6) 000574' 052760 004000 000010 BIS #MODEXH,DT.STO(R
```



ACTIV - ACTIVATE NEXT OPTION MODULE  
ACTIV.MAC 13-SEP-78 16:25

MACY11 30A(1052) 20-SEP-78 17:39 PAGE 22-1  
SET UP FOR THE NEXT PASS THRU MODULE LIST

SEQ 0254

```
800
801
802      ;+
803      ; RETURN TO CALLER
804      ;-
805      000602'      INLINE <1$:>
(2)      000602'
806      000602'      CALL RESREG
(3)      000602' 004767 000000G      JSR      PC,RESREG
807      000606'      ENDRTN
(3)      000606'
(3)      000606'      50000$:
(2)      000606' 000207      50001$:
808      RTS      PC
809      000001      .END
```

ACSR = 000102	CONFIG= 000056	ECCMEM= 000100	LPSTAT= 000001	PR7 = 000340
ACTBIT= 004000	CQOVF = 000001	ECCSTA= 000010	MAPSTA= 000200	PS = 177776
ACTIV 000146RG	CR = 000015	ENBEOB= 010000	MED = 076600	PSW = 177775
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	CTRLQ = 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	RUBCUT= 000177
BIT02 = 000004	DT.ERR= 000020	INDPAR= 000040	MSGPRM= 177776	RUNMOD= 100000
BIT03 = 000010	DT.ESI= 000044	INHDRP= 040000	MSGRES= 000001	RVALU= 001740
BIT04 = 000020	DT.EVN= 000000	INHPR= 020000	MSGSFT= 000006	SAM = 075464
BIT05 = 000040	DT.EXS= 000060	INHREL= 001000	MSGSKE= 000003	SAVREG= ***** G
BIT06 = 000100	DT.FCH= 000037	INHRE= 000400	MSGSMB= 000015	SBADR = 000102
BIT07 = 000200	DT.FCN= 000036	INIT = 000030	MSGSMH= 000014	SEKMOD= 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	NOPTY= 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	SRO = 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	KTERRO= 000040	PRI7 = 000340	SUSPND= 000001
CDERCT= 000146	DT.WBU= 000050	KTPRES= 000400	PRO = 000000	SVRO = 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\$GOD= 000400	\$NESTL= 177777	\$\$DST = 000000
SVR6 = 000076	WBSTAT= 000040	\$F\$IF = 000110	\$NSKO = 000300	\$\$ELOC= 000402
SYSCNT= 000052	WBUFEA= 000136	\$F\$INC= 000210	\$NSK1 = 000120	\$\$ERFL= 000000
YSERR= 000100	WBUFPA= 000134	\$F\$L00= 000200	\$NSK2 = 000110	\$\$FLAG= 000001
TMPID = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	\$NSK3 = 000110	\$\$FROM= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$ND = 000403	\$NSK4 = 000110	\$\$LOC = 000562R
UIPAR0= 177640	WDFR = 000116	\$F\$DR = 000320	\$NSK5 = 000110	\$\$LOCN= 000000
UIPAR1= 177642	WDT0 = 000114	\$F\$RTI= 000350	\$\$AVLE= 177777	\$\$REG = 177777
UIPAR2= 177644	WTINRE= 000352	\$F\$RTN= 000300	\$\$SK0 = 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	\$\$TD = 000000
UIPDR2= 177604	\$F\$BAD= 000401	\$ISKO = 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\$DD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:40  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0257

3 COMMON EQUATE MODULE  
558 COMMON DEFINITIONS AND REFERENCES FOR ARGCHK  
561 000000' .PRINT ;SPMAC: VERSION 1.1  
576 ARGCHK ROUTINE

```
508 .TITLE ARGCHK CHECK A KEYBOARD COMMAND ARGUMENT
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; ARGCHK
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE IS USED TO VERIFY THE EXISTENCE OF A KEYBOARD
517 ; COMMAND ARGUMENT. IT IS GIVEN THE DECODE BUFFER POINTER.
518 ; IT WILL SEARCH FOR THE FIRST NON SPACE
519 ; CHARACTER. IF IT IS A <CR> A RETURN WITH ERROR OCCURS. IN EITHER
520 ; CASE, THE UPDATED COMMAND DECODE BUFFER PTR IS RETURNED
521 ; POINTING TO THE FIRST NON SPACE CHARACTER.
522 ;
523 ; INPUTS:
524 ; 1. COMMAND DECODE BUFFER POINTER
525 ;
526 ; IMPLICIT INPUTS:
527 ; NONE
528 ;
529 ; OUTPUTS:
530 ; UPDATED COMMAND DECODE BUFFER POINTER
531 ; ERROR INDICATION
532 ;
533 ; IMPLICIT OUTPUTS:
534 ; NONE
535 ;
536 ; PATHOLOGICAL CONNECTIONS:
537 ; NONE
538 ;
539 ; SUBORDINATE ROUTINES CALLED:
540 ; NONE
541 ;
542 ; FUNCTIONAL SIDE EFFECTS:
543 ; NONE
544 ;
545 ; CALLING SEQUENCE:
546 ; CALL ARGCHK IN <PTR> OUT <NEWPTR>
547 ; WHERE PTR = CURRENT COMMAND DECODE BUFFER POINTER
548 ; NEWPTR = UPDATED COMMAND DECODE BUFFER POINTER
549 ;
550 ; VERSION:
551 ; 0.0
552 ;
553 ; EDIT DATE BY REASON
554 ;--
555
556
```

```
558          .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR ARGCHK
559
560          .MCALL STRUCT
561 000000'    STRUCT
562 (1) 000000' .PRINT  ;SPMAC: VERSION 1.1
563          000001 $LSTIN=1
564          000001 $LSTTAG=1
565
566          ;
567          ;*****
568          ;
569          ; REFERENCED BY OTHER MODULES:
570          ;
571          .GLOBL ARGCHK          ;MODULE ENTRY POINT
572          ;
573          ;*****
574
```

```

576          .SBTTL ARGCHK ROUTINE
577
578 000000'  ROUTINE ARGCHK <PTR,NEWPTR>
579          ARGCHK:
580
581          ;+
582          ; INITIALIZE AND SAVE CURRENT DECODE BUFFER POINTER
583          ; -
584
585 000000'  PUSH R0
586 (2) 000000' 010046          MOV      R0,-(SP)
587          LET R0 := PTR(R5)
588          MOV      PTR(R5),R0
589          ;+
590          ; WHILE SPACE, ADVANCE TO NEXT CHARACTER
591          ; -
592 000006'  WHILEB (R0) EQ #SPACE DO
593 (4) 000006'          50002$:
594 (6) 000006' 121027 000040          CMPB   (R0),#SPACE
595 (9) 000012' 001002          BNE   50003$
596          LET R0 := R0 + #1
597          INC   R0
598          ENDDD
599          BR   50002$
600          50003$:
601          ;+
602          ; UPDATE THE CMD DECODE BUFFER POINTER
603          ; AND IF A <CR> RETURN WITH ERROR. IF NOT A <CR> RETURN
604          ; NO ERROR
605          ; -
606 000020'  LET NEWPTR(R5) := R0
607 (4) 000020' 010065 000002          MOV      R0,NEWPTR(R5)
608          IFB (R0) EQ #CR THEN
609 (6) 000024' 121027 000015          CMPB   (R0),#CR
610 (9) 000030' 001003          BNE   50004$
611          POP R0
612 (2) 000032' 012600          MOV      (SP)+,R0
613          RETURN ERROR
614 (2) 000034' 000261          SEC
615 (4) 000036' 000403          BR   50001$
616          ELSE
617 (3) 000040'          50004$:
618          POP R0
619 (2) 000040' 012600          MOV      (SP)+,R0
620          RETURN NO.ERROR
621 (4) 000042' 000400          BR   50000$
622
623 000044'  ENDIF

```

(4) 000044'

614

615 000044'

(3) 000044'

(2) 000044' 000241

(3) 000046'

(2) 000046' 000207

616

617 000001

ENDRTN

.END

50005\$:

50000\$:

CLC

50001\$:

RTS

PC



ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RBUFSZ= 000132
ACTBIT= 004000	CTRLC = 000003	ERRTYP= 000106	MSGCKD= 000010	RBUFVA= 000124
ADDR22= 001000	CTRLQ = 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 = 000035	MSGMAP= 000021	RPTDAT= 002000
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RSTRT = 000112
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUBGUT= 000177
BIT0 = 000001	DT.CF0= 000014	IE = 000100	MSGPRM= 177776	RUNMOD= 100000
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	RSVALU= 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	INHRE= 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	NSKMOD= 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	SOPPAS= 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	KTERR0= 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	YSERR= 000100
CQCVF = 000001	ENBEOP= 010000	MEMPAS= 040000	RANNUM= 000054	TMPID = 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	\$SERFL= 000000
UIPAR2= 177644	WBUFQR= 000140	\$F\$GDD= 000400	\$LSTIN= 000001	\$FFLAG= 000001
UIPAR3= 177646	WBUFSZ= 000142	\$F\$IF = 000110	\$LSTTA= 000001	\$F\$FROM= 000000
UIPAR4= 177650	WDFR = 000116	\$F\$INC= 000210	\$NESTL= 177777	\$F\$LOC = 000030R
UIPAR5= 177652	WDTO = 000114	\$F\$LDD= 000200	\$NSKO = 000300	\$F\$LOCN= 000000
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$NSK1 = 000110	\$F\$REG = 177777
UIPAR7= 177656	WTWHMI= 000222	\$F\$ND = 000403	\$SAVLE= 177777	\$F\$RETU= 000001
UIPDR0= 177600	XFLAG = 000005	\$F\$DR = 000320	\$SSKO = 050003	\$F\$RTN1= 050000
UIPDR1= 177602	XOFF = 000023	\$F\$RTI= 000350	\$TAGLE= 177777	\$F\$RTN2= 050001
UIPDR2= 177604	XCN = 000021	\$F\$RTN= 000300	\$TAGNU= 050006	\$F\$SRC = 000000
UIPDR3= 177606	\$BGNLE= 177777	\$F\$SEL= 000140	\$TEMP = 000300	\$F\$TGSV= 000000
UIPDR4= 177610	\$ERFLG= 000000	\$F\$THE= 000330	\$TSKO = 050005	\$F\$TGS1= 000000
UIPDR5= 177612	\$F\$AND= 000310	\$F\$TRU= 000404	\$TSK1 = 050003	\$F\$TGS2= 000000
UIPDR6= 177614	\$F\$BAD= 000401	\$F\$UNT= 000130	\$F\$ARGC= 000004	\$F\$TO = 000000
UIPDR7= 177616	\$F\$BLA= 000170	\$F\$WHI= 000120	\$F\$BYTE= 000402	\$F\$TAG= 050000
WASADR= 000104	\$F\$CAS= 000150	\$F\$YES= 000402	\$F\$CASE= 000000	. = 000050R
WBSTAT= 000040	\$F\$DEC= 000220	\$IFLEV= 177777	\$F\$DST = 000000	
WBUFEA= 000136	\$F\$DDO = 000340	\$ISKO = 000001	\$F\$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)

```
508 .TITLE BACTIV - ACTIVATE NEXT BACKGROUND MODULE
509 .IDENT /V0.0/
510
511
512 ;++
513 ; MODULE NAME:
514 ; BACTIV
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ; THIS ROUTINE FETCHES THE ADDRESS OF THE NEXT BACKGROUND OPTION MODULE
518 ; TO BE ACTIVATED FROM THE BACKGROUND JOB LIST CONTAINED WITHIN THE MODULE
519 ; LIST, AND ACTIVATES THE BACKGROUND MODULE BY PLACING IT IN THE MONITOR'S
520 ; NEXT-TO-EXECUTE SLOT
521 ;
522 ; INPUTS:
523 ; 1. ADDRESS OF DATA TABLE
524 ;
525 ; IMPLICIT INPUTS:
526 ; 1. DT.BLST
527 ; 2. MODULE'S STATUS WORD (STAT)
528 ; 3. MODULE'S START ADDRESS (INIT)
529 ;
530 ; OUTPUTS:
531 ; NONE
532 ;
533 ; IMPLICIT OUTPUTS:
534 ; 1. ERROR/NO ERROR INDICATOR
535 ; WHERE ERROR - NO BKGRD MODULE TO ACTIVATE
536 ; NO ERROR - BKGRD MODULE HAS BEEN ACTIVATED
537 ;
538 ; PATHOLOGICAL CONNECTIONS:
539 ; 1. DX.HDR
540 ; 2. DX.RET
541 ;
542 ; SUBORDINATE ROUTINES CALLED:
543 ; SAVREG - SAVE REGISTERS
544 ; RESREG - RESTORE REGISTERS
545 ;
546 ; FUNCTIONAL SIDE EFFECTS:
547 ; NONE
548 ;
549 ; CALLING SEQUENCE:
550 ; CALL BACTIV IN <DTABLE>
551 ;
552 ; VERSION:
553 ; 0.0
554 ;
555 ; EDIT BY DATE REASON
556 ;--
```

```
558          .SBTTL  COMMON DEFINITIONS AND REFERENCES
559
560
561
562          .MCALL  STRUCT
563          000000' STRUCT
564          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
565          000001 000001 $LSTIN = 1
566          000001 000001 $LSTTAG = 1
567
568
569          ;
570          ;*****
571          ;
572          ; REFERENCED BY OTHER MODULES
573          ;
574          .GLOBL  BACTIV          ;MODULE'S ENTRY POINT
575          .GLOBL  BA.MPTP        ;MODULE'S BKGRD LIST POINTER
576          .GLOBL  BA.HDR        ;SUSPENDED BKGRD MODULE'S HEADER ADDRESS
577          .GLOBL  BA.RET        ;SUSPENDED BKGRD MODULE'S RETURN ADDRESS
578          .GLOBL  BA.STAT       ;MODULE'S STATUS WORD
579          ;
580          ;*****
581          ;
582          ; GLOBAL REFERENCES
583          ;
584          .GLOBL  DX.HDR        ;MONITOR'S NEXT-TO-EXECUTE
585          .GLOBL  DX.RET        ; SLOT
586          .GLOBL  SAVREG        ;SAVE REGISTERS
587          .GLOBL  RESREG        ;RESTORE REGISTERS
588          ;
589          ;*****
590          ;
591          ; GLOBAL STORAGE
592          ;
593          000000' 000000 BA.MPTR: .WORD 0 ;BACKGROUND-JOB-LIST POINTER
594          000002' 000000 BA.HDR: .WORD 0 ;STORAGE FOR SUSPENDED MODULE'S HEADER ADDRESS
595          000004' 000000 BA.RET: .WORD 0 ;STORAGE FOR SUSPENDED MODULE'S RESUME ADDRESS
596          000006' 000000 BA.STAT: .WORD 0 ;BACKGROUND LIST STATUS WORD
597          ;
598          ;*****
599          ;
```

```
601 .SBTTL ACTIVATE BACKGROUND MODULE ROUTINE
602
603
604 000010' ROUTINE BACTIV <DTABLE>
(2) 000010' BACTIV:
605
606 ;+
607 ; IS THERE A BACKGROUND MODULE CURRENTLY WORKING IT'S WAY THRU
608 ; THE QUEUING MECHANISM?
609 ; -
610
611 000010' IF #BKDEF SETIN BA.STAT THEN
(6) 000010' 032767 000002 177770 BIT #BKDEF,BA.STAT
(9) 000016' 001402 BEQ 50002$
612 000020' RETURN ERROR
(2) 000020' 000261 SEC
(4) 000022' 000513 BR 50001$
613 000024' ENDF
(4) 000024' 50002$:
614
615 ;+
616 ; DO THE NECESSARY SET UP STUFF
617 ; -
618
619 000024' CALL SAVREG
(3) 000024' 004767 000000G JSR PC,SAVREG
620 000030' LET R0 := DTABLE(R5)
(4) 000030' 016500 000000 MCV DTABLE(R5),R0
621
622 ;+
623 ; IS THERE A BACKGROUND MODULE WAITING IN A SUSPEND STATE? IF SO, ACTIVATE IT BY
624 ; ENTERING IT INTO THE MONITOR'S NEXT-TO-EXECUTE SLOT AND RETURN TO CALLER
625 ; -
626 000034' IF #SUSPND SETIN BA.STATUS THEN
(6) 000034' 032767 000001 177744 BIT #SUSPND,BA.STATU
(9) 000042' 001413 BEQ 50003$
627 000044' LET R1 := BA.HDR
(4) 000044' 016701 177732 MOV BA.HDR,R1
628 000050' LET DX.HDR := R1
(4) 000050' 010167 000000G MOV R1,DX.HDR
629 000054' LET DX.RET := BA.RET
(4) 000054' 016767 177724 000000G MOV BA.RET,DX.RET
630 000062' LET BA.STATUS := BA.STATUS CLR.BY #SUSPND
(6) 000062' 042767 000001 177716 BIC #SUSPND,BA.STATU
631 000070' INLINE <BR 1$>
(2) 000070' 000461 BR 1$
632 000072' ENDF
(4) 000072' 50003$:
633
634 ;+
635 ; OTHERWISE SEARCH THE BACKGROUND LIST LOOKING FOR A BACKGROUND MODULE TO
636 ; ACTIVATE
637 ; -
638
639 000072' IF @BA.MPTR EQ #ENDLST THEN
(6) 000072' 027727 177702 000000 CMP @BA.MPTR,#ENDLST
```

```
(9) 000100' 001003                                BNE      50004$
640
641
642          ;+
643          ; IF THE LIST ENTRY IS A ZERO, RESET IT TO THE TOP OF LIST
644          ; -
645          000102'                                LET BA.MPTR := DT.BLST(R0)
(4) 000102' 016067 000034 177670                    MOV      DT.BLST(R0),BA.M
646          000110'                                ENDF
(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'                                ENDF
(4) 000174'                                50006$:
677
```

```

678          ;+
679          ; UPDATE THE LIST POINTER AND CONTINUE SEARCHING
680          ;-
681 000174'          LET BA.MPTR := BA.MPTR + #2
(6) 000174' 062767 000002 177576          ADD      #2,BA.MPTR
682
683          ;+
684          ; IF ENTRY IS ZERO, RESET TO TOP OF LIST
685          ;-
686
687          IF @BA.MPTR EQ #0 THEN
(6) 000202' 005777 177572          TST      @BA.MPTR
(9) 000206' 001003          BNE      50007$
688          LET BA.MPTR := DT.BLST(R0)
(4) 000210' 016067 000034 177562          MOV      DT.BLST(R0),BA.M
689          ENDIF
(4) 000216'          50007$:
690 000216'          UNTIL BA.MPTR EQ R3
(3) 000216' 026703 177556          CMP      BA.MPTR,R3
(6) 000222' 001334          BNE      50005$
691
692
693          ;+
694          ; DD THE NECESSARY CLEAN-UP AND RETURN
695          ;-
696
697          CALL RESREG
(3) 000224' 004767 000000G          JSR      PC,RESREG
698          RETURN ERROR
(2) 000230' 000261          SEC
(4) 000232' 000407          BR      50001$
699
700          ;+
701          ; SET THE ACTBIT AND RETURN
702          ;-
703
704          INLINE <1$:>
(2) 000234'          1$:
705 000234'          LET STAT(R1) := STAT(R1) SET.BY #ACTBIT
(6) 000234' 052761 004000 000026          BIS      #ACTBIT,STAT(R1)
706
707          CALL RESREG
(3) 000242' 004767 000000G          JSR      PC,RESREG
708          RETURN NO.ERROR
(4) 000246' 000400          BR      50000$
709          ENDRTN
(3) 000250'          50000$:
(2) 000250' 000241          CLC
(3) 000252'          50001$:
(2) 000252' 000207          RTS      PC
710
711          .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	ENBEOB= 010000	MEMPAS= 040000	RBUFPA= 000126
APTPRE= 000200	CTRLC = C00003	ENBNUL= 000001	MODEXH= 004000	RBUFSZ= 000132
ASB = 000106	CTRLD = 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	RSVALU= 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	INHRE= 000400	MSGSNH= 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	NDAPTY= 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	SRO = 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	KTERR0= 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	\$\$BAD= 000401	\$\$WHI= 000120	\$\$CASE= 000000
SVR6 = 000076	UIPDR6= 177614	\$\$BLA= 000170	\$\$YES= 000402	\$\$DST = 000000
SYSCNT= 000052	UIPDR7= 177616	\$\$CAS= 000150	\$\$LEV= 177777	\$\$ELOC= 000403
SYSERR= 000100	WASADR= 000104	\$\$DEC= 000220	\$\$SKO = 000001	\$\$ERFL= 000000
TMPIO = 000002	WBSTAT= 000040	\$\$DDO = 000340	\$\$LOCTA= 177777	\$\$FLAG= 000001
TQOVF = 000002	WBUFEA= 000136	\$\$FAL= 000405	\$\$LSTIN= 000001	\$\$FROM= 000000
UIPAR0= 177640	WBUFPA= 000134	\$\$GDD= 000400	\$\$LSTA= 000001	\$\$LOC = 000222R
UIPAR1= 177642	WBUFRQ= 000140	\$\$IF = 000110	\$\$NESTL= 177777	\$\$LOCN= 000000
UIPAR2= 177644	WBUSZ= 000142	\$\$INC= 000210	\$\$NSKO = 000300	\$\$REG = 177777
UIPAR3= 177646	WDFR = 000116	\$\$LOD= 000200	\$\$NSK1 = 000130	\$\$RETU= 000001
UIPAR4= 177650	WDTO = 000114	\$\$NAM= 000160	\$\$NSK2 = 000110	\$\$RTN1= 050000
UIPAR5= 177652	WTINRE= 000352	\$\$ND = 000403	\$\$SAVLE= 177777	\$\$RTN2= 050001
UIPAR6= 177654	WTWHMI= 000222	\$\$DR = 000320	\$\$TAGLE= 177777	\$\$SRC = 000000
UIPAR7= 177656	XFLAG = 000005	\$\$RTI= 000350	\$\$TAGNU= 050010	\$\$TGSV= 000000
UIPDR0= 177600	XOFF = 000023	\$\$RTN= 000300	\$\$TEMP = 000300	\$\$TGS1= 000000
UIPDR1= 177602	XON = 000021	\$\$SEL= 000140	\$\$TSKO = 050005	\$\$TGS2= 000000
UIPDR2= 177604	\$\$GNLE= 177777	\$\$THE= 000330	\$\$TSK1 = 050007	\$\$TD = 000000
UIPDR3= 177606	\$\$ERFLG= 000000	\$\$TRU= 000404	\$\$ARGC= 000002	\$\$TAG= 050000
UIPDR4= 177610	\$\$SAND= 000310	\$\$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  
CCRE USED: 14K (27 PAGES)

```
508 .TITLE BADMEM - REPORT BAD MEMORY TRANSFER
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     BADMEM
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE IS USED TO REPORT A BAD MEMORY TRANSFER.
517 ;     IT IS PASSED THE ADDRESS A TABLE CONTAINING THE FAILING ADDRESS, ALONG
518 ;     WITH THE ACTUAL AND SHOULD-BE CONTENTS FOR THAT ADDRESS.  THE ROUTINE
519 ;     CONVERTS THESE VALUES TO ASCII AND OUTPUTS THEM.
520 ;
521 ; INPUTS:
522 ;     DATA TABLE ADDRESS
523 ;     ADDRESS OF TABLE CONTAINING LOW-ORDER AND HIGH-ORDER BITS OF PA
524 ;     ACTUAL CONTENTS
525 ;     SHOULD-BE CONTENTS
526 ;
527 ; IMPLICIT INPUTS:
528 ;     NONE
529 ;
530 ; OUTPUTS:
531 ;     NONE
532 ;
533 ; IMPLICIT OUTPUTS:
534 ;     NONE
535 ;
536 ; PATHOLOGICAL CONNECTIONS:
537 ;     NONE
538 ;
539 ; SUBORDINATE MODULES CALLED:
540 ;     BOAC ;BINARY TO OCTAL-ASCII CONVERSION
541 ;     BOA16 ;16-BIT BINARY TO OCTAL-ASCII CONVERSION
542 ;     MSGDHOOK ;HOOK MESSAGE TO PROPER DRIVER
543 ;     SAVREG
544 ;     RESREG
545 ;
546 ; FUNCTIONAL SIDE EFFECTS:
547 ;     NONE
548 ;
549 ; CALLING SEQUENCE:
550 ;     CALL BADMEM IN <A,B,C,D>
551 ;     A=ADDRESS OF DATA TABLE
552 ;     B=ADDRESS OF LOW ORDER 16 BITS OF PA
553 ;     C=ACTUAL CONTENTS
554 ;     D=SHOULD-BE CONTENTS
555 ;
556 ; VERSION:
557 ;     0.0
558 ;
559 ;     EDIT BY DATE REASON
560 ;
561 ;--
```

```
563 .SBTTL COMMON DEFINITIONS AND REFERENCES
564
565 .MCALL STRUCT
566 000000' STRUCT
567 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
568 000001 $LSTIN=1
569 000001 $LSTTAG=1
570 ;
571 ;*****
572 ;
573 ; REFERENCED BY OTHER MODULES
574 ;
575 .GLOBL BADMEM
576 ;
577 ;*****
578 ;
579 ; GLOBAL REFERENCES
580 ;
581 .GLOBL BOAC ;BINARY TO OCTAL-ASCII CONVERSION
582 .GLOBL BOA16 ;16-BIT BINARY TO OCTAL ASCII CONVERSION
583 .GLOBL MSGDHOOK ;HOOK MESSAGE TO PROPER DRIVER
584 .GLOBL SAVREG
585 .GLOBL RESREG
586 ;
587 ;*****
588 ;
589 ; LOCAL STORAGE
590 ;
591 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
592 000033' 000010 BM.ADR: .BLKB 10
593 000043' 040 020040 047503 .ASCII / CONTENTS: /
000050' 052116 047105 051524
000056' 020072
594 000060' 000006 BM.BD: .BLKB 6
595 000066' 020040 051440 047510 .ASCII / SHOULD BE: /
000074' 046125 020104 042502
000102' 020072
596 000104' 000006 BM.GD: .BLKB 6
597 000112' 000045 .ASCIIZ /%/
598
599 .EVEN
```

```
601 .SBTTL BADMEM ROUTINE
602
603 000114' ROUTINE BADMEM <TABL,PA,BD,GD>
604 (2) 000114' BADMEM:
605
606 ;+
607 ; SAVE REGISTERS
608 ; -
609 000114' CALL SAVREG
610 (3) 000114' 004767 000000G JSR PC, SAVREG
611
612 ;+
613 ; SET R0 TO THE START OF THE DATA TABLE, R1 TO THE ADDRESS OF THE PA TABLE,
614 ; R2 TO THE ACTUAL CONTENTS, AND R3 TO THE SHOULD-BE CONTENTS.
615 ; -
616
617 000120' LET R0 := TABL(R5)
618 (4) 000120' 016500 000000 MOV TABL(R5), R0
619 000124' LET R1 := PA(R5)
620 (4) 000124' 016501 000002 MOV PA(R5), R1
621 000130' LET R2 := BD(R5)
622 (4) 000130' 016502 000004 MOV BD(R5), R2
623 000134' LET R3 := GD(R5)
624 (4) 000134' 016503 000006 MOV GD(R5), R3
625
626 ;+
627 ; CONVERT THE PA TO ASCII.
628 ; -
629
630 000140' CALL BOAC IN <R0,(R1),2(R1),#BM.ADR>
631 (3) 000140' 010546 MOV R5, -(SP)
632 (7) 000142' 012745 000033' MOV #BM.ADR, -(R5)
633 (6) 000146' 016145 000002 MOV 2(R1), -(R5)
634 (5) 000152' 011145 MOV (R1), -(R5)
635 (4) 000154' 010045 MOV R0, -(R5)
636 (3) 000156' 004767 000000G JSR PC, BOAC
637 (3) 000162' 012605 MOV (SP)+, R5
638
639 ;+
640 ; CONVERT THE ACTUAL AND SHOULD-BE DATA TO ASCII.
641 ; -
642
643 000164' CALL BOA16 IN <R2,#BM.BD>
644 (3) 000164' 010546 MOV R5, -(SP)
645 (5) 000166' 012745 000060' MOV #BM.BD, -(R5)
646 (4) 000172' 010245 MOV R2, -(R5)
647 (3) 000174' 004767 000000G JSR PC, BOA16
648 (3) 000200' 012605 MOV (SP)+, R5
649
650 000202' CALL BOA16 IN <R3,#BM.GD>
651 (3) 000202' 010546 MOV R5, -(SP)
652 (5) 000204' 012745 000104' MOV #BM.GD, -(R5)
653 (4) 000210' 010345 MOV R3, -(R5)
```

636  
637  
638  
639  
640  
641  
642 000220' 004767 000000G  
(3) 000220' 010546  
(7) 000222' 012745 000252'  
(6) 000226' 012745 000000'  
(5) 000232' 012745 000002  
(4) 000236' 010045  
(3) 000240' 004767 000000G  
(3) 000244' 012605  
643 000246'  
(2) 000246' 000240  
644 000250'  
(2) 000250' 000776  
645 000252'  
(2) 000252'  
646  
647  
648  
649  
650  
651  
652 000252' 004767 000000G  
(3) 000252' 004767 000000G  
653  
654 000256'  
(3) 000256'  
(3) 000256'  
(2) 000256' 000207  
655 000001

;  
;+  
; OUTPUT THE MESSAGE.  
;-  
  
CALL MSGDHOOK IN <R0,#MSGPOP,#BM.MSG,#2\$>  
  
INLINE <1\$: NOP>  
  
INLINE <BR 1\$>  
  
INLINE <2\$:>  
  
;  
;+  
; RESTORE REGISTERS AND RETURN.  
;-  
  
CALL RESREG  
  
ENDRTN  
  
.END

JSR PC,BOA16  
MOV (SP)+,R5  
  
MOV R5,-(SP)  
MOV #2\$,-(R5)  
MOV #BM.MSG,-(R5)  
MOV #MSGPOP,-(R5)  
MOV R0,-(R5)  
JSR PC,MSGDHOOK  
MOV (SP)+,R5  
  
1\$: NOP  
  
BR 1\$  
  
2\$:  
  
JSR PC,RESREG  
  
50000\$:  
50001\$:  
RTS PC

ACSR = 000102	CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR5 = 000240
ACTBIT= 004000	CKTIM = 100000	DVID1 = 000014	LF = 000012	PR6 = 000300
ADDR22= 001000	CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PR7 = 000340
ADR = 000006	CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	PS = 177776
APTFER= 000004	CQOVF = 000001	ENBEOP= 010000	MED = 076600	PSW = 177776
APTPRE= 000200	CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RANNUM= 000054
ASB = 000106	CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFEA= 000130
ASSEMB= 000010	CSRC = 000102	EOPBIT= 000001	MODHQL= 002000	RBUFPA= 000126
ASTAT = 000104	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RBUFSZ= 000132
AUTO = 000010	CTRLO = 000017	EVNTBE= 000200	MSGCKD= 000010	RBUFVA= 000124
AUTOST= 020000	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RDSERV= 000101
AWAS = 000110	DCEVNT= 000011	EVNTPK= 000203	MSGDER= 000005	RDWHMI= 000022
BADMEM 000114RG	DEFRTN= 000400	EVNTPE= 000202	MSGDHO= ***** G	RELERR= 000020
BD = 000004	DIAGMC= 000000	FATERR= 100000	MSGDRP= 000017	RELMOD= 020000
BIT0 = 000001	DROPMQ= 100000	GD = 000006	MSGECH= 177777	RELTIM= 010000
BIT00 = 000001	DSEVNT= 000014	HRDCNT= 000044	MSGEOP= 000013	RESREG= ***** G
BIT01 = 000002	DT.ADD= 000042	HRDPAS= 000050	MSGHDR= 000004	RES1 = 000056
BIT02 = 000004	DT.AP = 000100	ICONT = 000036	MSGHNG= 000022	RES2 = 000060
BIT03 = 000010	DT.APK= 000076	ICOUNT= 000040	MSGHRD= 000007	RICHAR= 031060
BIT04 = 000020	DT.BLS= 000034	IDNUM = 000122	MSGMAP= 000021	RPTDAT= 002000
BIT05 = 000040	DT.CFO= 000014	IE = 000100	MSGNUL= 177775	RSTR = 000112
BIT06 = 000100	DT.CF1= 000016	INDPAR= 000040	MSGPOP= 000002	RUBOUT= 000177
BIT07 = 000200	DT.ERR= 000020	INHDRP= 040000	MSGPRM= 177776	RUNMOD= 100000
BIT08 = 000400	DT.ESI= 000044	INHEPR= 020000	MSGRES= 000001	RSVALU= 001740
BIT09 = 001000	DT.EVN= 000000	INHREL= 001000	MSGSFT= 000006	SAM = 075464
BIT1 = 000002	DT.EXS= 000060	INHRR= 000400	MSGSKE= 000003	SAVREG= ***** G
BIT10 = 002000	DT.FCH= 000037	INIT = 000030	MSGSMB= 000015	SBADR = 000102
BIT11 = 004000	DT.FCN= 000036	INTR = 000120	MSGSMH= 000014	SBKMOD= 000000
BIT12 = 010000	DT.HMX= 000104	IOMOD = 100000	MSGSMS= 000016	SBKSEL= 010000
BIT13 = 020000	DT.KBE= 000024	IOMODP= 102000	MSGSTD= 000000	SC.ADR= 000006
BIT14 = 040000	DT.KBP= 000026	IOMODR= 112000	MSGSYS= 000012	SC.ALC= 000014
BIT15 = 100000	DT.KBR= 000022	IGMODX= 110000	MSGVEC= 000020	SC.APC= 000016
BIT2 = 000004	DT.KBU= 000030	JACK = 035060	NBKMOD= 001000	SC.CKL= 000002
BIT3 = 000010	DT.MLS= 000032	KIPAR0= 172340	NCPJOP= 000020	SC.CKP= 000004
BIT4 = 000020	DT.MTI= 000110	KIPAR1= 172342	NDAPTY= 000002	SC.CLO= 000000
BIT5 = 000040	DT.OFF= 000070	KIPAR2= 172344	NULL = 000000	SC.HLD= 000010
BIT6 = 000100	DT.PAS= 000074	KIPAR3= 172346	OWEN = 024020	SC.SCA= 000012
BIT7 = 000200	DT.PC = 000002	KIPAR4= 172350	PA = 000002	SENDLS= 177777
BIT8 = 000400	DT.PFL= 000062	KIPAR5= 172352	PAERR = 000010	SCFCNT= 000042
BIT9 = 001000	DT.PSW= 000004	KIPAR6= 172354	PARPRE= 002000	SOFPAS= 000046
BKDEF = 000002	DT.PTA= 000064	KIPAR7= 172356	PARSTA= 000100	SPACE = 000040
BKMOD = 000020	DT.RCS= 000102	KIPDR0= 172300	PASCNT= 000034	SPOINT= 000032
BKMODE= 040000	DT.REL= 000040	KIPDR1= 172302	PDPLSI= 020000	SPVALU= 002200
BKSLSH= 000134	DT.SCT= 000066	KIPDR2= 172304	PDP60 = 004000	SR0 = 177572
BM.ADR 000033R	DT.SMX= 000106	KIPDR3= 172306	PDP70 = 010000	SR1 = 177574
BM.BD 00006CR	DT.SP = 000006	KIPDR4= 172310	PRI0 = 000000	SR2 = 177576
BM.GD 000104R	DT.SSI= 000046	KIPDR5= 172312	PRI1 = 000040	SR3 = 172516
BM.MSG 000000R	DT.ST0= 000010	KIPDR6= 172314	PRI4 = 000200	STAT = 000026
BOAC = ***** G	DT.ST1= 000012	KIPDR7= 172316	PRI5 = 000240	STATBI= 064757
BOA16 = ***** G	DT.SWR= 000056	KTERRO= 000040	PRI6 = 000300	STAT1 = 000027
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PRI7 = 000340	SUSPND= 000001
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PRO = 000000	SVRC = 000062
CDERCT= 000146	DT.WHL= 000054		PR4 = 000200	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	SLOCTA= 177777	\$REG = 177777
TMPID = 000002	WBSTAT= 000040	\$F\$DO = 000340	\$LSTIN= 000001	\$RETU= 000000
TQDVF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$LSTA= 000001	\$RTN1= 050001
UIPAR0= 177640	WBUFPA= 000134	\$F\$GOD= 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\$LDD= 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	\$STAG= 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  
CCRE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:41  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0279

3 COMMON EQUATE MODULE  
558 COMMON DEFINITIONS AND REFERENCES  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
594 BDACNV ROUTINE



```
508 .TITLE BDACNV - BINARY TO DECIMAL-ASCII CONVERSION ROUTINE
509 .IDENT /V0.0/
510
511
512
513 ;++
514 ; MODULE NAME:
515 ;     BDACNV
516 ;
517 ; FUNCTIONAL DESCRIPTION:
518 ;     THIS ROUTINE CONVERTS A 16-BIT UNSIGNED BINARY NUMBER
519 ;     TO A 5-DIGIT DECIMAL-ASCII NUMBER. THE TECHNIQUE USED
520 ;     IS THAT OF REPEATED DIVISION BY POWERS OF 10 (10), WITH THE RE-
521 ;     MAINDERS, WHICH COMPOSE THE DECIMAL NUMBER, BEING
522 ;     CONVERTED TO ASCII.
523 ;
524 ; INPUTS:
525 ;     1. 16-BIT BINARY NUMBER THAT IS TO BE CONVERTED
526 ;     2. ADDRESS AT WHICH THE RESULT IS TO BE STORED
527 ;
528 ; IMPLICIT INPUTS:
529 ;     NONE
530 ;
531 ; OUTPUTS:
532 ;     NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ;     1. 5-CHARACTER DECIMAL-ASCII RESULT
536 ;
537 ; PATHOLOGICAL CONNECTIONS:
538 ;     NONE
539 ;
540 ; SUBORDINATE MODULES CALLED:
541 ;     1. SAVREG          SAVE REGISTERS
542 ;     2. RESREG         RESTORE REGISTERS
543 ;
544 ; FUNCTIONAL SIDE EFFECTS:
545 ;     NONE
546 ;
547 ; CALLING SEQUENCE:
548 ;     CALL BDACNV IN <NUMBER,ADDRESS>
549 ;     NUMBER = 16-BIT BINARY NUMBER TO BE CONVERTED
550 ;     ADDRESS = ADDRESS TO STORE 5-CHAR RESULT
551 ;
552 ; VERSION:
553 ;     0.0
554 ;
555 ;     EDIT          BY          DATE          REASON
556 ;--
```

```

558 .SBTTL COMMON DEFINITIONS AND REFERENCES
559
560
561
562
563 .MCALL STRUCT
564 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
565 000001 $LSTIN = 1
566 000001 $LSTTAG = 1
567
568
569
570
571
572
573 ;
574 ;*****
575 ;
576 ; REFERENCED BY OTHER MODULES
577 ;
578 .GLOBL BDACNV ;MODULE'S ENTRY POINT
579 ;
580 ;*****
581 ;
582 ; GLOBAL REFERENCES
583 ;
584 .GLOBL SAVREG ;SAVE REGISTERS
585 .GLOBL RESREG ;RESTORE REGISTERS
586 ;
587 ;*****
588 ;
589 ; LOCAL STORAGE
590 ;
591 000000' 023420 001750 000144 BD.TEN: ^D10000,^D1000,^D100,^D10,^D1
000006' 000012 000001
592 ;

```

641	(4) 000040'	005003			CLR	R3
642	000042'		WHILE NUM(R5) HIS (R1) DO			
643	(4) 000042'			50004\$:		
644	(6) 000042'	026511	000000		CMP	NUM(R5), (R1)
645	(9) 000046'	103404			BLO	50005\$
646	000050'		LET NUM(R5) := NUM(R5) - (R1)		SUB	(R1), NUM(R5)
647	(6) 000050'	161165	000000		INC	R3
648	000054'	005203			BR	50004\$
649	(6) 000056'		ENDDO	50005\$:		
650	(4) 000056'	000771			ADD	#60, R3
651	(3) 000060'		LET R3 := R3 + #60		MOVB	R3, (R0)+
652	000064'	110320			ADD	#2, R1
653	(6) 000066'	062701	000002		BR	50002\$
654	(6) 000066'	062701	000002			
655	000072'		ENDDO	50003\$:		
656	(4) 000072'	000760				
657	(3) 000074'					
658						
659						
660						
661						
662						
663						
664						

  

655			;			
656			;	RESTORE REGISTERS AND RETURN.		
657			;			
658						
659	000074'		CALL RESREG		JSR	PC, RESREG
660	(3) 000074'	004767	000000G			
661						
662	000100'		ENDRTN	50000\$:		
663	(3) 000100'			50001\$:		
664	(3) 000100'				RTS	PC
665	(2) 000100'	000207				
666	000001		.END			

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

TQOVF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$LSTTA= 000001	\$\$ERFL= 000000
UIPAR0= 177640	WBUFPA= 000134	\$F\$G00= 000400	\$NESTL= 177777	\$\$FLAG= 000340
UIPAR1= 177642	WBUFRQ= 000140	\$F\$IF = 000110	\$NSK0 = 000300	\$\$FROM= 000000
UIPAR2= 177644	WBUFFSZ= 000142	\$F\$INC= 000210	\$NSK1 = 000120	\$\$LOC = 000046R
UIPAR3= 177646	WDFR = 000116	\$F\$L00= 000200	\$NSK2 = 000120	\$\$LOCN= 000000
UIPAR4= 177650	WDTO = 000114	\$F\$NAM= 000160	\$SAVLE= 177777	\$\$REG = 177777
UIPAR5= 177652	WTINRE= 000352	\$F\$ND = 000403	\$\$SK0 = 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	\$\$TD = 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)

.MAIN: MACY11 30A(1052) 20-SEP-78 17:42  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0286

3 COMMON EQUATE MODULE  
530 COMMON DEFINITIONS AND STORAGE  
533 000000' .PRINT ;SPMAC: VERSION 1.1  
572 BADVEC FIELD AN INTERRUPT THROUGH AN INCORRECT VECTOR  
632 BADVEC ROUTINE  
653 PBDVEC PROCESS EMTS(& INTERRUPTS THRU UNEXPECTED VECTORS)  
701 PBDVEC ROUTINE  
764 SETVEC INITIALIZE ALL UNUSED VECTOR LOCATIONS (0-774)  
810 SETVEC ROUTINE

```
508 .TITLE BDVSRV SERVICE INTERRUPTS THROUGH INCORRECT VECTORS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; BDVSRV
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS PACKAGE CONSISTS OF THE FOLLOWING ROUTINES:
517 ; 1. BADVEC - ISSUES AN EMT INSTRUCTION
518 ; 2. PBDVEC - FORMS BAD VECTOR MESSAGE
519 ; 3. SETVEC - SETS UP VECTORS
520 ;
521 ; VERSION:
522 ; 0.0
523 ;
524 ; EDIT DATE BY REASON
525 ;
526 ;
527 ;
```

```

529
530          .SBTTL COMMON DEFINITIONS AND STORAGE
531          ;
532          .MCALL STRUCT
533          STRUCT
534          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
535          000001 $LSTIN=1
536          000001 $LSTTAG=1
537
538          ;REFERENCED BY OTHER MODULES:
539          ;
540          .GLOBL BADVEC          ;MODULE ENTRY POINT
541          .GLOBL BADVC1        ;MCDULE ENTRY POINT
542          .GLOBL PBDVEC        ;MODULE ENTRY POINT
543          .GLOBL SETVEC        ;MGDULE ENTRY POINT
544
545          ;*****
546
547          ;*****
548          ; GLOBAL REFERENCES
549          ;
550          .GLOBL ENQTQ          ;MESSAGE ENQUEUER
551          .GLOBL DTABLE        ;ADDRESS OF DTABLE
552          .GLOBL SAVREG        ;SAVE REGISTERS
553          .GLOBL RESREG        ;RESTORE REGISTERS
554
555          ;
556          ;*****
557          ; LOCAL STORAGE
558          ;
559          ;*****
560
561          ;
562
563          000000' 000000 BD.VCT: .WORD 0          ;HIGH/LOW VECTOR FLAG
564          000002' 000000          .WORD 0          ;SAVE R5 STUFF HERE , NOT IN INTERRUPTED MODULE
565          000004' 000000          .WORD 0
566          000006' 000000          .WORD 0
567          000010' 000000          .WORD 0
568          000012' 000000          .WORD 0
569          000014'          SVR5S:
    
```



```
571
572      .SBTTL  BADVEC FIELD AN INTERRUPT THROUGH AN INCORRECT VECTOR
573      .IDENT /V0.0/
574
575      ;++
576      ; MODULE NAME:
577      ;     BADVEC
578      ;
579      ; FUNCTIONAL DESCRIPTION:
580      ;     THIS ROUTINE IS ENTERED AS A RESULT OF AN INTERRUPT THROUGH
581      ;     AN INCORRECT VECTOR. BY ISSUING AN EMT INSTRUCTION, THE PSW
582      ;     IS PUSHED ONTO THE STACK. THE PSW WILL
583      ;     CONTAIN THE LOCATION OF THE BAD VECTOR.
584      ;     THE ROUTINE INVOKED BY THE EMT DETERMINES THE BAD VECTOR
585      ;     AND QUEUES A BAD VECTOR MESSAGE. CONTROL IS RETURNED TO THIS
586      ;     ROUTINE AND A RETURN TO THE INTERRUPTED ROUTINE THEN OCCURS.
587      ;
588      ; INPUTS:
589      ;     NONE
590      ;
591      ; IMPLICIT INPUTS:
592      ;     NONE
593      ;
594      ; OUTPUTS:
595      ;     NONE
596      ;
597      ; IMPLICIT OUTPUTS:
598      ;     NONE
599      ;
600      ; PATHOLOGICAL CONNECTIONS:
601      ;     NONE
602      ;
603      ; SUBORDINATE ROUTINES CALLED:
604      ;     NONE
605      ;
606      ; FUNCTIONAL SIDE EFFECTS:
607      ;     SINCE THE HARDWARE HAS INTERRUPTED THROUGH THE WRONG VECTOR
608      ;     IT'S INTERRUPT ROUTINE WILL NOT BE EXECUTED.
609      ;     THE DECX OPTION MODULE CONTROLLING THE
610      ;     HARDWARE WHICH INTERRUPTED THROUGH THE WRONG VECTOR
611      ;     ADDRESS WILL BECOME SUSPENDED.
612      ;
613      ; CALLING SEQUENCE:
614      ;     CALLED BY A HARDWARE INTERRUPT THROUGH AN INCORRECT VECTOR
615      ;
616      ; VERSION:
617      ;     0.0
618      ;
619      ;     EDIT          DATE          BY          REASON
620      ; --
621
622
```

BDVSRV SERVICE INTERRUPTS THROUGH INCORRECT VECTORS MACY11 30A(1052) 20-SEP-78 17:42 PAGE 13-3  
BDVSRV.MAC 28-JUL-78 09:11 BADVEC FIELD AN INTERRUPT THROUGH AN INCORRECT VECTOR

SEQ 0290

624  
625  
626  
627  
628  
629  
630

;

```
653 .SBTTL PBDVEC PROCESS EMTS(& INTERRUPTS THRU UNEXPECTED VECTORS)
654 .IDENT /V0.0/
655
656 ;++
657 ; MODULE NAME:
658 ; PBDVEC
659 ;
660 ; FUNCTIONAL DESCRIPTION:
661 ; IF THE SYSTEM HARDWARE INTERRUPTS THRU AN UNEXPECTED VECTOR, AN
662 ; EMT INSTRUCTION IS EXECUTED WHICH PUSHES THE PSW AND PC
663 ; ONTO THE STACK AND THIS ROUTINE IS ENTERED. THIS PSW ON THE
664 ; STACK CONTAINS THE UNEXPECTED VECTOR ADDRESS. A MESSAGE
665 ; CONTAINING THE UNEXPECTED VECTOR IS QUEUED AND A RETURN TO THE
666 ; ROUTINE WHICH ISSUED THE EMT TAKES PLACE.
667 ;
668 ; INPUTS:
669 ; NONE
670 ;
671 ; IMPLICIT INPUTS:
672 ; NONE
673 ;
674 ; OUTPUTS:
675 ; NONE
676 ;
677 ; IMPLICIT OUTPUTS:
678 ; NONE
679 ;
680 ; PATHOLOGICAL CONNECTIONS:
681 ; ADDRESS OF DATA TABLE
682 ;
683 ; SUBORDINATE ROUTINES CALLED:
684 ; ENQTQ MESSAGE ENQUEUER
685 ;
686 ; FUNCTIONAL SIDE EFFECTS:
687 ; NONE
688 ;
689 ; CALLING SEQUENCE:
690 ; ISSUE AN EMT INSTRUCTION
691 ;
692 ; VERSION:
693 ; 0.0
694 ;
695 ; EDIT DATE BY REASON
696 ;--
697
698
699
```

```

701          .SBTTL PBDVEC ROUTINE
702
703 000024'  IROUTINE PBDVEC
(2) 000024'
704
705          ;+
706          ; INITIALIZATION - SAVE R0,R1 AND SAVE R5 HERE
707          ; -
708
709 000024'  PUSH R0,R1,R5
(2) 000024' 010046
(3) 000026' 010146
(4) 000030' 010546
710          ;+
711          ; INIT THE NEW R5 STACK
712          ; -
713
714 000032'  LET R5 := #SVR5S
(4) 000032' 012705 000014'
715
716          ;+
717          ; GET THE UNEXPECTED VECTOR CODE FROM THE STACK
718          ; SAVE IT IN R0,R1. CLEAR OUT ALL BUT BITS 0-3 IN R1
719          ; AND ALL BUT BITS 4-7 IN R0. SHIFT R1 TO LEFT.
720          ; AND COMBINE THE TWO WORDS INTO R0.
721          ; THIS IS DONE TO CONVERT THE SPECIAL VECTOR CODE BACK
722          ; INTO THE ACTUAL VECTOR IT REPRESENTS.
723          ; -
724
725
726
727 000036'  LET R0 := 10(SP)
(4) 000036' 016600 000010
728 000042'  LET R1 := 10(SP)
(4) 000042' 016601 000010
729 000046'  LET R1 := R1 CLR.BY #177760
(6) 000046' 042701 177760
730 000052'  LET R0 := R0 CLR.BY #177417
(6) 000052' 042700 177417
731 000056'  LET R1 := R1 SHIFT +1
(7) 000056' 006301
732 000060'  LET R0 := R0 SET.BY R1
(6) 000060' 050100
733
734          ;+
735          ; IF INTERRUPT VECTOR GREATER THEN 374 THEN ADD 400 TO BAD VECTOR
736          ; -
737
738 000062'  IF 6(SP) HI #BADVC1 AND 6(SP) LO #BADVEC THEN
(6) 000062' 026627 000006 000014'
(9) 000070' 101406
(6) 000072' 026627 000006 000020'
(9) 000100' 103002
739 000102'          LET R0 := R0 + #400
(6) 000102' 062700 000400
740 000106'  ENDIF
    
```

PBDVEC:

MOV R0,-(SP)  
 MOV R1,-(SP)  
 MOV R5,-(SP)

MOV #SVR5S,R5

MOV 10(SP),R0

MOV 10(SP),R1

BIC #177760,R1

BIC #177417,R0

ASL R1

BIS R1,R0

CMP 6(SP),#BADVC1  
 BLOS 50002\$  
 CMP 6(SP),#BADVEC  
 BHIS 50002\$

ADD #400,R0

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

```
764 .SBTTL SETVEC INITIALIZE ALL UNUSED VECTOR LOCATIONS (0-774)
765 .IDENT /V0.0/
766
767 ;++
768 ; MODULE NAME:
769 ; SETVEC
770 ;
771 ; FUNCTIONAL DESCRIPTION:
772 ; THIS ROUTINE WILL LOAD THE ADDRESS OF THE INCORRECT VECTOR
773 ; ROUTINE IN ALL UNUSED VECTOR LOCATIONS FROM 0 TO 774. IT WILL
774 ; PLACE A UNIQUE CODE IN ALL VECTOR + 2 LOCATIONS. THE CODE
775 ; REPRESENTS THE VECTOR LOCATION .
776 ;
777 ; INPUTS:
778 ; NONE
779 ;
780 ; IMPLICIT INPUTS:
781 ; NONE
782 ;
783 ; OUTPUTS:
784 ; NONE
785 ;
786 ; IMPLICIT OUTPUTS:
787 ; NONE
788 ;
789 ; PATHOLOGICAL CONNECTIONS:
790 ; NONE
791 ;
792 ; SUBORDINATE ROUTINES CALLED:
793 ; NONE
794 ;
795 ; FUNCTIONAL SIDE EFFECTS:
796 ; NONE
797 ;
798 ; CALLING SEQUENCE:
799 ; CALL <SETVEC>
800 ;
801 ; VERSION:
802 ; 0.0
803 ;
804 ; EDIT DATE BY REASON
805 ;--
806
807
```

```

309
310          .SBTTL SETVEC ROUTINE
311 000154'  ROUTINE SETVEC
312
313
314          ;+
315          ; SAVE REGISTERS AND FLAG TO LOW VECTORS
316          ; -
317 000154'  CALL SAVREG
318 (3) 000154' 004767 000000G          JSR    PC,SAVREG
319          LET BD.VCT := #0
320
321          ;+
322          ; LOAD LOC. 0 WITH BADVEC AND LOC. 2 WITH CODE
323          ; -
324 000164'  LET @#0 := #BADVEC
325 (4) 000164' 012737 000020' 000000          MOV    #BADVEC,@#0
326          LET @#2 := #0
327
328          ;+
329          ; LOAD TRACE TRAP VECTOR WITH BADVEC, AND TRACE TRAP'S PSW WITH CODE
330          ; -
331 000172'  LET @#2 := #0
332 (4) 000172' 005037 000002          CLR    @#2
333
334          ;+
335          ; LOAD TRACE TRAP VECTOR WITH BADVEC, AND TRACE TRAP'S PSW WITH CODE
336          ; -
337 000176'  LET @#14 := #BADVEC
338 (4) 000176' 012737 000020' 000014          MOV    #BADVEC,@#14
339          LET @#16 := #6
340 (4) 000204' 012737 000006 000016          MOV    #6,@#16
341
342          ;+
343          ; START AT LOC. 70 AND START LOADING CODES
344          ; -
345 000212'  LET R0 := #70
346 (4) 000212' 012700 000070          MOV    #70,R0
347
348          ;+
349          ; LOAD CODE UP TO LOCATION 1000 (WILL NOT LOAD 114,200 OR 250)
350          ; -
351 000216'  WHILE R0 NE #1000 DO
352 (4) 000216'
353 (6) 000216' 020027 001000          50002$: CMP    R0,#1000
354 (9) 000222' 001446          BEQ    50003$
355          IF R0 NE #114 AND R0 NE #200 AND R0 NE #250 THEN
356 (6) 000224' 020027 000114          CMP    R0,#114
357 (9) 000230' 001441          BEQ    50004$
358 (6) 000232' 020027 000200          CMP    R0,#200
359 (9) 000236' 001436          BEQ    50004$
360 (6) 000240' 020027 000250          CMP    R0,#250
361 (9) 000244' 001433          BEQ    50004$
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
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
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```





```

(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$:
901
902                                     ;+
903                                     ; CLEAN UP
904                                     ; -
905
906 000340'                                LET BD.VCT := #0
(4) 000340' 005067 177434                                CLR      BD.VCT
907 000344'                                CALL RESREG
(3) 000344' 0047C7 000000G                                JSR      PC,RESREG
908 000350'                                ENDRTN
(3) 000350'                                50000$:
(3) 000350'                                50001$:
(2) 000350' 000207                                RTS      PC
909 000001                                .END
    
```

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

SVR5S	000014R	UIPDR7=	177616	\$F\$DD =	000340	\$LOCTA=	177777	\$\$CASE=	000000
SVR6	= 000076	WASADR=	000104	\$F\$FAL=	000405	\$LSTIN=	000001	\$\$DST =	000000
SYSCNT=	000052	WBSTAT=	000040	\$F\$G00=	000400	\$LSTTA=	000001	\$\$ELOC=	000402
SYSERR=	000100	WBUFEA=	000136	\$F\$IF =	000110	\$NESTL=	177777	\$\$ERFL=	000000
TMPID =	000002	WBUFPA=	000134	\$F\$INC=	000210	\$NSKO =	000300	\$\$FLAG=	000001
TQOVF =	000002	WBUFRQ=	000140	\$F\$L00=	000200	\$NSK1 =	000120	\$\$FROM=	000000
UIPAR0=	177640	WBUFSZ=	000142	\$F\$NAM=	000160	\$NSK2 =	000110	\$\$LOC =	000272R
UIPAR1=	177642	WDFR =	000116	\$F\$ND =	000403	\$NSK3 =	000110	\$\$LOCN=	000000
UIPAR2=	177644	WDTO =	000114	\$F\$DR =	000320	\$NSK4 =	000110	\$\$REG =	177777
UIPAR3=	177646	WTINRE=	000352	\$F\$RTI=	000350	\$\$AVLE=	177777	\$\$RETU=	000000
UIPAR4=	177650	WTWHMI=	000222	\$F\$RTN=	000300	\$\$SK0 =	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 =	050005	\$\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:43  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0301

3 COMMON EQUATE MODULE  
582 COMMON DEFINITIONS AND REFERENCES FOR BOAC  
585 000000' .PRINT ;SPMAC: VERSION 1.1  
606 BOAC ROUTINE  
682 22 BIT ADDRESSING CONVERSION

```
508
509 .TITLE BOAC BINARY TO OCTAL ASCII CONVERSION FOR 18 & 22 BIT NUMBERS
510 .IDENT /V0.0/
511
512 ;++
513 ; MODULE NAME:
514 ; BOAC
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ; THIS ROUTINE WILL CONVERT AN 18 OR 22 BIT BINARY NUMBER TO A
518 ; 8 CHARACTER OCTAL ASCII STRING STORED AT THE CALLER SPECIFIED
519 ; ADDRESS.
520 ;
521 ; INPUTS:
522 ; 1. ADDRESS OF DATA TABLE.
523 ; 2. BITS 0 THRU 15 OF THE NUMBER TO BE CONVERTED
524 ; 3. BITS 16 & 17 OR BITS 16-21 OF THE NUMBER TO BE CONVERTED
525 ; (SEE CALLING SEQUENCE BELOW)
526 ; 4. STARTING ADDRESS OF THE 8 BYTE STORAGE AREA WHERE
527 ; THE OCTAL ASCII CHARACTERS ARE TO BE STORED.
528 ;
529 ; IMPLICIT INPUTS:
530 ; NONE
531 ;
532 ; OUTPUTS:
533 ; NONE
534 ;
535 ; IMPLICIT OUTPUTS:
536 ; OCTAL ASCII CHARACTERS
537 ;
538 ; PATHOLOGICAL CONNECTIONS:
539 ; NONE
540 ;
541 ; SUBORDINATE ROUTINES CALLED:
542 ; BOA16 16 BIT BINARY TO OCTAL ASCII CONVERT ROUTINE
543 ; SAVREG
544 ; RESREG
545 ;
546 ; FUNCTIONAL SIDE EFFECTS:
547 ; NONE
548 ;
549 ; CALLING SEQUENCE:
550 ; CALL BOAC IN <DTADR,NUM1,NUM2,ADDR>
551 ; WHERE DTADR = ADDRESS OF DTABLE
552 ; NUM1 = BITS 0-15 OF THE OCTAL NUMBER TO BE CONVERTED
553 ;
554 ; FOR 18 BITS
555 ;
556 ; NUM2 = BITS 16 & 17 OF THE OCTAL NUMBER TO BE CONVERTED
557 ; ( BITS 16 & 17 MUST BE LINED UP IN BIT POSITIONS
558 ; 4 AND 5 I.E. 0 000 000 000 XYC 000 WHERE
559 ; X = BIT 17 AND Y = BIT 16 )
560 ;
561 ; FOR 22 BITS
562 ;
563 ; NUM2 = BITS 16 - 21 OF THE # TO BE CONVERTED.
```



```
582          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR BOAC
583
584          .MCALL STRUCT
585          STRUCT
586          (1) 000000'
587          000001'
588          000001'
589
590          ;*****
591          ;
592          ;REFERENCED BY OTHER MODULES
593          ;
594          .GLOBL BOAC          ;MODULE ENTRY POINT
595          ;
596          ;*****
597          ;
598          ;GLOBAL REFERENCES
599          ;
600          .GLOBL BOA16        ;CALLED TO CONVERT BITS 0-15 TO ASCII
601          .GLOBL SAVREG
602          .GLOBL RESREG
603          ;
604          ;
```

```

606          .SBTTL BOAC ROUTINE
607
608 000000'    ROUTINE BOAC <DTADR,NUMB1,NUMB2,ADDR>
609 (2) 000000'
610
611          ;+
612          ;SAVE REGISTERS
613          ;+
614
615 000000'    CALL SAVREG
616 (3) 000000' 004767 000000G
617
618          ;+
619          ;GET DTABLE ADDRESS TO R0
620          ;+
621 000004'    LET R4 := DTADR(R5)
622 (4) 000004' 016504 000000
623
624          ;+
625          ; PUT THE LOW ORDER BITS IN R0, THE HIGH ORDER BITS IN R3, AND
626          ; THE ADDRESS OF THESTORAGE AREA IN R1.
627          ;+
628 000010'    LET R0 := NUMB1(R5)
629 (4) 000010' 016500 000002
630 000014'    LET R3 := NUMB2(R5)
631 (4) 000014' 016503 000004
632 000020'    LET R1 := ADDR(R5)
633 (4) 000020' 016501 000006
634
635          ;+
636          ; IF NOT 18-BIT ADDRESSING GO TO 22-BIT SECTION.
637          ;+
638 000024'    IF #MAPSTAT NOTSETIN DT.STO(R4) THEN
639 (6) 000024' 032764 000200 000010
640 (9) 000032' 001023
641
642          ;+
643          ;SINCE ONLY 18 BITS REQUIRED, ZERO THE UPPER 2 MOST SIGNIFICANT
644          ;ASCII LOCATIONS AND UPDATE TABLE ADDRESS
645          ;+
646 000034'    LET (R1)+ :B= #60
647 (4) 000034' 112721 000060
648 000040'    LET (R1)+ :B= #60
649 (4) 000040' 112721 000060
650
651          ;+
652          ;CONVERT BITS 0-15 TO ASCII
    
```

BOAC:

JSR PC, SAVREG

MOV DTADR(R5), R4

MOV NUMB1(R5), R0

MOV NUMB2(R5), R3

MOV ADDR(R5), R1

BIT #MAPSTAT, DT.STO(

BNE 50002\$

MOVB #60, (R1)+

MOVB #60, (R1)+





```

682      .SBTTL 22 BIT ADDRESSING CONVERSION
683      ;+
684      ;           2 2  B I T  A D D R E S S I N G
685      ;+
686
687
688      ;+
689      ;INITIALIZE AND SAVE ADDRESS + 2 OF ASCII STORAGE AREA
690      ;TO CONVERT LOWER BITS(0-15) TO OCTAL ASCII
691      ;ALSO CLEAN UP ANY UNWANTED BITS FROM BITS 16-21
692      ;+
693
694
695      000102'          LET R1 := R1 + #2
696      (6) 000102' 062701 000002          ADD      #2,R1
697      000106'          LET R3 := R3 CLR.BY #177700
698      (6) 000106' 042703 177700          BIC      #177700,R3
699
700      ;+
701      ;SAVE BITS 0-15 AND CONVERT TO OCTAL ASCII
702      ;+
703
704      000112'          CALL BDA16 IN <R0,R1>
705      (3) 000112' 010546          MOV      R5,-(SP)
706      (5) 000114' 010145          MOV      R1,-(R5)
707      (4) 000116' 010045          MOV      R0,-(R5)
708      (3) 000120' 004767 000000G      JSR      PC,BDA16
709      (3) 000124' 012605          MOV      (SP)+,R5
710
711      ;+
712      ;SHIFT BITS 16-21 LEFT ONE TIME
713      ;AND SAVE TO ALIGN BITS 16,17 WITH BITS 1,2
714      ;+
715
716      000126'          LET R3 := R3 SHIFT +1
717      (7) 000126' 006303          ASL      R3
718      000130'          LET R0 := R3
719      (4) 000130' 010300          MOV      R3,R0
720
721      ;+
722      ;MASK OFF ALL BUT BITS 0-2 IN REG AND ADD TO MOST
723      ;SIGNIFICANT OCTAL ASCII CHARACTER
724      ;THE SEQUENCE SHOWN IS FOR HANDLING ODD ADDRESSES AS WELL AS EVEN
725      ;+
726
727      000132'          LET R0 := R0 CLR.BY #177770
728      (6) 000132' 042700 177770          BIC      #177770,R0
729      000136'          LET R2 :=B= (R1)
730      (4) 000136' 111102          MOVB    (R1),R2
    
```

```

727 000140'          LET R2 := R2 + R0
(6) 000140' 060002          ADD      R0,R2
728 000142'          LET (R1) :B= R2
(4) 000142' 110211          MOVWB   R2,(R1)
729
730                    ;+
731                    ;SHIFT BITS 16-21 3 TIMES TO RIGHT
732                    ;--
733
734 000144'          LET R3 := R3 SHIFT -3
(7) 000144' 006203          ASR      R3
(7) 000146' 006203          ASR      R3
(7) 000150' 006203          ASR      R3
735
736
737                    ;+
738                    ;SET UP COUNTER
739                    ;--
740
741 000152'          LET R2 := #2
(4) 000152' 012702 000002          MOV      #2,R2
742
743                    ;+
744                    ;WHILE COUNTER NOT = 0 ,SAVE BITS 18-21 IN R0,MASK OFF ALL
745                    ;BUT BITS 0-2 IN R0,ADD #60 TO R0, STORE R0 IN ASCII AREA
746                    ;SHIFT BITS 18-21 3 TIMES TO RIGHT AND DECREMENT COUNTER
747                    ;--
748
749                    WHILE R2 NE #0 DO
750                    50004$:
(4) 000156'          TST      R2
(6) 000156' 005702          BEQ      50005$
(9) 000160' 001413
751                    ;+
752                    ;SAVE BITS 18-21
753                    ;--
754
755 000162'          LET R0 := R3
(4) 000162' 010300          MOV      R3,R0
756
757                    ;+
758                    ;MASK OFF UNNEEDED BITS AND ADD # 60
759                    ;--
760
761 000164'          LET R0 := R0 CLR.BY #177770
(6) 000164' 042700 177770          BIC      #177770,R0
762 000170'          LET R0 := R0 + #60
(6) 000170' 062700 000060          ADD      #60,R0
763
764                    ;+
765                    ;STORE IN ASCII AREA, SHIFT BITS AND DECREMENT COUNTER
766                    ;--
767
768 000174'          LET -(R1) :B= R0
(4) 000174' 110041          MOVWB   R0,-(R1)
769 000176'          LET R3 := R3 SHIFT -3
    
```

```

(7) 000176' 006203
(7) 000200' 006203
(7) 000202' 006203
770 000204'
(6) 000204' 005302
771
772 000206' ENDDO
(4) 000206' 000763
(3) 000210'
773
774
775
776
777
778 000210' ENDIF
(4) 000210'
779
780
781
782
783
784 000210' CALL RESREG
(3) 000210' 004767 000000G
785
786 000214' ENDRTN
(3) 000214'
(3) 000214'
(2) 000214' 000207
787 000001 .END
    
```

LET R2 := R2 - #1

ASR R3  
 ASR R3  
 ASR R3  
 DEC R2  
 BR 50004\$  
 50005\$:  
 50003\$:  
 JSR PC,RESREG  
 50000\$:  
 50001\$:  
 RTS PC

ACSR = 000102	CR = 000015	ENBEOB= 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	CTRLQ = 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.CF0= 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	RSVALU= 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	INHRR= 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	ICMODP= 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	NCPUGP= 000020	SC.CKL= 000002
BIT13 = 020000	DT.KBU= 000030	JACK = 035060	NDAPTY= 000002	SC.CKP= 000004
BIT14 = 040000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.CLD= 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
BDAC 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
CASSTAT= 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	\$ISKO = 000001	\$\$DST = 000000
TMPID = 000002	WBSTAT= 000040	\$F\$DO = 000340	\$LOCTA= 177777	\$\$ELDC= 000402
TQDYF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$ERFL= 000000
UIPAR0= 177640	WBUFPA= 000134	\$F\$G00= 000400	\$LSTTA= 000001	\$\$FLAG= 000001
UIPAR1= 177642	WBUFRQ= 000140	\$F\$IF = 000110	\$NESTL= 177777	\$\$FROM= 000000
UIPAR2= 177644	WBUFSZ= 000142	\$F\$INC= 000210	\$NSKO = 000300	\$\$LDC = 000160R
UIPAR3= 177646	WDFR = 000116	\$F\$L00= 000200	\$NSK1 = 000110	\$\$LOCN= 000000
UIPAR4= 177650	WDT0 = 000114	\$F\$NAM= 000160	\$NSK2 = 000120	\$\$REG = 177777
UIPAR5= 177652	WTINRE= 000352	\$F\$NO = 000403	\$\$AVLE= 177777	\$\$RETU= 000000
UIPAR6= 177654	WTWHMI= 000222	\$F\$DR = 000320	\$\$SK0 = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:43  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0312

3 COMMON EQUATE MODULE  
559 COMMON DEFINITIONS AND REFERENCES FOR BOA16  
562 000000' .PRINT ;SPMAC: VERSION 1.1  
578 BOA16 ROUTINE

```
508
509 .TITLE BOA16 OCTAL TO ASCII CONVERSION ROUTINE FOR 16 BIT NUMBERS
510 .IDENT /V0.0/
511
512 ;++
513 ; MODULE NAME:
514 ;     BOA16
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ;     THIS ROUTINE WILL CONVERT A SIX CHARACTER (SIXTEEN BIT) OCTAL
518 ;     NUMBER TO A SIX CHARACTER ASCII STRING STORED AT THE CALLER
519 ;     SPECIFIED ADDRESS.
520 ;
521 ; INPUTS:
522 ;     1. NUMBER TO BE CONVERTED.
523 ;     2. START ADDRESS OF 6 BYTE STORAGE AREA WHERE
524 ;     ASCII CHARACTERS ARE TO BE STORED.
525 ;
526 ; IMPLICIT INPUTS:
527 ;     NONE
528 ;
529 ; OUTPUTS:
530 ;     NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ;     NONE
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ;     NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ;     SAVREG - SAVE GPRS
540 ;     RESREG - RESTORE GPRS
541 ;
542 ; FUNCTIONAL SIDE EFFECTS:
543 ;     NONE
544 ;
545 ; CALLING SEQUENCE:
546 ;     CALL BOA16 IN <A,B>
547 ;     WHERE A = NUMBER TO BE CONVERTED
548 ;           B = START ADDRESS OF 6 BYTE STORAGE AREA WHERE ASCII
549 ;           CHARACTERS ARE TO BE STORED
550 ;
551 ;
552 ;     EDIT          DATE          BY          REASON
553 ;     ----          -
554 ;
555
556
557
```



```
559          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR BOA16
560
561          .MCALL STRUCT
562          STRUCT
563          (1) 000000'
564          000001
565          000001
566          ; *****
567          ; REFERENCED BY OTHER MODULES:
568          ;
569          .GLOBL BOA16          ;ENTRY POINT FOR THIS MODULE
570          ;
571          ; *****
572          ; GLOBAL REFERENCES
573          ;
574          .GLOBL SAVREG        ;SAVE GPRS ROUTINE
575          .GLOBL RESREG        ;RESTORE GPRS ROUTINE
```

```

577
578 .SBTTL BOA16 ROUTINE
579 000000' ROUTINE BOA16 <NUM,ADRESS>
580 (2) 000000' BOA16:
581 ;+
582 ; SAVE REGISTERS
583 ; -
584
585 000000' CALL SAVREG
586 (3) 000000' 004767 000000G JSR PC, SAVREG
587 ;+
588 ; GET 6 BYTE STORAGE ADDRESS + 6 AND SET CHARACTER COUNT
589 ; -
590
591 000004' LET R0 := ADRESS(R5) + #6
592 (4) 000004' 016500 000002 MOV ADDRESS(R5),R0
593 (6) 000010' 062700 000006 ADD #6,R0
594 000014' LET R1 := #6
595 (4) 000014' 012701 000006 MOV #6,R1
596 000020' LET R3 := NUM(R5)
597 (4) 000020' 016503 000000 MOV NUM(R5),R3
598 ;+
599 ; WHILE R1 IS NOT = 0,GET THE OCTAL #,CLEAR OUT ALL BUT LSB 0-2,
600 ; ADD 60 (CONVERT TO ASCII), STORE THE ASCII ,UPDATE POINTER,CLEAR
601 ; OUT 3 BITS JUST CONVERTED, SHIFT NEXT 3 BITS, DECREMENT COUNTER
602 ; -
603 WHILE R1 NE #0 DO
604 (4) 000024' 50002$:
605 (6) 000024' 005701 TST R1
606 (9) 000026' 001415 BEQ 50003$
607 LET R2 := R3
608 (4) 000030' 010302 MOV R3,R2
609 LET R2 := R2 CLR.BY #177770
610 (6) 000032' 042702 177770 BIC #177770,R2
611 LET R2 := R2 + #60
612 (6) 000036' 062702 000060 ADD #60,R2
613 LET -(R0) :B= R2
614 (4) 000042' 110240 MOVB R2,-(R0)
615 LET R3 := R3 CLR.BY #7
616 (6) 000044' 042703 000007 BIC #7,R3
617 LET R3 := R3 ROTATE -3
618 (7) 000050' 006003 ROR R3
619 (7) 000052' 006003 ROR R3
620 (7) 000054' 006003 ROR R3
621 LET R1 := R1 - #1
622 (6) 000056' 005301 DEC R1
623 ENDDO
624 (4) 000060' 000761 50003$: BR 50002$
625 (3) 000062'
626 ;+
627

```

```
613          ; CLEAN UP
614          ; -
615
616 000062'   CALL RESREG
(3) 000062' 004767 000000G          JSR      PC,RESREG
617
618 000066'   ENDRTN
(3) 000066'
(3) 000066'          50000S:
(2) 000066' 000207          50001S:
619          000001          RTS      PC
        .END
```

SYMBOL TABLE	
ACSR = 000102	CSRA = 000100
ACTBIT = 004000	CSRC = 000102
ADDR22 = 001000	CTRLC = 000003
ADR = 000006	CTRLQ = 000017
ADRESS = 000002	CTRLU = 000025
APTFER = 000004	DCEVNT = 000011
APTPRE = 000200	DEFRTN = 000400
ASB = 000106	DIAGMC = 000000
ASSEMB = 000010	DROPMO = 100000
ASTAT = 000104	DSEVNT = 000014
AUTO = 000010	DT.ADD = 000042
AUTOST = 020000	DT.AP = 000100
AWAS = 000110	DT.APK = 000076
BIT0 = 000001	DT.BLS = 000034
BIT01 = 000002	DT.CF0 = 000014
BIT02 = 000004	DT.CF1 = 000016
BIT03 = 000010	DT.ERR = 000020
BIT04 = 000020	DT.ESI = 000044
BIT05 = 000040	DT.EVN = 000000
BIT06 = 000100	DT.EXS = 000060
BIT07 = 000200	DT.FCH = 000037
BIT08 = 000400	DT.FCN = 000036
BIT09 = 001000	DT.HMX = 000104
BIT1 = 000002	DT.KBE = 000024
BIT10 = 002000	DT.KBP = 000026
BIT11 = 004000	DT.KBR = 000022
BIT12 = 010000	DT.KBU = 000030
BIT13 = 020000	DT.MLS = 000032
BIT14 = 040000	DT.MTI = 000110
BIT15 = 100000	DT.OFF = 000070
BIT2 = 000004	DT.PAS = 000074
BIT3 = 000010	DT.PC = 000002
BIT4 = 000020	DT.PFL = 000062
BIT5 = 000040	DT.PSW = 000004
BIT6 = 000100	DT.PTA = 000064
BIT7 = 000200	DT.RCS = 000102
BIT8 = 000400	DT.REL = 000040
BIT9 = 001000	DT.SCT = 000066
BKDEF = 000002	DT.SMX = 000106
BKMOD = 000020	DT.SP = 000006
BKMODE = 040000	DT.SSI = 000046
BKSLSH = 000134	DT.ST0 = 000010
BOA16 000000RG	DT.ST1 = 000012
CAPRES = 000004	DT.SWR = 000056
CASAT = 000004	DT.SYP = 000072
CDERCT = 000146	DT.WBU = 000050
CDWDCT = 000144	DT.WHL = 000054
CKTIM = 100000	DT.WLL = 000052
CLKPRE = 000001	DVID1 = 000014
CONFIG = 000056	ECCMEM = 000100
CQOVF = 000001	ECCSTA = 000010
CR = 000015	ENBEOP = 010000
	ENBNUL = 000001
	ENDLST = 000000
	EOPBIT = 000001
	ERRTP = 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
	INHPR = 020000
	INHREL = 001000
	INHRE = 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
	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
	PRO = 000000
	PR4 = 000200
	PR5 = 000240
	PR6 = 000300
	PR7 = 000340
	PS = 177776
	PSW = 177776
	RANNUM = 000054
	RBUFEA = 000130
	RBUFPA = 000126
	RBUFSZ = 000132
	RSUFVA = 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
	RVALU = 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
	SDFCNT = 000042
	SDFPAS = 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
	YSERR = 000100
	TMPIO = 000002

TQDVF = 000002	WBSTAT= 000040	\$FSDEC= 000220	\$IFLEV= 177777	\$SELOC= 000000
UIPAR0= 177640	WBUFEA= 000136	\$FSDO = 000340	\$LOCTA= 177777	\$\$SERFL= 000000
UIPAR1= 177642	WBUFPA= 000134	\$FFAL= 000405	\$LSTIN= 000001	\$\$FLAG= 000340
UIPAR2= 177644	WBUFQR= 000140	\$FGOO= 000400	\$LSTTA= 000001	\$\$FROM= 000000
UIPAR3= 177646	WBUFSZ= 000142	\$FSIF = 000110	\$NESTL= 177777	\$\$LOC = 000026R
UIPAR4= 177650	WDFR = 000116	\$FSINC= 000210	\$NSKO = 000300	\$\$LCCN= 000000
UIPAR5= 177652	WDTO = 000114	\$FL00= 000200	\$NSK1 = 000120	\$\$REG = 177777
UIPAR6= 177654	WTINRE= 000352	\$FSNAM= 000160	SSAVLE= 177777	\$\$RETU= 000000
UIPAR7= 177656	WTWHMI= 000222	\$FSND = 000403	SSKO = 050003	\$\$RTN1= 050000
UIPDR0= 177600	XFLAG = 000005	\$FSOR = 000320	\$TAGLE= 177777	\$\$RTN2= 050001
UIPDR1= 177602	XOFF = 000023	\$FSRTI= 000350	\$TAGNU= 050004	\$\$SRC = 000000
UIPDR2= 177604	XON = 000021	\$FSRTN= 000300	\$TEMP = 000300	\$\$TGSV= 000000
UIPDR3= 177606	\$BGNLE= 177777	\$FSSEL= 000140	\$TSKO = 050002	\$\$TGS1= 000000
UIPDR4= 177610	\$ERFLG= 000400	\$FSTHE= 000330	\$TSK1 = 050003	\$\$TGS2= 000000
UIPDR5= 177612	\$FSAND= 000310	\$FSTRU= 000404	\$\$ARGC= 000004	\$\$TO = 000000
UIPDR6= 177614	\$FSBAD= 000401	\$FSUNT= 000130	\$\$BYTE= 000403	\$\$TAG= 050000
UIPDR7= 177616	\$FBLA= 000170	\$F\$WHI= 000120	\$\$CASE= 000000	. = 000070R
WASADR= 000104	\$FCAS= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:44  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0319

3 COMMON EQUATE MODULE  
552 COMMON DEFINITIONS AND REFERENCES  
554 000000' .PRINT ;SPMAC: VERSION 1.1  
572 CLREOP ROUTINE

CLREOP - CLEAR END-OF-PASS INDICATORS  
CLREOP.MAC 28-JUL-78 09:12

MACY11 30A(1052) 20-SEP-78 17:44 PAGE 19  
COMMON EQUATE MODULE

SEQ 0320

```
508 .TITLE CLREOP - CLEAR END-OF-PASS INDICATORS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     CLREOP
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CLEARS THE 1ST-END-OF-PASS INDICATOR IN EACH OPTION
517 ;     MODULE OF THE RUNTIME EXERCISER
518 ;
519 ; INPUTS:
520 ;     DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ;     DT.MLST
524 ;
525 ; OUTPUTS:
526 ;     NONE
527 ;
528 ; IMPLICIT OUTPUTS:
529 ;     MODULES' 1ST-EOP INDICATOR, LOCATED IN MODULE'S HEADER WORD, XFLAG
530 ;
531 ; PATHOLOGICAL CONNECTIONS:
532 ;     NONE
533 ;
534 ; SUBORDINATE MODULES CALLED:
535 ;     SAVREG - SAVE REGISTERS
536 ;     RESREG - RESTORE REGISTERS
537 ;
538 ; FUNCTIONAL SIDE EFFECTS:
539 ;     NONE
540 ;
541 ; CALLING SEQUENCE:
542 ;     CALL CLREOP IN <A>
543 ;         A=DATA TABLE ADDRESS
544 ;
545 ; VERSION:
546 ;     0.0
547 ;
548 ;     EDIT             BY             DATE             REASON
549 ;
550 ;--
```

CLREOP - CLEAR END-OF-PASS INDICATORS  
CLREOP.MAC 28-JUL-78 09:12

MACY11 30A(1052) 20-SEP-78 17:44 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 0321

```
552 .SBTTL COMMON DEFINITIONS AND REFERENCES
553 .MCALL STRUCT
554 000000'
(1) 000000'
555 000001
556 000001
557
558 ;
559 ;*****
560 ;
561 ; REFERENCED BY OTHER MODULES:
562 ;
563 .GLOBL CLREOP ;MODULE ENTRY POINT
564 ;
565 ;*****
566 ;
567 ; GLOBAL REFERENCES
568 ;
569 .GLOBL SAVREG ;SAVE GPR'S
570 .GLOBL RESREG ;RESTORE GPR'S
```



```

572          .SBTTL CLREOP ROUTINE
573
574 000000'  ROUTINE CLREOP <TABL>
(2) 000000'
575
576
577          ;+
578          ; SAVE REGISTERS
579          ; -
580
581 000000'  CALL SAVREG
(3) 000000' 004767 000000G
582
583
584          ;+
585          ; SET R0 TO THE START OF THE DATA TABLE
586          ; -
587
588 000004'  LET R0 := TABL(R5)
(4) 000004' 016500 000000
589
590
591          ;+
592          ; GET THE MODULE LIST
593          ; -
594
595 000010'  LET R1 := DT.MLST(R0)
(4) 000010' 016001 000032
596
597
598          ;+
599          ; CLEAR EOPBIT IN XFLAG IN EACH MODULE HEADER
600          ; -
601
602          WHILE (R1) NE #ENDLST DO
(4) 000014'
(6) 000014' 021127 000000
(9) 000020' 001405
603          LET R2 := (R1)+
(4) 000022' 012102
604          LET XFLAG(R2) :B= XFLAG(R2) CLR.BY #EOPBIT
(6) 000024' 142762 000001 000005
605          ENDDO
(4) 000032' 000770
(3) 000034'
606
607
608          ;+
609          ; RESTORE REGISTERS AND RETURN
610          ; -
611
612 000034'  CALL RESREG
(3) 000034' 004767 000000G
613
614          ENDRTN
(3) 000040'

```

CLREOP:

JSR PC, SAVREG

MOV TABL(R5), R0

MOV DT.MLST(R0), R1

50002\$:

CMP (R1), #ENDLST  
 BEQ 50003\$

MOV (R1)+, R2

BICB #EOPBIT, XFLAG(R2)

BR 50002\$

50003\$:

50000\$:

CLREOP - CLEAR END-OF-PASS INDICATORS  
CLREOP.MAC 28-JUL-78 09:12

MACY11 30A(1052) 20-SEP-78 17:44 PAGE 19-3  
CLREOP ROUTINE

SEQ 0323

(3) 000040'  
(2) 000040' 000207  
615  
616 000001

.END

50001\$:  
RTS PC

ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RDSERV= 000101
ACTBIT= 004000	CTRLC = 000003	ERRTP= 000106	MSGCKD= 000010	RDWHMI= 000022
ADDR22= 001000	CTRL0 = 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	DROPMQ= 100000	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
ASTAT = 000104	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AUTJ = 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	RVALU= 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	INHRE= 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	IQMODP= 102000	MSGVEC= 000020	SC.CKL= 000002
BIT1 = 000002	DT.KBP= 000026	IQMODR= 112000	NEKMOD= 001000	SC.CKP= 000004
BIT10 = 002000	DT.KBR= 000022	IQMODX= 110000	NPCUOP= 000020	SC.CLO= 000000
BIT11 = 004000	DT.KBU= 000030	JACK = 035060	NOPTY= 000002	SC.HLD= 000010
BIT12 = 010000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT13 = 020000	DT.MTI= 000110	KIPAR1= 172342	GWEN = 024020	SENDLS= 177777
BIT14 = 040000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SQFCNT= 000042
BIT15 = 100000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SQFPAS= 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	SRO = 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	PRO = 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	TMPID = 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\$GDO= 000400	\$LSTTA= 000001	\$\$FROM= 000000
UIPAR3= 177646	WBUFFSZ= 000142	\$F\$IF = 000110	\$NESTL= 177777	\$\$LOC = 000020R
UIPAR4= 177650	WDFR = 000116	\$F\$INC= 000210	\$NSKO = 000300	\$\$LOCN= 000000
UIPAR5= 177652	WDTO = 000114	\$F\$LDO= 000200	\$NSK1 = 000120	\$\$REG = 177777
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$SAVLE= 177777	\$\$RETU= 000000
UIPAR7= 177656	WTWHMI= 000222	\$F\$ND = 000403	\$SSKO = 050003	\$\$RTN1= 050000
UIPDR0= 177600	XFLAG = 000005	\$F\$QR = 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	\$TSKO = 050002	\$\$TGS1= 000000
UIPDR4= 177610	\$ERFLG= 000400	\$F\$THE= 000330	\$TSK1 = 050003	\$\$TGS2= 000000
UIPDR5= 177612	\$F\$AND= 000310	\$F\$TRU= 000404	\$\$ARGC= 000002	\$\$TD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:44  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0326

3 COMMON EQUATE MODULE  
552 COMMON DEFINITIONS AND REFERENCES  
557 000000' .PRINT ;SPMAC: VERSION 1.1  
584 CMDCPY ROUTINE

```
508 .TITLE CMDCPY - COMMAND COPY
509 .IDENT /V0.0/
510 ;++
511 ; MODULE NAME:
512 ;     CMDCPY
513 ;
514 ; FUNCTIONAL DESCRIPTION:
515 ;     COPIES A COMMAND FROM THE INPUT BUFFER
516 ;     TO THE DECODE BUFFER, EXCLUDING RUBBED OUT CHARACTERS
517 ;
518 ; INPUTS:
519 ;     DATA TABLE ADDRESS, DECODE BUFFER ADDRESS
520 ;
521 ; IMPLICIT INPUTS:
522 ;     DT.KBUF
523 ;
524 ; OUTPUTS:
525 ;     ABORT FLAG
526 ;
527 ; IMPLICIT OUTPUTS:
528 ;     NONE
529 ;
530 ; PATHOLOGICAL CONNECTIONS:
531 ;     OV.HIKB
532 ;
533 ; SUBORDINATE MODULES CALLED:
534 ;     SAVREG
535 ;     RESREG
536 ;
537 ; FUNCTIONAL SIDE EFFECTS:
538 ;     NONE
539 ;
540 ; CALLING SEQUENCE:
541 ;     CALL CMDCPY IN <A,B> OUT <C>
542 ;           A=DATA TABLE ADDRESS
543 ;           B=DECODE BUFFER ADDRESS
544 ;           C=ABORT FLAG
545 ;
546 ; VERSION:
547 ;     0.0
548 ;
549 ;           EDIT           BY           DATE           REASON
550 ;--
```

```
552          .SBTTL COMMON DEFINITIONS AND REFERENCES
553
554
555
556          .MCALL STRUCT
557          STRUCT
558          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
559          000001 $LSTIN=1
560          000001 $LSTTAG=1
561
562
563          ;
564          ;*****
565          ;
566          ; REFERENCED BY OTHER MODULES
567          ;
568          .GLOBL CMDCPY          ;MODULE ENTRY POINT
569          ;
570          ;*****
571          ;
572          ; GLOBAL REFERENCES
573          ;
574          .GLOBL SAVREG
575          .GLOBL RESREG
576          .GLOBL OV.HIKB          ;KB BUF HIGH ADDR
577          ;
578          ;*****
579          ; LOCAL STORAGE:
580          ;
581          000000' 000000G CM.KSIZ:          OV.HIKB          ;KB HIGH LIMIT
582          ;
```

584				.SBTTL CMDCPY ROUTINE		
585	000002'			ROUTINE CMDCPY <TABL,DECBUF,ABORT>		
(2)	000002'					CMDCPY:
586						
587				;++		
588				;SAVE REGISTERS		
589				;-		
590						
591	000002'			CALL SAVREG		
(3)	000002'	004767	000000G		JSR	PC,SAVREG
592						
593				;++		
594				;SET R0 TO START OF DTABLE		
595				;-		
596						
597	000006'			LET R0 := TABL(R5)		
(4)	000006'	016500	000000		MOV	TABL(R5),R0
598						
599				;++		
600				;CLEAR THE ABORT FLAG		
601				;-		
602						
603	000012'			LET ABORT(R5) := #0		
(4)	000012'	005065	000004		CLR	ABORT(R5)
604						
605				;++		
606				;INITIALIZE INPUT BUFFER POINTER		
607				;-		
608						
609	000016'			LET R1 := DT.KBUF(R0)		
(4)	000016'	016001	000030		MOV	DT.KBUF(R0),R1
610						
611				;++		
612				;INITIALIZE DECODE BUFFER POINTER		
613				;-		
614						
615	000022'			LET R2 := DECBUF(R5)		
(4)	000022'	016502	000002		MOV	DECBUF(R5),R2
616						
617				;++		
618				;IF FIRST CHARACTER IS <CR> OR <RUBOUT>, SET		
619				;ABORT FLAG		
620				;-		
621						
622	000026'			IFB (R1) EQ #CR ORB (R1) EQ #RUBOUT THEN		
(6)	000026'	121127	000015		CMPB	(R1),#CR
(8)	000032'	001403			BEQ	50002\$
(6)	000034'	121127	000177		CMPB	(R1),#RUBOUT
(9)	000040'	001004			BNE	50003\$
(6)	000042'					50002\$:
623						
624	000042'			LET ABORT(R5) := #1		
(4)	000042'	012765	000001 000004		MOV	#1,ABORT(R5)
625						
626	000050'			ELSE		
(4)	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                                     BHS      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,ABJRT(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                                     ;+

```

```

661
662 000126'          LET (R2)+ :B= (R1)
(4) 000126' 111122          MOVB   (R1), (R2)+
663 000130'          ENDIF          50014$:
(4) 000130'
664
665          ;+
665          ;GET NEXT INPUT BUFFER CHARACTER
667          ;--
668
669 000130'          LET R1 := R1 + #1
(6) 000130' 005201          INC    R1
670
671          ENDIF          50010$:
(4) 000132'
672          ENDDO          BR     50005$
(4) 000132' 000747          50006$:
(3) 000134'
673
674
675          ;+
676          ;IF THE ENTIRE INPUT STRING WAS RUBBED OUT, SET THE ABORT FLAG
677          ;(FIRST, BACK UP THE DECODE BUFFER POINTER TO ALLOW
678          ;FOR THE CR AND LF, THEN SEE IF THE POINTER IS AT THE
679          ;BEGINNING OF THE BUFFER.)
680          ;--
681
682 000134'          LET R2 := R2 - #2
(6) 000134' 162702 000002          SUB   #2, R2
683 000140'          IF R2 LOS DECBUF(R5) THEN
(6) 000140' 020265 000002          CMP   R2, DECBUF(R5)
(9) 000144' 101003          BHI   50015$
684 000146'          LET ABORT(R5) := #1
(4) 000146' 012765 000001 000004          MOV   #1, ABORT(R5)
685 000154'          ENDIF          50015$:
(4) 000154'
686
687
688 000154'          ENDIF          50004$:
(4) 000154'
689          ;+
690          ;RESTORE REGISTERS AND RETURN
691          ;--
692
693 000154'          INLINE <1$:>
(2) 000154'          1$:
694 000154'          CALL RESREG
(3) 000154' 004767 000000G          JSR   PC, RESREG
695
696 000160'          ENDRTN          50000$:
(3) 000160'          50001$:
(3) 000160'
(2) 000160' 000207          RTS   PC
697 000001          .END
  
```

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

SYSERR= 000100	WBSTAT= 000040	\$F\$G00= 000400	\$LSTTA= 000001	\$\$BYTE= 000403
TABL = 000000	WBUFEA= 000136	\$F\$IF = 000110	\$NESTL= 177777	\$\$CASE= 000000
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSK0 = 000300	\$\$DST = 000000
TQOVF = 000002	WBUFRQ= 000140	\$F\$L00= 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	\$\$AVLE= 177777	\$\$LOCN= 000000
UIPAR5= 177652	XFLAG = 000005	\$F\$SEL= 000140	\$\$SK0 = 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	\$ISKO = 000001	\$TSK3 = 050010	\$\$TGS2= 000000
UIPDR5= 177612	\$F\$CAS= 000150	\$ISK1 = 000001	\$TSK4 = 050014	\$\$TD = 000000
UIPDR6= 177614	\$F\$DEC= 000220	\$ISK2 = 000001	\$TSK5 = 050012	\$\$TAG= 050000
UIPDR7= 177616	\$F\$DD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:45  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0334

3 COMMON EQUATE MODULE  
554 COMMON DEFINITIONS AND REFERENCES  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
581 CMDDCD ROUTINE

```
508 .TITLE CMDDCD - DECODE COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ;
513 ; MODULE NAME:
514 ; CMDDCD
515 ;
516 ; FUNTIONAL DESCRIPTION:
517 ; DECODES THE CONTENTS OF THE DECODE BUFFER AND
518 ; RETURNS A COMMAND CODE
519 ;
520 ; INPUTS:
521 ; DECODE BUFFER ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ; NONE
525 ;
526 ; OUTPUTS:
527 ; COMMAND CODE, UPDATED DECODE BUFFER POINTER
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; NONE
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ; NONE
537 ;
538 ; FUNCTIONAL SIDE EFFECTS:
539 ; NONE
540 ;
541 ; CALLING SEQUENCE:
542 ; CALL CMDDCD IN <A> OUT <B,C>
543 ; A=DECODE BUFFER ADDRESS
544 ; B=UPDATED DECODE BUFFER POINTER
545 ; C=COMMAND CODE
546 ;
547 ; VERSION:
548 ; 0.0
549 ; EDIT BY DATE REASON
550 ;
551 ;--
552 ;
```

```
554 .SBTTL COMMON DEFINITIONS AND REFERENCES
555
556
557
558 .MCALL STRUCT
559 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
560 000001 $LSTIN=1
561 000001 $LSTTAG=1
562
563
564
565 ;
566 ;*****
567 ;
568 ; REFERENCED BY OTHER MODULES
569 ;
570 .GLOBL CMDDCD ;MODULE ENTRY POINT
571 ;
572 ;*****
573 ;
574 ; GLOBAL REFERENCES
575 ;
576 .GLOBL CM.TBL ;VALID COMMAND TABLE
577 ;
578 ;*****
579 ;
```

```

581 .SBTTL CMDDCD ROUTINE
582 .EVEN
583
584 000000' ROUTINE CMDDCD <DECBUF,DBFPTR,CDCODE>
585 (2) 000000'
586
587 ;+
588 ;SAVE REGISTERS
589 ;+
590
591 000000' PUSH R1,R2
592 (2) 000000' 010146 MOV R1,-(SP)
593 (3) 000002' 010246 MOV R2,-(SP)
594
595 ;+
596 ;LET R1 BE THE POINTER FOR THE VALID COMMAND TABLE
597 ;+
598 LET R1 := #CM.YBL
599 (4) 000004' 012701 000000G MOV #CM.TBL,R1
600
601 ;+
602 ;SET THE COMMAND CODE TO 0
603 ;+
604 LET CDCODE(R5) := #0
605 (4) 000010' 005065 000004 CLR CDCODE(R5)
606
607 ;+
608 ;GO THROUGH THE VALID COMMAND TABLE UNTIL THE END
609 ;+
610 REPEAT
611 (3) 000014' 50002$:
612
613 ;+
614 ;LET R2 BE THE DECODE BUFFER POINTER
615 ;+
616 LET R2 := DECBUF(R5)
617 (4) 000014' 016502 000000 MOV DECBUF(R5),R2
618
619 ;+
620 ;FIND THE FIRST CHARACTER IN THE DECODE BUFFER THAT
621 ;IS NOT A SPACE
622 ;+
623 WHILEB (R2) EQ #SPACE DO
624 (4) 000020' 50003$:
625 (6) 000020' 121227 000040 CMPB (R2),#SPACE
626 (9) 000024' 001002 BNE 50004$
627 (6) 000026' 005202 INC R2
628 000030' ENDDO

```



```

(4) 000030' 000773
(3) 000032'
626
627
628          ;+
629          ;COMPARE CHARACTERS OF VALID COMMAND TO CHARACTERS IN
630          ;DECODE BUFFER UNTIL A MISMATCH IS FOUND
631          ;-
632          WHILEB (R1) EQ (R2) DO
(4) 000032'
(6) 000032' 121112
(9) 000034' 001003
633          LET R1 := R1 + #1
(6) 000036' 005201
634          LET R2 := R2 + #1
(6) 000040' 005202
635          ENDDO
(4) 000042' 000773
(3) 000044'
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          IFB (R1) EQ #0 THEN
(6) 000044' 105711
(9) 000046' 001007
644
645          IFB (R2) EQ #SPACE ORB (R2) EQ #CR THEN
(6) 000050' 121227 000040
(8) 000054' 001403
(6) 000056' 121227 000015
(9) 000062' 001001
(6) 000064'
646          INLINE <BR      1$>
(2) 000064' 000415
647          ENDIF
(4) 000066'
648
649          ENDIF
(4) 000066'
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          WHILEB (R1) NE #0 DO
(4) 000066'
(6) 000066' 105711
(9) 000070' 001402
657          LET R1 := R1 + #1
(6) 000072' 005201
658          ENDDO

```

```

BR      50003$
50004$:
50005$:
CMPB   (R1),(R2)
BNE    50006$
INC    R1
INC    R2
BR      50005$
50006$:
TSTB   (R1)
BNE    50007$
CMPB   (R2),#SPACE
BEQ    50010$
CMPB   (R2),#CR
BNE    50011$
50010$:
BR      1$
50011$:
50007$:
50012$:
TSTB   (R1)
SEQ    50013$
INC    R1

```

659	(4) 000074' 000774						
660	(3) 000076'				50013\$:	BR	50012\$
661	000076'			LET CDCODE(R5) := CDCODE(R5) + #1			
662	(6) 000076' 005265 000004					INC	CDCODE(R5)
663	000102'			LET R1 := R1 + #1			
664	(6) 000102' 005201					INC	R1
665				;++			
666				;END OF TABLE YET?			
667				;-			
668							
669	000104'			UNTILB (R1) EQ #-1			
670	(3) 000104' 121127 177777					CMPB	(R1), #-1
671	(6) 000110' 001341					BNE	50002\$
672				;++			
673				;YES - ILLEGAL COMMAND			
674				;SET COMMAND CODE TO -1			
675				;-			
676	000112'			LET CDCODE(R5) := #-1			
677	(4) 000112' 012765 177777 000004					MOV	#-1, CDCODE(R5)
678				;++			
679				;SAVE THE DECODE BUFFER POINTER			
680				;-			
681							
682	000120'			INLINE <1\$:>			
683	(2) 000120'					1\$:	
684	000120'			LET DBFPTR(R5) := R2			
685	(4) 000120' 010265 000002					MOV	R2, DBFPTR(R5)
686				;++			
687				;SAVE REGISTERS AND RETURN			
688				;-			
689							
690	000124'			POP R2, R1			
691	(2) 000124' 012602					MOV	(SP)+, R2
692	(3) 000126' 012601					MOV	(SP)+, R1
693							
694				ENDRTN			
695	(3) 000130'				50000\$:		
696	(3) 000130'				50001\$:		
697	(2) 000130' 000207					RTS	PC
698				;++			
699	000001			;VALID COMMAND TABLE			
				;-			
				.END			

ACSR = 000102	CR = 000015	ECCSTA= 000010	MED = 076600	RBUFEA= 000130
ACTBIT= 004000	CSRA = 000100	ENBEDP= 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	CTRLD = 000017	EOPBIT= 000001	MODSEL= 001000	RDSERVA= 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	EVNTPD= 000202	MSGECH= 177777	RES1 = 000056
AUTOST= 020000	DIAGMC= 000000	EVNTRE= 000201	MSGGEP= 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	RVALU= 0001740
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	INHRR= 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	ICMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT12 = 010000	DT.KBE= 000024	ICMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT13 = 020000	DT.KBP= 000026	ICMODR= 112000	NBKMOD= 001000	SC.HLD= 000010
BIT14 = 040000	DT.KBR= 000022	ICMODX= 110000	NCPUPQ= 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
CQDVF = 000001	ECCMEM= 000100	MAPSTA= 000200	RANNUM= 000054	TMPID = 000002

TQDVF = 000002	WBUFEA= 000136	\$FSFAL= 000405	\$LOCTA= 177777	\$\$DST = 000000
UIPAR0= 177640	WBUFPA= 000134	\$FSGOD= 000400	\$LSTIN= 000001	\$\$ELOC= 000402
UIPAR1= 177642	WBUFRQ= 000140	\$FSIF = 000110	\$LSTTA= 000001	\$\$ERFL= 000000
UIPAR2= 177644	WBUFSZ= 000142	\$FSINC= 000210	\$NESTL= 177777	\$\$FLAG= 000340
UIPAR3= 177646	WDFR = 000116	\$FSLOO= 000200	\$NSK0 = 000300	\$\$FROM= 000000
UIPAR4= 177650	WDT0 = 000114	\$FSNAM= 000160	\$NSK1 = 000130	\$\$LOC = 000110R
UIPAR5= 177652	WTINRE= 000352	\$FSNO = 000403	\$NSK2 = 000120	\$\$LQCN= 000000
UIPAR6= 177654	WTWHMI= 000222	\$FSOR = 000320	\$NSK3 = 000110	\$\$REG = 177777
UIPAR7= 177656	XFLAG = 000005	\$FSRTI= 000350	\$SAVLE= 177777	\$\$RETU= 000000
UIPDR0= 177600	XOFF = 000023	\$FSRTN= 000300	\$\$SSK0 = 050013	\$\$RTN1= 050000
UIPDR1= 177602	XON = 000021	\$FSSEL= 000140	\$TAGLE= 177777	\$\$RTN2= 050001
UIPDR2= 177604	\$BGNLE= 177777	\$FSTHE= 000330	\$TAGNU= 050014	\$\$SRC = 000000
UIPDR3= 177606	\$ERFLG= 000400	\$FSTRU= 000404	\$TEMP = 000300	\$\$TGSV= 000000
UIPDR4= 177610	\$FSAND= 000310	\$FSUNT= 000130	\$TSK0 = 050002	\$\$TGS1= 000000
UIPDR5= 177612	\$FSBAD= 000401	\$FSWHI= 000120	\$TSK1 = 050012	\$\$TGS2= 000000
UIPDR6= 177614	\$FSBLA= 000170	\$FSYES= 000402	\$TSK2 = 050013	\$\$TO = 000000
UIPDR7= 177616	\$FCAS = 000150	\$IFLEV= 177777	\$\$ARGC= 000006	\$\$TAG = 050000
WASADR= 000104	\$F\$DEC= 000220	\$ISKO = 000001	\$\$BYTE= 000402	. = 000132R
WBSTAT= 000040	\$F\$DD = 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  
CCRE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:46  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0342

3 COMMON EQUATE MODULE  
553 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
622 CMDPRC ROUTINE

```
508 .TITLE CMDPRC - PROCESS KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     CMDPRC
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CONTROLS THE DECODING AND PROCESSING OF KEYBOARD COMMANDS
517 ;     BY IDENTIFYING THE COMMAND AND THEN CALLING THE PROPER
518 ;     SERVICE ROUTINE
519 ;
520 ; INPUTS:
521 ;     DATA TABLE ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ;     NONE
525 ;
526 ; OUTPUTS:
527 ;     NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ;     NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ;     NONE
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ;     CMDCPY, CMDDCD, CMDRST, ALL COMMAND SERVICE ROUTINES
537 ;     SAVRES
538 ;     RESREG
539 ;
540 ; FUNCTIONAL SIDE EFFECTS:
541 ;     NONE
542 ;
543 ; CALLING SEQUENCE:
544 ;     CALL CMDPRC IN <A>
545 ;         A=ADDRESS OF DATA TABLE
546 ;
547 ; VERSION:
548 ;     0.0
549 ;
550 ;
551 ;--
```

EDIT	BY	DATE	REASON
------	----	------	--------

```
553 .SBTTL COMMON DEFINITIONS AND REFERENCES
554
555
556
557 .MCALL STRUCT
558 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
559 000001 $LSTIN=1
560 000001 $LSTTAG=1
561
562
563
564 ;
565 ;*****
566 ;
567 ; REFERENCED BY OTHER MODULES
568 ;
569 .GLOBL CMDPRC ;MODULE ENTRY POINT
570 .GLOBL CM.BADNAME ; - MESSAGES
571 .GLOBL CM.ARG ;
572 .GLOBL CM.NUM ;
573 .GLOBL CM.ODD ;
574 .GLOBL CM.RUN ;
575 .GLOBL CM.ADR ;
576 .GLOBL CM.NBAD ;
577 ;
578 ;*****
579 ;
580 ; GLOBAL REFERENCES
581 ;
582 .GLOBL CMDCPY ;COMMAND COPY MODULE ENTRY POINT
583 .GLOBL CMDDCD ;COMMAND DECODE MODULE ENTRY POINT
584 .GLOBL CMDRST ;COMMAND RESET MODULE ENTRY POINT
585 .GLOBL CM.DIS ;COMMAND DISPATCH TABLE
586 .GLOBL SAVREG
587 .GLOBL RESREG
588 ;
589 ;*****
590 ;
591 ;*****
592 ;
593 ; LOCAL EQUATES:
594 ;
595 000122 CM.BUFSIZ=^D<82> ;BUFFER SIZE 82 (10) BYTES
596 ;MUST ALWAYS=KEYBD. BUF SIZE
597 ;
598 ;
599 ;*****
600 ;
601 ; LOCAL STORAGE:
602 ;
603 000000' 000000 CM.ABORT: .WORD 0 ;ABORT FLAG
604 000002' 000000 CM.CDCODE: .WORD 0 ;COMMAND CODE
605 000004' 000000 CM.DBFPTR: .WORD 0 ;DECODE BUFFER POINTER
606 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 ;
```



622				.SBTTL CMDPRC ROUTINE		
623	000412'			ROUTINE CMDPRC <TABL>		
(2)	000412'					CMDPRC:
624						
625				;		
626				SAVE REGISTERS		
627				;		
628	000412'			CALL SAVREG		
(3)	000412'	004767	000000G			JSR PC, SAVREG
629						
630				;		
631				SET R0 TO START OF DTABLE		
632				;		
633	000416'			LET R0 := TABL(R5)		
(4)	000416'	016500	000000			MOV TABL(R5), R0
634						
635				;		
636				CLEAR THE DECODE BUFFER AND LOCAL STORAGE WORDS		
637				;		
638						
639						
640	000422'			LET R1 := #CM.DECBUF		
(4)	000422'	012701	000265'			MOV #CM.DECBUF, R1
641	000426'			LET R2 := R1 + #CM.BUFSIZ		
(4)	000426'	010102				MOV R1, R2
(6)	000430'	062702	000122			ADD #CM.BUFSIZ, R2
642	000434'			WHILEB R1 NE R2 DO		
(4)	000434'					50002\$:
(6)	000434'	120102				CMPB R1, R2
(9)	000436'	001402				BEQ 50003\$
643	000440'			LET (R1)+ :B= #0		
(4)	000440'	105021				CLRB (R1)+
644	000442'			ENDDO		
(4)	000442'	000774				BR 50002\$
(3)	000444'					50003\$:
645	000444'			LET CM.ABORT := #0		
(4)	000444'	005067	177330			CLR CM.ABORT
646	000450'			LET CM.CDCODE := #0		
(4)	000450'	005067	177326			CLR CM.CDCODE
647	000454'			LET CM.DBFPTR := #0		
(4)	000454'	005067	177324			CLR CM.DBFPTR
648						
649				;		
650				COPY INPUT BUFFER INTO DECODE BUFFER		
651				;		
652						
653	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
654						
655				;		

```

656 ; IF ABORT FLAG NOT SET, DECODE COMMAND
657 ; -
658
659 000506' IF CM.ABORT NE #1 THEN
(6) 000506' 026727 177266 000001 CMP CM.ABORT,#1
(9) 000514' 001436 BEQ 50004$
660
661
662
663 000516' 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' MJV #CM.DECBUF,-(R5)
(3) 000530' 004767 000000G JSR PC,CMDDCD
(3) 000534' 012605 MJV (SP)+,R5
(4) 000536' 012567 177242 MOV (R5)+,CM.DBFPTR
(4) 000542' 012567 177234 MOV (R5)+,CM.CDCODE
664
665 ; +
666 ; IF THERE WAS AN ERROR (CM.CDCODE=-1), THEN OUTPUT THE DECODE BUFFER
667 ; ALONG WITH AN ERROR MESSAGE. IF THE COMMAND
668 ; WAS VALID, DISPATCH CONTROL TO THE PROPER ROUTINE.
669 ; -
670
671 000546' IF CM.CDCODE EQ #-1 THEN
(6) 000546' 026727 177230 177777 CMP CM.CDCODE,#-1
(9) 000554' 001004 BNE 50005$
672
673 000556' LET DT.KBRSP(R0) := #CM.INVCMD
(4) 000556' 012760 000243' 000022 MOV #CM.INVCMD,DT.KB
674
675 000564' ELSE
(4) 000564' 000412 BR 50006$
(3) 000566'
676 000566' LET R1 := CM.CDCODE SHIFT #+1
(4) 000566' 016701 177210 MOV CM.CDCODE,R1
(7) 000572' 006301 ASL R1
677 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
678 000612' ENDIF
(4) 000612' 50006$:
679 000612' ENDIF
(4) 000612' 50004$:
680
681
682
683 ; +
684 ; CALL COMMAND RESET MODULE WHICH WILL OUTPUT PROMPT
685 ; -
686
687 000612' CALL CMDRST IN <R0>
(3) 000612' 010546 MOV R5,-(SP)

```

CMDPRC - PROCESS KEYBOARD COMMAND  
CMDPRC.MAC 06-SEP-78 14:32

MACY11 30A(1052) 20-SEP-78 17:46 PAGE 19-5  
CMDPRC ROUTINE

SEQ 0348

(4) 000614' 010045  
(3) 000616' 004767 000000G  
(3) 000622' 012605  
688  
689  
590  
691  
692  
693 000624'  
(3) 000624' 004767 000000G  
694  
695  
696  
697  
698  
699 000630'  
(3) 000630'  
(3) 000630'  
(2) 000630' 000207  
700 000001

;  
;+  
;RESTORE REGISTERS  
;-  
CALL RESREG  
;  
;+  
;RETURN  
;-  
ENDRTN  
.  
END

MOV R0,-(R5)  
JSR PC,CMDRST  
MOV (SP)+,R5

JSR PC,RESREG

500005:  
50001\$:

RTS PC

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	PRO = 000000
ASB = 000106	CM.DEC 000265R	DT.SWR= 000056	KTERR0= 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 = 177775
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	ENBEOB= 010000	MEMPAS= 040000	RBUFPA= 000126
BIT02 = 000004	CR = 000015	ENBNUL= 000001	MGDEXH= 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	CTRLD = 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	EVNTPB= 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	INHRE= 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	NCPUCP= 000020	SC.CLO= 000000
CAPRES= 000004	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SC.CLD= 000000
CASSTAT= 000004	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.HLD= 000010
CDERCT= 000146	DT.MTI= 000110	KIPAR1= 172342	QWEN = 024020	SC.SCA= 000012
CDWDCT= 000144	DT.DFF= 000070	KIPAR2= 172344	PAERR = 000010	SENCLS= 177777
CKTIM = 100000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SOFcnt= 000042
CLKPRE= 000001	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SCFPAS= 000046
CMDCPY= ***** G	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPACE = 000040
CMDDCD= ***** G	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SPOINT= 000032
CMDPRC 000412RG	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SPVALU= 002200
CMDRST= ***** G	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SRO = 177572
CM.ABO 000000R	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	SR1 = 177574
				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	\$\$ELDC= 000402
SUSPND= 000001	UIPDR1= 177602	\$F\$AND= 000310	\$IFLEV= 177777	\$\$SERFL= 000000
SVR0 = 000062	UIPDR2= 177604	\$F\$BAD= 000401	\$ISKO = 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	\$\$REU= 000000
SVRS = 000076	WASADR= 000104	\$F\$GDO= 000400	\$NSKO = 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\$LDD= 000200	\$\$AVLE= 177777	\$\$TGSV= 000000
TMPID = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	\$\$SKO = 050003	\$\$TGS1= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$ND = 000403	\$TAGLE= 177777	\$\$TGS2= 000000
UIPAR0= 177640	WDFR = 000116	\$F\$DR = 000320	\$TAGNU= 050007	\$\$TD = 000000
UIPAR1= 177642	WDTQ = 000114	\$F\$RTI= 000350	\$TEMP = 000300	\$\$\$TAG= 050000
UIPAR2= 177644	WTINRE= 000352	\$F\$RTN= 000300	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:46  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0351

3 COMMON EQUATE MODULE  
549 COMMON DEFINITIONS AND REFERENCES  
554 000000' .PRINT ;SPMAC: VERSION 1.1  
576 CMDRST ROUTINE

```
508 .TITLE CMDRST - KEYBOARD RESET
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     CMDRST
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     OUTPUTS A PROMPT TO THE KEYBOARD
517 ;
518 ; INPUTS:
519 ;     ADDRESS OF DATA TABLE
520 ;
521 ; IMPLICIT INPUTS:
522 ;     DT.STO
523 ;
524 ; OUTPUTS:
525 ;     NONE
526 ;
527 ; IMPLICIT OUTPUTS:
528 ;     DT.KBPRM
529 ;
530 ; PATHOLOGICAL CONNECTIONS:
531 ;     NONE
532 ;
533 ; SUBORDINATE MODULES CALLED:
534 ;     KBINI
535 ;
536 ; FUNCTIONAL SIDE EFFECTS:
537 ;     NONE
538 ;
539 ; CALLING SEQUENCE:
540 ;     CALL CMDRST IN <A>
541 ;         A=ADDRESS OF DATA TABLE
542 ;
543 ; VERSION:
544 ;     0.0
545 ;
546 ;     EDIT          BY          DATE          REASON
547 ;--
```

```
549          .SBTTL  COMMON DEFINITIONS AND REFERENCES
550
551
552
553          .MCALL STRUCT
554 000000'    STRUCT
(1) 000000'    .PRINT  ;SPMAC: VERSION 1.1
555          000001    $LSTIN=1
556          000001    $LSTTAG=1
557
558
559          ;
560          ;*****
561          ;
562          ; REFERENCED BY OTHER MODULES
563          ;
564          .GLOBL  CMDRST          ;MODULE ENTRY POINT
565          ;
566          ;*****
567          ;
568          ; LOCAL STORAGE
569          ;
570 000000' 051502 037131 000 CM.RUNMSG:  .ASCIZ /BSY>/          ;RUN MODE PROMPT
571 000005' 103 042115 000076 CM.CMDMSG:  .ASCIZ /CMD>/          ;COMMAND MODE PROMPT
572          ;
573          ;*****
574          ;
```



```

576          .SBTTL CMDRST ROUTINE
577
578 000012'    ROUTINE CMDRST <TABL>
579 (2) 000012'
580
581          ;+
582          ;SET R0 TO START OF DATA TABLE
583          ;-
584 000012'    PUSH R0
585 (2) 000012' 010046
586 000014'    LET R0 := TABL(R5)
587 (4) 000014' 016500 000000
588          ;+
589          ;IF IN RUN MODE, OUTPUT RUN PROMPT
590          ;-
591
592 000020'    IF #RUNMOD SETIN DT.STO(R0) THEN
593 (6) 000020' 032760 100000 000010
594 (9) 000026' 001404
595          LET DT.KBPRM(R0) := #CM.RUNMSG
596 (4) 000030' 012760 000000' 000026
597          ;+
598          ;ELSE OUTPUT COMMAND MODE PROMPT
599          ;-
600          ELSE
601 (4) 000036' 000403
602 (3) 000040'
603          LET DT.KBPRM(R0) := #CM.CMDMSG
604 (4) 000040' 012760 000005' 000026
605          ENDIF
606 (4) 000046'
607          POP R0
608 (2) 000046' 012600
609          ENDRTN
610 (3) 000050'
611 (3) 000050'
612 (2) 000050' 000207
613          .END
614 000001'

```

CMDRST:

```

MOV      R0,-(SP)
MOV      TABL(R5),R0
BIT      #RUNMOD,DT.STO(R
BEQ      50002$
MOV      #CM.RUNMSG,DT.KB
BR       50003$
50002$:
MOV      #CM.CMDMSG,DT.KB
50003$:
MOV      (SP)+,R0
50000$:
50001$:
RTS     PC

```

ACSR = 000102	CR = 000015	ENBNUL= 000001	MODEXH= 004000	RBUFV= 000132
ACTBIT= 004000	CSRA = 000100	ENDLST= 000000	MODHQL= 002000	RBUFVA= 000124
ADDR22= 001000	CSRC = 000102	EQPBIT= 000001	MODSEL= 001000	RDSERV= 000101
ADR = 000006	CTRLC = 000003	ERRTYP= 000106	MSGCKD= 000010	RDWHMI= 000022
APTFER= 000004	CTRLD = 000017	EVNTBE= 000200	MSGCKS= 000011	RELERR= 000020
APTPRE= 000200	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RELMDD= 020000
ASB = 000106	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RELTIM= 010000
ASSEMB= 000010	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	RES1 = 000056
ASTAT = 000104	DIAGMC= 000000	EVNTRE= 000201	MSGEOP= 000013	RES2 = 000060
AUTO = 000010	DROPMO= 100000	FATERR= 100000	MSGHDR= 000004	RICHAR= 031060
AUTOJST= 020000	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RPTDAT= 002000
AWAS = 000110	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RSTRT = 000112
BIT0 = 000001	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RUBOUT= 000177
BIT00 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUNMOD= 100000
BIT01 = 000002	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RSVALU= 001740
BIT02 = 000004	DT.CF0= 000014	IE = 000100	MSGPRM= 177776	SAM = 075464
BIT03 = 000010	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SBADR = 000102
BIT04 = 000020	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBKMOD= 000000
BIT05 = 000040	DT.ESI= 000044	INHEPR= 020000	MSGSKE= 000003	SBKSEL= 010000
BIT06 = 000100	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SC.ADR= 000006
BIT07 = 000200	DT.EXS= 000060	INHRR= 000400	MSGSMH= 000014	SC.ALC= 000014
BIT08 = 000400	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.APC= 000016
BIT09 = 001000	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.CKL= 000002
BIT1 = 000002	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT10 = 002000	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT11 = 004000	DT.KBP= 000026	IOMODR= 112000	NEKMOD= 001000	SC.HLD= 000010
BIT12 = 010000	DT.KBR= 000022	IOMODX= 110000	NCPUPP= 000020	SC.SCA= 000012
BIT13 = 020000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SENDLS= 177777
BIT14 = 040000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SQFCNT= 000042
BIT15 = 100000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SQFPAS= 000046
BIT2 = 000004	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT3 = 000010	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPOINT= 000032
BIT4 = 000020	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPVALU= 002200
BIT5 = 000040	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SR0 = 177572
BIT6 = 000100	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SR1 = 177574
BIT7 = 000200	DT.PTA= 000064	KIPAR7= 172356	POP60 = 004000	SR2 = 177576
BIT8 = 000400	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
BIT9 = 001000	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	STAT = 000026
BKDEF = 000002	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	STATBI= 064757
BKMOD = 000020	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STAT1 = 000027
BKMODE= 040000	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	SUSPND= 000001
BKSLSH= 000134	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	SVR0 = 000062
CAPRES= 000004	DT.ST0= 000010	KIPDR6= 172314	PRI7 = 000340	SVR1 = 000064
CASTAT= 000004	DT.ST1= 000012	KIPDR7= 172316	PRO = 000000	SVR2 = 000066
CDERCT= 000146	DT.SWR= 000056	KTERRO= 000040	PR4 = 000200	SVR3 = 000070
CDWDCT= 000144	DT.SYP= 000072	KTPRES= 000400	PR5 = 000240	SVR4 = 000072
CKTIM = 100000	DT.WBU= 000050	KTSTAT= 000020	PR6 = 000300	SVR5 = 000074
CLKPRE= 000001	DT.WHL= 000054	KTXTND= 040000	PR7 = 000340	SVR6 = 000076
CMDRST 000012RG	DT.WLL= 000052	LF = 000012	PS = 177776	SYSCNT= 000052
CM.CMD 000005R	DVID1 = 000014	LPSTAT= 000001	PSW = 177776	YSERR= 000100
CM.RUN 000000R	ECCMEM= 000100	MAPSTA= 000200	RANUM= 000054	TABL = 000000
CONFIG= 000056	ECCSTA= 000010	MED = 076500	RBUFEA= 000130	TMPID = 000002
CQOVF = 000001	ENBEOP= 010000	MEMPAS= 040000	RBUFPA= 000126	TQOVF = 000002

UIPAR0= 177640	WBUFEA= 000136	\$F\$DO = 000340	\$ISKO = 000001	\$\$FLAG= 000001
UIPAR1= 177642	WBUFPA= 000134	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$FROM= 000000
UIPAR2= 177644	WBUFRQ= 000140	\$F\$GDO= 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\$LDO= 000200	\$NSKO = 000300	\$\$RETU= 000000
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR7= 177656	WTWHMI= 000222	\$F\$ND = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:47  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0357

3 COMMON EQUATE MODULE

CMDTB1 - COMMAND TABLE 1  
CMDTB1.MAC 28-JUL-78 09:13

MACY11 30A(1052) 20-SEP-78 17:47 PAGE 19  
COMMON EQUATE MODULE

SEQ 0358

```
508 .TITLE CMDTB1 - COMMAND TABLE 1
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     CMDTB1
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CONTAINS THE COMPLETE VALID COMMAND AND COMMAND
517 ;     DISPATCH TABLES
518 ;
519 ; VERSION:
520 ;     0.0
521 ;
522 ;     EDIT             BY             DATE             REASON
523 ;
524 ;--
```

```

526
527 ;
528 ;*****
529 ;
530 ; REFERENCED BY OTHER MODULES
531 ;
532 .GLOBL CM.DIS
533 .GLOBL CM.TBL
534 ;
535 ;*****
536 ;
537 ; GLOBAL REFERENCES
538 ;
539 .GLOBL KCOFF ;CACHE OFF ROUTINE
540 .GLOBL KCON ;CACHE ON ROUTINE
541 .GLOBL KDES ;DESELECT ROUTINE
542 .GLOBL KEXAM ;EXAM ROUTINE
543 .GLOBL KFILL ;FILL ROUTINE
544 .GLOBL KKTOFF ;KT OFF ROUTINE
545 .GLOBL KKTON ;KT ON ROUTINE
546 .GLOBL KLPOFF ;LP OFF ROUTINE
547 .GLOBL KLPON ;LP ON ROUTINE
548 .GLOBL KMAP ;MAP ROUTINE
549 .GLOBL KMOD ;MODIFY ROUTINE
550 .GLOBL KMOFF ;UNIBUS MAP OFF ROUTINE
551 .GLOBL KMON ;UNIBUS MAP ON ROUTINE
552 .GLOBL KPOFF ;PARITY OFF ROUTINE
553 .GLOBL KPON ;PARITY ON ROUTINE
554 .GLOBL KROTOF ;WRITE BUFFER ROTATION OFF ROUTINE
555 .GLOBL KROTON ;WRITE BUFFER ROTATION ON ROUTINE
556 .GLOBL KRUN ;RUN ROUTINE
557 .GLOBL KRUNL ;RUN LOCKED ROUTINE
558 .GLOBL KSEL ;SELECT ROUTINE
559 .GLOBL KSUM ;SUMMARY ROUTINE
560 .GLOBL KSWR ;MODIFY SWITCH REGISTER ROUTINE
561 ;
562 ;*****
563 ;
564
565 000000' CM.DIS: ;COMMAND DISPATCH TABLE
566 000000' 000000G .WORD KCOFF
567 000002' 000000G .WORD KCON
568 000004' 000000G .WORD KDES
569 000006' 000000G .WORD KEXAM
570 000010' 000000G .WORD KFILL
571 000012' 000000G .WORD KKTOFF
572 000014' 000000G .WORD KKTON
573 000016' 000000G .WORD KLPOFF
574 000020' 000000G .WORD KLPON
575 000022' 000000G .WORD KMAP
576 000024' 000000G .WORD KMOD
577 000026' 000000G .WORD KMOFF
578 000030' 000000G .WORD KMON
579 000032' 000000G .WORD KPOFF
580 000034' 000000G .WORD KPON
581 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 /CDN/  
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	ERRTP= 000106	KMON = ***** G	PDPLSI= 020000
ACTBIT= 004000	CTRLC = 000003	EVNTBE= 000200	KPOFF = ***** G	PDP60 = 004000
ADDR22= 001000	CTRLD = 000017	EVNTHD= 000200	KPON = ***** G	PDP70 = 010000
ADR = 000006	CTRLU = 000025	EVNKT= 000203	KROTOF= ***** G	PRI0 = 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	PRC = 000000
AWAS = 000110	DT.APK= 000076	ICOUNT= 000040	KTERRO= 000040	PR4 = 000200
BIT0 = 000001	DT.BLS= 000034	IDNUM = 000122	KTPRES= 000400	PR5 = 000240
BIT01 = 000002	DT.CF0= 000014	IE = 000100	KTSTAT= 000020	PR6 = 000300
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	KTXTND= 040000	PR7 = 000340
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	LF = 000012	PS = 177776
BIT04 = 000020	DT.EI= 000044	INHEPR= 020000	LPSTAT= 000001	PSW = 177776
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MAPSTA= 000200	RANNUM= 000054
BIT06 = 000100	DT.EXS= 000060	INHRR= 000400	MED = 075600	RBUFEA= 000130
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MEMPAS= 040000	RBUFPA= 000126
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MODEXH= 004000	RBUFSZ= 000132
BIT09 = 001000	DT.HMX= 000104	ICMOD = 100000	MODHOL= 002000	RBUFVA= 000124
BIT1 = 000002	DT.KBE= 000024	ICMODP= 102000	MODSEL= 001000	RDSERV= 000101
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	MSGCKD= 000010	RDWHMI= 000022
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	MSGCKS= 000011	RELERR= 000020
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	MSGDER= 000005	RELMOD= 020000
BIT13 = 020000	DT.MLS= 000032	KCOFF = ***** G	MSGDRP= 000017	RELTIM= 010000
BIT14 = 040000	DT.MTI= 000110	KCON = ***** G	MSGECH= 177777	RES1 = 000056
BIT15 = 100000	DT.OFF= 000070	KDES = ***** G	MSGEOP= 000013	RES2 = 000060
BIT2 = 000004	DT.PAS= 000074	KEXAM = ***** G	MSGHDR= 000004	RICHAR= 031060
BIT3 = 000010	DT.PC = 000002	KFILL = ***** G	MSGHNG= 000022	RPTDAT= 002000
BIT4 = 000020	DT.PFL= 000062	KIPAR0= 172340	MSGHRD= 000007	RSTR = 000112
BIT5 = 000040	DT.PSW= 000004	KIPAR1= 172342	MSGMAP= 000021	RUBOUT= 000177
BIT6 = 000100	DT.PTA= 000064	KIPAR2= 172344	MSGNUL= 177775	RUNMOD= 100000
BIT7 = 000200	DT.RCS= 000102	KIPAR3= 172346	MSGPOP= 000002	R5VALU= 001740
BIT8 = 000400	DT.REL= 000040	KIPAR4= 172350	MSGPRM= 177776	SAM = 075464
BIT9 = 001000	DT.SCT= 000066	KIPAR5= 172352	MSGRES= 000001	SBADR = 000102
BKDEF = 000002	DT.SMX= 000106	KIPAR6= 172354	MSGSFT= 000006	SBKMOD= 000000
BKMOD = 000020	DT.SP = 000006	KIPAR7= 172356	MSGSKE= 000003	SBKSEL= 010000
BKMODE= 040000	DT.SSI= 000046	KIPDR0= 172300	MSGSMB= 000015	SC.ADR= 000006
BKSLSH= 000134	DT.ST0= 000010	KIPDR1= 172302	MSGSMH= 000014	SC.ALC= 000014
CAPRES= 000004	DT.ST1= 000012	KIPDR2= 172304	MSGSMS= 000016	SC.APC= 000016
CASAT= 000004	DT.SWR= 000056	KIPDR3= 172306	MSGSTD= 000000	SC.CKL= 000002
CDERCT= 000146	DT.SYP= 000072	KIPDR4= 172310	MSGSYS= 000012	SC.CKP= 000004
CDWDC= 000144	DT.WBU= 000050	KIPDR5= 172312	MSGVEC= 000020	SC.CLD= 000000
CKTIM = 100000	DT.WHL= 000054	KIPDR6= 172314	NBKMOD= 001000	SC.HLD= 000010
CLKPRE= 000001	DT.WLL= 000052	KIPDR7= 172316	NCPUOP= 000020	SC.SCA= 000012
CM.DIS 000000RG	DVID1 = 000014	KKTOFF= ***** G	NOPTY= 000002	SENDSL= 177777
CM.TBL 000054RG	ECCMEM= 000100	KKTCN = ***** G	NULL = 000000	SQFCNT= 000042
CONFIG= 000056	ECCSTA= 000010	KLPOFF= ***** G	OWEN = 024020	SQFPAS= 000046
CQQVF = 000001	ENBEDP= 010000	KLPON = ***** G	PAERR = 000010	SPACE = 000040
CR = 000015	ENBNUL= 000001	KMAP = ***** G	PARPRE= 002000	SPOINT= 000032
CSRA = 000100	ENDLST= 000000	KMOD = ***** G	PARSTA= 000100	SPVALU= 002200
	EOPBIT= 000001	KMOFF = ***** G	PASCNT= 000034	SRO = 177572



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	TMPID = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:47  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0363

3 COMMON EQUATE MODULE

CMDTB2 - COMMAND TABLE 2  
CMDTB2.MAC 28-JUL-78 09:13

MACY11 30A(1052) 20-SEP-78 17:47 PAGE 19  
COMMON EQUATE MODULE

SEQ 0364

```
508          .TITLE CMDTB2 - COMMAND TABLE 2
509          .IDENT /V0.0/
510
511          ;++
512          ; MODULE NAME:
513          ;         CMDTB2
514          ;
515          ; FUNCTIONAL DESCRIPTION:
516          ;         CONTAINS SHORTENED VERSIONS OF THE VALID COMMAND
517          ;         AND COMMAND DISPATCH TABLES.
518          ;
519          ;
520          ; VERSION:
521          ;         0.0
522          ;
523          ;         EDIT             BY             DATE             REASON
524          ;
525          ;--
```

```
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572
```

```
;  
;*****  
; 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  
;  
;*****  
;  
CM.DIS: ;COMMAND DISPATCH TABLE  
      .WORD KDES  
      .WORD KFill  
      .WORD KMOD  
      .WORD KRUN  
      .WORD KSEL  
      .WORD KSUM  
      .WORD KSWR  
  
CM.TBL: ;VALID COMMAND TABLE  
      .ASCIZ /DES/  
      .ASCIZ /FILL/  
      .ASCIZ /MOD/  
      .ASCIZ /RUN/  
      .ASCIZ /SEL/  
      .ASCIZ /SUM/  
      .ASCIZ /SWR/  
      .BYTE -1  
  
.EVEN  
.END  
000001
```

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

SYSERR= 000100	UIPAR4= 177650	UIPDR3= 177605	WBUFEA= 000136	WTWHMI= 000222
TMPIO = 000002	UIPAR5= 177652	UIPDR4= 177610	WBUFPA= 000134	XFLAG = 000005
TQDVF = 000002	UIPAR6= 177654	UIPDR5= 177612	WBUFRQ= 000140	XOFF = 000023
UIPAR0= 177640	UIPAR7= 177656	UIPDR6= 177614	WBUSFSZ= 000142	XON = 000021
UIPAR1= 177642	UIPDR0= 177600	UIPDR7= 177616	WDFR = 000116	. = 000054R
UIPAR2= 177644	UIPDR1= 177602	WASADR= 000104	WDT0 = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:47

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0368

3 COMMON EQUATE MODULE

CMDTB3 - COMMAND TABLE 3  
CMDTB3.MAC 28-JUL-78 09:13

MACY11 30A(1052) 20-SEP-78 17:47 PAGE 19  
COMMON EQUATE MODULE

SEQ 0369

```
508 .TITLE CMDTB3 - COMMAND TABLE 3
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     CMDTB3
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CONTAINS THE VALID COMMAND AND COMMAND
517 ;     DISPATCH TABLES EXCEPT:
518 ;     CON, COFF, KTON, KTOFF, MON, MOFF, RUNL
519 ;
520 ; VERSION:
521 ;     0.0
522 ;
523 ;     EDIT             BY             DATE             REASON
524 ;
525 ;--
```



```

527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582

```

```

;
;*****
;
; 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 KRTOF ;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
;
;*****
;
;*****
;
CM.DIS: ;COMMAND DISPATCH TABLE
.WORD KDES
.WORD KEXAM
.WORD KFILL
.WORD KLPOFF
.WORD KLPON
.WORD KMAP
.WORD KMOD
.WORD KPOFF
.WORD KPON
.WORD KRTOF
.WORD KROTON
.WORD KRUN
.WORD KSEL
.WORD KSUM
.WORD KSWR

CM.TBL: ;VALID COMMAND TABLE
.ASCIZ /DES/
.ASCIZ /EXAM/
.ASCIZ /FILL/
.ASCIZ /LPPOFF/
.ASCIZ /LPON/

```

```

000000'
000000' 000000G
000002' 000000G
000004' 000000G
000006' 000000G
000010' 000000G
000012' 000000G
000014' 000000G
000016' 000000G
000020' 000000G
000022' 000000G
000024' 000000G
000026' 000000G
000030' 000000G
000032' 000000G
000034' 000000G

000036' 042504 000123
000042' 054105 046501 000
000047' 106 046111 000114
000054' 050114 043117 000106
000062' 050114 047117 000

```

CMDTB3 - COMMAND TABLE 3  
CMDTB3.MAC 28-JUL-78 09:13

MACY11 30A(1052) 20-SEP-78 17:47 PAGE 19-2  
COMMON EQUATE MODULE

SEQ 0371

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	CTRLD = 000017	EVNTHD= 000200	KSWR = ***** G	PRO = 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= 100030	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
AUTO3T= 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	INHRR= 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	MSGECP= 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	NDAPTY= 000002	SENDLS= 177777
BKSLSH= 000134	DT.ST1= 000012	KIPDR4= 172310	NULL = 000000	SQFCNT= 000042
CAPRES= 000004	DT.SWR= 000056	KIPDR5= 172312	OWEN = 024020	SQFPAS= 000046
CASAT= 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	SRO = 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	TMPID = 000002	UIPAR7= 177653	WASADR= 000104	WTWHMI= 000222
SVR1 = 000064	TQCVF = 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	WDIO = 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)

.MAIN. MACY11 GOA(1052) 20-SEP-78 17:47

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0374

3	COMMON EQUATE MODULE	
537	COMMON DEFINITIONS & REFERENCES	
540	000000'	.PRINT ;SPMAC: VERSION 1.1
573	CQINI (FUNCTIONAL DESCRIPTION)	
616	CQINI (CODE)	
633	ENQCQ (FUNCTIONAL DESCRIPTION)	
683	ENQCQ (CODE)	
748	DEQCQ (FUNCTIONAL DESCRIPTION)	
797	DEQCQ (CODE)	

```
508 .TITLE CTLQUE (CONTROL QUEUE MODULES)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; CTLQUE
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE CONTAINS THREE MODULES. THE FIRST ONE
517 ; INITIALIZES THE CONTROL QUEUE AND THE NEXT TWO ACTUALLY DO THE
518 ; ENQUEING AND DEQUEING OF ITEMS FROM THE CONTROL QUEUE.
519 ; THE DESCRIPTION IS GIVEN UNDER THE RESPECTIVE MODULES.
520 ; THESE MODULES ARE:
521 ;
522 ; 1. CQINI - INITIALIZES THE CONTROL QUEUE.
523 ; 2. ENQCQ - ENQUES ITEMS ON THE CONTROL QUEUE.
524 ; 3. DEQCQ - DEQUES ITEMS FROM THE CONTROL QUEUE.
525 ;
526 ; THE CONTROL QUEUE AND THE QUEUE POINTERS ARE CONTAINED IN
527 ; THIS MODULE FOR USE BY THE ABOVE-NAMED MODULES LOCALLY.
528 ;
529 ; VERSION:
530 ; 0.0
531 ;
532 ; EDIT DATE BY REASON
533 ;--
534
535
```

```

537          .SBTTL COMMON DEFINITIONS & REFERENCES
538
539          .MCALL STRUCT
540 000000'    STRUCT
541 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
542          000001 $LSTIN=1
543          000001 $LSTTAG=1
544
545          ;*****
546          ; REFERENCED BY OTHER MODULES:
547          ;
548          .GLOBL CQINI          ;INITIALIZE CONTROL QUEUE
549          .GLOBL ENQCQ        ;ENQUE ITEMS ON THE CONTROL QUEUE
550          .GLOBL DEQCQ        ;DEQUE ITEMS FROM THE CONTROL QUEUE
551          ;
552          ; *****
553          ;
554          ; GLOBAL REFERENCES:
555          ;
556          .GLOBL GETPSW        ;GET CALLER'S PS WORD ROUTINE
557
558          .GLOBL OV.CQ         ;CONTROL QUEUE BUFFER
559          .GLOBL OV.HICQ
560          .GLOBL OV.CQSIZ
561          ;*****
562          ; LOCAL STORAGE
563          ;
564 000000' 000000 CQ.PSW: 0          ;LOC. TO SAVE CALLER'S PS WORD
565 000002' 000000G CQ.INP: OV.CQ      ;ENQUE POINTER
566 000004' 000000G CQ.OTP: OV.CQ      ;DEQUE POINTER
567 000006' 000000 CQ.CT: 0          ;CURRENT ENTRY COUNT
568 000010' 000000G CQ.HI: OV.HICQ     ;QUEUE'S HIGH ADDR
569 000012' 000000G CQ.MAX: OV.CQSIZ  ;MAX NO. OF ENTRIES ALLOWED
570 000014' 000000G CQ.ICQ: OV.CQ     ;START OF CQ
571

```

```
573          .SBTTL CQINI (FUNCTIONAL DESCRIPTION)
574
575          ;++
576          ; MODULE NAME:
577          ;     CQINI
578          ;
579          ; FUNCTIONAL DESCRIPTION:
580          ;     THIS MODULE INITIALIZES THE CONTROL QUEUE. IT RESETS THE ENQUE
581          ;     AND DEQUE POINTERS TO THE START OF THE QUEUE, AND SETS THE
582          ;     COUNT TO ZERO.
583          ;
584          ; INPUTS:
585          ;     NONE
586          ;
587          ; IMPLICIT INPUTS:
588          ;     NONE
589          ;
590          ; OUTPUTS:
591          ;     NONE
592          ;
593          ; IMPLICIT OUTPUTS:
594          ;     NONE
595          ;
596          ; PATHOLOGICAL CONNECTIONS:
597          ;     NONE
598          ;
599          ; SUBORDINATE ROUTINES CALLED:
600          ;     NONE
601          ;
602          ; FUNCTIONAL SIDE EFFECTS:
603          ;     NONE
604          ;
605          ; CALLING SEQUENCE:
606          ;     CALL CQINI
607          ;
608          ; VERSION:
609          ;     0.0
610          ;
611          ;     EDIT      DATE      BY      REASON
612          ;     -----
613          ;
614          ;
```



```
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
(2) 000016'
627 000016' LET CQ.INP := CQ.ICQ
(4) 000016' 016767 177772 177756
528 000024' LET CQ.OTP := CQ.ICQ
(4) 000024' 016767 177764 177752
629 000032' LET CQ.CT := #0
(4) 000032' 005067 177750
530 000036' ENDRTN
(3) 000036'
(3) 000036'
(2) 000036' 000207
631
```

CQINI:

```
MOV CQ.ICQ,CQ.INP
MOV CQ.ICQ,CQ.OTP
CLR CQ.CT
```

50000S:

50001S:

```
RTS PC
```

```
633 .SBTTL ENQCQ (FUNCTIONAL DESCRIPTION)
634
635 ;++
636 ; MODULE NAME:
637 ; ENQCQ
638 ;
639 ; FUNCTIONAL DESCRIPTION:
640 ; THIS MODULE HANDLES QUEUEING OF ITEMS ON THE CONTROL
641 ; QUEUE.
642 ;
643 ; INPUTS:
644 ; 1. DATA TABLE ADDRESS
645 ; 2. MODULE'S HEADER ADDRESS
646 ; 3. RETURN ADDRESS
647 ;
648 ; IMPLICIT INPUTS:
649 ; NONE
650 ;
651 ; OUTPUTS:
652 ; NONE
653 ;
654 ; IMPLICIT OUTPUTS:
655 ; DT.ERR
656 ;
657 ; PATHOLOGICAL CONNECTIONS:
658 ; NONE
659 ;
660 ; SUBORDINATE ROUTINES CALLED:
661 ; GETPSW ;GET CALLER'S PS WORD
662 ;
663 ; FUNCTIONAL SIDE EFFECTS:
664 ; ENQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
665 ; SERVES AS A LOCK MECHANISM.
666 ;
667 ; CALLING SEQUENCE:
668 ; CALL ENQCQ IN <DTADR,HDRADR,RETADR>
669 ;
670 ; WHERE DTADR = DATA TABLE ADDRESS
671 ; HDRADR = HEADER ADDRESS
672 ; RETADR = RETURN ADDRESS
673 ;
674 ; VERSION:
675 ; 0.0
676 ;
677 ; EDIT DATE BY REASON
678 ; -----
679 ;
680 ;
681 ;
```

```

683          .SBTTL ENQCQ (CODE)
684
685 000040'   ROUTINE ENQCQ <DT,HDR,RET>
(2) 000040'
686
687          ;+
688          ; SAVE R0
689          ; -
690
691 000040'   PUSH R0
(2) 000040' 010046
692
693          ;+
694          ; IF THE COUNT IS EQUAL TO MAXIMUM, THE
695          ; QUEUE IS FULL. SET ERROR FLAG IN THE
696          ; STATUS INDICATOR AND RETURN WITH ERROR.
697          ; -
698
699 000042'   IF CQ.CT GE CQ.MAX THEN
(6) 000042' 026767 177740 177742
(9) 000050' 002410
700 000052'   LET R0 := DT(R5)
(4) 000052' 016500 000000
701 000056'   LET DT.ERR(R0) := DT.ERR(R0) SET.BY #CQOVF!FATERR
(6) 000056' 052760 100001 000020
702 000064'   POP R0
(2) 000064' 012600
703 000066'   RETURN ERROR
(2) 000066' 000261
(4) 000070' 000443
704 000072'   ENDIF
(4) 000072'
705
706          ;+
707          ; SAVE CALLER'S PS WORD AND CHANGE
708          ; PRIORITY LEVEL TO 7
709          ; -
710
711 000072'   CALL GETPSW OUT <CQ.PSW>
(4) 000072' 162705 000002
(3) 000076' 004767 000000G
(4) 000102' 012567 177672
712 000106'   PUSH #PR7
(2) 000106' 012746 000340
713 000112'   PUSH #1$
(2) 000112' 012746 000120'
714 000116'   INLINE <RTI>
(2) 000116' 000002
715
716
717          ;+
718          ; IF THE ENQUE POINTER IS AT HIGH
719          ; ADDRESS, RE-INITIALIZE IT TO THE
720          ; START ADDRESS OF THE QUEUE.
721          ; -
722 000120'   INLINE <1$:>
    
```

ENQCQ:

MOV R0,-(SP)

CMP CQ.CT,CQ.MAX  
BLT 50002\$

MOV DT(R5),R0

BIS #CQOVF!FATERR,DT

MOV (SP)+,R0

SEC  
BR 50001\$

50002\$:

SUB #1\*2,R5  
JSR PC,GETPSW  
MOV (R5)+,CQ.PSW

MOV #PR7,-(SP)

MOV #1\$,-(SP)

RTI

(2)	000120'					1\$:	
723	000120'				IF CQ.INP EQ CQ.HI THEN		
(6)	000120'	026767	177656	177662		CMP	CQ.INP,CQ.HI
(9)	000126'	001003				BNE	50003\$
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		
(4)	000136'	016700	177640			MOV	CQ.INP,R0
732	000142'				LET (R0)+ := HDR(R5)		
(4)	000142'	016520	000002			MOV	HDR(R5),(R0)+
733	000146'				LET (R0)+ := RET(R5)		
(4)	000146'	016520	000004			MOV	RET(R5),(R0)+
734	000152'				LET CQ.INP := R0		
(4)	000152'	010067	177624			MOV	R0,CQ.INP
735	000156'				LET CQ.CT := CQ.CT + #1		
(6)	000156'	005267	177624			INC	CQ.CT
736							
737					:+		
738					; RESTORE CALLER'S PS WORD AND RETURN		
739					;-		
740							
741	000162'				PUSH CQ.PSW		
(2)	000162'	016746	177612			MOV	CQ.PSW,-(SP)
742	000166'				PUSH #2\$		
(2)	000166'	012746	000174'			MOV	#2\$,-(SP)
743	000172'				INLINE <RTI>		
(2)	000172'	000002				RTI	
744	000174'				INLINE <2\$:>		
(2)	000174'					2\$:	
745	000174'				POP R0		
(2)	000174'	012600				MOV	(SP)+,R0
746	000176'				ENDRTN		
(3)	000176'					50000\$:	
(2)	000176'	000241				CLC	
(3)	000200'					50001\$:	
(2)	000200'	000207				RTS	PC

```
748 .SBTTL DEQCQ (FUNCTIONAL DESCRIPTION)
749
750
751 ;++
752 ; MODULE NAME:
753 ; DEQCQ
754 ;
755 ; FUNCTIONAL DESCRIPTION:
756 ; THIS MODULE HANDLES DEQUEUEING OF ITEMS FROM THE CONTROL
757 ; QUEUE.
758 ;
759 ; INPUTS:
760 ; NONE
761 ;
762 ; IMPLICIT INPUTS:
763 ; NONE
764 ;
765 ; OUTPUTS:
766 ; 1. MODULE'S HEADER ADDRESS
767 ; 2. RETURN ADDRESS
768 ;
769 ; IMPLICIT OUTPUTS:
770 ; ERROR RETURN INDICATES QUEUE UNDERFLOW.
771 ;
772 ; PATHOLOGICAL CONNECTIONS:
773 ; NONE
774 ;
775 ; SUBORDINATE ROUTINES CALLED:
776 ; GETPSW
777 ;
778 ; FUNCTIONAL SIDE EFFECTS:
779 ; DEQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
780 ; SERVES AS A LOCK MECHANISM.
781 ;
782 ; CALLING SEQUENCE:
783 ; CALL DEQCQ OUT <HDRADR,RETADR>
784 ;
785 ; WHERE HDRADR = HEADER ADDRESS
786 ; RETADR = RETURN ADDRESS
787 ;
788 ; VERSION:
789 ; 0.0
790 ;
791 ; EDIT DATE BY REASON
792 ; -----
793 ;
794 ;
795 ;
```

```

797          .SBTTL DEQCQ (CODE)
798
799 000202'   ROUTINE DEQCQ <HDR,RET>
(2) 000202'
800
801          ;+
802          ; IF COUNT IS EQUAL TO ZERO, THE QUEUE
803          ; IS EMPTY. RETURN WITH ERROR.
804          ; -
805
805 000202'   IF CQ.CT LE #0 THEN
(6) 000202' 005767 177600
(9) 000206' 003002
807 000210'   RETURN ERROR
(2) 000210' 000261
(4) 000212' 000444
808 000214'   ENDIF
(4) 000214'
809
810          ;+
811          ; SAVE CALLER'S PS WORD AND
812          ; CHANGE PRIORITY LEVEL TO 7
813          ; -
814
815 000214'   CALL GETPSW OUT <CQ.PSW>
(4) 000214' 162705 000002
(3) 000220' 004767 000000G
(4) 000224' 012567 177550
816 000230'   PUSH #PR7
(2) 000230' 012746 000340
817 000234'   PUSH #3$
(2) 000234' 012746 000242'
818 000240'   INLINE <RTI>
(2) 000240' 000002
819
820          ;+
821          ; IF DEQUE POINTER IS AT HIGH ADDRESS,
822          ; RE-INITIALIZE IT TO THE QUEUE'S
823          ; START ADDRESS.
824          ; -
825
826 000242'   INLINE <3$:>
(2) 000242'
827 000242'   IF CQ.OTP EQ CQ.HI THEN
(6) 000242' 026767 177536 177540
(9) 000250' 001003
828 000252'   LET CQ.OTP := #OV.CQ
(4) 000252' 012767 000000G 177524
829 000260'   ENDIF
(4) 000260'
830
831          ;+
832          ; NOW DEQUEUE THE ENTRY.
833          ; -
834
835 000260'   PUSH R0
    
```

DEQCQ:

TST CQ.CT  
 BGT 50002\$  
 SEC  
 BR 50001\$

50002\$:

SUB #1\*2,R5  
 JSR PC,GETPSW  
 MOV (R5)+,CQ.PSW  
 MOV #PR7,-(SP)  
 MOV #3\$,-(SP)  
 RTI

3\$:  
 CMP CQ.OTP,CQ.HI  
 ENE 50003\$  
 MOV #OV.CQ,CQ.OTP

50003\$:

836	(2)	000260'	010046		MOV	R0,-(SP)
837	(4)	000262'	016700 177516	LET R0 := CQ.OTP	MOV	CQ.OTP,R0
838	(4)	000266'	012065 000000	LET HDR(R5) := (R0)+	MOV	(R0)+,HDR(R5)
839	(4)	000272'	012065 000002	LET RET(R5) := (R0)+	MOV	(R0)+,RET(R5)
840	(4)	000276'	010067 177502	LET CQ.OTP := R0	MOV	R0,CQ.OTP
841	(6)	000302'	005367 177500	LET CQ.CT := CQ.CT - #1	DEC	CQ.CT
842				;		
843				; RESTORE CALLER'S PS WORD AND RETURN		
844				;-		
845						
846		000306'		PUSH CQ.PSW	MOV	CQ.PSW,-(SP)
847	(2)	000312'	016746 177466	PUSH #4\$	MOV	#4\$,-(SP)
848	(2)	000316'	012746 000320'	INLINE <RTI>	RTI	
849	(2)	000320'	000002	INLINE <4\$:>	4\$:	
850	(2)	000320'		POP R0	MOV	(SP)+,R0
851	(3)	000322'	012600	ENDRTN		
852	(2)	000324'	000241		50000\$:	CLC
	(3)	000324'			50001\$:	RTS
	(2)	000324'	000207			PC
			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	PRO = 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	ENBEOB= 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	CTRL0 = 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	DR0PM0= 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	INHPR= 020000	MSGSKE= 000003	R5VALU= 001740
BIT15 = 100000	DT.FCN= 000036	INHREL= 001000	MSGSMB= 000015	SAM = 075464
BIT2 = 000004	DT.HMX= 000104	INHRR= 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	S3KSEL= 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	DV.CQ = ***** G	SC.HLD= 000010
BKMODE= 040000	DT.PFL= 000062	KIPAR2= 172344	DV.CQS= ***** G	SC.SCA= 000012
BKSLSH= 000134	DT.PSW= 000004	KIPAR3= 172346	DV.HIC= ***** G	SENDLS= 177777
CAPRES= 000004	DT.PTA= 000064	KIPAR4= 172350	OWEN = 024020	S0FCNT= 000042
CASTAT= 000004	DT.RCS= 000102	KIPAR5= 172352	PAERR = 000010	S0FPAS= 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	SRO = 177572
CONFIG= 000056	DT.SSI= 000046	KIPDR2= 172304	PDP60 = 004000	SR1 = 177574
CQINI 000016RG	DT.ST0= 000010	KIPDR3= 172306	PDP70 = 010000	SR2 = 177576
CQCWF = 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	XGN = 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	\$\$ELDC= 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	\$ISKO = 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	\$\$RETN= 000001
TMPID = 000002	WBUFEA= 000136	\$F\$GOO= 000400	\$NESTL= 177777	\$\$RTN1= 050000
TQDVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NSKO = 000300	\$\$RTN2= 050001
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	\$NSK1 = 000110	\$\$SRC = 000000
UIPAR1= 177642	WBUFSZ= 000142	\$F\$LDO= 000200	\$\$AVLE= 177777	\$\$TGSV= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$TAGLE= 177777	\$\$TGS1= 000000
UIPAR3= 177646	WDTO = 000114	\$F\$ND = 000403	\$TAGNU= 050004	\$\$TGS2= 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	\$TEMP = 000300	\$\$TD = 000000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	\$TSKO = 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:CTLOUE=SPMAC/ML,EQUATE,CTLQUE  
RUN-TIME: 14 4 .4 SECONDS  
RUN-TIME RATIO: 40/19=2.0  
CORE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:48  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0387

3 COMMON EQUATE MODULE  
531 COMMON DEFINITIONS AND REFERENCES FOR CONOF  
534 000000' .PRINT ;SPMAC: VERSION 1.1  
577 KCON PROCESS THE 'CON'(CACHE ON) KEYBOARD COMMAND  
629 KCON ROUTINE  
722 KCOFF PROCESS THE 'COFF' KEYBOARD COMMAND  
772 KCOFF PROCESS THE KEYBOARD COMMANDS 'COFF'

```
508 .TITLE CONOF PROCESS THE 'CON' AND 'COFF' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511
512
513 ;++
514 ; MODULE PACKAGE NAME:
515 ; CONOF
516 ;
517 ; FUNCTIONAL DESCRIPTION:
518 ; THIS MODULE PACKAGE CONTAINS:
519 ; 1. KCON - PROCESS THE 'CON' KEYBOARD COMMAND
520 ; 2. KCOFF - PROCESS THE 'COFF' KEYBOARD COMMAND
521 ;
522 ; VERSION:
523 ; 0.0
524 ;
525 ; EDIT BY DATE REASON
526 ;--
527
528
529
```

```

531          .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR CONOF
532
533          .MCALL  STRUCT
534 000000'   STRUCT
535 (1) 000000' .PRINT  ;SPMAC: VERSION i.1
536          000001   $LSTTAG=1
537          000001   $LSTIN=1
538          ;*****
539          ;
540          ; REFERENCED BY OTHER MODULES:
541          ;
542          .GLOBL  KCON           ;KCON MODULE ENTRY POINT
543          .GLOBL  KCOFF          ;KCOFF MODULE ENTRY POINT
544          ;
545          ; GLOBAL REFERENCES:
546          ;
547          .GLOBL  CCNTRL          ;CONTROL WORD FOR CACHE REG.
548          .GLOBL  SAVREG          ;SAVE REGISTERS
549          .GLOBL  RESREG          ;RESTORE REGISTERS
550          .GLOBL  ARGCHK          ;CHECK AN ARGUMENT
551          .GLOBL  CM.ARG          ;'INVALID ARGUMENT' MESSAGE
552          .GLOBL  KONTRL          ;CONTROL WORD
553          ;
554          ;*****
555          ;*****
556          ;
557          ;
558          ;*****
559          ;
560          ; LOCAL EQUATES:
561          ;
562          ;
563          000352   CFLUSH=352           ;FLUSH 11/60 CACHE
564          ;
565          ;*****
566          ;
567          ; LOCAL STORAGE:
568          ;
569          ;
570 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
571 000036' 040503 044103 020105 CO.ON:  .ASCIZ  /CACHE ON%/
000044' 047117 000045
572 000050' 047516 041440 041501 CO.NUN: .ASCIZ  /NO CACHE%/
000056' 042510 000045
573 000062' 040503 044103 020105 CO.OFF: .ASCIZ  /CACHE OFF%/
000070' 043117 022506      000
574          000076'   .EVEN
575

```

```
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.CFO
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
```

```

629          .SBTTL KCON ROUTINE
630
631 000076'    ROUTINE KCON <DTADR,BUFPTR>
(2) 000076'
632
633          ;+
634          ; INIT AND SAVE DTABLE,BUFPTR
635          ; -
636
637 000076'    CALL SAVREG
(3) 000076' 004767 000000G          JSR      PC,SAVREG
638 000102'    LET R1 := DTADR(R5)
(4) 000102' 016501 000000          MOV      DTADR(R5),R1
639 000106'    LET R2 := BUFPTR(R5)
(4) 000106' 016502 000002          MOV      BUFPTR(R5),R2
640
641          ;+
642          ; CHECK FOR JUNK ARGUMENTS
643          ; -
644
645 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
646
647 000132'    IF.ERROR THEN
(6) 000132' 103045          BCC      50002$
648
649
650          ;+
651          ; NO JUNK ARGS...CONTINUE... SEE IF CACHE EXISTS AT ALL
652          ; -
653
654          IF #CAPRES SETIN DT.CF0(R1) THEN
(6) 000134' 032761 000004 000014          BIT      #CAPRES,DT.CF0(R
(9) 000142' 001435          BEQ      50003$
655
656          ;+
657          ; THERE IS CACHE... SEE WHAT FLAVOR... PDP 11/60 OR OTHER
658          ; -
659
660          IF #PDP60 SETIN DT.CF0(R1) THEN
(6) 000144' 032761 004000 000014          BIT      #PDP60,DT.CF0(R1
(9) 000152' 001404          BEQ      50004$
661
662          ;+
663          ; IT'S AN 11/60 SO,USING THE 'MED' INSTRUCTION FEATURE OF IT, DO A
664          ; CACHE SWEEP. CODE OF 200 TO R0,352 TRAILING TO GET INIT REG(INTERNAL
665          ; MICROCODE). CLEAR THE FORCE MISS BITS....
666          ; -
667
668          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$:
685
686                                     ;+
687                                     ; NOW STUFF THE CACHE ON MSG, SET THE CACHE ENABLE BIT IN STATUS WORD...
688                                     ;-
689
690 000220'                               LET DT.KBRSP(R1) := #CO.ON
(4) 000220' 012761 000036' 000022                               MOV #CO.ON,DT.KBRSP(
691 000226'                               LET DT.STO(R1) := DT.STO(R1) SET.BY #CASTAT
(6) 000226' 052761 000004 000010                               BIS #CASTAT,DT.STO(R
692
693 000234'                               ENDIF
(4) 000234'                               50006$:
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
```



```
722 .SBTTL KCOFF PROCESS THE 'COFF' KEYBOARD COMMAND
723 .IDENT /V0.0/
724
725 ;++
726 ; MODULE NAME:
727 ; KCOFF
728 ;
729 ; FUNCTIONAL DESCRIPTION:
730 ; THIS ROUTINE PROCESSES THE 'COFF'(CACHE OFF) KEYBOARD COMMAND. IT
731 ; WILL FIRST CHECK FOR THE PRESENCE
732 ; OF ILLEGAL ARGUMENTS. IF ANY, AN ERROR MESSAGE IS OUTPUTTED
733 ; AND IF NOT THE FORCE MISS BITS IN THE CONTROL
734 ; REGISTER ARE SET.
735 ; IF NO CACHE AN ERROR MESSAGE IS STUFFED.
736 ;
737 ; INPUTS:
738 ; 1. ADDRESS OF DATA TABLE
739 ; 2. DECODE BUFFER PTR
740 ;
741 ; IMPLICIT INPUTS:
742 ; DT.CFO
743 ;
744 ; OUTPUTS:
745 ; NONE
746 ;
747 ; IMPLICIT OUTPUTS:
748 ; 1. DT.KBRSP
749 ;
750 ; PATHOLOGICAL CONNECTIONS:
751 ; NONE
752 ;
753 ; SUBORDINATE ROUTINES CALLED:
754 ; 1. ARGCHK - CHECK AN ARGUMENT
755 ;
756 ; FUNCTIONAL SIDE EFFECTS:
757 ; NONE
758 ;
759 ; CALLING SEQUENCE:
760 ; CALL KCOFF IN <DTADR,BUFPTR>
761 ; WHERE DTADR = ADDRESS OF DATA TABLE
762 ; BUFPTR = DECODE BUFFER POINTER
763 ;
764 ; VERSION:
765 ; 0.0
766 ;
767 ; EDIT DATE BY REASON
768 ;--
769
770
```

```

772 .SBTTL KCOFF PROCESS THE KEYBOARD COMMANDS 'COFF'
773
774 000262' ROUTINE KCOFF <DTADR,BUFPTR>
(2) 000262' KCOFF:
775
776 ;+
777 ; INITIALIZE AND SAVE DATA TABLE ADDRESS AND BUFPTR
778 ; -
779
780 000262' PUSH R0,R1
(2) 000262' 010046 MOV R0,-(SP)
(3) 000264' 010146 MOV R1,-(SP)
781 000266' LET R0 := DTADR(R5)
(4) 000266' 016500 000000 MOV DTADR(R5),R0
782 000272' LET R1 := BUFPTR(R5)
(4) 000272' 016501 000002 MOV BUFPTR(R5),R1
783
784 ;+
785 ; SEE IF ANY JUNK ARGUMENTS
786 ; -
787
788 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
789
790 000316' IF.ERROR THEN
(6) 000316' 103025 BCC 50002$
791
792 ;+
792 ; NO JUNK ARGUMENTS FOUND SO SEE IF CACHE EXISTS AT ALL
794 ; -
795
796
797 000320' IF #CAPRES SETIN DT.CFO(R0) THEN
(6) 000320' 032760 000004 000014 BIT #CAPRES,DT.CFO(R
(9) 000326' 001415 BEQ 50003$
798
799
800
801 ;+
802 ; CACHE DOES EXIST SO TURN OFF CACHE AND STUFF MSG
803 ; -
804 000330' LET KONTRL := KONTRL SET.BY #14
(6) 000330' 052767 000014 000000G BIS #14,KONTRL
805 000336' LET @CCNTRL := KONTRL
(4) 000336' 016777 000000G 000000G MOV KONTRL,@CCNTRL
806 000344' LET DT.KBRSP(R0) := #CO.OFF
(4) 000344' 012760 000062' 000022 MOV #CO.OFF,DT.KBRSP
807 000352' LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #CASTAT
(6) 000352' 042760 000004 000010 BIC #CASTAT,DT.ST0(R
808
809 000360' ELSE

```

```

(4) 000360' 000403
(3) 000362'
810
811
812 ;+
813 ; NO CACHE ON SYSTEM - GO INSULT OPERATOR....
814 ; STUFF ERROR MSG...
815 ;-
815 000362' LET DT.KBRSP(R0) := #CO.NUN
(4) 000362' 012760 000050' 000022 MOV #CO.NUN,DT.KBRSP
816
817 000370' ENDIF
(4) 000370'
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'
(3) 000404'
(2) 000404' 000207 50000$:
835 50001$:
836 000001 .END RTS PC
  
```

ACSR = 000102	CONFIG= 000056	DT.WBU= 000050	KONTRL= ***** G	PR0 = 000000
ACTBIT= 004000	CO.BAD 000000R	DT.WHL= 000054	KTERR0= 000340	PR4 = 000200
ADDR22= 001000	CO.NUN 000050R	DT.WLL= 000052	KTPRES= 000400	PR5 = 000240
ADR = 000006	CO.OFF 000052R	DVID1 = 000014	KTSTAT= 000020	PR6 = 000300
APTFER= 000004	CD.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	ENENUL= 000001	MAPSTA= 000200	RANNUM= 000054
ASSEMB= 000010	CSRC = 000102	ENDLST= 000000	MED = 076600	RBUFEA= 000130
ASAT = 000104	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	RBUFPA= 000126
AUTJ = 000010	CTRLO = 000017	ERRTYP= 000106	MODEXH= 004000	RBUFSZ= 000132
AUTQST= 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.CFO= 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	INHRE= 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	NDAPTY= 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	SDFCNT= 000042
BKMODE= 040000	DT.PSW= 000004	KIPAR4= 172350	PARPRE= 002000	SDFPAS= 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 = 000000
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= 000001	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	\$S\$BYTE= 000403
SVR1 = 000064	UIPDR3= 177606	\$F\$BAD= 000401	\$IFLEV= 177777	\$S\$CASE= 000000
SVR2 = 000066	UIPDR4= 177610	\$F\$BLA= 000170	\$ISKO = 000001	\$S\$DST = 000000
SVR3 = 000070	UIPDR5= 177612	\$F\$CAS= 000150	\$ISK1 = 000001	\$S\$ELOC= 000402
SVR4 = 000072	UIPDR6= 177614	\$F\$DEC= 000220	\$ISK2 = 000001	\$S\$ERFL= 000000
SVR5 = 000074	UIPDR7= 177616	\$F\$DO = 000340	\$LOCTA= 177777	\$S\$FLAG= 000001
SVR6 = 000076	WASADR= 000104	\$F\$FAL= 000405	\$LSTIN= 000001	\$S\$FROM= 000001
SYSCNT= 000052	WBSTAT= 000040	\$F\$GDD= 000400	\$LSTTA= 000001	\$S\$LDC = 000326R
SYSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$NESTL= 177777	\$S\$LDCN= 000000
TMPIO = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSKO = 000300	\$S\$REG = 177777
TQOVF = 000002	WBUFRQ= 000140	\$F\$LDD= 000200	\$NSK1 = 000110	\$S\$RETU= 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$NSK2 = 000110	\$S\$RTN1= 050000
UIPAR1= 177642	WDFR = 000116	\$F\$NO = 000403	\$NSK3 = 000110	\$S\$RTN2= 050001
UIPAR2= 177644	WDTO = 000114	\$F\$OR = 000320	\$SAVLE= 177777	\$S\$SRC = 000000
UIPAR3= 177646	WTINRE= 000352	\$F\$RTI= 000350	\$TAGLE= 177777	\$S\$TGSV= 000000
UIPAR4= 177650	WTWHMI= 000222	\$F\$RTN= 000300	\$TAGNU= 050006	\$S\$TGS1= 000000
UIPAR5= 177652	XFLAG = 000005	\$F\$SEL= 000140	\$TEMP = 000300	\$S\$TGS2= 000000
UIPAR6= 177654	XOFF = 000023	\$F\$THE= 000330	\$TSKO = 050005	\$S\$TD = 000001
UIPAR7= 177656	XON = 000021	\$F\$TRU= 000404	\$TSK1 = 050004	\$S\$TAG= 050000
UIPDRO= 177600	\$BGNLE= 177777	\$F\$UNT= 000130	\$TSK2 = 050006	. = 000406R
UIPDR1= 177602	\$ERFLG= 000400	\$F\$WHI= 000120	\$SARGC= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:48  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0399

3 COMMON EQUATE MODULE  
555 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
595 DPEQP ROUTINE

```
508 .TITLE DPEOP - XXDP/ACT-11 END-OF-PASS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     DPEOP
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CONTROLS RETURN TO AND RESTART FROM XXDP/ACT-11
517 ;
518 ; INPUTS:
519 ;     DATA TABLE ADDRESS
520 ;
521 ; IMPLICIT INPUTS:
522 ;     DT.ESIZ, DT.ADDR, DT.BLST, DT.SSIZ, DT.ST1
523 ;
524 ; OUTPUTS:
525 ;     NONE
526 ;
527 ; IMPLICIT OUTPUTS:
528 ;     NONE
529 ;
530 ; PATHOLOGICAL CONNECTIONS:
531 ;     MD.COD                ;MESSAGE CODE IN MSGDEQ
532 ;     DX.R5                 ;R5 STACK POINTER
533 ;     DX.SP                 ;R6 STACK POINTER
534 ;     BA.MPTR              ;BACKGROUND MODULE LIST POINTER
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ;     RSTRCY                ;RECOVER FROM RESET
538 ;     MSGDEQ               ;MESSAGE DEQUEUER
539 ;     DX.INI               ;MONITOR INIT. ROUTINE
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ;     CAUSES PROGRAM CONTROL TO LEAVE THE EXERCISER
543 ;
544 ; CALLING SEQUENCE:
545 ;     CALL DPEOP IN <A>
546 ;         A=ADDRESS OF DATA TABLE
547 ;
548 ; VERSION:
549 ;     0.0
550 ;     EDIT                BY                DATE                REASON
551 ;
552 ;
553 ;
```

```
555 .SBTTL COMMON DEFINITIONS AND REFERENCES
556
557 .MCALL STRUCT
558 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
559 000001 $LSTIN=1
560 000001 $LSTTAG=1
561
562 ;
563 ;*****
564 ;
565 ; REFERENCED BY OTHER MODULES
566 ;
567 .GLOBL DPEOP ;MODULE ENTRY POINT
568 .GLOBL $ENDAD ;XXDP/ACT-11 "HOOK"
569 ;
570 ;*****
571 ;
572 ; GLOBAL REFERENCES
573 ;
574 .GLOBL RSTRCY ;RECOVER FROM RESET
575 .GLOBL DX.INI ;MONITOR INIT. ROUTINE
576 .GLOBL DX.SP ;POINTER TO MONITOR'S R6 STACK
577 .GLOBL DX.R5 ;POINTER TO MONITOR'S R5 STACK
578 .GLOBL DX.DEQ ;MONITOR'S DEQUEUEING ROUTINE
579 .GLOBL BA.MPTR ;BACKGROUND MODULE LIST POINTER
580 .GLOBL MSGDEQ ;MESSAGE DEQUEUER
581 .GLOBL MD.COD ;MESSAGE TYPE CODE
582 ;
583 ;*****
584 ;
585 ; LOCAL STORAGE
586 ;
587 000000' 000000 DP.R0: .WORD 0 ;R0 STORAGE
588 000002' 000000 DP.R1: .WORD 0 ;R1 STORAGE
589 000004' 000000 DP.R2: .WORD 0 ;R2 STORAGE
590 ;
591 ;*****
592 ;
593 ;
```



```

595          .SBTTL DPEOP ROUTINE
596
597 000006'    ROUTINE DPEOP <TABL>
(2) 000006'
598
599
600          ;+
601          ; SET R0 TO THE START OF THE DATA TABLE
602          ; -
603
604 000006'    LET R0 := TABL(R5)
(4) 000006' 016500 000000
605
606
607          ;+
608          ; WAIT FOR THE TYPE QUEUE TO EMPTY.
609          ; -
610
611 000012'    REPEAT
(3) 000012'
612 000012'    CALL MSGDEQ IN <R0>
(3) 000012' 010546
(4) 000014' 010045
(3) 000016' 004767 000000G
(3) 000022' 012605
613 000024'    UNTIL MD.COD EQ #MSGNUL
(3) 000024' 026727 000000G 177775
(6) 000032' 001367
614
615
616          ;+
617          ; MOVE THE XXDP MONITOR (EVEN IF IT DOESN'T EXIST, BECAUSE WE DON'T KNOW
618          ; IF WE HAVE XXDP OF ACT-11) TO THE TOP 1.5K OF MEMORY (BUT NOT HIGHER
619          ; THAN 28K SINCE XXDP CAN'T USE KT).
620          ; -
621
622 000034'    IF DT.SSIZ(R0) HIS #1600 THEN
(6) 000034' 026027 000046 001600
(9) 000042' 103403
623          LET R2 := #160000
(4) 000044' 012702 160000
624          ELSE
(4) 000050' 000410
(3) 000052'
625          LET R2 := DT.SSIZ(R0) SHIFT #6
(4) 000052' 016002 000046
(7) 000056' 006302
(7) 000060' 006302
(7) 000062' 006302
(7) 000064' 006302
(7) 000066' 006302
(7) 000070' 006302
626          ENDIF
(4) 000072'
627
628

```

DPEOP:

MOV TABL(R5),R0

50002\$:

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,MSGDEQ  
MOV (SP)+,R5  
CMP MD.COD,#MSGNUL  
BNE 50002\$

50003\$:

CMP DT.SSIZ(R0),#160  
BLO 50003\$  
MOV #160000,R2  
BR 50004\$  
MOV DT.SSIZ(R0),R2  
ASL R2  
ASL R2  
ASL R2  
ASL R2  
ASL R2  
ASL R2

50004\$:

```

629          ;+
630          ; SET R1 TO THE 1ST ADDRESS ABOVE THE EXERCISER, SET R3 TO
631          ; 1.5K, AND MOVE THE CODE.
632          ; (DON'T FORGET THAT DT.ESIZ IS THE SIZE OF THE EXERCISER PLUS THE XXDP
633          ; MONITOR.)
634          ; -
635
636 000072'   LET R1 := DT.ESIZ(R0) + #2
637 (4) 000072' 016001 000044
638 (6) 000076' 062701 C00002
639          MOV      DT.ESIZ(R0),R1
640          ADD      #2,R1
641
642 000102'   LET R3 := #^D1536
643 (4) 000102' 012703 003000
644          MOV      #^D1536,R3
645
646          WHILE R3 NE #0 DO
647
648          50005$:
649          TST      R3
650          BEQ      50006$
651
652          LET -(R2) := -(R1)
653          MOV      -(R1),-(R2)
654
655          LET R3 := R3 - #1
656          DEC      R3
657
658          ENDDO
659          BR      50005$
660          50006$:
661
662          ;+
663          ; SAVE REGISTERS FOR AFTER THE RETURN FROM XXDP.
664          ; -
665
666 000120'   LET DP.R0 := R0
667 (4) 000120' 010067 177654
668          MOV      R0,DP.R0
669
670 000124'   LET DP.R1 := R1
671 (4) 000124' 010167 177652
672          MOV      R1,DP.R1
673
674 000130'   LET DP.R2 := R2
675 (4) 000130' 010267 177650
676          MOV      R2,DP.R2
677
678
679          ;+
680          ; *****
681          ; HERE IS THE ACTUAL XXDP/ACT-11 STUFF.
682          ; -
683
684 000134'   INLINE <      RESET>
685 (2) 000134' 000005
686          RESET
687
688 000136'   INLINE <      MOV      @#42,R0>
689 (2) 000136' 013700 000042
690          MOV      @#42,R0
691
692 000142'   INLINE <$ENDAD: JSR      PC,(R0)>
693 (2) 000142' 004710
694          $ENDAD: JSR      PC,(R0)
695
696 000144'   INLINE <      .WORD NOP,NOP,NOP>
697 (2) 000144' 000240 000240 000240
698          .WORD NOP,NOP,NO
699
700          ;+
701          ; *****
702          ; -

```

```

668      ;+
669      ; RESTORE THE REGISTERS WE WERE USING.
670      ; -
671
672      000152'      LET R0 := DP.R0
(4)      000152' 016700 177622      MOV      DP.R0,R0
673      000156'      LET R1 := DP.R1
(4)      000156' 016701 177620      MOV      DP.R1,R1
674      000162'      LET R2 := DP.R2
(4)      000162' 016702 177616      MOV      DP.R2,R2
675
676
677      ;+
678      ; NOW MOVE THE XXDP MONITOR BACK TO THE TOP OF THE EXERCISER.
679      ; -
680
681      000166'      LET R3 := #^D1536
(4)      000166' 012703 003000      MOV      #^D1536,R3
682      000172'      WHILE R3 NE #0 DO
(4)      000172'
(6)      000172' 005703      50007$: TST      R3
(9)      000174' 001403      BEQ      50010$
683      000176'      LET (R1)+ := (R2)+
(4)      000176' 012221      MOV      (R2)+,(R1)+
684      000200'      LET R3 := R3 - #1
(6)      000200' 005303      DEC      R3
685      000202'      ENDDO
(4)      000202' 000773      BR      50007$
(3)      000204'      50010$:
686
687
688      ;+
689      ; RE-INITIALIZE THE STACKS.
690      ; -
691
692      000204'      LET SP := #SPVALUE
(4)      000204' 012706 002200      MOV      #SPVALUE,SP
693      000210'      LET DX.SP := SP
(4)      000210' 010667 000000G      MOV      SP,DX.SP
694      000214'      LET R5 := #R5VALUE
(4)      000214' 012705 001740      MOV      #R5VALUE,R5
695      000220'      LET DX.R5 := R5
(4)      000220' 010567 000000G      MOV      R5,DX.R5
696
697
698      ;+
699      ; RE-INITIALIZE THE BACKGROUND LIST POINTER
700      ; -
701
702      000224'      LET BA.MPTR := DT.BLST(R0)
(4)      000224' 016067 000034 000000G      MOV      DT.BLST(R0),BA.M
703
704
705      ;+
706      ; SINCE ACT-11 REQUIRES AT LEAST 200 MILLISECONDS BETWEEN RESETS,
707      ; 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'
(6) 000236' 005704          50011$: TST      R4
(9) 000240' 003410          BLE      50012$
712 000242'   LET R3 := #177777
(4) 000242' 012703 177777          MOV      #177777,R3
713 000246'   WHILE R3 NE #0 DO
(4) 000246'          50013$:
(6) 000246' 005703          1ST      R3
(9) 000250' 001402          BEQ      50014$
714 000252'   LET R3 := R3 - #1
(6) 000252' 005303          DEC      R3
715 000254'   ENDDO
(4) 000254' 000774          50013$: BR      50013$
(3) 000256'   LET R4 := R4 - #1
716 000256'          50014$:
(6) 000256' 005304          DEC      R4
717 000260'   ENDDO
(4) 000260' 000766          50014$: BR      50011$
(3) 000262'   ;
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'
(3) 000314' 50000$:
(2) 000314' 000207 50001$:
758 000001 .END RTS PC
```

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	ENBEO= 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	RBUFBSZ= 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.CFO= 000014	HRDPAS= 000050	MSGHDR= 000004	RICHAR= 031060
BIT04 = 000020	DT.CF1= 000016	ICONT = 000036	MSGHNG= 000022	RPTDAT= 020000
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	INHRR= 000400	MSGSKE= 000003	SBKMOD= 000000
BIT12 = 010000	DT.KBP= 000026	INIT = 000030	MSGSM3= 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	ICMODP= 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	NOPTY= 000002	SC.SCA= 000012
BIT7 = 000200	DT.PSW= 000004	KIPAR2= 172344	NULL = 000000	SENDLS= 177777
BIT8 = 000400	DT.PFA= 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.STO= 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
CQDVF = 000001	DT.WLL= 000052	KTERRO= 000040	PRI7 = 000340	SVR0 = 000062
CR = 000015	DVID1 = 000014	KTPRES= 000400	PRO = 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	\$ISKO = 000001	\$SELOC= 000402
SVR5 = 000074	UIPDR6= 177614	\$F\$CAS= 000150	\$LOCTA= 177777	\$SERFL= 000000
SVRS = 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	\$SLOC = 000250R
TABL = 000000	WBUFEA= 000136	\$F\$GDD= 000400	\$NSKO = 000300	\$SLOCN= 000000
TMPIO = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NSK1 = 000120	\$REG = 177777
TQDVF = 000002	WBUFRQ= 000140	\$F\$INC= 000210	\$NSK2 = 000120	\$RETN= 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$LDD= 000200	\$SAVLE= 177777	\$RTN1= 050000
UIPAR1= 177642	WDFR = 000116	\$F\$NAM= 000160	\$SSKO = 050012	\$RTN2= 050001
UIPAR2= 177644	WDTO = 000114	\$F\$ND = 000403	\$TAGLE= 177777	\$SRC = 000000
UIPAR3= 177646	WTINRE= 000352	\$F\$DR = 000320	\$TAGNU= 050015	\$TGSV= 000000
UIPAR4= 177650	WTWHMI= 000222	\$F\$RTI= 000350	\$TEMP = 000300	\$TGS1= 000000
UIPAR5= 177652	XFLAG = 000005	\$F\$RTN= 000300	\$TSKO = 050011	\$TGS2= 000000
UIPAR6= 177654	XOFF = 000023	\$F\$SEL= 000140	\$TSK1 = 050012	\$TO = 000000
UIPAR7= 177656	XGN = 000021	\$F\$THE= 000330	\$TSK2 = 050013	\$STAG= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:49  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0409

3 COMMON EQUATE MODULE  
552 COMMON DEFINITIONS AND REFERENCES  
555 000000' .PRINT ;SPMAC: VERSION 1.1  
578 DPSTRT ROUTINE



DPSTRT - XXDP/ACT-11 START-UP  
DPSTRT.MAC 28-JUL-78 09:14

MACY11 30A(1052) 20-SEP-78 17:49 PAGE 19  
COMMON EQUATE MODULE

SEQ 0410

```
508 .TITLE DPSTRT - XXDP/ACT-11 START-UP
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     DPSTRT
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     MOVES XXDP MONITOR AND FAKES A "RUN" COMMAND
517 ;
518 ; INPUTS:
519 ;     DATA TABLE ADDRESS
520 ;
521 ; IMPLICIT INPUTS:
522 ;     DT.ESIZ,DT.SSIZ,DT.REL
523 ;
524 ; OUTPUTS:
525 ;     NONE
526 ;
527 ; IMPLICIT OUTPUTS:
528 ;     DT.ESIZ,DT.REL
529 ;
530 ; PATHOLOGICAL CONNECTIONS:
531 ;     NONE
532 ;
533 ; SUBORDINATE MODULES CALLED:
534 ;     SAVREG                ;SAVE REGISTERS
535 ;     RESREG                ;RESTORE REGISTERS
536 ;     KRUN                  ;PROCESS "RUN" COMMAND
537 ;
538 ; FUNCTIONAL SIDE EFFECTS:
539 ;     STARTS THE EXERCISING OF THE SYSTEM
540 ;
541 ; CALLING SEQUENCE:
542 ;     CALL DPSTRT IN <A>
543 ;         A=DATA TABLE ADDRESS
544 ;
545 ; VERSION:
546 ;     0.0
547 ;     EDIT                BY                DATE                REASON
548 ;
549 ;--
550
```

DPSTRT - XXDP/ACT-11 START-UP  
DPSTRT.MAC 28-JUL-78 09:14

MACY11 SOA(1052) 20-SEP-78 17:49 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 0411

```
552          .SBTTL COMMON DEFINITIONS AND REFERENCES
553
554          .MCALL STRUCT
555          STRUCT
556          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
557          000001 $LSTIN=1
558          000001 $LSTTAG=1
559          ;
560          ;*****
561          ;
562          ; REFERENCED BY OTHER MODULES
563          ;
564          .GLOBL DPSTRT          ;MODULE ENTRY POINT
565          ;
566          ;*****
567          ;
568          ; GLOBAL REFERENCES
569          ;
570          .GLOBL SAVREG          ;SAVE REGISTERS
571          .GLOBL RESREG          ;RESTORE REGISTERS
572          .GLOBL KRUN            ;PROCESS "RUN" COMMAND
573          ;
574          ;*****
575          ;
576          ;
```

```
578 .SBTTL DPSTRT ROUTINE
579
580 000000' ROUTINE DPSTRT <TABL>
581 (2) 000000' DPSTRT:
582
583 ;+
584 ; SAVE REGISTERS
585 ;-
586
587 000000' CALL SAVREG
588 (3) 000000' 004767 000000G JSR PC,SAVREG
589
590 ;+
591 ; SET R0 TO THE START OF THE DATA TABLE.
592 ;-
593
594 000004' LET R0 := TABL(R5)
595 (4) 000004' 016500 000000 MOV TABL(R5),R0
596
597 ;+
598 ; MOVE THE XXDP MONITOR (EVEN IF IT DOESN'T EXIST, SINCE WE DON'T KNOW
599 ; IF WE'RE UNDER XXDP OR ACT-11) FROM ITS LOAD AREA (HIGHEST NON-KT
600 ; ADDRESSABLE MEMORY IN THE SYSTEM) TO JUST ABOVE THE EXERCISER.
601 ; (R1 IS INITIALIZED TO THE 1ST FREE ADDRESS ABOVE THE EXERCISER, AND R2 IS
602 ; INITIALIZED TO THE LOWEST ADDRESS OF XXDP.)
603 ;-
604
605 000010' IF DT.SSIZ(R0) HIS #1600 THEN
606 (6) 000010' 026027 000046 001600 CMP DT.SSIZ(R0),#160
607 (9) 000016' 103403 BLO 50002$
608 000020' LET R3 := #160000
609 (4) 000020' 012703 160000 MOV #160000,R3
610 000024' ELSE
611 (4) 000024' 000410 BR 50003$
612 (3) 000026'
613 000026' LET R3 := DT.SSIZ(R0) SHIFT #+6
614 (4) 000026' 016003 000046 MOV DT.SSIZ(R0),R3
615 (7) 000032' 006303 ASL R3
616 (7) 000034' 006303 ASL R3
617 (7) 000036' 006303 ASL R3
618 (7) 000040' 006303 ASL R3
619 (7) 000042' 006303 ASL R3
620 (7) 000044' 006303 ASL R3
621 000046' ENDF
622 (4) 000046'
623 000046' LET R2 := R3 - #^D3074
624 (4) 000046' 010302 MOV R3,R2
625 (6) 000050' 162702 006002 SUB #^D3074,R2
626 611
627 000054' LET R1 := DT.ESIZ(R0) + #2
628 (4) 000054' 016001 000044 MOV DT.ESIZ(R0),R1
629 (6) 000060' 062701 000002 ADD #2,R1
630 000064' WHILE R2 LO R3 DO
```

614	000070'			LET (R1)+ := (R2)+		
615	000072'	012221		ENDDDO		
616	000074'	000774			50004\$:	
617						
618						
619				;+ ; ADJUST THE EXERCISER SIZE TO INCLUDE THE XXDP MONITOR.		
620				;-		
621						
622	000074'			LET DT.ESIZ(R0) := R1 - #2		
623	000074'	010160	000044			
624	000100'	162760	000002 000044		MOV	R1,DT.ESIZ(R0)
625					SUB	#2,DT.ESIZ(R0)
626				;+ ; INCREASE THE RELOCATION CONSTANT TO 28K.		
627				;-		
628						
629	000106'			LET DT.REL(R0) := #1600		
630	000106'	012760	001600 000040		MOV	#1600,DT.REL(R0)
631						
632				;+ ; NOW PRETEND A "RUN" COMMAND HAS BEEN TYPED.		
633				;-		
634						
635						
636	000114'			CALL KRUN IN <R0,#0>		
637	000114'	010546			MOV	R5,-(SP)
638	000116'	012745	000000		MOV	#0,-(R5)
639	000122'	010045			MOV	R0,-(R5)
640	000124'	004767	000000G		JSR	PC,KRUN
641	000130'	012605			MOV	(SP)+,R5
642						
643				;+ ; RESTORE REGISTERS AND RETURN.		
644				;-		
645	000132'			CALL RESREG		
646	000132'	004767	000000G		JSR	PC,RESREG
647	000136'			ENDRTN		
648	000136'				50000\$:	
649	000136'				50001\$:	
650	000136'	000207			RTS	PC
651						
652	000001			.END		

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ACTBIT= 004000	CTRLO = 000017	ERRTYP= 000105	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	RICHR= 031060
AUTJST= 020000	DT.AP = 000100	ICCNT = 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	RVALU= 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	INHRR= 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	ICMOD = 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	ICMODX= 110000	NBKMOD= 001000	SC.CKP= 000004
BIT11 = 004000	DT.KBU= 000030	JACK = 035060	NPCUOP= 000020	SC.CLD= 000000
BIT12 = 010000	DT.MLS= 000032	KIPAR0= 172340	NCPTY= 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	SGFCNT= 000042
BIT2 = 000004	DT.PC = 000002	KIPAR4= 172350	PARPRE= 002000	SGFPAS= 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	SRC = 177572
BIT7 = 000200	DT.REL= 000040	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
BIT8 = 000400	DT.SCT= 000056	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	PRO = 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	TMPID = 000002

TQOVF = 000002	WBSTAT= 000040	\$F\$DEC= 000220	\$IFLEV= 177777	\$\$DST = 000000
UIPAR0= 177640	WBUFEA= 000136	\$F\$DD = 000340	\$ISKO = 000001	\$\$ELDC= 000402
UIPAR1= 177642	WBUFPA= 000134	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$ERFL= 000000
UIPAR2= 177644	WBUFRQ= 000140	\$F\$GCO= 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\$L00= 000200	\$NSKO = 000300	\$\$L0CN= 000000
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$NSK1 = 000120	\$\$REG = 177777
UIPAR7= 177656	WTWHMI= 000222	\$F\$ND = 000403	\$\$AVLE= 177777	\$\$RETU= 000000
UIPDR0= 177600	XFLAG = 000005	\$F\$DR = 000320	\$\$SK0 = 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	\$TSKO = 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	\$\$TD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:50  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0416

3 COMMON EQUATE MODULE  
572 COMMON DEFINITIONS AND REFERENCES FOR DRPMOD  
575 000000' .PRINT ;SPMAC: VERSION 1.1  
599 DRPMOD ROUTINE

```
508 .TITLE DRPMD DROP A MODULE FROM AN EXERCISER
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; DRPMD
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE WILL DROP A MODULE FROM AN EXERCISER. IT WILL
517 ; FIRST INSURE THAT THE MODULE BEING DROPPED HAS NOT ALREADY
518 ; BEEN DROPPED. IF THE MODULE IS AN SBK OR NBK, IT WILL CLEAR
519 ; THE MODHOLD BIT IN THE STATUS INDICATOR. THEN CLEAR THE
520 ; MODULES ACTIVE BIT IN THE STATUS WORD AND SET THE DROPPED BIT
521 ; IN THE MODULE'S STATUS WORD. THEN, SET THE CKTIM BIT.
522 ; IF THE MODULE BEING DROPPED IS THE SYSTEM CLOCK MODULE, THE
523 ; CLOCK PRESENT BIT IN THE CONFIGURATION WORD WILL BE CLEARED
524 ; AND THE LOCATION SC.CLO WILL BE CLEARED.
525 ;
526 ; INPUTS:
527 ; 1. DATA TABLE ADDRESS
528 ; 2. MODULE HEADER ADDRESS
529 ;
530 ; IMPLICIT INPUTS:
531 ; DT.ST0
532 ; DT.ST1
533 ; DT.PC
534 ; DT.ADDR
535 ; DT.SCT
536 ; DT.CFO
537 ; DT.MLST
538 ;
539 ; OUTPUTS:
540 ; NONE
541 ;
542 ; IMPLICIT OUTPUTS:
543 ; 1. DT.MACT - NUMBER OF MODULES ACTIVE
544 ; 2. DT.ST0 - STATUS INDICATOR
545 ; 3. DT.MEOP - MODULES END OF PASS COUNT
546 ; 4. DT.CFO - CONFIGURATION WORD 0
547 ; 5. DT.ST1 - STATUS INDICATOR 1
548 ;
549 ; PATHOLOGICAL CONNECTIONS:
550 ; NONE
551 ;
552 ; SUBORDINATE ROUTINES CALLED:
553 ; 1. SAVREG
554 ; 2. RESREG
555 ;
556 ; FUNCTIONAL SIDE EFFECTS:
557 ; NONE
558 ;
559 ; CALLING SEQUENCE:
560 ; CALL DRPMD IN <DTADR,MODHDR>
561 ; WHERE
562 ; 1. DTADR = ADDRESS OF DATA TABLE
563 ; 2. MODHDR - MODULE BEING DROPPED HEADER ADDRESS
```



564  
565  
566  
567  
568  
569  
570

```
;  
; VERSION:  
; 0.0  
;  
; EDIT DATE BY REASON  
;--
```

```
572 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR DRPMDR
573
574 .MCALL STRUCT
575 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
576
577 000001 $LSTTAG=1
578 000001 $LSTIN=1
579
580 ;*****
581 ;
582 ; REFERENCED BY OTHER MODULES:
583 ;
584 .GLOBL DRPMDR ;MODULE ENTRY POINT
585 ;
586 ;*****
587 ;
588 ; GLOBAL REFERENCES:
589 ;
590 .GLOBL ENQTQ ;ENQUEUE A MESSAGE
591 .GLOBL DUNCHK ;SEE IF ANY OTHER MODULES CAN RUN
592 .GLOBL SAVREG ;SAVE THE REGISTERS
593 .GLOBL RESREG ;RESTORE THE REGISTERS
594 ;
595 ;*****
596 ;
597 ;*****
```

```
599          .SBTTL DRPMOD ROUTINE
600
601 000000'          ROUTINE DRPMOD <DTADR,MODHDR>
(2) 000000'
602
603          ;+
604          ; INIT AND SAVE DATA TABLE ADDRESS AND MODULE HEADER ADDRESS
605          ; -
606
607 000000'          CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC, SAVREG
608 000004'          LET R0 := DTADR(R5)
(4) 000004' 016500 000000          MOV      DTADR(R5), R0
609 000010'          LET R1 := MODHDR(R5)
(4) 000010' 016501 000002          MOV      MODHDR(R5), R1
610
611
612
613          ;+
614          ; SET CKTIM BIT TO INDICATE THAT MODULE HAS BEEN
615          ; DROPPED. SET DROPPED BIT IN MODULE STAT WORD
616          ; -
617
618 000014'          LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
(6) 000014' 052760 100000 000012          BIS      #CKTIM, DT.ST1(R0)
619
620
621 000022'          LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
(6) 000022' 042761 004000 000026          BIC      #ACTBIT, STAT(R1)
622 000030'          LET STAT(R1) := STAT(R1) SET.BY #BIT13
(6) 000030' 052761 020000 000026          BIS      #BIT13, STAT(R1)
623
624          ;+
625          ; GET A COPY OF THE STATUS WORD AND SEE IF NBK OR SBK MOD
626          ; -
627
628 000036'          LET R2 := STAT(R1)
(4) 000036' 016102 000026          MOV      STAT(R1), R2
629 000042'          LET R2 := R2 CLR.BY #STATBITS
(6) 000042' 042702 064757          BIC      #STATBITS, R2
630
631
632 000046'          IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
(6) 000046' 020227 000000          CMP      R2, #SBKMOD
(8) 000052' 001403          BEQ      50002$
(6) 000054' 020227 001000          CMP      R2, #NBKMOD
(9) 000060' 001003          BNE      50003$
(6) 000062'          50002$:
633
634
635          ;+
636          ; IT'S AN SBK OR NBK SO CLEAR THE MODHOLD BIT IN DT.ST0
637          ; -
638
639 000062'          LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #MODHOLD
(6) 000062' 042760 002000 000010          BIC      #MODHOLD, DT.ST0(
```

```
640
641 000070'          ENDIF
642 (4) 000070'          50003$:
643
644
645 ;+
646 ; GET ADDRESS OF SYS CLOCK TABLE
647 ;-
648 000070'          LET R2 := DT.SCT(R0)
649 (4) 000070' 016002 000066          MOV      DT.SCT(R0),R2
650
651 ;+
652 ; IF SYS CLOCK IS THE ONE BEING DROPPED - CLEAR OUT SC.CLO
653 ; AND CLOCK PRESENT BIT IN DT.CFO
654 ;-
655
656 000074'          IF R1 EQ SC.SCA(R2) THEN
657 (6) 000074' 020162 000012          CMP      R1,SC.SCA(R2)
658 (9) 000100' 001005          BNE      50004$
659
660 ;+
661 ; YUP - IT'S THE SYS CLOCK
662 ;-
663 000102'          LET SC.CLO(R2) := #0
664 (4) 000102' 005062 000000          CLR      SC.CLO(R2)
665 000106'          LET DT.CFO(R0) := DT.CFO(R0) CLR.BY #CLKPRES
666 (6) 000106' 042760 000001 000014          BIC      #CLKPRES,DT.CFO(
667
668 ENDIF
669 (4) 000114'          50004$:
670
671 ;+
672 ; SEE IF ANY MORE MODULES CAN RUN
673 ;-
674
675 000114'          CALL DUNCHK IN <R0,R1>
676 (3) 000114' 010546          MOV      R5,-(SP)
677 (5) 000116' 010145          MOV      R1,-(R5)
678 (4) 000120' 010045          MOV      R0,-(R5)
679 (3) 000122' 004767 000000G          JSR      PC,DUNCHK
680 (3) 000126' 012605          MOV      (SP)+,R5
681
682 ;+
683 ; ENQUE THE MESSAGE
684 ;-
685
686 000130'          LET R3 := DT.PC(R0) - #2
687 (4) 000130' 016003 000002          MOV      DT.PC(R0),R3
688 (6) 000134' 162703 000002          SUB      #2,R3
689
690 000140'          CALL ENQTQ IN <R0,#MSGDRP,R3,R1,#0>
691 (3) 000140' 010546          MOV      R5,-(SP)
692 (8) 000142' 012745 000000          MOV      #0,-(R5)
693 (7) 000146' 010145          MOV      R1,-(R5)
```



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

SYSERR= 000100	UIPDR7= 177616	\$FSBLA= 000170	\$FSWHI= 000120	\$\$DST = 000000
TMPID = 000002	WASADR= 000104	\$FSCAS= 000150	SF\$YES= 000402	\$\$ELOC= 000402
TQOVF = 000002	WBSTAT= 000040	\$FSDEC= 000220	\$IFLEV= 177777	\$\$ERFL= 000000
UIPAR0= 177640	WBUFEA= 000136	\$FSDO = 000340	SISK0 = 000001	\$\$FLAG= 000001
UIPAR1= 177642	WBUFPA= 000134	\$FSFAL= 000405	SLOCTA= 177777	\$\$FROM= 000000
UIPAR2= 177644	WBUFQ= 000140	\$FSGOO= 000400	SLSTIN= 000001	\$\$LOC = 000100R
UIPAR3= 177646	WBUFSZ= 000142	\$FSIF = 000110	SLSTTA= 000001	\$\$LCCN= 000000
UIPAR4= 177650	WDFR = 000116	\$FSINC= 000210	\$NESTL= 177777	\$\$REG = 177777
UIPAR5= 177652	WDT0 = 000114	\$F\$LOO= 000200	\$NSKO = 000300	\$\$RETU= 000000
UIPAR6= 177654	WTINRE= 000352	\$FSNAM= 000160	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR7= 177656	WTWHMI= 000222	\$F\$NO = 000403	SSAVLE= 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:51  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0425

3 COMMON EQUATE MODULE  
563 COMMON DEFINITIONS AND REFERENCES FOR DUNCHK  
566 000000' .PRINT ;SPMAC: VERSION 1.1



508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561

```
.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
; --
```

```
563 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR DUNCHK
564
565 .MCALL STRUCT
566 000000' STRUCT
567 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
568 000001 $LSTTAG=1
569 000001 $LSTIN=1
570
571 ;*****
572 ;
573 ; REFERENCED BY OTHER MODULES:
574 ;
575 .GLOBL DUNCHK
576 ;
577 ;*****
578 ;
579 ; GLOBAL REFERENCES:
580 ;
581 .GLOBL SAVREG
582 .GLOBL RESREG
583 .GLOBL PRRLOC ;RELOCATE CODE
584 .GLOBL MSGDHOOK ;O/P A MESSAGE
585 .GLOBL DX.RSTRT ;RESTART ADDRESS
586 .GLOBL APTDED ;PUT APT IN THE COFFIN
587 .GLOBL APTKIL ;KILL APT FATALLY
588 ;
589 ;
590 ;*****
591 ;
592 ;
593 ;*****
594 ;
595 ;
596 ; LOCAL STORAGE:
597 ;
598 000000' 045 DR.WOW: .ASCII /%/
599 000001' 000005 DR.NAM: .BLKB 5
600 000006' 042040 047522 050120 .ASCII / DROPPED/
601 000014' 042105
602 000016' 020445 020041 052522 .ASCIZ /%!! RUN OVER - ALL MODULES DROPPED !!%/
603 000024' 020116 053117 051105
604 000032' 026440 040440 046114
605 000040' 046440 042117 046125
000046' 051505 042040 047522
000054' 050120 042105 020440
000062' 022441 000
602 000066' .EVEN
603 ;
604
605
```

```

607 000066' ROUTINE DUNCHK <DTB,MOD>
(2) 000066'
608 000066' CALL SAVREG
(3) 000066' 004767 000000G JSR PC,SAVREG
609 000072' LET R0 := DTB(R5) MOV DTB(R5),R0
(4) 000072' 016500 000000 MOV MOD(R5),R1
610 000076' LET R1 := MOD(R5)
(4) 000076' 016501 000002
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 BIT #APTPRES,DT.CF0(
(6) 000102' 032760 000200 000014 BEQ 50002$
(9) 000110' 001413
620 000112' CALL APTDED IN <R0,R1> MOV R5,-(SP)
(3) 000112' 010546 MOV R1,-(R5)
(5) 000114' 010145 MOV R0,-(R5)
(4) 000116' 010045 JSR PC,APTDDED
(3) 000120' 004767 000000G MOV (SP)+,R5
(3) 000124' 012605 CALL APTKIL IN <R0> MOV R5,-(SP)
621 000126' MOV R0,-(R5)
(3) 000126' 010546 JSR PC,APTKIL
(4) 000130' 010045 MOV (SP)+,R5
(3) 000132' 004767 000000G
(3) 000136' 012605
622 000140' ENDF
(4) 000140' 50002$:
623
624 ;+
625 ; SEE IF THERE ARE ANY POSSIBLE MODULES LEFT TO RUN...
626 ; -
627
628 000140' LET R2 := DT.MLST(R0) MOV DT.MLST(R0),R2
(4) 000140' 016002 000032
629 000144' LET R4 := #0 CLR R4
(4) 000144' 005004
630 000146' WHILE (R2) NE #0 DO 50003$:
(4) 000146' TST (R2)
(6) 000146' 005712 BEQ 50004$
(9) 000150' 001413
631 000152' LET R3 := (R2)+ MOV (R2)+,R3
(4) 000152' 012203
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 INC R4
(6) 000174' 005204
634 000176' ENDF
(4) 000176' 50005$:

```

```
635 000176' ENDDO
(4) 000176' 000763
(3) 000200'
636
637
638
639
640 ;+
641 ; IF NO MORE CAN RUN, GET THE MODULE NAME INTO THE
642 ; MESSAGE THAT SAYS !! RUN OVER !! ETC. AND THEN O/P MESSAGE.
643 ; -
644 000200' IF R4 EQ #0 THEN
(6) 000200' 005704
(9) 000202' 001046
645 000204' LET R2 := #DR.NAM
(4) 000204' 012702 000001' MOV #DR.NAM,R2
646 000210' LET R3 := #5 MOV #5,R3
(4) 000210' 012703 000005
647 000214' WHILE R3 NE #0 DO
(4) 000214'
(6) 000214' 005703
(9) 000216' 001403
648 000220' LET (R2)+ :B= (R1)+
(4) 000220' 112122 MOV (R1)+,(R2)+
649 000222' LET R3 := R3 - #1
(6) 000222' 005303 DEC R3
650 000224' ENDDO
(4) 000224' 000773
(3) 000226'
651 000226' CALL MSGDHOOK IN <R0,#MSGPOP,#DR.WOW,#443$>
(3) 000226' 010546
(7) 000230' 012745 000260'
(6) 000234' 012745 000000'
(5) 000240' 012745 000002
(4) 000244' 010045
(3) 000246' 004767 000000G
(3) 000252' 012605
652 000254' INLINE <1$: 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 SEQ 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
(3) 000304' 012605
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          50000$:
(3) 000324'          50001$:
(3) 000324'
(2) 000324' 000207          RTS PC
680          000001          .END
```

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

SYMBOL TABLE

SVR3 = 000070	UIPDR4= 177610	\$F\$BAD= 000401	\$F\$YES= 000402	\$S\$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\$GDD= 000400	\$LSTTA= 000001	\$\$FROM= 000000
TQDVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NESTL= 177777	\$\$LOC = 000266R
UIPAR0= 177640	WBUFQR= 000140	\$F\$INC= 000210	\$NSKO = 000300	\$\$LOCN= 000000
UIPAR1= 177642	WBUFSS= 000142	\$F\$LOD= 000200	\$NSK1 = 000110	\$\$REG = 177777
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$RETN= 000000
UIPAR3= 177646	WDTO = 000114	\$F\$ND = 000403	\$\$AVLE= 177777	\$\$RTN1= 050000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	\$\$SKO = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:51  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0433

3 COMMON EQUATE MODULE  
562 COMMON DEFINITIONS AND REFERENCES  
565 000000' .PRINT ;SPMAC: VERSION 1.1  
628 ERREC ROUTINE



```
508 .TITLE ERREC - ERROR RECOVERY
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; ERREC
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; ON FATAL ERRORS, THE CURRENT RUN OF THE EXERCISER IS
517 ; ABORTED. IF THE ERROR IS NOT FATAL, THE EXERCISER IS
518 ; RESTARTED.
519 ;
520 ; INPUTS:
521 ; DATA TABLE ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ; DT.ERR, DT.ADDR, DT.CFO
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; DT.STO
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; BUSY FLAG IN MSGDEQ (MD.BSY)
534 ; MESSAGE CODE IN MSGDEQ (MD.COD)
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ; MSGDHOOK ;HOOK MESSAGE TO PROPER DRIVER
538 ; PRRLOC ;PROCESS RELOCATION
539 ; WBFLIM ;DETERMINE WRITE BUFFER LIMITS
540 ; UNIMAP ;LOAD UNICUS MAP
541 ; MSGDEQ ;MESSAGE DEQUEUER
542 ; CLKON ;TURN ON SYSTEM CLOCK
543 ; KBINI ;INITIALIZE KEYBOARD
544 ; APTKIL ;KILL AN APT RUN
545 ; SAVREG ;SAVE REGISTERS
546 ; RESREG ;RESTORE REGISTERS
547 ;
548 ; FUNCTIONAL SIDE EFFECTS:
549 ; NONE
550 ;
551 ; CALLING SEQUENCE:
552 ; CALL ERREC IN <A>
553 ; A=DATA TABLE ADDRESS
554 ;
555 ; VERSION:
556 ; 0.0
557 ;
558 ; EDIT BY DATE REASON
559 ;
560 ;--
```

```

562 .SBTTL COMMON DEFINITIONS AND REFERENCES
563
564 .MCALL STRUCT
565 000000' STRUCT
566 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
567 000001 $LSTIN=1
568 000001 $LSTTAG=1
569
570 ;
571 ;*****
572 ; REFERENCED BY OTHER MODULES:
573 ;
574 .GLOBL ERREC ;MODULE ENTRY POINT
575 ;
576 ;*****
577 ;
578 ; GLOBAL REFERENCES
579 ;
580 .GLOBL MSGDHOOK
581 .GLOBL APTKIL
582 .GLOBL PRRLOC
583 .GLOBL WBFLIM
584 .GLOBL UNIMAP
585 .GLOBL MSGDEQ
586 .GLOBL CLKON
587 .GLOBL KBINI
588 .GLOBL DX.RSTST ;MONITOR'S RESTART LOCATION
589 .GLOBL MD.BSY ;MSGDEQ BUSY FLAG
590 .GLOBL MD.COD ;MESSAGE CODE IN MSGDEQ
591 .GLOBL SAVREG
592 .GLOBL RESREG
593 ;
594 ;*****
595 ;
596 ; LOCAL STORAGE
597 ;
598 000000' ER.TBL: ;MESSAGE ADDRESS TABLE
599 000000' 000072' .WORD ER.CQD
600 000002' 000121' .WORD ER.TQD
601 000004' 000145' .WORD ER.APT
602 000006' 000151' .WORD ER.PAR
603 000010' 000167' .WORD ER.REL
604 000012' 000202' .WORD ER.KT
605 000014' 000205' .WORD ER.SYS
606
607 ; - MESSAGES -
608 000016' 043045 052101 046101 ER.MG1: .ASCIZ /%FATAL ERROR%TOO MANY /
609 000024' 042440 051122 051117
610 000032' 052045 047517 046440
611 000040' 047101 02013: 000
612 000045' 040 051105 047522 ER.MG2: .ASCIZ / ERRORS%RUN ABORTED%/
613 000052' 051522 051045 047125
614 000060' 040440 047502 052122
615 000066' 042105 000045

```

```
611 ;+
612 ; THE FOLLOWING MESSAGES MUST BE IN THE SAME ORDER AS THEY ARE
613 ; IN THE ERROR WORD OF THE DATA TABLE, I. E., THE ERROR
614 ; MESSAGE ASSOCIATED WITH BIT 0 MUST BE FIRST, THEN THE ERROR MESSAGE
615 ; ASSOCIATED WITH BIT 1, ETC.
616 ;-
617
618 000072' 047503 052116 047522 ER.CQD: .ASCIZ /CONTROL QUEUE OVERFLOW/
    000100' 020114 052521 052505
    000106' 020105 053117 051105
    000114' 046106 053517 000
619 000121' 124 050131 020105 ER.TQD: .ASCIZ /TYPE QUEUE OVERFLOW/
    000126' 052521 052505 020105
    000134' 053117 051105 046106
    000142' 053517 000
620 000145' 101 052120 000 ER.APT: .ASCIZ /APT/
621 000151' 120 051101 052111 ER.PAR: .ASCIZ /PARITY OR ECC/
    000156' 020131 051117 042440
    000164' 041503 000
622 000167' 122 046105 041517 ER.REL: .ASCIZ /RELOCATION/
    000174' 052101 047511 000116
623 000202' 052113 000 ER.KT: .ASCIZ /KT/
624 000205' 123 051531 042524 ER.SYS: .ASCIZ /SYSTEM/
    000212' 000115
625
626 .EVEN
```

```

628 .SBTTL ERREC ROUTINE
629
630
631 000214' ROUTINE ERREC <TABL>
632 (2) 000214' ERREC:
633
634 ;+
635 ; SAVE REGISTERS
636 ; -
637
638 000214' CALL SAVREG
639 (3) 000214' 004767 000000G JSR PC,SAVREG
640
641 ;+
642 ; SET R0 TO THE START OF THE DATA TABLE.
643 ; -
644
645 000220' LET R0 := TABL(R5)
646 (4) 000220' 016500 000000 MOV TABL(R5),R0
647
648 ;+
649 ; SEE IF THE ERROR WAS FATAL.
650 ; -
651 000224' IF #FATERR SETIN DT.ERR(R0) THEN
652 (6) 000224' 032760 100000 000020 BIT #FATERR,DT.ERR(R
653 (9) 000232' 001532 BEQ 50002$
654
655 ;+
656 ; THE ERROR WAS FATAL.
657 ; -
658
659 ;+
660 ; DETERMINE THE TYPE OF ERROR. THIS IS DONE BY FIRST COPYING THE CONTENTS
661 ; OF THE ERROR WORD (EXCEPT THE FATAL ERROR BIT) INTO TEMPORARY
662 ; STORAGE (R2). THEN THE BIT NUMBER (R4) OF THE SET ERROR INDICATOR
663 ; IS DETERMINED.
664 ; -
665
666 000234' LET R2 := DT.ERR(R0) CLR.BY #100000
667 (4) 000234' 016002 000020 MOV DT.ERR(R0),R2
668 (6) 000240' 042702 100000 BIC #100000,R2
669 000244' LET R3 := #000001
670 (4) 000244' 012703 000001 MOV #000001,R3
671 000250' LET R4 := #-1
672 (4) 000250' 012704 177777 MOV #-1,R4
673 000254' REPEAT
674 (3) 000254' 50003$:
675 (6) 000254' 040302 LET R2 := R2 CLR.BY R3 BIC R3,R2
676 000256' LET R4 := R4 + #1
677 (6) 000256' 005204 INC R4

```

672	000260'		LET R3 := R3 SHIFT #+1		
(7)	000260'	006303		ASL	R3
673	000262'		UNTIL R2 EQ #0		
(3)	000262'	005702		TST	R2
(6)	000264'	001373		BNE	50003\$
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		
(7)	000266'	006304		ASL	R4
682					
683	000270'		LET R1 := #ER.TBL + R4		
(4)	000270'	012701	000000'	MOV	#ER.TBL,R1
(6)	000274'	060401		ADD	R4,R1
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
695 000410'
(2) 000410' 000776
696
697 000412'
(2) 000412'
698
699
700
701
702
703
704
705 000412'
(6) 000412' 026027 000042 000200
(9) 000420' 001425
706 000422'
(3) 000422' 010546
(5) 000424' 012745 000200
(4) 000430' 010045
(3) 000432' 004767 000000G
(3) 000436' 012605
707 000440'
(3) 000440' 010546
(4) 000442' 010045
(3) 000444' 004767 000000G
(3) 000450' 012605
708 000452'
(6) 000452' 032760 001000 000014
(9) 000460' 001405
709 000462'
(3) 000462' 010546
(4) 000464' 010045
(3) 000466' 004767 000000G
(3) 000472' 012605
710 000474'
(4) 000474'
711 000474'
(4) 000474'
712
713
714
715
716
717
718 000474'
(6) 000474' 042760 100000 000010
719
720
721
722
723
724
725
726
727 000502'

```

```

                    5$: NOP
                    BR 5$
                    6$:
;+
; IF THE EXERCISER IS NOT AT THE LOWEST PART OF MEMORY, MOVE IT, RESET
; THE WRITE BUFFER, AND RELOAD THE UNIBUS MAP IF IT EXISTS.
;-

                    IF DT.ADDR(R0) NE #200 THEN
CMP                DT.ADDR(R0),#200
BEQ                50004$
                    CALL PRRLOC IN <R0,#200>
MOV                R5,-(SP)
MOV                #200,-(R5)
MOV                R0,-(R5)
JSR                PC,PRRLOC
MOV                (SP)+,R5
                    CALL WBFLIM IN <R0>
MOV                R5,-(SP)
MOV                R0,-(R5)
JSR                PC,WBFLIM
MOV                (SP)+,R5
                    IF #ADDR22 SETIN DT.CF0(R0) THEN
BIT                #ADDR22,DT.CF0(R
BEQ                50005$
                    CALL UNIMAP IN <R0>
MOV                R5,-(SP)
MOV                R0,-(R5)
JSR                PC,UNIMAP
MOV                (SP)+,R5
                    ENDIF
                    50005$:
                    ENDIF
                    50004$:
;+
; CLEAR THE RUN MODE BIT.
;-

                    LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RUNMODE
BIC                #RUNMODE,DT.ST0(
;+
; IF AN APT FATAL ERROR ... CALL THE APT KILL ROUTINE
;-

                    IF #APTFER SETIN DT.ERR(R0) THEN

```

728	000512'		CALL APTKIL		BIT	#APTFER,DT.ERR(R
729	000516'		ENDIF		BEQ	50006\$
730	000516'			50006\$:		
731	000516'		ELSE			
732	000520'	000431			BR	50007\$
733				50002\$:		
734			;+ ; THE ERROR WAS NOT FATAL.			
735			;-			
736						
737						
738			;+ ; ENABLE THE KEYBOARD INPUT.			
739			;-			
740						
741						
742			CALL KBINI IN <R0>			
743	000520'				MOV	R5,-(SP)
744	000520'	010546			MOV	R0,-(R5)
745	000522'	010045			JSR	PC,KBINI
746	000524'	004767	000000G		MOV	(SP)+,R5
747	000530'	012605				
748						
749						
750	000532'		LET MD.BSY := #0		CLR	MD.BSY
751	000532'	005067	000000G			
752	000536'		REPEAT	50010\$:		
753	000536'		CALL MSGDEQ IN <R0>			
754	000536'	010546			MOV	R5,-(SP)
755	000540'	010045			MOV	R0,-(R5)
756	000542'	004767	000000G		JSR	PC,MSGDEQ
757	000546'	012605			MOV	(SP)+,R5
758	000550'		UNTIL MD.COD EQ #MSGNUL			
759	000550'	026727	000000G 177775		CMP	MD.COD,#MSGNUL
760	000556'	001367			BNE	50010\$
761						
762						
763						
764						
765						
766						
767						
768						
769						
770						
771						
772						
773						
774						
775						
776						
777						
778						
779						
780						
781						
782						
783						
784						
785						
786						
787						
788						
789						
790						
791						
792						
793						
794						
795						
796						
797						
798						
799						
800						
801						
802						
803						
804						
805						
806						
807						
808						
809						
810						
811						
812						
813						
814						
815						
816						
817						
818						
819						
820						
821						
822						
823						
824						
825						
826						
827						
828						
829						
830						
831						
832						
833						
834						
835						
836						
837						
838						
839						
840						
841						
842						
843						
844						
845						
846						
847						
848						
849						
850						
851						
852						
853						
854						
855						
856						
857						
858						
859						
860						
861						
862						
863						
864						
865						
866						
867						
868						
869						
870						
871						
872						
873						
874						
875						
876						
877						
878						
879						
880						
881						
882						
883						
884						
885						
886						
887						
888						
889						
890						
891						
892						
893						
894						
895						
896						
897						
898						
899						
900						
901						
902						
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						
950						
951						
952						
953						
954						
955						
956						
957						
958						
959						
960						
961						
962						
963						
964						
965						
966						
967						
968						
969						
970						
971						
972						
973						
974						
975						
976						
977						
978						
979						
980						
981						
982				</		

762	000574'	004767	000000G						
763	(3)	000600'	012605					JSR	PC,CLKON
764	000602'			ENDIF				MOV	(SP)+,R5
765	(4)	000602'				50011S:			
766	000602'			ENDIF					
767	(4)	000602'				50007S:			
768				;+					
769				; RESTORE REGISTERS					
770				;-					
771	000602'			CALL RESREG					
772	(3)	000602'	004767	000000G				JSR	PC,RESREG
773									
774				;+					
775				; RESTART THE MONITOR.					
776				;-					
777	000606'			INLINE <JMP DX.RSTRT>					
778	(2)	000606'	000167	000000G				JMP	DX.RSTRT
779									
780	000612'			ENDRTN					
781	(3)	000612'				50000S:			
782	(3)	000612'				50001S:			
	(2)	000612'	000207					RTS	PC
	000001			.END					



ACSR = 000102	CSRA = 000100	ENBNUL= 000001	KIPDR5= 172312	PRI0 = 000000
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	KIPDR6= 172314	PRI1 = 000040
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	KIPDR7= 172316	PRI4 = 000200
ADR = 000006	CTRLD = 000017	ERREC 000214RG	KTERR0= 000040	PRI5 = 000240
APTFER= 000004	CTRLU = 000025	ERRTYP= 000106	KTPRES= 000400	PRI6 = 000300
APTKIL= ***** G	DCEVNT= 000011	ER.APT 000145R	KTSTAT= 000020	PRI7 = 000340
APTPRE= 000200	DEFRTN= 000400	ER.CQD 000072R	KTXTND= 040000	PRRLOC= ***** G
ASB = 000106	DIAGMC= 000000	ER.KT 000202R	LF = 000012	PR0 = 000000
ASSEMB= 000010	DROPMO= 100000	ER.MG1 000016R	LPSTAT= 000001	PR4 = 000200
ASTAT = 000104	DSEVNT= 000014	ER.MG2 000045R	MAPSTA= 000200	PR5 = 000240
AUTO = 000010	DT.ADD= 000042	ER.PAR 000151R	MD.BSY= ***** G	PR6 = 000300
AUTOST= 020000	DT.AP = 000100	ER.REL 000167R	MD.COD= ***** G	PR7 = 000340
AWAS = 000110	DT.APK= 000076	ER.SYS 000205R	MED = 076600	PS = 177776
BIT0 = 000001	DT.BLS= 000034	ER.TBL 000000R	MEMPAS= 040000	PSW = 177776
BIT00 = 000001	DT.CFO= 000014	ER.TQD 000121R	MODEXH= 004000	RANNUM= 000054
BIT01 = 000002	DT.CF1= 000016	EVNTBE= 000200	MODHOL= 002000	RBUFEA= 000130
BIT02 = 000004	DT.ERR= 000020	EVNTHD= 000200	MODSEL= 001000	RBUFPA= 000126
BIT03 = 000010	DT.ESI= 000044	EVNTKT= 000203	MSGCKD= 000010	RBUFSZ= 000132
BIT04 = 000020	DT.EVN= 000000	EVNTPE= 000202	MSGCKS= 000011	RBUFVA= 000124
BIT05 = 000040	DT.EXS= 000060	EVNTRE= 000201	MSGDEQ= ***** G	RDSERV= 000101
BIT06 = 000100	DT.FCH= 000037	FATERR= 100000	MSGDER= 000005	RDWHMI= 000022
BIT07 = 000200	DT.FCN= 000036	HRDCNT= 000044	MSGDHO= ***** G	RELERR= 000020
BIT08 = 000400	DT.HMX= 000104	HRDPAS= 000050	MSGDRP= 000017	RELMOD= 020000
BIT09 = 001000	DT.KBE= 000024	ICONT = 000036	MSGECH= 177777	RELTIM= 010000
BIT1 = 000002	DT.KBP= 000026	ICOUNT= 000040	MSGEDP= 000013	RESREG= ***** G
BIT10 = 002000	DT.KBR= 000022	IDNUM = 000122	MSGHDR= 000004	RES1 = 000056
BIT11 = 004000	DT.KBU= 000030	IE = 000100	MSGHNG= 000022	RES2 = 000060
BIT12 = 010000	DT.MLS= 000032	INDPAR= 000040	MSGHRD= 000007	RICHAR= 031060
BIT13 = 020000	DT.MTI= 000110	INHDRP= 040000	MSGMAP= 000021	RPTDAT= 002000
BIT14 = 040000	DT.OFF= 000070	INHEPR= 020000	MSGNUL= 177775	RSTRT = 000112
BIT15 = 100000	DT.PAS= 000074	INHREL= 001000	MSGPOP= 000002	RJBOUT= 000177
BIT2 = 000004	DT.PC = 000002	INHRR= 000400	MSGPRM= 177776	RUNMOD= 100000
BIT3 = 000010	DT.PFL= 000062	INIT = 000030	MSGRES= 000001	R5VALU= 001740
BIT4 = 000020	DT.PSW= 000004	INTR = 000120	MSGSFT= 000006	SAM = 075464
BIT5 = 000040	DT.PTA= 000064	IOMOD = 100000	MSGSKE= 000003	SAVREG= ***** G
BIT6 = 000100	DT.RCS= 000102	IOMODP= 102000	MSGSMB= 000015	SBADR = 000102
BIT7 = 000200	DT.REL= 000040	IOMODR= 112000	MSGSMH= 000014	SBKMOD= 000000
BIT8 = 000400	DT.SCT= 000066	IOMODX= 110000	MSGSMS= 000016	SBKSEL= 010000
BIT9 = 001000	DT.SMX= 000106	JACK = 035060	MSGSTD= 000000	SC.ADR= 000006
BKDEF = 000002	DT.SP = 000006	KBINI = ***** G	MSGSYS= 000012	SC.ALC= 000014
BKMOD = 000020	DT.SSI= 000046	KIPAR0= 172340	MSGVEC= 000020	SC.APC= 000016
BKMODE= 040000	DT.ST0= 000010	KIPAR1= 172342	NBKMOD= 001000	SC.CKL= 000002
BKSLSH= 000134	DT.ST1= 000012	KIPAR2= 172344	NCPUOP= 000020	SC.CKP= 000004
CAPRES= 000004	DT.SWR= 000056	KIPAR3= 172346	NOPTY= 000002	SC.CLO= 000000
CASSTAT= 000004	DT.SYP= 000072	KIPAR4= 172350	NULL = 000000	SC.HLD= 000010
CDERCT= 000146	DT.WBU= 000050	KIPAR5= 172352	OWEN = 024020	SC.SCA= 000012
CDWDCT= 000144	DT.WHL= 000054	KIPAR6= 172354	PAERR = 000010	SENDLS= 177777
CKTIM = 100000	DT.WLL= 000052	KIPAR7= 172356	PARPRE= 002000	SOFCNT= 000042
CLKON = ***** G	DVID1 = 000014	KIPDR0= 172300	PARSTA= 000100	SOFPAS= 000046
CLKPRE= 000001	DX.RST= ***** G	KIPDR1= 172302	PASCNT= 000034	SPACE = 000040
CONFIG= 000056	ECCMEM= 000100	KIPDR2= 172304	PDPLSI= 020000	SPOINT= 000032
CQCVF = 000001	ECCSTA= 000010	KIPDR3= 172306	PDP60 = 004000	SPVALU= 002200
CR = 000015	ENBEOP= 010000	KIPDR4= 172310	PDP70 = 010000	SRO = 177572

SR1 = 177574	UIPAR4= 177650	WTWHMI= 000222	\$F\$THE= 000330	\$TSK2 = 050005
SR2 = 177576	UIPAR5= 177652	XFLAG = 000005	\$F\$TRU= 000404	\$SARGC= 000002
SR3 = 172516	UIPAR6= 177654	XOFF = 000023	\$F\$UNT= 000130	\$S\$BYTE= 000403
STAT = 000026	UIPAR7= 177656	XON = 000021	\$F\$WHI= 000120	\$SCASE= 000000
STATBI= 064757	UIPDR0= 177600	\$BGNLE= 177777	\$F\$YES= 000402	\$SDST = 000000
STAT1 = 000027	UIPDR1= 177602	\$ERFLG= 000400	\$IFLEV= 177777	\$SELOC= 000402
SUSPND= 000001	UIPDR2= 177604	\$F\$AND= 000310	\$ISKO = 000001	\$SERFL= 000000
SVRO = 000062	UIPDR3= 177606	\$F\$BAD= 000401	\$ISK1 = 000001	\$SFLAG= 000001
SVR1 = 000064	UIPDR4= 177610	\$F\$BLA= 000170	\$ISK2 = 000001	\$S\$FROM= 000000
SVR2 = 000066	UIPDR5= 177612	\$F\$CAS= 000150	SLOCTA= 177777	\$S\$LOC = 000566R
SVR3 = 000070	UIPDR6= 177614	\$F\$DEC= 000220	\$LSTIN= 000001	\$S\$LOCN= 000000
SVR4 = 000072	UIPDR7= 177616	\$F\$DD = 000340	\$LSTTA= 000001	\$S\$REG = 177777
SVR5 = 000074	UNIMAP= ***** G	\$F\$FAL= 000405	\$NESTL= 177777	\$S\$RETU= 000000
SVR6 = 000076	WASADR= 000104	\$F\$GDD= 000400	\$NSKO = 000300	\$S\$RTN1= 050000
SYSCNT= 000052	WBFLIM= ***** G	\$F\$IF = 000110	\$NSK1 = 000110	\$S\$RTN2= 050001
YSERR= 000100	WBSTAT= 000040	\$F\$INC= 000210	\$NSK2 = 000110	\$S\$SRC = 000000
TABL = 000000	WBUFEA= 000136	\$F\$LCC= 000200	\$NSK3 = 000110	\$S\$TGSV= 000000
TMPIO = 000002	WBUFPA= 000134	\$F\$NAM= 000160	\$SAVLE= 177777	\$S\$TGS1= 000000
TQOVF = 000002	WBUFRQ= 000140	\$F\$NO = 000403	\$TAGLE= 177777	\$S\$TGS2= 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$DR = 000320	\$TAGNU= 050012	\$S\$TO = 000000
UIPAR1= 177642	WDFR = 000116	\$F\$RTI= 000350	\$TEMP = 000300	\$S\$TAG= 050000
UIPAR2= 177644	WDTO = 000114	\$F\$RTN= 000300	\$TSKO = 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)

3	COMMON EQUATE MODULE
562	COMMON DEFINITIONS AND REFERENCES
566	000000' .PRINT ;SPMAC: VERSION 1.1
613	MESSAGE - END-OF-PASS
624	MESSAGE - SUM COMMAND HEADER LINE (JUST A CR FOR THIS MONITOR)
628	MESSAGE - SUM COMMAND MODULE LINE
645	MESSAGE - SUM COMMAND SUMMARY LINE
654	MESSAGE - COMMON HEADER
667	MESSAGE - HRDERR,SOFERR
693	MESSAGE - HRDERR,SOFERR,EXTENSION
700	MESSAGE - DROP MODULE
709	MESSAGE - CKDATA,DATAACK,DATERR
734	MESSAGE - CKDATA/DATAACK SUMMARY LINE
747	FILLMSG - FILL-IN THE BLANKS ROUTINE
769	PROCESS END-OF-PASS TYPE MESSAGE
796	PROCESS COMMON HEADER TYPE MESSAGE
836	PROCESS HRDERR,SOFERR
919	PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?
973	PROCESS SUM COMMAND TYPE MESSAGES
986	PROCESS SUM COMMAND MODULE LINE MESSAGE
1016	PROCESS SUM COMMAND SUMMARY LINE MESSAGE
1036	PROCESS DROP MODULE TYPE MESSAGE
1068	PROCESS CKDATA/DATAACK/DATERR
1189	PROCESS CKDATA/DATAACK SUMMARY LINE
1218	DO THE NECESSARY CLEAN-UP AND THEN RETURN
1234	GET MODULE NAME
1269	CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII

```
508 .TITLE FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; FILLMSG
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE IS RESPONSIBLE FOR FIILLING IN THE BLANK FIELDS
517 ; OF THE MONITOR GENERATED SKELETAL MESSAGES.
518 ; THE TYPES OF SKELETAL MESSAGES SUPPORTED ARE:
519 ; 1. END OF PASS 2. HARD ERROR
520 ; 3. SUM COMMAND 4. SOFT ERROR
521 ; 5. COMMON HEADER 6. DROP MODULE
522 ; 7. DATA ERROR 8. BAD VECTOR
523 ; 9. MAP SUMMARY 10. CKDATA CALL
524
525 ; INPUTS:
526 ; 1. DATA TABLE
527 ; 2. MSG TYPE CODE
528 ; 3. MSG ADDR/SPECIAL VALUE
529 ; 4. MODULE'S HDR ADDR
530
531 ; IMPLICIT INPUTS:
532 ; DT.STO
533
534 ; OUTPUTS:
535 ; 1. ADDRESS OF FILLED-IN MESSAGE
536
537 ; IMPLICIT OUTPUTS:
538 ; NONE
539
540 ; PATHOLOGICAL CONNECTIONS:
541 ; NONE
542
543 ; SUBORDINATE ROUTINES CALLED:
544 ; 1. BDACNV ;BINARY TO DECIMAL ASCII CONVERSION
545 ; 2. BOAC ;BINARY TO OCTAL-ASCII CONVERSION
546 ; 3. GPA ;GET PHYSICAL ADDRESS
547 ; 4. SAVREG ;
548 ; 5. RESREG ;
549
550 ; FUNCTIONAL SIDE EFFECTS:
551 ; NONE
552
553 ; CALLING SEQUENCE:
554 ; CALL FILLMSG IN <DTABLE, TYPCODE, MSGADDR, HDRADDR> OUT <FILMSGADR>
555
556 ; VERSION:
557 ; 0.0
558
559 ; EDIT BY DATE REASON
560 ;
```

```

562          .SBTTL  COMMON DEFINITIONS AND REFERENCES
563
564
565          .MCALL  STRUCT
566 000000'    STRUCT
567 (1) 000000' .PRINT  ;SPMAC: VERSION 1.1
568          000001    ;$LSTIN = 1
569          000001    ;$LSTTAG = 1
570
571
572          ;
573          ;*****
574          ;
575          ; REFERENCED BY OTHER MODULES
576          ;
577          .GLOBL  FILLMSG          ;MODULE'S ENTRY POINT
578          ;
579          ;*****
580          ;
581          ; GLOBAL REFERENCES
582          ;
583          .GLOBL  BDACNV          ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
584          .GLOBL  BOA16          ;BINARY TO OCTAL-ASCII CONVERSION ROUTINE
585          .GLOBL  BOAC           ;BINARY TO OCTAL ASCII CONVERSION
586          .GLOBL  CKHUNG        ;CHECK FOR HUNG MODULES ROUTINE
587          .GLOBL  GPA           ;GET PHYSICAL ADDRESS ROUTINE
588          .GLOBL  SAVREG        ;
589          .GLOBL  RESREG        ;
590          ;
591          ;*****
592          ;
593          ; LOCAL EQUATES:
594          ;
595          000010' FLDMAX = ^D<8>          ;MAX FIELDS PER LINE
596          ;
597          ;*****
598          ;
599          ; LOCAL STORAGE
600          ;
601 000000' 000000' FLD CNT: .WORD 0          ;FIELD COUNTER
602 000002' 000000' LINECNT: .WORD 0        ;LINE COUNTER
603 000004'      TMP1:                ;STORAGE WORD
604 000004' 000000' TMPVA: .WORD 0        ;STORAGE/VIRTUAL ADDRESS
605 000006' 000000' TMPPA: .WORD 0        ;STORAGE/PHYSICAL ADDRESS
606 000010' 000000' TMPEA: .WORD 0        ;STORAGE/EA BITS
607 000012' 000000' FM.PA: .WORD 0        ;STORAGE - PHYS. ADDR.
608 000014' 000000' FM.EA: .WORD 0        ;STORAGE - EA BITS
609          ;
610          ;*****
611          ;
    
```

613 .SBTTL MESSAGE - END-OF-PASS  
614  
615  
616  
617 000016' 045 EOPMSG: .ASCII /%/  
618 000017' 000005 EOPNAM: .BLKB ^D<5>  
619 000024' 042440 042116 050040 .ASCII / END PASS #/  
000032' 051501 020123 043  
620 000037' 000005 EOPPAS: .BLKB ^D<5>  
621 000044' 000045 .ASCIZ /%/  
622

FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS MACY11 30A(1052) 20-SEPT-78 17:52 PAGE 19 3  
FILLML.MAC 16-AUG-78 14:36 MESSAGE - SUM COMMAND HEADER LINE (JUST A CR FOR THIS MONITOR)

SEQ 0448

624 .SBTTL MESSAGE - SUM COMMAND HEADER LINE (JUST A CR FOR THIS MONITOR)  
625  
626 000046' 000045 RNHMSG: .ASCIZ /%/





```
645 .SBTTL MESSAGE - SUM COMMAND SUMMARY LINE
646
647 000161' 045 054523 052123 RNSMSG: .ASCII /%SYSTEM ERRORS: /
    000166' 046505 042440 051122
    000174' 051117 035123 040
648 000201' 000005 RNSERR: .BLKB ^D<5>
649 000206' 020040 020040 050040 .ASCII / POWER FAILS: /
    000214' 053517 051105 043040
    000222' 044501 051514 020072
650 000230' 000005 RNSPWR: .BLKB ^D<5>
651 000235' 045 000 .ASCIZ /%/
652
```

```
654 .SBTTL MESSAGE - COMMON HEADER
655
656 000237' 045 HDRMSG: .ASCII /%/
657 000240' 000005 HDRNAM: .BLKB ^D<5>
658 000245' 040 040520 020072 .ASCII / PA: /
659 000252' 000010 HDRPA: .BLKB ^D<8>
660 000262' 040440 041520 020072 .ASCII / APC: /
661 000270' 000006 HDRPC: .BLKB 6
662 000276' 050040 051501 020123 .ASCII / PASS #/
    000304' 043
663 000305' 000005 HDRPAS: .BLKB ^D<5>
664 000312' 000045 .ASCIZ /%/
665
```

```

667 .SBTTL MESSAGE - HRDERR,SOFERR
668
669 000314' 045 DATAMSG: .ASCII /%/
670 000315' 000005 DATNAM: .BLKB ^D<5>
671 000322' 050040 035101 040 .ASCII / PA: /
672 000327' 000010 DATPC: .BLKB ^D<8>
673 000337' 040 050101 035103 .ASCII / APC: /
    000344' 040
674 000345' 000006 DATAPC: .BLKB ^D<6>
675 000353' 040 040520 051523 .ASCII / PASS #/
    000360' 021440
676 000362' 000005 DATPAS: .BLKB ^D<5>
677 000367' 040 .ASCII / /
678 000370' 000004 DATTYP: .BLKB ^D<4> ;DATA, HARD, SOFT
679 000374' 042440 051122 043 .ASCII / ERR#/
680 000401' 000005 DATNUM: .BLKB ^D<5>
681 000406' 041445 051123 035101 .ASCII /%CSRA: /
    000414' 040
682 000415' 000006 DATADD: .BLKB ^D<6>
683 000423' 040 051503 041522 .ASCII / CSRC: /
    000430' 020072
684 000432' 000006 DATCON: .BLKB ^D<6>
685 000440' 040440 052123 052101 .ASCII / ASTAT: /
    000446' 020072
686 000450' 000006 DATSTAT: .BLKB ^D<6>
687 000456' 042440 051122 054524 .ASCII / ERRTP: /
    000464' 035120 040
688 000467' 000006 DATERR: .BLKB ^D<6>
689 000475' 045 .ASCII /%/
690 000476' 000 DATEND: .BYTE 0 ;END OF MESSAGE INDICATOR
691
    
```

FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS MACY11 30A(1052) 20-SEP-78 17:52 PAGE 19-8  
FILLML.MAC 16-AUG-78 14:36 MESSAGE - HRDERR,SOFERR,EXTENSION

SEQ 0453

693 .SBTTL MESSAGE - HRDERR,SOFERR,EXTENSION  
694  
695 000477' EXTMSG:  
696 000477' 000344 .BLKB ^D<228>  
697 001043' 000 EXTEND: .BYTE 0  
698

FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS  
FILLML.MAC 16-AUG-78 14:36 MESSAGE - DROP MODULE

MACY11 30A(1052) 20-SEP-78 17:52 PAGE 19-9

SEQ 0454

```
700 .SBTTL MESSAGE - DROP MODULE
701
702 001044' 045 DRPMSG: .ASCII /%/
703 001045' 000005 DRPNAM: .BLKB ^D<5>
704 001052' 042040 047522 050120 .ASCII / DROPPED AT APC: /
    001060' 042105 040440 020:24
    001066' 050101 035103 040
705 001073' 000006 DRPPC: .BLKB ^D<6>
706 001101' 045 000 .ASCIZ /%/
707
```

```

709 .SBTTL MESSAGE - CKDATA,DATAACK,DATERR
710
711 001103' 045 CKDMSG: .ASCII /%/
712 001104' 000005 CKDNAM: .BLKB ^D<5>
713 001111' 040 040520 020072 .ASCII / PA: /
714 001116' 000010 CKDPA: .BLKB ^D<8>
715 001126' 040440 041520 020072 .ASCII / APC: /
716 001134' 000006 CKDVA: .BLKB ^D<6>
717 001142' 050040 051501 020123 .ASCII / PASS #/
    001150' 043
718 001151' 000005 CKDPAS: .BLKB ^D<5>
719 001156' 042440 051122 021440 .ASCII / ERR #/
720 001164' 000005 CKDNUM: .BLKB ^D<5>
721 001171' 045 051503 040522 .ASCII /%CSRA: /
    001176' 020072
722 001200' 000006 CKDADD: .BLKB ^D<6>
723 001206' 051440 041057 020072 .ASCII " S/B: "
724 001214' 000006 CKDSB: .BLKB ^D<6>
725 001222' 053440 051501 020072 .ASCII / WAS: /
726 001230' 000006 CKDWAS: .BLKB ^D<6>
727 001236' 053440 040522 051104 .ASCII / WRADR /
    001244' 040
728 001245' 000010 CKDWRA: .BLKB ^D<8>
729 001255' 040 042122 042101 .ASCII / RDADR /
    001262' 020122
730 001264' 000010 CKDRDA: .BLKB ^D<8>
731 001274' 000045 .ASCIZ /%/
732
  
```

```
734                                    .SBTTL MESSAGE - CKDATA/DATAACK SUMMARY LINE
735
736 001276'                            CKDSMSG:
737 001276' 000005                    CKDSNAM: .BLKB ^D<5>
738 001303'                            .ASCII / HAD /
739 001310' 000005                    CKDSCNT: .BLKB ^D<5>
740 001315'                            .ASCII / ERRORS OUT OF /
     001322' 051522 047440 052125
     001330' 047440 020106
741 001334' 000005                    CKDSRD: .BLKB ^D<5>
742 001341'                            .ASCIZ / WORDS READ%/
     001346' 020123 042522 042101
     001354' 000045
743
744                                    .EVEN
745
```

```
747 .SBTTL FILLMSG - FILL-IN THE BLANKS ROUTINE
748
749
750
751
752 001356' ROUTINE FILLMSG <DTABLE,TYP CODE, MSGADR, HDRADR,FLMSADR>
(2) 001356' FILLMSG:
753
754
755
756
757 ;+
758 ; SAVE REGISTERS AND SAVE DTABLE,ADR,MSGADR AND TYP CODE
759 ; -
760
761 001356' CALL SAVREG
(3) 001356' 004767 000000G JSR PC,SAVREG
762 001362' LET R0 := DTABLE(R5) MOV DTABLE(R5),R0
(4) 001362' 016500 000000 LET R1 := HDRADR(R5) MOV HDRADR(R5),R1
763 001366' LET R2 := MSGADR(R5) MOV MSGADR(R5),R2
(4) 001366' 016501 000006 LET R4 := TYP CODE(R5) MOV TYP CODE(R5),R4
764 001372' LET R4 := TYP CODE(R5)
(4) 001372' 016502 000004
765 001376' LET R4 := TYP CODE(R5)
(4) 001376' 016504 000002
766
767
```



```

769          .SBTTL  PROCESS END-OF-PASS TYPE MESSAGE
770
771
772
773          ;+
774          ; IS IT AN END-OF-PASS TYPE? IF SO, FILL IN THE BLANK FIELDS,
775          ; UPDATE THE MODULE'S PASS TIME, AND CHECK FOR ANY "HUNG" MODULES
776          ; -
777
778
779          IF R4 EQ #MSGEOP THEN
780
781          (6) 001402' 020427 000013          CMP      R4,#MSGEOP
782          (9) 001406' 001024          BNE      50002$
783
784          ;+
785          ; GET MODULE NAME
786          ; -
787
788          CALL GETNAM IN <R1, #EOPNAM>
789
790          (3) 001410' 010546          MOV      R5,-(SP)
791          (5) 001412' 012745 000017'    MOV      #EOPNAM,-(R5)
792          (4) 001416' 010145          MOV      R1,-(R5)
793          (3) 001420' 004767 002032    JSR      PC,GETNAM
794          (3) 001424' 012605          MOV      (SP)+,R5
795
796          ;+
797          ; CONVERT PASS COUNT TO DECIMAL ASCII
798          ; -
799          CALL BDACNV IN <PASCNT(R1), #EOPPAS>
800
801          (3) 001426' 010546          MOV      R5,-(SP)
802          (5) 001430' 012745 000037'    MOV      #EOPPAS,-(R5)
803          (4) 001434' 016145 000034    MOV      PASCNT(R1),-(R5)
804          (3) 001440' 004767 000000G   JSR      PC,BDACNV
805          (3) 001444' 012605          MOV      (SP)+,R5
806
807          LET FLMSADR(R5) := #EOPMSG
808          (4) 001446' 012765 000016' 000010  MOV      #EOPMSG,FLMSADR(
809          793 001454'          INLINE <JMP 1$>
810          (2) 001454' 000167 001770          JMP 1$
811          794 001460'          ENDIF
812          (4) 001460'          50002$:

```

```

796 .SBTTL PROCESS COMMON HEADER TYPE MESSAGE
797
798 ;+
799 ; IS IT A COMMON HEADER TYPE MESSAGE?
800 ; -
801
802 001460' IF R4 EQ #MSGHDR THEN
(6) 001460' 020427 000004 CMP R4,#MSGHDR
(9) 001464' 001045 BNE 50003$
803
804 ;+
805 ; GET MODULE NAME
806 ; -
807 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
808
809 ;+
810 ; GET PHYSICAL ADDRESS
811 ; -
812 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
813
814 ;+
815 ; FORM APC
816 ; -
817
818 001524' LET R3 := R2 - R1
(4) 001524' 010203 MOV R2,R3
(6) 001526' 160103 SUB R1,R3
819
820 ;+
821 ; AND CONVERT IT TO ASCII
822 ; -
823
824 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
825
826 ;+
827 ; CONVERT PASCNT TO ASCII
828 ; -
829
830 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							

```

836          .SBTTL PROCESS HRDERR,SOFERR
837
838
839          ;+
840          ; IF NOT A HARD OR SOFT ERROR, CONTINUE CHECKING
841          ; -
842
843
844          IF R4 NE #MSGSFY AND R4 NE #MSGHRD THEN
845          (6) 001600' 020427 000006          CMP      R4,#MSGSFY
846          (9) 001604' 001405          BEQ      50004$
847          (6) 001606' 020427 000007          CMP      R4,#MSGHRD
848          (9) 001612' 001402          BEQ      50004$
849
850          INLINE <JMP 100$>
851          (2) 001614' 000167 000474          JMP 100$
852          (4) 001620'
853          ENDIF
854
855          50004$:
856          ;+
857          ; INSERT THE END OF MESSAGE CHARACTER AND ASSUME THAT THERE IS NO EXTENDED PRINTOUT
858          ; -
859
860          LET DATEND :B= #0
861          (4) 001620' 105067 176652          CLRB    DATEND
862
863          ;+
864          ; GET MODULE NAME
865          ; -
866
867          CALL GETNAM IN <R1, #DATNAM>
868          (3) 001624' 010546          MOV     R5,-(SP)
869          (5) 001626' 012745 000315'      MOV     #DATNAM,-(R5)
870          (4) 001632' 010145          MOV     R1,-(R5)
871          (3) 001634' 004767 001616          JSR    PC,GETNAM
872          (3) 001640' 012605          MOV     (SP)+,R5
873
874          ;+
875          ; GET P.A.
876          ; -
877
878          CALL GETAPA IN <R0,R2,#DATPC>
879          (3) 001642' 010546          MOV     R5,-(SP)
880          (6) 001644' 012745 000327'      MOV     #DATPC,-(R5)
881          (5) 001650' 010245          MOV     R2,-(R5)
882          (4) 001652' 010045          MOV     R0,-(R5)
883          (3) 001654' 004767 001636          JSR    PC,GETAPA
884          (3) 001660' 012605          MOV     (SP)+,R5
885
886          ;+
887          ; FORM APC
888          ; -
889
890          LET R3 := R2 - R1
891          (4) 001662' 010203          MOV     R2,R3
892          (6) 001664' 160103          SUB     R1,R3
893
894          ;+
    
```

```
872 ; CONVERT IT TO ASCII
873 ; -
874
875 001666' CALL BDA16 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
876
877 ;+
878 ; CONVERT PASS COUNT TO ASCII
879 ; -
880
881 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
882
883 001724' LET R3 := #DATTYP
(4) 001724' 012703 000370' MOV #DATTYP,R3
884
885 ;+
886 ; IF SOFT, FILL IN WORD 'SOFT'. IF HARD, FILL IN WORD 'HARD'.
887 ; -
888 001730' IF R4 EQ #MSGSFY THEN
(6) 001730' 020427 000006 CMP R4,#MSGSFY
(9) 001734' 001013 BNE 50005$
889 001736' LET (R3)+ :B= #'S MOVB #'S,(R3)+
(4) 001736' 112723 000123 LET (R3)+ :B= #'O MOVB #'O,(R3)+
890 001742' LET (R3)+ :B= #'F MOVB #'F,(R3)+
(4) 001742' 112723 000117 LET (R3)+ :B= #'T MOVB #'T,(R3)+
891 001746' LET R0 := SOFCNT(R1) MOV SOFCNT(R1),R0
(4) 001746' 112723 000106 ELSE BR 50006$
892 001752' LET R0 := HRDCNT(R1) MOV HRDCNT(R1),R0
(4) 001752' 112723 000124
893 001756' LET (R3)+ :B= #'H MOVB #'H,(R3)+
(4) 001764' 112723 000110 LET (R3)+ :B= #'A MOVB #'A,(R3)+
896 001770' LET (R3)+ :B= #'R MOVB #'R,(R3)+
(4) 001774' 112723 000122 LET (R3)+ :B= #'D MOVB #'D,(R3)+
898 002000' LET R0 := HRDCNT(R1) MOV HRDCNT(R1),R0
(4) 002000' 112723 000104
899 002004' ENDIF
(4) 002004' 016100 000044
900 002010'
(4) 002010'
901
```

```
902
903          ;+
904          ; CONVERT HARD OR SOFT TO ASCII
905          ; -
906
907          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
908
909          ;+
910          ; CONVERT CSRA,CSRC,ASTAT,ERRTYP TO ASCII
911          ; -
912
913          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
914          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
915          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
916          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
917
```

```

919 .SBTTL PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?
920
921
922
923 ;+
924 ; IS THERE A HRDERR/SOFERR EXTENDED PRINTOUT ASSOCIATED WITH THIS MESSAGE?
925 ; -
926
927 IF R4 EQ #MSGSFY OR R4 EQ #MSGHRD THEN
(6) 002126' 020427 000006 CMP R4,#MSGSFY
(8) 002132' 001403 BEQ 50007$
(6) 002134' 020427 000007 CMP R4,#MSGHRD
(9) 002140' 001065 BNE 50010$
(6) 002142' 50007$:
928 002142' LET R2 := R2 + #4 ADD #4,R2
(6) 002142' 062702 000004 LET R2 := (R2) MOV (R2),R2
929 002146' (4) 002146' 011202 IF R2 NE #0 THEN TST R2
930 002150' (6) 002150' 005702 BEQ 50011$
(9) 002152' 001453 LET DATEND :B= #<CR>
931 002154' (4) 002154' 112767 000015 176314 MOVB #<CR>,DATEND
932
933 ;+
934 ; GO THROUGH THE MESSAGE TABLE AND, FOR EACH ADDRESS IN THE TABLE, FIND
935 ; THE CONTENTS OF THAT ADDRESS, CONVERT IT TO ASCII, STORE THE ASCII IN THE
936 ; EXTENDED PRINTOUT TABLE (EXTMSG) FOLLOWED BY A SPACE AND, IF THE MAXIMUM
937 ; NUMBER OF ENTRIES PER LINE HAS BEEN REACHED, LOAD A CR, LF (%) INTO THE
938 ; TABLE. CONTINUE THIS UNTIL EITHER THE LAST ENTRY IN THE MESSAGE
939 ; TABLE OR THE END OF THE PRINTOUT STORAGE AREA (EXTEND)
940 ; IS REACHED.
941 ; -
942
943
944
945 LET R3 := #EXTMSG MOV #EXTMSG,R3
(4) 002162' 012703 000477' LET FLD CNT := #0 CLR FLD CNT
946 002166' (4) 002166' 005067 175606 WHILE (R2) NE #-1 DO
947 002172' 50012$:
(4) 002172' (6) 002172' 021227 177777 CMP (R2),#-1
(9) 002176' 001436 BEQ 50013$
948 002200' WHILE R3 LO #EXTEND DO
(4) 002200' 50014$:
(6) 002200' 020327 001043' CMP R3,#EXTEND
(9) 002204' 103032 BHIS 50015$
949 002206' WHILE FLD CNT LT #FLD MAX DO
(4) 002206' 50016$:
(6) 002206' 026727 175566 000010 CMP FLD CNT,#FLD MAX
(9) 002214' 002021 SGE 50017$
950 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
951	002232'				LET R3 := R3 + #6		
(6)	002232'	062703	000006			ADD	#6,R3
952	002236'				LET (R3)+ :B= #<SPACE>		
(4)	002236'	112723	000040			MOVB	#<SPACE>,(R3)+
953	002242'				IF (R2) EQ #-1 THEN		
(6)	002242'	021227	177777			CMP	(R2),#-1
(9)	002246'	001001				BNE	50020\$
954	002250'				INLINE <BR		1000\$>
(2)	002250'	000411				BR	1000\$
955	002252'				ENDIF		
(4)	002252'					50020\$:	
956	002252'				LET FLDCNT := FLDCNT + #1		
(6)	002252'	005267	175522			INC	FLDCNT
957	002256'				ENDDO		
(4)	002256'	000753				BR	50016\$
(3)	002260'					50017\$:	
958	002260'				LET (R3)+ :B= #'%		
(4)	002260'	112723	000045			MOVB	#'%,(R3)+
959	002264'				LET FLDCNT := #0		
(4)	002264'	005067	175510			CLR	FLDCNT
960	002270'				ENDDO		
(4)	002270'	000743				BR	50014\$
(3)	002272'					50015\$:	
961	002272'				ENDDO		
(4)	002272'	000737				BR	50012\$
(3)	002274'					50013\$:	
962	002274'				INLINE <1000\$:>		
(2)	002274'					1000\$:	
963	002274'				LET (R3)+ :B= #'%		
(4)	002274'	112723	000045			MOVB	#'%,(R3)+
964	002300'				LET (R3)+ :B= #0		
(4)	002300'	105023				CLRB	(R3)+
965							
966							
967	002302'				ENDIF		
(4)	002302'					50011\$:	
968	002302'				LET FLMSADR(R5) := #DATAMS		
(4)	002302'	012765	000314' 000010			MOV	#DATAMS,FLMSADR(
969	002310'				INLINE <JMP 1\$>		
(2)	002310'	000167	001134			JMP	1\$
970	002314'				ENDIF		
(4)	002314'					50010\$:	
971	002314'				INLINE <100\$:>		
(2)	002314'					100\$:	



```
973 .SBTTL PROCESS SUM COMMAND TYPE MESSAGES
974
975
976
977 ;+
978 ; IS IT A SUM COMMAND HEADER LINE TYPE MESSAGE?
979 ;-
980
981 IF R4 EQ #MSGSMH THEN
(6) 002314' 020427 000014 CMP R4,#MSGSMH
(9) 002320' 001005 ENE 50021$
982 002322' LET FLMSADR(R5) := #RNHMSG
(4) 002322' 012765 000046' 000010 MOV #RNHMSG,FLMSADR(
983 002330' INLINE <JMP 1$> JMP 1$
(2) 002330' 000167 001114
984 002334' ENDIF
(4) 002334' 50021$:
```

```

986 .SBTTL PROCESS SUM COMMAND MODULE LINE MESSAGE
987
988
989
990 ;+
991 ; IS IT A "SUM COMMAND-MODULE-LINE" MESSAGE?
992 ; -
993
994 002334' IF R4 EQ #MSGSMB THEN
(6) 002334' 020427 000015 CMP R4,#MSGSMB
(9) 002340' 001063 BNE 50022$
995
996 ;+
997 ; GET MODULE NAME
998 ; -
999
1000 002342' CALL GETNAM IN <R1, #RBNAM>
(3) 002342' 010546 MOV R5,-(SP)
(5) 002344' 012745 000051' MOV #RBNAM,-(R5)
(4) 002350' 010145 MOV R1,-(R5)
(3) 002352' 004767 001100 JSR PC,GETNAM
(3) 002356' 012605 MOV (SP)+,R5
1001
1002 ;+
1003 ; CONVERT PC,STAT,PASS COUNT,HARD ERRORS AND SOFT ERRORS
1004 ; TO ASCII
1005 ; -
1006 002360' CALL BOA16 IN <R1,#RNBPC>
(3) 002360' 010546 MOV R5,-(SP)
(5) 002362' 012745 000066' MOV #RNBPC,-(R5)
(4) 002366' 010145 MOV R1,-(R5)
(3) 002370' 004767 000000G JSR PC,BOA16
(3) 002374' 012605 MOV (SP)+,R5
1007 002376' CALL BOA16 IN <STAT(R1), #RNBSTAT>
(3) 002376' 010546 MOV R5,-(SP)
(5) 002400' 012745 000102' MOV #RNBSTAT,-(R5)
(4) 002404' 016145 000026 MOV STAT(R1),-(R5)
(3) 002410' 004767 000000G JSR PC,BOA16
(3) 002414' 012605 MOV (SP)+,R5
1008 002416' CALL BDACNV IN <PASCNT(R1), #RNBPAS>
(3) 002416' 010546 MOV R5,-(SP)
(5) 002420' 012745 000117' MOV #RNBPAS,-(R5)
(4) 002424' 016145 000034 MOV PASCNT(R1),-(R5)
(3) 002430' 004767 000000G JSR PC,BDACNV
(3) 002434' 012605 MOV (SP)+,R5
1009 002436' CALL BDACNV IN <HRDCNT(R1), #RNBHRD>
(3) 002436' 010546 MOV R5,-(SP)
(5) 002440' 012745 000135' MOV #RNBHRD,-(R5)
(4) 002444' 016145 000044 MOV HRDCNT(R1),-(R5)
(3) 002450' 004767 000000G JSR PC,BDACNV
(3) 002454' 012605 MOV (SP)+,R5
1010 002456' CALL BDACNV IN <SOF CNT(R1), #RNB SFT>
(3) 002456' 010546 MOV R5,-(SP)
(5) 002460' 012745 000153' MOV #RNB SFT,-(R5)
(4) 002464' 016145 000042 MOV SOF CNT(R1),-(R5)
(3) 002470' 004767 000000G JSR PC,BDACNV
    
```

```
(3) 002474' 012605
1011 002476'
(4) 002476' 012765 000050' 000010
1012 002504'
(2) 002504' 000167 000740
1013 002510'
(4) 002510'
1014
```

LET FLMSADR(R5) := #RNBMSG

INLINE <JMP 1\$>

ENDIF

MOV (SP)+,R5

MOV #RNBMSG,FLMSADR(

JMP 1\$

50022\$:

```

1016          .SBTTL PROCESS SUM COMMAND SUMMARY LINE MESSAGE
1017
1018
1019
1020          ;+
1021          ; IS IT A SUM COMMAND SUMMARY LINE MESSAGE?
1022          ; -
1023
1024 002510'          IF R4 EQ #MSGSMS THEN
(6) 002510' 020427 000016          CMP      R4,#MSGSMS
(9) 002514' 001025          BNE      50023$
1025
1026          ;+
1027          ; CONVERT SYSTEM ERRORS AND POWER FAILS TO ASCII
1028          ; -
1029 002516'          CALL BDACNV IN <DT.EXS(R0), #RNSERR>
(3) 002516' 010546          MOV      R5,-(SP)
(5) 002520' 012745 000201'          MOV      #RNSERR,-(R5)
(4) 002524' 016045 000060          MOV      DT.EXS(R0),-(R5)
(3) 002530' 004767 000000G          JSR      PC,BDACNV
(3) 002534' 012605          MOV      (SP)+,R5
1030 002536'          CALL BDACNV IN <DT.PFL(R0), #RNSPWR>
(3) 002536' 010546          MOV      R5,-(SP)
(5) 002540' 012745 000230'          MOV      #RNSPWR,-(R5)
(4) 002544' 016045 000062          MOV      DT.PFL(R0),-(R5)
(3) 002550' 004767 000000G          JSR      PC,BDACNV
(3) 002554' 012605          MOV      (SP)+,R5
1031 002556'          LET FLMSADR(R5) := #RNSMSG
(4) 002556' 012765 000161' 000010          MOV      #RNSMSG,FLMSADR(
1032 002564'          INLINE <JMP 1$>
(2) 002564' 000167 000660          JMP 1$
1033 002570'          ENDIF
(4) 002570'          50023$:
1034

```

```

1036          .SBTTL  PROCESS DROP MODULE TYPE MESSAGE
1037
1038
1039
1040
1041          ;+
1042          ; IS IT A DROP MODULE MESSAGE?
1043          ; -
1044
1045          IF R4 EQ #MSGDRP THEN
1046          (6) 002570' 020427 000017          CMP      R4,#MSGDRP
1047          (9) 002574' 001025          BNE      50024$
1048
1049          ;+
1050          ; GET MODULE NAME
1051          ; -
1052          CALL GETNAM IN <R1,#DRPNAM>
1053          (3) 002576' 010546          MOV      R5,-(SP)
1054          (5) 002600' 012745 001045'          MOV      #DRPNAM,-(R5)
1055          (4) 002604' 010145          MOV      R1,-(R5)
1056          (3) 002606' 004767 000644          JSR      PC,GETNAM
1057          (3) 002612' 012605          MOV      (SP)+,R5
1058
1059          ;+
1060          ; FORM APC
1061          ; -
1062          LET R3 := R2 - R1
1063          (4) 002614' 010203          MOV      R2,R3
1064          (6) 002616' 160103          SUB      R1,R3
1065
1066          ;+
1067          ; CONVERT DROP PC TO ASCII
1068          ; -
1069          CALL BOA16 IN <R3,#DRPPC>
1070          (3) 002620' 010546          MOV      R5,-(SP)
1071          (5) 002622' 012745 001073'          MOV      #DRPPC,-(R5)
1072          (4) 002626' 010345          MOV      R3,-(R5)
1073          (3) 002630' 004767 000000G          JSR      PC,BOA16
1074          (3) 002634' 012605          MOV      (SP)+,R5
1075
1076          LET FLMSADR(R5) := #DRPMSG
1077          (4) 002636' 012765 001044' 000010          MOV      #DRPMSG,FLMSADR(
1078          002644'          INLINE <JMP 1$>
1079          (2) 002644' 000167 000600          JMP 1$
1080
1081          ENDIF
1082          (4) 002650'
1083          (4) 002650'
1084
1085          50024$:
    
```

```

1068          .SBTTL PROCESS CKDATA/DATAACK/DATERR
1069
1070
1071          ;+
1072          ; IS IT A CKDATA/DATAACK/DATERR TYPE MESSAGE?
1073          ;+
1074          ;-
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          ;
1075 002664'          INLINE <JMP      339$>
(2) 002664' 000167 000460          JMP      339$
1076          ;
1076 002670'          ENDIF
(4) 002670'          ;
1077          ;
1077          ;+
1078          ; GET MODULE NAME
1079          ;+
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

```

1101	(3) 002746' 012605		MOV	(SP)+,R5
1102		;		
1103		;		
1104		;		
1105		;		
1106	002750'	CALL BDACNV IN <PASCNT(R1), #CKDPAS>		
1107	(3) 002750' 010546		MOV	R5,-(SP)
1108	(5) 002752' 012745 001151'		MOV	#CKDPAS,-(R5)
1109	(4) 002756' 016145 000034		MOV	PASCNT(R1),-(R5)
1110	(3) 002762' 004767 000000G		JSR	PC,BDACNV
1111	(3) 002766' 012605		MOV	(SP)+,R5
1112		;		
1113		;		
1114		;		
1115		;		
1116		;		
1117		;		
1118	002770'	CALL BDACNV IN <SOFcnt(R1), #CKDNUM>		
1119	(3) 002770' 010546		MOV	R5,-(SP)
1120	(5) 002772' 012745 001164'		MOV	#CKDNUM,-(R5)
1121	(4) 002776' 016145 000042		MOV	SOFcnt(R1),-(R5)
1122	(3) 003002' 004767 000000G		JSR	PC,BDACNV
1123	(3) 003006' 012605		MOV	(SP)+,R5
1124		;		
1125		;		
1126		;		
1127		;		
1128		;		
1129	003010'	CALL BOA16 IN <CSRA(R1), #CKDADD>		
1130	(3) 003010' 010546		MOV	R5,-(SP)
1131	(5) 003012' 012745 001200'		MOV	#CKDADD,-(R5)
1132	(4) 003016' 016145 000100		MOV	CSRA(R1),-(R5)
1133	(3) 003022' 004767 000000G		JSR	PC,BOA16
1134	(3) 003026' 012605		MOV	(SP)+,R5
1135		;		
1136		;		
1137		;		
1138	003030'	CALL BOA16 IN <ASB(R1), #CKDSB>		
1139	(3) 003030' 010546		MOV	R5,-(SP)
1140	(5) 003032' 012745 001214'		MOV	#CKDSB,-(R5)
1141	(4) 003036' 016145 000106		MOV	ASB(R1),-(R5)
1142	(3) 003042' 004767 000000G		JSR	PC,BOA16
1143	(3) 003046' 012605		MOV	(SP)+,R5
1144		;		
1145		;		
1146		;		
1147	003050'	CALL BOA16 IN <AWAS(R1), #CKDWAS>		
1148	(3) 003050' 010546		MOV	R5,-(SP)
1149	(5) 003052' 012745 001230'		MOV	#CKDWAS,-(R5)
1150	(4) 003056' 016145 000110		MOV	AWAS(R1),-(R5)
1151	(3) 003062' 004767 000000G		JSR	PC,BOA16
1152	(3) 003066' 012605		MOV	(SP)+,R5
1153		;		
1154		;		
1155		;		
1156		;		
1157		;		
1158	003070'	IF R4 NE #MSGDER THEN		
1159	(6) 003070' 020427 000005		CMP	R4,#MSGDER
1160	(9) 003074' 001476		BEQ	50026\$

```

1129          ;+
1130          ; CLEAR MSB OF WORD COUNT
1131          ;-
1132
1133          LET R3 := CDWDCT(R1) CLR.BY #BIT15
1134          (4) 003076' 016103 000144
1135          (6) 003102' 042703 100000
1136          MOV          CDWDCT(R1),R3
1137          BIC          #BIT15,R3
1138
1139          ;+
1140          ; FORM WRITE BUFFER FAILING ADDRESS
1141          ;-
1142
1143          LET R3 := R3 - #1
1144          (6) 003106' 005303
1145          LET R3 := R3 SHIFT 1
1146          (7) 003110' 006303
1147          LET R4 := R3 + WBUFPA(R1)
1148          (4) 003112' 010304
1149          (6) 003114' 066104 000134
1150          MOV          R3,R4
1151          ADD          WBUFPA(R1),R4
1152
1153          LET TMPPA := R4
1154          (4) 003120' 010467 174662
1155          MOV          R4,TMPPA
1156          LET TMPEA := WBUFEA(R1)
1157          (4) 003124' 016167 000136 174656
1158          MOV          WBUFEA(R1),TMPEA
1159          LET FM.PA := TMPPA
1160          (4) 003132' 016767 174650 174652
1161          MOV          TMPPA,FM.PA
1162          LET FM.EA := TMPEA
1163          (4) 003140' 016767 174644 174646
1164          MOV          TMPEA,FM.EA
1165
1166          ;+
1167          ; CONVERT WRITE ADDRESS TO ASCII
1168          ;-
1169
1170          CALL BOAC IN <R0,FM.PA,FM.EA, #CKDWRA>
1171          (3) 003146' 010546
1172          (7) 003150' 012745 001245'
1173          (6) 003154' 016745 174634
1174          (5) 003160' 016745 174626
1175          (4) 003164' 010045
1176          (3) 003166' 004767 000000G
1177          (3) 003172' 012605
1178          MOV          R5,-(SP)
1179          MOV          #CKDWRA,-(R5)
1180          MOV          FM.EA,-(R5)
1181          MOV          FM.PA,-(R5)
1182          MOV          R0,-(R5)
1183          JSR          PC,BOAC
1184          MOV          (SP)+,R5
1185
1186          LET R4 := R2 + #4
1187          (4) 003174' 010204
1188          (6) 003176' 062704 000004
1189          MOV          R2,R4
1190          ADD          #4,R4
1191
1192          LET R3 := R3 + @(R4)
1193          (6) 003202' 067403 000000
1194          ADD          @(R4),R3
1195
1196          LET R4 := (R4) + #2
1197          (4) 003206' 011404
1198          (6) 003210' 062704 000002
1199          MOV          (R4),R4
1200          ADD          #2,R4
1201
1202          LET R4 := (R4)
1203          (4) 003214' 011404
1204          MOV          (R4),R4
1205
1206          LET TMPPA := R3
1207          (4) 003216' 010367 174564
1208          MOV          R3,TMPPA
    
```



```

1161 003222'          LET TMPEA := R4
      (4) 003222' 010467 174562
1162                                     MOV      R4, TMPEA
1163 003226'          LET FM.PA := TMPPA
      (4) 003226' 016767 174554 174556
1164 003234'          LET FM.EA := TMPEA
      (4) 003234' 016767 174550 174552
1165                                     MOV      TMPPA, FM.PA
1166                                     MOV      TMPEA, FM.EA
1167                                     ;+
1168                                     ; CONVERT READ ADDRESS TO ASCII
1169                                     ;-
1170 003242'          CALL BOAC IN <R0, FM.PA, FM.EA, #CKDRDA>
      (3) 003242' 010546
      (7) 003244' 012745 001264'
      (6) 003250' 016745 174540
      (5) 003254' 016745 174532
      (4) 003260' 010045
      (3) 003262' 004767 000000G
      (3) 003266' 012605
1171                                     MOV      R5, -(SP)
1172                                     MOV      #CKDRDA, -(R5)
1173                                     MOV      FM.EA, -(R5)
1174                                     MOV      FM.PA, -(R5)
1175                                     MOV      R0, -(R5)
1176                                     JSR      PC, BOAC
1177                                     MOV      (SP)+, R5
1178                                     ;+
1179                                     ; IT'S A DATERR, SO GET SBADR AND WASADR
1180                                     ;-
1181 ELSE
1182 003270'          BR      50027$
      (4) 003270' 000422
      (3) 003272'
1183 003272'          CALL GETAPA IN <R0, SBADR(R1), #CKDWRA>
      (3) 003272' 010546
      (6) 003274' 012745 001245'
      (5) 003300' 016145 000102
      (4) 003304' 010045
      (3) 003306' 004767 000204
      (3) 003312' 012605
1184 003314'          CALL GETAPA IN <R0, WASADR(R1), #CKDRDA>
      (3) 003314' 010546
      (6) 003316' 012745 001264'
      (5) 003322' 016145 000104
      (4) 003326' 010045
      (3) 003330' 004767 000162
      (3) 003334' 012605
1185 003314'          MOV      R5, -(SP)
1186 003316'          MOV      #CKDRDA, -(R5)
1187 003322'          MOV      WASADR(R1), -(R5)
1188 003326'          MOV      R0, -(R5)
1189 003330'          JSR      PC, GETAPA
1190 003334'          MOV      (SP)+, R5
1191 ENDIF
1192 003336'          BR      50027$
      (4) 003336'
1193
1194 003336'          LET FLMSADR(R5) := #CKDMSG
      (4) 003336' 012765 001103' 000010
1195 003344'          MOV      #CKDMSG, FLMSADR(
      (2) 003344' 000167 000100
1196 003350'          JUMP 1$
      (2) 003350'
1197 003350'          MOV      339$, >
1198                                     JUMP 1$
1199                                     339$:

```

```

1189          .SBTTL PROCESS CKDATA/DATAACK SUMMARY LINE
1190
1191
1192          ;+
1193          ; IS IT THE CKDATA/DATAACK SUMMARY LINE?
1194          ; -
1195
1196          IF R4 EQ #MSGCKS THEN
1197          (6) 003350' 020427 000011          CMP      R4,#MSGCKS
1198          (9) 003354' 001035          BNE      50030$
1199          LET R3 := CDERCT(R1) CLR.BY #BIT15
1200          (4) 003356' 016103 000146          MOV      CDERCT(R1),R3
1201          (6) 003362' 042703 100000          BIC      #BIT15,R3
1202
1203          ;+
1204          ; GET THE MODULE NAME
1205          ; -
1206          CALL GETNAM IN <R1,#CKDSNAM>
1207          (3) 003366' 010546          MOV      R5,-(SP)
1208          (5) 003370' 012745 001276'          MOV      #CKDSNAM,-(R5)
1209          (4) 003374' 010145          MOV      R1,-(R5)
1210          (3) 003376' 004767 000054          JSR      PC,GETNAM
1211          (3) 003402' 012605          MOV      (SP)+,R5
1212
1213          ;+
1214          ; CONVERT SUMMARY COUNT TO ASCII
1215          ; -
1216          CALL BDACNV IN <R3,#CKDSCNT>
1217          (3) 003404' 010546          MOV      R5,-(SP)
1218          (5) 003406' 012745 001310'          MOV      #CKDSCNT,-(R5)
1219          (4) 003412' 010345          MOV      R3,-(R5)
1220          (3) 003414' 004767 000000G          JSR      PC,BDACNV
1221          (3) 003420' 012605          MOV      (SP)+,R5
1222
1223          ;+
1224          ; CONVERT NUMBER OF LOCATIONS READ TO ASCII
1225          ; -
1226          CALL BDACNV IN <RBUFSZ(R1),#CKDSRD>
1227          (3) 003422' 010546          MOV      R5,-(SP)
1228          (5) 003424' 012745 001334'          MOV      #CKDSRD,-(R5)
1229          (4) 003430' 016145 000132          MOV      RBUFSZ(R1),-(R5)
1230          (3) 003434' 004767 000000G          JSR      PC,BDACNV
1231          (3) 003440' 012605          MOV      (SP)+,R5
1232
1233          LET FLMSADR(R5) := #CKDSMSG
1234          (4) 003442' 012765 001276' 000010          MOV      #CKDSMSG,FLMSADR
1235          1216 003450'          ENDIF
1236          (4) 003450'
    
```

50030\$:

```
1218 .SBTTL DO THE NECESSARY CLEAN-UP AND THEN RETURN
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$:>
(2) 003450'
1230 003450' CALL RESREG 1$:
(3) 003450' 004767 000000G JSR PC,RESREG
1231 003454' ENDRTN
(3) 003454' 50000$:
(3) 003454' 50001$:
(2) 003454' 000207 RTS PC
1232
```

```

1234          .SBTTL  GET MODULE NAME
1235
1236          ;+
1237          ; THIS ROUTINE RETRIEVES THE MODULE NAME FROM THE MODULE'S HEADER AND
1238          ; STORES IT IN THE CALLER SUPPLIED STORAGE AREA
1239          ; -
1240
1241 003456'    ROUTINE GETNAM <HDRADR, RSLTADR>
1242          (2) 003456'
1243
1244          ;+
1245          ; SAVE REGISTERS
1246          ; -
1247 003456'    CALL SAVREG
1248          (3) 003456' 004767 000000G
1249
1250          ;+
1251          ; GET HEADER ADDR., MOVE 5 CHARS TO RESULT ADDR.
1252          ; -
1253 003462'    LET R0 := HDRADR(R5)
1254          (4) 003462' 016500 000000
1255          003466'    LET R1 := RSLTADR(R5)
1256          (4) 003466' 016501 000002
1257          003472'    LET R2 := #^D<5>
1258          (4) 003472' 012702 000005
1259          003476'    WHILE R2 GT #0 DO
1260          (4) 003476'
1261          (6) 003476' 005702
1262          (9) 003500' 003403
1263          003502'    LET (R1)+ :B= (R0)+
1264          (4) 003502' 112021
1265          003504'    LET R2 := R2 - #1
1266          (6) 003504' 005302
1267          003506'    ENDDO
1268          (4) 003506' 000773
1269          (3) 003510'
1270          ;+
1271          ; RESTORE REGISTERS
1272          ; -
1273          003510'    CALL RESREG
1274          (3) 003510' 004767 000000G
1275          003514'    ENDRTN
1276          (3) 003514'
1277          (3) 003514'
1278          (2) 003514' 000207
1279
1280          GETNAM:
1281
1282          JSR      PC, SAVREG
1283
1284          MOV      HDRADR(R5), R0
1285          MOV      RSLTADR(R5), R1
1286          MOV      #^D<5>, R2
1287
1288          50002$:
1289          TST      R2
1290          BLE      50003$
1291          MOV      (R0)+, (R1)+
1292          DEC      R2
1293          BR       50002$
1294
1295          50003$:
1296
1297          50000$:
1298          50001$:
1299          RTS      PC

```

```

1269          .SBTTL  CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII
1270
1271
1272          ;+
1273          ; THIS ROUTINE ACCEPTS A 16-BIT VIRTUAL ADDRESS AND RETURNS AN 8-BYTE
1274          ; (22 BIT) PHYSICAL ADDRESS IN THE CALLER SUPPLIER STORAGE AREA
1275          ; -
1276
1277          003516'          ROUTINE GETAPA <DTABLE, VA, RSLTADR>
1278          (2) 003516'
1279
1280          ;+
1281          ; SAVE REGISTERS
1282          ; -
1283
1284          003516'          CALL SAVREG
1285          (3) 003516' 004767 000000G          JSR          PC, SAVREG
1286
1287          ;+
1288          ; SAVE DTABLE ADDR, RESULT ADDR
1289          ; -
1290
1291          003522'          LET R0 := DTABLE(R5)
1292          (4) 003522' 016500 000000          MOV          DTABLE(R5), R0
1293
1294          003526'          LET R1 := RSLTADR(R5)
1295          (4) 003526' 016501 000004          MOV          RSLTADR(R5), R1
1296
1297          003532'          LET TMPVA := VA(R5)
1298          (4) 003532' 016567 000002 174244          MOV          VA(R5), TMPVA
1299
1300          ;+
1301          ; CONVERT TO PA AND THEN TO ASCII
1302          ; -
1303
1304          003540'          CALL GPA IN <R0, #TMPVA>
1305          (3) 003540' 010546          MOV          R5, -(SP)
1306          (5) 003542' 012745 000004'          MOV          #TMPVA, -(R5)
1307          (4) 003546' 010045          MOV          R0, -(R5)
1308          (3) 003550' 004767 000000G          JSR          PC, GPA
1309          (3) 003554' 012605          MOV          (SP)+, R5
1310
1311          003556'          CALL BOAC IN <R0, TMPPA, TMPEA, R1>
1312          (3) 003556' 010546          MOV          R5, -(SP)
1313          (7) 003560' 010145          MOV          R1, -(R5)
1314          (6) 003562' 016745 174222          MOV          TMPEA, -(R5)
1315          (5) 003566' 016745 174214          MOV          TMPPA, -(R5)
1316          (4) 003572' 010045          MOV          R0, -(R5)
1317          (3) 003574' 004767 000000G          JSR          PC, BOAC
1318          (3) 003600' 012605          MOV          (SP)+, R5
1319
1320          ;+
1321          ; RESTORE REGISTERS
1322          ; -
1323
1324          003602'          CALL RESREG
1325          (3) 003602' 004767 000000G          JSR          PC, RESREG
1326
1327          003606'          ENDRTN
1328          (3) 003606'          50000$:
1329          (3) 003606'          50001$:
1330          (2) 003606' 000207          RTS          PC
1331
1332

```

FILLML - FILL MESSAGE (LITTLE) - TO FILL IN SKELETAL MSGS MACY11 30A(1052) 20-SEP-78 17:52 PAGE 19-34  
FILLML.MAC 16-AUG-78 14:36 CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII

SEQ 0479

1304 000001 .END

SYMBOL TABLE

ACSR = 000102	CKDPA 001116R	DT.FCH= 000037	GETAPA 003516R	MODEXH= 004000
ACTBIT= 004000	CKDPAS 001151R	DT.FCN= 000033	GETNAM 003456R	MODHOL= 002000
ADDR22= 001000	CKDRDA 001264R	DT.HMX= 000104	GPA = ***** G	MODSEL= 001000
ADR = 000006	CKDSB 001214R	DT.KBE= 000024	HDRADR= 000000	MSGADR= 000004
APTFER= 000004	CKDSCN 001310R	DT.KBP= 000026	HDRMSG 000237R	MSGCKD= 000010
APTPRE= 000200	CKDSMS 001276R	DT.KBR= 000022	HDRNAM 000240R	MSGCKS= 000011
ASB = 000106	CKDSNA 001276R	DT.KBU= 000030	HDRPA 000252R	MSGDER= 000005
ASSEMB= 000010	CKDSRD 001334R	DT.MLS= 000032	HDRPAS 000305R	MSGDRP= 000017
ASTAT = 000104	CKDVA 001134R	DT.MTI= 000110	HDRPC 000270R	MSGECH= 177777
AUTO = 000010	CKDWAS 001230R	DT.OFF= 000070	HRDCNT= 000044	MSGEDP= 000013
AUTDST= 020000	CKDWRA 001245R	DT.PAS= 000074	HRDPAS= 000050	MSGHDR= 000004
AWAS = 000110	CKHUNG= ***** G	DT.PC = 000002	ICQNT = 000036	MSGHNG= 000022
BDACNV= ***** G	CKTIM = 100000	DT.PFL= 000062	ICOUNT= 000040	MSGHRD= 000007
BIT0 = 000001	CLKPRE= 000001	DT.PSW= 000004	IDNUM = 000122	MSGMAP= 000021
BIT00 = 000001	CONFIG= 000056	DT.PTA= 000064	IE = 000100	MSGNUL= 177775
BIT01 = 000002	CQDVF = 000001	DT.RCS= 000102	INDPAR= 000040	MSGPOP= 000002
BIT02 = 000004	CR = 000015	DT.REL= 000040	INHDRP= 040000	MSGPRM= 177776
BIT03 = 000010	CSRA = 000100	DT.SCT= 000066	INHEPR= 020000	MSGRES= 000001
BIT04 = 000020	CSRC = 000102	DT.SMX= 000106	INHREL= 001000	MSGSFT= 000006
BIT05 = 000040	CTRLC = 000003	DT.SP = 000006	INHRE= 000400	MSGSKE= 000003
BIT06 = 000100	CTRL0 = 000017	DT.SSI= 000046	INIT = 000030	MSGSMB= 000015
BIT07 = 000200	CTRLU = 000025	DT.ST0= 000010	INTR = 000120	MSGSMH= 000014
BIT08 = 000400	DATADD 000415R	DT.ST1= 000012	IOMOD = 100000	MSGSMS= 000016
BIT09 = 001000	DATAMS 000314R	DT.SWR= 000056	IOMODP= 102000	MSGSTD= 000000
BIT1 = 000002	DATAPC 000345R	DT.SYP= 000072	IOMODR= 112000	MSGSYS= 000012
BIT10 = 002000	DATCON 000432R	DT.WBU= 000050	IOMODX= 110000	MSGVEC= 000020
BIT11 = 004000	DATEND 000476R	DT.WHL= 000054	JACK = 035060	NRKMOD= 001000
BIT12 = 010000	DATERR 000467R	DT.WLL= 000052	KIPAR0= 172340	NCPUDP= 000020
BIT13 = 020000	DATNAM 000315R	DVID1 = 000014	KIPAR1= 172342	NCAPTY= 000002
BIT14 = 040000	DATNUM 000401R	ECCMEM= 000100	KIPAR2= 172344	NULL = 000000
BIT15 = 100000	DATPAS 000362R	ECCSTA= 000010	KIPAR3= 172346	OWEN = 024020
BIT2 = 000004	DATPC 000327R	ENBEDP= 010000	KIPAR4= 172350	PAERR = 000010
BIT3 = 000010	DATSTA 000450R	ENBNUL= 000001	KIPAR5= 172352	PARPRE= 002000
BIT4 = 000020	DATTYP 000370R	ENDLST= 000000	KIPAR6= 172354	PARSTA= 000100
BIT5 = 000040	DCEVNT= 000011	EOPBIT= 000001	KIPAR7= 172356	PASCNT= 000034
BIT6 = 000100	DEFRTN= 000400	EOPMSG 000016R	KIPDR0= 172300	PDPLSI= 020000
BIT7 = 000200	DIAGMC= 000000	EOPNAM 000017R	KIPDR1= 172302	PDP60 = 004000
BIT8 = 000400	DROPWO= 100000	EOPPAS 000037R	KIPDR2= 172304	PDP70 = 010000
BIT9 = 001000	DRPMSG 001044R	ERRTYP= 000106	KIPDR3= 172306	PRI0 = 000000
BKDEF = 000002	DRPNAM 001045R	EVNTBE= 000200	KIPDR4= 172310	PRI1 = 000040
BKMOD = 000020	DRPPC 001073R	EVNTHD= 000200	KIPDR5= 172312	PRI4 = 000200
BKMODE= 040000	DSEVNT= 000014	EVNTKT= 000203	KIPDR6= 172314	PRI5 = 000240
BKSLSH= 000134	DTABLE= 000000	EVNTPE= 000202	KIPDR7= 172316	PRI6 = 000300
BOAC = ***** G	DT.ADD= 000042	EVNTRE= 000201	KTERRO= 000040	PRI7 = 000340
BOA16 = ***** G	DT.AP = 000100	EXTEND 001043R	KTPRES= 000400	PRO = 000000
CAPRES= 000004	DT.APK= 000076	EXTMSG 000477R	KTSTAT= 000020	PR4 = 000200
CASAT= 000004	DT.BLS= 000034	FATERR= 100000	KTXTND= 040000	PR5 = 000240
CDERCT= 000146	DT.CFO= 000014	FILLMS 001356RG	LF = 000012	PR6 = 000300
CDWDCT= 000144	DT.CF1= 000016	FLDCNT 000000R	LINECN 000002R	PR7 = 000340
CKDADD 001200R	DT.ERR= 000020	FLDMAX= 000010	LPSTAT= 000001	PS = 177776
CKDMSG 001103R	DT.ESI= 000044	FLMSAD= 000010	MAPSTA= 000200	PSW = 177776
CKDNAM 001104R	DT.EVN= 000000	FM.EA 000014R	MED = 076600	RANNUM= 000054
CKDNUM 001164R	DT.EXS= 000060	FM.PA 000012R	MEMPAS= 040000	RBUFEA= 000130

RBUFFA= 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	SDFCNT= 000042	UIPAR7= 177656	\$F\$NAM= 000160	\$TSK4 = 050014
RESREG= ***** G	SDFPAS= 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	\$SARGC= 000006
RNBHRD 000135R	SR0 = 177572	UIPDR4= 177610	\$F\$SEL= 000140	\$SBYTE= 000403
RNBMSG 000050R	SR1 = 177574	UIPDR5= 177612	\$F\$THE= 000330	\$SCASE= 000000
RNBNAM 000051R	SR2 = 177576	UIPDR6= 177614	\$F\$TRU= 000404	\$SDST = 000000
RNBPAS 000117R	SR3 = 172516	UIPDR7= 177616	\$F\$UNT= 000130	\$SELOC= 000402
RNBPC 000066R	STAT = 000026	VA = 000002	\$F\$WHI= 000120	\$SERFL= 000000
RNBST 000153R	STATBI= 064757	WASADR= 000104	\$F\$YES= 000402	\$SFLAG= 000340
RNBSTA 000102R	STAT1 = 000027	WBSTAT= 000040	\$IFLEV= 177777	\$SFROM= 000000
RNHMSG 000046R	SUSPND= 000001	WBUFEA= 000136	\$ISKO = 000001	\$SLOC = 003500R
RNSERR 000201R	SVRO = 000062	WBUFFA= 000134	\$ISK1 = 000001	\$SLOCN= 000000
RNSMSG 000161R	SVR1 = 000064	WBUFRQ= 000140	\$ISK2 = 000001	\$SREG = 177777
RNSPWR 000230R	SVR2 = 000066	WBUFSZ= 000142	\$LOCTA= 177777	\$SRETU= 000000
RPTDAT= 002000	SVR3 = 000070	WDFR = 000116	\$LSTIN= 000001	\$SRTN1= 050000
RSLTAD= 000004	SVR4 = 000072	WDTO = 000114	\$LSTTA= 000001	\$SRTN2= 050001
RSTRT = 000112	SVR5 = 000074	WTINRE= 000352	\$NESTL= 177777	\$SSRC = 000000
RUBOUT= 000177	SVR6 = 000076	WTWHMI= 000222	\$NSKO = 000300	\$STGSV= 000000
RUNMOD= 100000	SYSCNT= 000052	XFLAG = 000005	\$NSK1 = 000120	\$STGS1= 000000
R5VALU= 001740	SYSERR= 000100	XOFF = 000023	\$NSK2 = 000110	\$STGS2= 000000
SAM = 075464	TMPEA 000010R	XCN = 000021	\$NSK3 = 000120	\$STD = 000000
SAVREG= ***** G	TMPIO = 000002	\$BGNLE= 177777	\$NSK4 = 000120	\$SSTAG= 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	\$SSKO = 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)



3	COMMON EQUATE MODULE
565	COMMON DEFINITIONS AND REFERENCES
569	000000' .PRINT ;SPMAC: VERSION 1.1
620	MESSAGE - END-OF-PASS
635	MESSAGE - SUM COMMAND HEADER LINE
642	MESSAGE - SUM COMMAND MODULE LINE
659	MESSAGE - SUM COMMAND SUMMARY LINE
668	MESSAGE - COMMON HEADER
681	MESSAGE - HRDERR,SOFERR
707	MESSAGE - HRDERR,SOFERR,EXTENSION
714	MESSAGE - DROP MODULE
723	MESSAGE - BAD VECTOR
730	MESSAGE - MAP COMMAND
741	MESSAGE - HUNG MODULE
748	MESSAGE - CKDATA,DATAACK,DATERR
773	MESSAGE - CKDATA/DATAACK SUMMARY LINE
786	FILLMSG - FILL-IN THE BLANKS ROUTINE
808	PROCESS END-OF-PASS TYPE MESSAGE
860	PROCESS COMMON HEADER TYPE MESSAGE
900	PROCESS HRDERR,SOFERR
983	PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?
1037	PROCESS SUM COMMAND TYPE MESSAGES
1051	PROCESS SUM COMMAND MODULE LINE MESSAGE
1081	PROCESS SUM COMMAND SUMMARY LINE MESSAGE
1101	PROCESS DROP MODULE TYPE MESSAGE
1133	PROCESS HUNG MODULE MESSAGE
1148	PROCESS BAD VECTOR TYPE MESSAGE
1173	PROCESS MAP COMMAND TYPE MESSAGE
1201	PROCESS CKDATA/DATAACK/DATERR
1335	PROCESS CKDATA/DATAACK SUMMARY LINE
1364	DO THE NECESSARY CLEAN-UP AND THEN RETURN
1380	GET MODULE NAME
1415	GET ELAPSED TIME
1466	GET MODULE'S PASS TIME
1520	CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII

```

508 .TITLE FILLMS - MODULE TO FILL IN BLANK FIELDS IN SKELETAL MESSAGES
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; FILLMSG
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE IS RESPONSIBLE FOR FIILLING IN THE BLANK FIELDS
517 ; OF THE MONITOR GENERATED SKELETAL MESSAGES.
518 ; THE TYPES OF SKELETAL MESSAGES SUPPORTED ARE:
519 ; 1. END OF PASS 2. HARD ERROR
520 ; 3. SUM COMMAND 4. SOFT ERROR
521 ; 5. COMMON HEADER 6. DROP MODULE
522 ; 7. DATA ERROR 8. BAD VECTOR
523 ; 9. MAP SUMMARY 10. CKDATA CALL
524
525 ; INPUTS:
526 ; 1. DATA TABLE
527 ; 2. MSG TYPE CODE
528 ; 3. MSG ADDR/SPECIAL VALUE
529 ; 4. MODULE'S HDR ADDR
530
531 ; IMPLICIT INPUTS:
532 ; STATUS INDICATOR WORD 0
533
534 ; OUTPUTS:
535 ; 1. ADDRESS OF FILLED-IN MESSAGE
536
537 ; IMPLICIT OUTPUTS:
538 ; NONE
539
540 ; PATHOLOGICAL CONNECTIONS:
541 ; NONE
542
543 ; SUBORDINATE ROUTINES CALLED:
544 ; 1. BDACNV ;BINARY TO DECIMAL ASCII CONVERSION
545 ; 2. UPDTIM ;UPDATE MODULE'S END-OF-PASS TIME
546 ; 3. CKHUNG ;CHECK FOR HUNG OPTION MODULES
547 ; 4. BOAC ;BINARY TO OCTAL-ASCII CONVERSION
548 ; 5. GPA ;GET PHYSICAL ADDRESS
549 ; 6. SAVREG ;
550 ; 7. RESREG ;
551 ; 8. UNIPA ;GET UNIBUS PHYSICAL ADDRESS
552
553 ; FUNCTIONAL SIDE EFFECTS:
554 ; NONE
555
556 ; CALLING SEQUENCE:
557 ; CALL FILLMSG IN <DTABLE,TPCODE,MSGADDR,HDRADDR> OUT <FILMSGADR>
558
559 ; VERSION:
560 ; 0.0
561
562 ; EDIT BY DATE REASON
563 ;

```

```

565 .SBTTL COMMON DEFINITIONS AND REFERENCES
566
567
568 .MCALL STRUCT
569 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
570 000001 $LSTIN = 1
571 000001 $LSTTAG = 1
572
573
574
575 ;
576 ;*****
577 ;
578 ; REFERENCED BY OTHER MODULES
579 ;
580 .GLOBL FILLMSG ;MODULE'S ENTRY POINT
581 ;
582 ;*****
583 ;
584 ; GLOBAL REFERENCES
585 ;
586 .GLOBL BDACNV ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
587 .GLOBL BOA16 ;BINARY TO OCTAL-ASCII CONVERSION ROUTINE
588 .GLOBL BOAC ;BINARY TO OCTAL ASCII CONVERSION
589 .GLOBL CKHUNG ;CHECK FOR HUNG MODULES ROUTINE
590 .GLOBL GPA ;GET PHYSICAL ADDRESS ROUTINE
591 .GLOBL PRHMS ;GET TIME ROUTINE
592 .GLOBL PRPSCN ;GET PASS TIME ROUTINE
593 .GLOBL UPDTIM ;UPDATE MODULES PASS TIME ROUTINE
594 .GLOBL SAVREG ;
595 .GLOBL RESREG ;
596 .GLOBL UNIPA ;GET UNIBUS PHYSICAL ADDRESS
597 ;
598 ;*****
599 ;
600 ; LOCAL EQUATES:
601 ;
602 000010 FLDMAX = ^D<8> ;MAX FIELDS PER LINE
603 ;
604 ;*****
605 ;
606 ; LOCAL STORAGE
607 ;
608 000000' 000000 FLD CNT: .WORD 0 ;FIELD COUNTER
609 000002' 000000 LINECNT: .WORD 0 ;LINE COUNTER
610 000004' TMP1: ;STORAGE WORD
611 000004' 000000 TMPVA: .WORD 0 ;STORAGE/VIRTUAL ADDRESS
612 000006' 000000 TMPPA: .WORD 0 ;STORAGE/PHYSICAL ADDRESS
613 000010' 000000 TMPEA: .WORD 0 ;STORAGE/EA BITS
614 000012' 000000 FM.PA: .WORD 0 ;STORAGE - PHYS. ADDR.
615 000014' 000000 FM.EA: .WORD 0 ;STORAGE - EA BITS
616 ;
617 ;*****
618 ;
    
```

```
620 .SBTTL MESSAGE - END-OF-PASS
621
622
623
624 000016' 045 EOPMSG: .ASCII /%/
625 000017' 000005 EOPNAM: .BLKB ^D<5>
626 000024' 042440 042116 050040 .ASCII / END PASS #/
    000032' 051501 020123 043
627 000037' 000005 EOPPAS: .BLKB ^D<5>
628 000044' 020056 052522 052116 .ASCII /. RUNTIME: /
    000052' 046511 035105 040
629 000057' 000011 EOPTIM: .BLKB ^D<9>
630 000070' 050040 052123 046511 .ASCII / PTIME: /
    000076' 035105 040
631 000101' 000011 EOPPST: .BLKB ^D<9>
632 000112' 000045 .ASCIZ /%/
633
```

```
635 .SBTTL MESSAGE - SUM COMMAND HEADER LINE
636
637 000114' 051445 046525 040515 RNHMSG: .ASCII /%SUMMARY AT RUNTIME: /
    000122' 054522 040440 020124
    000130' 052522 052116 046511
    000136' 035105 040
638 000141' 000011 RNHTIM: .BLKB ^D<9>
639 000152' 000 .BYTE 0 ;ASCII TERMINATOR
640
```

```
642 .SBTTL MESSAGE - SUM COMMAND MODULE LINE
643
644 000153' 045 RNBMSG: .ASCII /%/
645 000154' 000005 RBNBAM: .BLKB ^D<5>
646 000161' 040 052101 053040 .ASCII / AT VA: /
    000166' 035101 040
647 000171' 000006 RNBPC: .BLKB ^D<6>
648 000177' 040 052123 052101 .ASCII / STAT /
    000204' 040
649 000205' 000006 RNBSTAT: .BLKB ^D<6>
650 000213' 040 040520 051523 .ASCII / PASS #/
    000220' 021440
651 000222' 000005 RNBPAS: .BLKB ^D<5>
652 000227' 040 051110 042504 .ASCII / HRDERRS /
    000234' 051122 020123
653 000240' 000005 RNBHRD: .BLKB ^D<5>
654 000245' 040 043123 042524 .ASCII / SFTERRS /
    000252' 051122 020123
655 000256' 000005 RNBSFT: .BLKB ^D<5>
656 000263' 000 .BYTE 0
657
```

```
659 .SBTTL MESSAGE - SUM COMMAND SUMMARY LINE
660
661 000264' 051445 051531 042524 RNSMSG: .ASCII /%SYSTEM ERRORS: /
    000272' 020115 051105 047522
    000300' 051522 020072
662 000304' 000005 RNSERR: .BLKB ^D<5>
663 000311' 040 020040 020040 .ASCII / POWER FAILS: /
    000316' 047520 042527 020122
    000324' 040506 046111 035123
    000332' 040
664 000333' 000005 RNSPWR: .BLKB ^D<5>
665 000340' 000045 .ASCIZ /%/
666
```

```
668 .SBTTL MESSAGE - COMMON HEADER
669
670 000342' 045 HDRMSG: .ASCII /%/
671 000343' 000005 HDRNAM: .BLKB ^D<5>
672 000350' 050040 035101 040 .ASCII / PA: /
673 000355' 000010 HDRPA: .BLKB ^D<8>
674 000365' 040 050101 035103 .ASCII / APC: /
    000372' 040
675 000373' 000006 HDRPC: .BLKB 6
676 000401' 040 040520 051523 .ASCII / PASS #/
    000406' 021440
677 000410' 000005 HDRPAS: .BLKB ^D<5>
678 000415' 045 000 .ASCIZ /%/
679
```



```

681 .SBTTL MESSAGE - HRDERR,SOFERR
682
683 000417' 045 DATAMSG: .ASCII %/
684 000420' 000005 DATNAM: .BLKB ^D<5>
685 000425' 040 040520 020072 .ASCII / PA: /
686 000432' 000010 DATPC: .BLKB ^D<8>
687 000442' 040440 041520 020072 .ASCII / APC: /
688 000450' 000006 DATAPC: .BLKB ^D<6>
689 000456' 050040 051501 020123 .ASCII / PASS #/
    000464' 043
690 000465' 000005 DATPAS: .BLKB ^D<5>
691 000472' 040 .ASCII / /
692 000473' 000004 DATTYP: .BLKB ^D<4> ;DATA, HARD, SOFT
693 000477' 040 051105 021522 .ASCII / ERR#/
694 000504' 000005 DATNUM: .BLKB ^D<5>
695 000511' 045 051503 040522 .ASCII /%CSRA: /
    000516' 020072
696 000520' 000006 DATADD: .BLKB ^D<6>
697 000526' 041440 051123 035103 .ASCII / CSRC: /
    000534' 040
698 000535' 000006 DATCON: .BLKB ^D<6>
699 000543' 040 051501 040524 .ASCII / ASTAT: /
    000550' 035124 040
700 000553' 000006 DATSTAT: .BLKB ^D<6>
701 000561' 040 051105 052122 .ASCII / ERRTP: /
    000566' 050131 020072
702 000572' 000006 DATERR: .BLKB ^D<6>
703 000600' 045 .ASCII %/
704 000601' 000 DATEND: .BYTE 0 ;END OF MESSAGE INDICATOR
705
    
```

707  
708  
709 000602'  
710 000602' 000344  
711 001146' 000  
712

.SBTTL MESSAGE - HRDERR,SOFERR,EXTENSION

EXTMSG:  
.BLKB ^D<228>  
EXTEND: .BYTE 0

```
714 .SBTTL MESSAGE - DROP MODULE
715
716 00:1147' 045 DRPMSG: .ASCII /%/
717 00:1150' 000005 DRPNAM: .BLKB ^D<5>
718 00:1155' 040 051104 050117 .ASCII / DROPPED AT APC: /
    00:1162' 042520 020104 052101
    00:1170' 040440 041520 020072
719 00:1176' 000006 DRPPC: .BLKB ^D<6>
720 00:1204' 000045 .ASCIZ /%/
721
```

```
723 .SBTTL MESSAGE - BAD VECTOR
724
725 001206' 041045 042101 053040 BADVMSG: .ASCII /%BAD VECTOR: /
    001214' 041505 047524 035122
    001222' 040
726 001223' 000006 BADVCT: .BLKB ^D<6>
727 001231' 045 000 .ASCIZ /%/
728
```

```
730 .SBTTL MESSAGE - MAP COMMAND
731
732 001233' 045 MAPMSG: .ASCII /%/
733 001234' 000005 MAPNAM: .BLKB ^D<5>
734 001241' 040 052101 053040 .ASCII / AT VA: /
    001246' 035101 040
735 001251' 000006 MAPPC: .BLKB ^D<6>
736 001257' 040 052123 052101 .ASCII / STAT: /
    001264' 020072
737 001266' 000006 MAPSTT: .BLKB ^D<6>
738 001274' 000 .BYTE 0
739
```

741 .SBTTL MESSAGE - HUNG MODULE  
742  
743 001275' 045 HNGMSG: .ASCII /%/  
744 001276' 000005 HNGNAM: .BLKB ^D<5>  
745 001303' 040 051511 044040 .ASCIZ ' IS HUNG/DROPPED%'  
001310' 047125 027507 051104  
001316' 050117 042520 022504  
001324' 000  
746

```
748 .SBTTL MESSAGE - CKDATA,DATAACK,DATERR
749
750 001325' 045 CKDMSG: .ASCII /%/
751 001326' 000005 CKDNAM: .BLKB ^D<5>
752 001333' 040 040520 020072 .ASCII / PA: /
753 001340' 000010 CKDPA: .BLKB ^D<8>
754 001350' 040440 041520 020072 .ASCII / APC: /
755 001356' 000006 CKDVA: .BLKB ^D<6>
756 001364' 050040 051501 020123 .ASCII / PASS #/
    001372' 043
757 001373' 000005 CKDPAS: .BLKB ^D<5>
758 001400' 042440 051122 021440 .ASCII / ERR #/
759 001406' 000005 CKDNUM: .BLKB ^D<5>
760 001413' 045 051503 040522 .ASCII /%CSRA: /
    001420' 020072
761 001422' 000006 CKDADD: .BLKB ^D<6>
762 001430' 051440 041057 020072 .ASCII " S/B: "
763 001436' 000006 CKDSB: .BLKB ^D<6>
764 001444' 053440 051501 020072 .ASCII / WAS: /
765 001452' 000006 CKDWAS: .BLKB ^D<6>
766 001460' 053440 040522 051104 .ASCII / WRADR /
    001466' 040
767 001467' 000010 CKDWRA: .BLKB ^D<8>
768 001477' 040 042122 042101 .ASCII / RDADR /
    001504' 020122
769 001506' 000010 CKDRDA: .BLKB ^D<8>
770 001516' 000045 .ASCIZ /%/
771
```

```
773 .SBTTL MESSAGE - CKDATA/DATAACK SUMMARY LINE
774
775 001520' CKDSMSG:
776 001520' 000005 CKDSNAM: .BLKB ^D<5>
777 001525' 040 040510 020104 .ASCII / HAD /
778 001532' 000005 CKDSCNT: .BLKB ^D<5>
779 001537' 040 051105 047522 .ASCII / ERRORS OUT OF /
    001544' 051522 047440 052125
    001552' 047440 020106
780 001556' 000005 CKDSRD: .BLKB ^D<5>
781 001563' 040 047527 042122 .ASCIZ / WORDS READ%/
    001570' 020123 042522 042101
    001576' 000045
782
783 .EVEN
784
```



```
786 .SBTTL FILLMSG - FILL-IN THE BLANKS ROUTINE
787
788
789
790
791 001600' ROUTINE FILLMSG <DTABLE,TYP CODE, MSGADR, HDRADR,FLMSADR>
(2) 001600' FILLMSG:
792
793
794
795
796 ;+
797 ; SAVE REGISTERS AND SAVE DTABLE,ADR,MSGADR AND TYP CODE
798 ; -
799
800 001600' CALL SAVREG
(3) 001600' 004767 000000G JSR PC, SAVREG
801 001604' LET R0 := DTABLE(R5) MOV DTABLE(R5),R0
(4) 001604' 016500 000000 MOV HDRADR(R5),R1
802 001610' LET R1 := HDRADR(R5) MOV HDRADR(R5),R1
(4) 001610' 016501 000006 MOV MSGADR(R5),R2
803 001614' LET R2 := MSGADR(R5) MOV MSGADR(R5),R2
(4) 001614' 016502 000004 MOV TYP CODE(R5),R4
804 001620' LET R4 := TYP CODE(R5) MOV TYP CODE(R5),R4
(4) 001620' 016504 000002
805
806
```

```

808 .SBTTL PROCESS END-OF-PASS TYPE MESSAGE
809
810
811
812 ;+
813 ; IS IT AN END-OF-PASS TYPE? IF SO, FILL IN THE BLANK FIELDS,
814 ; UPDATE THE MODULE'S PASS TIME, AND CHECK FOR ANY "HUNG" MODULES
815 ; -
816
817
818 001624' IF R4 EQ #MSGEOP THEN
(6) 001624' 020427 000013 CMP R4,#MSGEOP
(9) 001630' 001057 BNE 50002$
819
820 ;+
821 ; GET MODULE NAME
822 ; -
823
824 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
825
826 ;+
827 ; CONVERT PASS COUNT TO DECIMAL ASCII
828 ; -
829 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
830
831 ;+
832 ; GET ELAPSED TIME CONVERTED TO ASCII
833 ; -
834
835 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
836
837 ;+
838 ; CONVERT PASS TIME TO ASCII
839 ; -
840 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
    
```

```

841
842
843
844           ;+
845           ; IF A CLOCK... UPDATE TIME
846           ; -
847 001726'   IF #CLKPRES SETIN DT.CFO(R0) THEN
(6) 001726' 032760 000001 000014
(9) 001734' 001410
848 001736'   CALL UPDTIM IN <R0,R1>
(3) 001736' 010546
(5) 001740' 010145
(4) 001742' 010045
(3) 001744' 004767 000000G
(3) 001750' 012605
849
850           ;+
851           ; SEE IF ANY MODULE IS HUNG
852           ; -
853
854 001752'   CALL CKHUNG
(3) 001752' 004767 000000G
855 001756'   ENDIF
(4) 001756'
856 001756'   LET FLMSADR(R5) := #EOPMSG
(4) 001756' 012765 000016' 000010
857 001764'   INLINE <JMP 1$>
(2) 001764' 000167 002256
858 001770'   ENDIF
(4) 001770'

```

```

BIT #CLKPRES,DT.CFO(
BEQ 50003$
MOV R5,-(SP)
MOV R1,-(R5)
MOV R0,-(R5)
JSR PC,UPDTIM
MOV (SP)+,R5
JSR PC,CKHUNG
50003$:
MOV #EOPMSG,FLMSADR(
JMP 1$
50002$:

```

```

860 .SBTTL PROCESS COMMON HEADER TYPE MESSAGE
861
862 ;+
863 ; IS IT A COMMON HEADER TYPE MESSAGE?
864 ; -
865
866 IF R4 EQ #MSGHDR THEN
(6) 001770' 020427 000004 CMP R4,#MSGHDR
(9) 001774' 001045 BNE 50004$
867
868 ;+
869 ; GET MODULE NAME
870 ; -
871 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
872
873 ;+
874 ; GET PHYSICAL ADDRESS
875 ; -
876 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
877
878 ;+
879 ; FORM APC
880 ; -
881
882 LET R3 := R2 - R1
(4) 002034' 010203 MOV R2,R3
(6) 002036' 160103 SUB R1,R3
883
884 ;+
885 ; AND CONVERT IT TO ASCII
886 ; -
887
888 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
889
890 ;+
891 ; CONVERT PASCNT TO ASCII
892 ; -
893
894 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'          50004S:
898
```

```

900 .SBTTL PROCESS HRDERR,SOFERR
901
902
903 ;+
904 ; IF NOT A HARD OR SOFT ERROR, CONTINUE CHECKING
905 ; -
906
907
908 IF R4 NE #MSGSFY AND R4 NE #MSGHRD THEN
(6) 002110' 020427 000006 CMP R4,#MSGSFY
(9) 002114' 001405 BEQ 50005$
(6) 002116' 020427 000007 CMP R4,#MSGHRD
(9) 002122' 001402 BEQ 50005$
909 002124' INLINE <JMP 100$> JMP 100$
(2) 002124' 000167 000474
910 002130' ENDFIF
(4) 002130' 50005$:
911 ;+
912 ; INSERT THE END OF MESSAGE CHARACTER AND ASSUME THAT THERE IS NO EXTENDED PRINTOUT
913 ; -
914
915 002130' LET DATEND :B= #0
(4) 002130' 105067 176445 CLRB DATEND
916
917 ;+
918 ; GET MODULE NAME
919 ; -
920
921 002134' CALL GETNAM IN <R1, #DATNAM>
(3) 002134' 010546 MOV R5,-(SP)
(5) 002136' 012745 000420' MOV #DATNAM,-(R5)
(4) 002142' 010145 MOV R1,-(R5)
(3) 002144' 004767 002104 JSR PC,GETNAM
(3) 002150' 012605 MOV (SP)+,R5
922
923 ;+
924 ; GET P.A.
925 ; -
926
927 002152' CALL GETAPA IN <R0,R2,#DATPC>
(3) 002152' 010546 MOV R5,-(SP)
(6) 002154' 012745 000432' MOV #DATPC,-(R5)
(5) 002160' 010245 MOV R2,-(R5)
(4) 002162' 010045 MOV R0,-(R5)
(3) 002164' 004767 002312 JSR PC,GETAPA
(3) 002170' 012605 MOV (SP)+,R5
928
929 ;+
930 ; FORM APC
931 ; -
932
933 002172' LET R3 := R2 - R1
(4) 002172' 010203 MOV R2,R3
(6) 002174' 160103 SUB R1,R3
934
935 ;+
    
```

```

936 ; CONVERT IT TO ASCII
937 ; -
938
939 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
940
941 ; +
942 ; CONVERT PASS COUNT TO ASCII
943 ; -
944
945 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
946
947 LET R3 := #DATTYP
(4) 002234' 012703 000473' MOV #DATTYP, R3
948
949 ; +
950 ; IF SOFT, FILL IN WORD 'SOFT'. IF HARD, FILL IN WORD 'HARD'.
951 ; -
952 IF R4 EQ #MSGSFY THEN
(6) 002240' 020427 000006 CMP R4, #MSGSFY
(9) 002244' 001013 BNE 50006$
953 LET (R3)+ :B= #'S MOVB #'S, (R3)+
(4) 002246' 112723 000123 LET (R3)+ :B= #'D MOVB #'D, (R3)+
954 LET (R3)+ :B= #'F MOVB #'F, (R3)+
(4) 002252' 112723 000117 LET (R3)+ :B= #'T MOVB #'T, (R3)+
955 LET R0 := SOFCNT(R1) MOV SOFCNT(R1), R0
(4) 002256' 112723 000106 ELSE
956 LET (R3)+ :B= #'H BR 50007$
(4) 002262' 112723 000124 LET (R3)+ :B= #'A MOVB #'A, (R3)+
957 LET (R3)+ :B= #'R MOVB #'R, (R3)+
(4) 002266' 016100 000042 LET (R3)+ :B= #'D MOVB #'D, (R3)+
958 ELSE MOV HRDCNT(R1), R0
(4) 002272' 000412 LET (R3)+ :B= #'H MOVB #'H, (R3)+
(3) 002274' LET (R3)+ :B= #'A MOVB #'A, (R3)+
959 LET (R3)+ :B= #'R MOVB #'R, (R3)+
(4) 002274' 112723 000110 LET (R3)+ :B= #'D MOVB #'D, (R3)+
960 LET R0 := HRDCNT(R1) MOV HRDCNT(R1), R0
(4) 002300' 112723 000101
961 LET (R3)+ :B= #'D MOVB #'D, (R3)+
(4) 002304' 112723 000122
962 LET (R3)+ :B= #'D MOVB #'D, (R3)+
(4) 002310' 112723 000104
963 LET R0 := HRDCNT(R1) MOV HRDCNT(R1), R0
(4) 002314' 016100 000044
964 ENDIF
(4) 002320'
965

```

```

966
967          ;+
968          ; CONVERT HARD OR SOFT TO ASCII
969          ; -
970
971 002320'          CALL BDACNV IN <R0,#DATNUM>
(3) 002320' 010546
(5) 002322' 012745 000504'
(4) 002326' 010045
(3) 002330' 004767 000000G
(3) 002334' 012605
972
973          ;+
974          ; CONVERT CSRA,CSRC,ASTAT,ERRTYP TO ASCII
975          ; -
976
977 002336'          CALL BOA16 IN <CSRA(R1), #DATADD>
(3) 002336' 010546
(5) 002340' 012745 000520'
(4) 002344' 016145 000100
(3) 002350' 004767 000000G
(3) 002354' 012605
978 002356'          CALL BOA16 IN <CSRC(R1), #DATCON>
(3) 002356' 010546
(5) 002360' 012745 000535'
(4) 002364' 016145 000102
(3) 002370' 004767 000000G
(3) 002374' 012605
979 002376'          CALL BOA16 IN <ASTAT(R1),#DATSTAT>
(3) 002376' 010546
(5) 002400' 012745 000553'
(4) 002404' 016145 000104
(3) 002410' 004767 000000G
(3) 002414' 012605
980 002416'          CALL BOA16 IN <ERRTYP(R1),#DATERR>
(3) 002416' 010546
(5) 002420' 012745 000572'
(4) 002424' 016145 000106
(3) 002430' 004767 000000G
(3) 002434' 012605
981
    
```

```

MOV R5,-(SP)
MOV #DATNUM,-(R5)
MOV R0,-(R5)
JSR PC,BDACNV
MOV (SP)+,R5

MOV R5,-(SP)
MOV #DATADD,-(R5)
MOV CSRA(R1),-(R5)
JSR PC,BOA16
MOV (SP)+,R5

MOV R5,-(SP)
MOV #DATCON,-(R5)
MOV CSRC(R1),-(R5)
JSR PC,BOA16
MOV (SP)+,R5

MOV R5,-(SP)
MOV #DATSTAT,-(R5)
MOV ASTAT(R1),-(R5)
JSR PC,BOA16
MOV (SP)+,R5

MOV R5,-(SP)
MOV #DATERR,-(R5)
MOV ERRTYP(R1),-(R5)
JSR PC,BOA16
MOV (SP)+,R5
    
```



```

983 .SBTTL PROCESS HRDERR, SOFERR EXTENDED PRINTOUT MESSAGES?
984
985
986
987
988 ;+
989 ; IS THERE A HRDERR/SOFERR EXTENDED PRINTOUT ASSOCIATED WITH THIS MESSAGE?
990 ; -
991 IF R4 EQ #MSGSFY OR R4 EQ #MSGHRD THEN
(6) 002436' 020427 000006 CMP R4,#MSGSFY
(8) 002442' 001403 BEQ 50010$
(6) 002444' 020427 000007 CMP R4,#MSGHRD
(9) 002450' 001065 BNE 50011$
(6) 002452' 50010$:
992 002452' LET R2 := R2 + #4 ADD #4,R2
(6) 002452' 062702 000004 LET R2 := (R2) MOV (R2),R2
993 002456' (4) 002456' 011202 IF R2 NE #0 THEN TST R2
994 002460' (6) 002460' 005702 BEQ 50012$
(9) 002462' 001453 LET DATEND :B= #<CR>
995 002464' (4) 002464' 112767 000015 176107 MOVB #<CR>,DATEND
996
997 ;+
998 ; GO THROUGH THE MESSAGE TABLE AND, FOR EACH ADDRESS IN THE TABLE, FIND
999 ; THE CONTENTS OF THAT ADDRESS, CONVERT IT TO ASCII, STORE THE ASCII IN THE
1000 ; EXTENDED PRINTOUT TABLE (EXTMSG) FOLLOWED BY A SPACE AND, IF THE MAXIMUM
1001 ; NUMBER OF ENTRIES PER LINE HAS BEEN REACHED, LOAD A CR, LF (%) INTO THE
1002 ; TABLE. CONTINUE THIS UNTIL EITHER THE LAST ENTRY IN THE MESSAGE
1003 ; TABLE OR THE END OF THE PRINTOUT STORAGE AREA (EXTEND)
1004 ; IS REACHED.
1005 ; -
1006
1007
1008
1009 002472' LET R3 := #EXTMSG MOV #EXTMSG,R3
(4) 002472' 012703 000602' LET FLD CNT := #0 CLR FLD CNT
1010 002475' (4) 002476' 005067 175276 WHILE (R2) NE #-1 DO 50013$:
1011 002502' (4) 002502' CMP (R2), #-1
(6) 002502' 021227 177777 BEQ 50014$
(9) 002506' 001436 WHILE R3 LO #EXTEND DO 50015$:
1012 002510' (4) 002510' CMP R3,#EXTEND
(6) 002510' 020327 001146' BHIS 50016$
(9) 002514' 103032 WHILE FLD CNT LT #FLD MAX DO 50017$:
1013 002516' (4) 002516' CMP FLD CNT,#FLD MAX
(6) 002516' 026727 175256 000010 BGE 50020$
(9) 002524' 002021 CALL BOA16 IN <@(R2)+,R3>
1014 002526' (3) 002526' 010546 MOV R5,-(SP)
(5) 002530' 010345 MOV R3,-(R5)
(4) 002532' 013245 MOV @(R2)+,-(R5)
    
```

(3)	002534'	004767	000090G			JSR	PC,BOA16
(3)	002540'	012605				MOV	(SP)+,R5
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          IF R4 EQ #MSGSMH THEN
(6) 002624' 020427 000014          CMP      R4,#MSGSMH
(9) 002630' 001014          BNE      50022$
1046          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          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$:
    
```



```
(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
```

```

1081 .SBTTL PROCESS SUM COMMAND SUMMARY LINE MESSAGE
1082
1083
1084
1085 ;+
1086 ; IS IT A SUM COMMAND SUMMARY LINE MESSAGE?
1087 ;-
1088
1089 003036' IF R4 EQ #MSGSMS THEN
(6) 003036' 020427 000016 CMP R4,#MSGSMS
(9) 003042' 001025 BNE 50024$
1090
1091 ;+
1092 ; CONVERT SYSTEM ERRORS AND POWER FAILS TO ASCII
1093 ;-
1094 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
1095 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
1096 003104' LET FLMSADR(R5) := #RNSMSG
(4) 003104' 012765 000264' 000010 MOV #RNSMSG,FLMSADR(
1097 003112' INLINE <JMP 1$>
(2) 003112' 000167 001130 JMP 1$
1098 003116'
(4) 003116'
1099
    
```

ENDIF

50024\$:



```
1133 .SBTTL PROCESS HUNG MODULE MESSAGE
1134
1135
1136 ;+
1137 ; IS IT A HUNG MODULE TYPE MESSAGE?
1138 ;-
1139
1140 003176' IF R4 EQ #MSGHNG THEN
(6) 003176' 020427 000022 CMP R4,#MSGHNG
(9) 003202' 001014 BNE 50026$
1141 003204' CALL GETNAM IN <R1, #HNGNAM>
(3) 003204' 010546 MOV R5,-(SP)
(5) 003206' 012745 001276' MOV #HNGNAM,-(R5)
(4) 003212' 010145 MOV R1,-(R5)
(3) 003214' 004767 001034 JSR PC,GETNAM
(3) 003220' 012605 MOV (SP)+,R5
1142 003222' LET FLMSADR(R5) := #HNGMSG
(4) 003222' 012765 001275' 000010 MOV #HNGMSG,FLMSADR(
1143 003230' INLINE <JMP 1$>
(2) 003230' 000167 001012 JMP 1$
1144 003234' ENDF
(4) 003234' 50026$:
1145
1146
```



```

1148          .SBTTL  PROCESS BAD VECTOR TYPE MESSAGE
1149
1150
1151
1152
1153          ;+
1154          ; IS IT A BAD VECTOR MESSAGE?
1155          ; -
1156
1157          003234'          IF R4 EQ #MSGVEC THEN
1158          (6) 003234' 020427 000020          CMP      R4,#MSGVEC
1159          (9) 003240' 001016          BNE      50027$
1160
1161          ;+
1162          ; SAVE BAD VECTOR ADDRESS IN R2
1163          ; -
1164          003242'          LET R2 := MSGADR(R5)
1165          (4) 003242' 016502 000004          MOV      MSGADR(R5),R2
1166          ;+
1167          ; CONVERT BAD VECTOR TO ASCII
1168          ; -
1169          003246'          CALL BOA16 IN <R2, #BADVCT>
1170          (3) 003246' 010546          MOV      R5,-(SP)
1171          (5) 003250' 012745 001223'          MOV      #BADVCT,-(R5)
1172          (4) 003254' 010245          MOV      R2,-(R5)
1173          (3) 003256' 004767 000000G          JSR      PC,BOA16
1174          (3) 003262' 012605          MOV      (SP)+,R5
1175          003264'          LET FLMSADR(R5) := #BADVMSG
1176          (4) 003264' 012765 001206' 000010          MOV      #BADVMSG,FLMSADR
1177          003272'          INLINE <JMP 1$>
1178          (2) 003272' 000167 000750          JMP 1$
1179          003276'          ENDIF
1180          (4) 003276'
50027$:
    
```

```

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$> JMP 1$
(2) 003366' 000167 000654
1198 003372' ENDIF
(4) 003372' 50030$:
1199
    
```

```

1201 .SBTTL PROCESS CKDATA/DATAACK/DATERR
1202
1203
1204 ;+
1205 ; IS IT A CKDATA/DATAACK/DATERR TYPE MESSAGE?
1206 ;-
1207 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$
1208 003406' INLINE <JMP 339$> JMP 339$
(2) 003406' 000167 000534
1209 003412' ENDIF
(4) 003412' 50031$:
1210
1211 ;+
1212 ; GET MODULE NAME
1213 ;-
1214
1215 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
1216
1217 ;+
1218 ; GET A PA
1219 ;-
1220
1221 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
1222
1223 ;+
1224 ; FORM APC
1225 ;-
1226
1227 003450' LET R3 := R2 - R1
(4) 003450' 010203 MOV R2,R3
(6) 003452' 160103 SUB R1,R3
1228
1229 ;+
1230 ; CONVERT VA TO ASCII
1231 ;-
1232
1233 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
    
```

1234	(3) 003470' 012605		MOV	(SP)+,R5
1235				
1236				
1237				
1238				
1239	003472'		CALL BDACNV IN <PASCNT(R1), #CKDPAS>	
	(3) 003472' 010546		MOV	R5,-(SP)
	(5) 003474' 012745 001373'		MOV	#CKDPAS,-(R5)
	(4) 003500' 016145 000034		MOV	PASCNT(R1),-(R5)
	(3) 003504' 004767 000000G		JSR	PC,BDACNV
	(3) 003510' 012605		MOV	(SP)+,R5
1240				
1241				
1242				
1243				
1244				
1245	003512'		CALL BDACNV IN <SOFcnt(R1), #CKDNUM>	
	(3) 003512' 010546		MOV	R5,-(SP)
	(5) 003514' 012745 001406'		MOV	#CKDNUM,-(R5)
	(4) 003520' 016145 000042		MOV	SOFcnt(R1),-(R5)
	(3) 003524' 004767 000000G		JSR	PC,BDACNV
	(3) 003530' 012605		MOV	(SP)+,R5
1246				
1247				
1248				
1249				
1250				
1251	003532'		CALL BOA16 IN <CSRA(R1), #CKDADD>	
	(3) 003532' 010546		MOV	R5,-(SP)
	(5) 003534' 012745 001422'		MOV	#CKDADD,-(R5)
	(4) 003540' 016145 000100		MOV	CSRA(R1),-(R5)
	(3) 003544' 004767 000000G		JSR	PC,BOA16
	(3) 003550' 012605		MOV	(SP)+,R5
1252	003552'		CALL BOA16 IN <ASB(R1), #CKDSB>	
	(3) 003552' 010546		MOV	R5,-(SP)
	(5) 003554' 012745 001436'		MOV	#CKDSB,-(R5)
	(4) 003560' 016145 000106		MOV	ASB(R1),-(R5)
	(3) 003564' 004767 000000G		JSR	PC,BOA16
	(3) 003570' 012605		MOV	(SP)+,R5
1253	003572'		CALL BOA16 IN <AWAS(R1), #CKDWAS>	
	(3) 003572' 010546		MOV	R5,-(SP)
	(5) 003574' 012745 001452'		MOV	#CKDWAS,-(R5)
	(4) 003600' 016145 000110		MOV	AWAS(R1),-(R5)
	(3) 003604' 004767 000000G		JSR	PC,BOA16
	(3) 003610' 012605		MOV	(SP)+,R5
1254				
1255				
1256				
1257				
1258				
1259	003612'		IF R4 NE #MSGDER THEN	
	(6) 003612' 020427 000005		CMP	R4,#MSGDER
	(9) 003616' 001524		BEQ	50032\$
1260				
1261				

```

1262          ;+
1263          ; CLEAR MSB OF WORD COUNT
1264          ; -
1265
1266          LET R3 := CDWDCT(R1) CLR.BY #BIT15
1267          (4) 003620' 016103 000144          MOV      CDWDCT(R1),R3
1268          (6) 003624' 042703 100000          BIC      #BIT15,R3
1269
1270          ;+
1271          ; FORM WRITE BUFFER FAILING ADDRESS
1272          ; -
1273
1274          LET R3 := R3 - #1
1275          (6) 003630' 005303          DEC      R3
1276          LET R3 := R3 SHIFT 1
1277          (7) 003632' 006303          ASL      R3
1278          LET R4 := R3 + WBUFPA(R1)
1279          (4) 003634' 010304          MOV      R3,R4
1280          (6) 003636' 066104 000134          ADD      WBUFPA(R1),R4
1281
1282          LET TMPPA := R4
1283          (4) 003642' 010467 174140          MOV      R4, TMPPA
1284          LET TMPEA := WBUFEA(R1)
1285          (4) 003646' 016167 000136 174134          MOV      WBUFEA(R1), TMPEA
1286
1287          ;+
1288          ; IF MAPPING ON, CALL UNIPA
1289          ; -
1290
1291          IF #MAPSTAT SETIN DT.STO(R0) THEN
1292          (6) 003654' 032760 000200 000010          BIT      #MAPSTAT,DT.STO(
1293          (9) 003662' 001407          BEQ      50033$
1294          CALL UNIPA IN <#TMPPA>
1295          (3) 003664' 010546          MOV      R5,-(SP)
1296          (4) 003666' 012745 000006'          MOV      #TMPPA,-(R5)
1297          (3) 003672' 004767 000000G          JSR      PC,UNIPA
1298          (3) 003676' 012605          MOV      (SP)+,R5
1299          ELSE
1300          (4) 003700' 000406          BR       50034$
1301          (3) 003702'          50033$:
1302          LET FM.PA := TMPPA
1303          (4) 003702' 016767 174100 174102          MOV      TMPPA,FM.PA
1304          LET FM.EA := TMPEA
1305          (4) 003710' 016767 174074 174076          MOV      TMPEA,FM.EA
1306          ENDIF
1307          (4) 003716'          50034$:
1308          (4) 003716'
1309
1310          ;+
1311          ; CONVERT WRITE ADDRESS TO ASCII
1312          ; -
1313
1314          CALL BOAC IN <R0,FM.PA,FM.EA, #CKDWRA>
1315          (3) 003716' 010546          MOV      R5,-(SP)
1316          (7) 003720' 012745 001467'          MOV      #CKDWRA,-(R5)
1317          (6) 003724' 016745 174064          MOV      FM.EA,-(R5)
    
```

```

(5) 003730' 016745 174056          MOV      FM.PA,-(R5)
(4) 003734' 010045          MOV      R0,-(R5)
(3) 003736' 004767 000000G        JSR      PC,BOAC
(3) 003742' 012605          MOV      (SP)+,R5
1296 003744'                LET R4 := R2 + #4
(4) 003744' 010204          MOV      R2,R4
(6) 003746' 062704 000004          ADD      #4,R4
1297 003752'                LET R3 := R3 + @(R4)
(6) 003752' 067403 000000          ADD      @(R4),R3
1298 003756'                LET R4 := (R4) + #2
(4) 003756' 011404          MOV      (R4),R4
(6) 003760' 062704 000002          ADD      #2,R4
1299 003764'                LET R4 := (R4)
(4) 003764' 011404          MOV      (R4),R4
1300
1301 003766'                LET TMPPA := R3
(4) 003766' 010367 174014          MOV      R3,TMPPA
1302 003772'                LET TMPEA := R4
(4) 003772' 010467 174012          MOV      R4,TMPEA
1303
1304                               ;+
1305                               ; IF MAPPING ON CALL UNIPA
1306                               ; -
1307
1302 003776'                IF #MAPSTAT SETIN DT.STO(R0) THEN
(6) 003776' 032760 000200 000010          BIT      #MAPSTAT,DT.STO(
(9) 004004' 001407          BEQ      50035$
1309 004006'                CALL UNIPA IN <#TMPPA>
(3) 004006' 010546          MOV      R5,-(SP)
(4) 004010' 012745 000006'          MOV      #TMPPA,-(R5)
(3) 004014' 004767 000000G        JSR      PC,UNIPA
(3) 004020' 012605          MOV      (SP)+,R5
1310 004022'                ELSE
(4) 004022' 000406          BR      50036$
(3) 004024'                BR      50035$
1311 004024'                LET FM.PA := TMPPA
(4) 004024' 016767 173756 173760          MOV      TMPPA,FM.PA
1312 004032'                LET FM.EA := TMPEA
(4) 004032' 016767 173752 173754          MOV      TMPEA,FM.EA
1313 004040'                ENDIF
(4) 004040'                BR      50036$
1314
1315                               ;+
1316                               ; CONVERT READ ADDRESS TO ASCII
1317                               ; -
1318
1319 004040'                CALL BOAC IN <R0, FM.PA,FM.EA, #CKDRDA>
(3) 004040' 010546          MOV      R5,-(SP)
(7) 004042' 012745 001506'          MOV      #CKDRDA,-(R5)
(6) 004046' 016745 173742          MOV      FM.EA,-(R5)
(5) 004052' 016745 173734          MOV      FM.PA,-(R5)
(4) 004056' 010045          MOV      R0,-(R5)
(3) 004060' 004767 000000G        JSR      PC,BOAC
(3) 004064' 012605          MOV      (SP)+,R5
1320 004066'                ELSE
(4) 004066' 000422          BR      50037$

```

50032S:

(3) 004070'  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326

```

;+
; GET A P.A.
;-
    
```

1327 004070' 010546  
 (3) 004070' 010546  
 (6) 004072' 012745 001467'  
 (5) 004076' 016145 000102  
 (4) 004102' 010045  
 (3) 004104' 004767 000372  
 (3) 004110' 012605

CALL GETAPA IN <R0,SBADR(R1),#CKDWRA>

```

MOV R5,-(SP)
MOV #CKDWRA,-(R5)
MOV SBADR(R1),-(R5)
MOV R0,-(R5)
JSR PC,GETAPA
MOV (SP)+,R5
    
```

1328 004112' 010546  
 (3) 004112' 010546  
 (6) 004114' 012745 001506'  
 (5) 004120' 016145 000104  
 (4) 004124' 010045  
 (3) 004126' 004767 000350  
 (3) 004132' 012605

CALL GETAPA IN <R0,WASADR(R1),#CKDRDA>

```

MOV R5,-(SP)
MOV #CKDRDA,-(R5)
MOV WASADR(R1),-(R5)
MOV R0,-(R5)
JSR PC,GETAPA
MOV (SP)+,R5
    
```

1329 004134'  
 (4) 004134'  
 1330 004134'  
 (4) 004134' 012765 00132E' 000010  
 1331 004142'  
 (2) 004142' 000167 000100  
 1332 004146'  
 (2) 004146'  
 1333

```

ENDIF
LET FLMSADR(R5) := #CKDMSG
INLINE <JMP 1$>
INLINE <339$:>
    
```

50037S:

```

MOV #CKDMSG,FLMSADR(
JMP 1$
339$:
    
```

```

1335 .SBTTL PROCESS CKDATA/DATAACK SUMMARY LINE
1336
1337
1338 ;+
1339 ; IS IT THE CKDATA/DATAACK SUMMARY LINE?
1340 ; -
1341
1342 004146' IF R4 EQ #MSGCKS THEN
(6) 004146' 020427 000011 CMP R4,#MSGCKS
(9) 004152' 001035 BNE 50040$
1343 004154' LET R3 := CDERCT(R1) CLR.BY #BIT15
(4) 004154' 016103 000146 MOV CDERCT(R1),R3
(6) 004160' 042703 100000 BIC #BIT15,R3
1344
1345 ;+
1346 ; GET THE MODULE NAME
1347 ; -
1348
1349 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
1350
1351 ;+
1352 ; CONVERT SUMMARY COUNT TO ASCII
1353 ; -
1354
1355 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
1356
1357 ;+
1358 ; CONVERT NUMBER OF LOCATIONS READ TO ASCII
1359 ; -
1360 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
1361 004240' LET FLMSADR(R5) := #CKDSMSG
(4) 004240' 012765 001520' 000010 MOV #CKDSMSG,FLMSADR
1362 004246' ENDIF
(4) 004246'

```

50040\$:



1364 .SBTTL DO THE NECESSARY CLEAN-UP AND THEN RETURN

1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375 004246'  
(2) 004246'  
1376 004246'  
(3) 004246' 004767 000000G  
1377 004252'  
(3) 004252'  
(3) 004252'  
(2) 004252' 000207  
1378

;+  
; NOW THAT WE'VE FILLED-IN THE BLANK FIELDS, DO THE CLEAN UP AND RETURN  
;-

INLINE <1\$:>

1\$:

CALL RESREG

JSR PC,RESREG

ENDRTN

50000\$:

50001\$:

RTS PC

```

1380 .SBTTL GET MODULE NAME
1381
1382 ;+
1383 ; THIS ROUTINE RETRIEVES THE MODULE NAME FROM THE MODULE'S HEADER AND
1384 ; STORES IT IN THE CALLER SUPPLIED STORAGE AREA
1385 ;-
1386
1387 004254' ROUTINE GETNAM <HDRADR, RSLTADR>
(2) 004254' GETNAM:
1388
1389 ;+
1390 ; SAVE REGISTERS
1391 ;-
1392
1393 004254' CALL SAVREG
(3) 004254' 004767 000000G JSR PC, SAVREG
1394
1395 ;+
1396 ; GET HEADER ADDR., MOVE 5 CHARS TO RESULT ADDR.
1397 ;-
1398
1399 004260' LET R0 := HDRADR(R5)
(4) 004260' 016500 000000 MOV HDRADR(R5), R0
1400 004264' LET R1 := RSLTADR(R5)
(4) 004264' 016501 000002 MOV RSLTADR(R5), R1
1401 004270' LET R2 := #^D<5>
(4) 004270' 012702 000005 MOV #^D<5>, R2
1402 004274' WHILE R2 GT #0 DO
(4) 004274' 50002$:
(6) 004274' 005702 TST R2
(9) 004276' 003403 BLE 50003$
1403 004300' LET (R1)+ := (R0)+
(4) 004300' 112021 MOVB (R0)+, (R1)+
1404 004302' LET R2 := R2 - #1
(6) 004302' 005302 DEC R2
1405 004304' ENDDO
(4) 004304' 000773 BR 50002$
(3) 004306' 50003$:
1406 ;+
1407 ; RESTORE REGISTERS
1408 ;-
1409
1410 004306' CALL RESREG
(3) 004306' 004767 000000G JSR PC, RESREG
1411 004312' ENDRTN
(3) 004312' 50000$:
(3) 004312' 50001$:
(2) 004312' 000207 RTS PC
1412
1413

```

```

1415          .SBTTL GET ELASPED TIME
1416
1417
1418          ;+
1419          ; THIS ROUTINE RETRIEVES THE EXERCISER'S ELASPED TIME AND RETURNS IT IN
1420          ; THE 9-BYTE FORMAT: "HHH:MM:SS"
1421          ;-
1422
1423 004314'    ROUTINE GETETIM <DTABLE, RSLTADR>
1424 (2) 004314'                                GETETIM:
1425
1426          ;+
1427          ; SAVE REGISTERS
1428          ;-
1429 004314'    CALL SAVREG
1430 (3) 004314' 004767 000000G                JSR      PC, SAVREG
1431
1432          ;+
1433          ; SAVE DTABLE ADDR., RESULT ADDR.
1434          ;-
1435 004320'    LET R0 := DTABLE(R5)
1436 (4) 004320' 016500 000000                MOV      DTABLE(R5), R0
1437 004324'    LET R1 := RSLTADR(R5)
1438 (4) 004324' 016501 000002                MOV      RSLTADR(R5), R1
1439
1440          ;+
1441          ; IF A CLOCK ON SYSTEM, CONVERT ELAPSED TIME INTO HRS,
1442          ; MIN., SECS.
1443          ;-
1444 004330'    IF #CLKPRES SETIN DT.CFO(R0) THEN
1445 (6) 004330' 032760 000001 000014        BIT      #CLKPRES, DT.CFO(
1446 (9) 004336' 001407                        BEQ      50002$
1447 004340'    CALL PRHMS IN <R0, R1>
1448 (3) 004340' 010546                        MOV      R5, -(SP)
1449 (5) 004342' 010145                        MOV      R1, -(R5)
1450 (4) 004344' 010045                        MOV      R0, -(R5)
1451 (3) 004346' 004767 000000G                JSR      PC, PRHMS
1452 (3) 004352' 012605                        MOV      (SP)+, R5
1453 004354'    ELSE
1454 (4) 004354' 000410                        BR       50003$
1455 (3) 004356'                                50002$:
1456
1457          ;+
1458          ; IF NOT, FILL IN BLANKS WITH SPACES
1459          ;-
1460
1461 004356'    LET R2 := #^D<9>
1462 (4) 004356' 012702 000011                MOV      #^D<9>, R2
1463 004362'    WHILE R2 GT #0 DD
1464 (4) 004362'                                50004$:
1465 (6) 004362' 005702                        TST      R2
1466 (9) 004364' 003404                        BLE      50005$
    
```

```
1454 004366'          LET (R1)+ :B= #<SPACE>
(4) 004366' 112721 000040
1455 004372'          LET R2 := R2 - #1
(6) 004372' 005302
1456 004374'          ENDDO
(4) 004374' 000772
(3) 004376'          50005$: BR      50004$
1457 004376'          ENDIF
(4) 004376'          50003$:
1458
1459 ;+
1460 ; RESTORE REGISTERS
1461 ; -
1462 004376'          CALL RESREG
(3) 004376' 004767 000000G
1463 004402'          ENDRTN
(3) 004402'          50000$:
(3) 004402'          50001$:
(2) 004402' 000207          RTS      PC
1464
```

```

1466 .SBTTL GET MODULE'S PASS TIME
1467
1468
1469 ;+
1470 ; THIS ROUTINE GETS THE OPTION MODULE'S PASS TIME AND RETURNS IT IN THE
1471 ; 9-BYTE FORMAT: "HHH:MM:SS"
1472 ;-
1473
1474 004404' ROUTINE GETPTIM <DTABLE,HDRADR,RSLTADR>
(2) 004404' GETPTIM:
1475
1476 ;+
1477 ; SAVE REGISTERS
1478 ;-
1479
1480 004404' CALL SAVREG
(3) 004404' 004767 000000G JSR PC,SAVREG
1481
1482 ;+
1483 ; SAVE DTABLE ADDR,HEADER ADDR,RESULT ADDR.
1484 ;-
1485
1486 004410' LET R0 := DTABLE(R5)
(4) 004410' 016500 000000 MOV DTABLE(R5),R0
1487 004414' LET R1 := HDRADR(R5)
(4) 004414' 016501 000002 MOV HDRADR(R5),R1
1488 004420' LET R2 := RSLTADR(R5)
(4) 004420' 016502 000004 MOV RSLTADR(R5),R2
1489
1490 ;+
1491 ; IF A CLOCK ON SYSTEM, UPDATE MODULE'S PASS TIME
1492 ;-
1493
1494 004424' IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 004424' 032760 000001 000014 BIT #CLKPRES,DT.CF0(
(9) 004432' 001410 BEQ 50002$
1495 004434' CALL PRPSCNT IN <R0,R1,R2>
(3) 004434' 010546 MOV R5,-(SP)
(6) 004436' 010245 MOV R2,-(R5)
(5) 004440' 010145 MOV R1,-(R5)
(4) 004442' 010045 MOV R0,-(R5)
(3) 004444' 004767 000000G JSR PC,PRPSCNT
(3) 004450' 012605 MOV (SP)+,R5
1496 004452' ELSE
(4) 004452' 000410 BR 50003$
(3) 004454' 50002$:
1497
1498 ;+
1499 ; INSERT SPACES
1500 ;-
1501
1502 004454' LET R3 := #^D<9>
(4) 004454' 012703 000011 MOV #^D<9>,R3
1503 004460' WHILE R3 GT #0 DO
(4) 004460' 50004$:
(6) 004460' 005703 TST R3
    
```



```
1509
1510          ;+
1511          ; RESTORE REGISTERS
1512          ; -
1513
1514 004474'          CALL RESREG
   (3) 004474' 004767 000000G          JSR          PC,RESREG
1515 004500'          ENDRTN
   (3) 004500'          50000$:
   (3) 004500'          50001$:
   (2) 004500' 000207          RTS          PC
1516
1517
1518
```

```

1520 .SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII
1521
1522
1523 ;+
1524 ; THIS ROUTINE ACCEPTS A 16-BIT VIRTUAL ADDRESS AND RETURNS AN 8-BYTE
1525 ; (22 BIT) PHYSICAL ADDRESS IN THE CALLER SUPPLIER STORAGE AREA
1526 ; -
1527
1528 004502' ROUTINE GETAPA <DTABLE, VA, RSLTADR>
(2) 004502' GETAPA:
1529
1530
1531 ;+
1532 ; SAVE REGISTERS
1533 ; -
1534
1535 004502' CALL SAVREG
(3) 004502' 004767 000000G JSR PC, SAVREG
1536
1537 ;+
1538 ; SAVE DTABLE ADDR, RESULT ADDR
1539 ; -
1540 004506' LET R0 := DTABLE(R5)
(4) 004506' 016500 000000 MOV DTABLE(R5), R0
1541 004512' LET R1 := RSLTADR(R5)
(4) 004512' 016501 000004 MOV RSLTADR(R5), R1
1542 004516' LET TMPVA := VA(R5)
(4) 004516' 016567 000002 173260 MOV VA(R5), TMPVA
1543 ;+
1544 ; CONVERT TO PA AND THEN TO ASCII
1545 ; -
1546 004524' CALL GPA IN <R0, #TMPVA>
(3) 004524' 010546 MOV R5, -(SP)
(5) 004526' 012745 000004' MOV #TMPVA, -(R5)
(4) 004532' 010045 MOV R0, -(R5)
(3) 004534' 004767 000000G JSR PC, GPA
(3) 004540' 012605 MOV (SP)+, R5
1547 004542' CALL BOAC IN <R0, TMPPA, TMPEA, R1>
(3) 004542' 010546 MOV R5, -(SP)
(7) 004544' 010145 MOV R1, -(R5)
(6) 004546' 016745 173236 MOV TMPEA, -(R5)
(5) 004552' 016745 173230 MOV TMPPA, -(R5)
(4) 004556' 010045 MOV R0, -(R5)
(3) 004560' 004767 000000G JSR PC, BOAC
(3) 004564' 012605 MOV (SP)+, R5
1548
1549 ;+
1550 ; RESTORE REGISTERS
1551 ; -
1552 004566' CALL RESREG
(3) 004566' 004767 000000G JSR PC, RESREG
1553 004572' ENDRTN
(3) 004572' 50000$:
(3) 004572' 50001$:
(2) 004572' 000207 RTS PC
1554
    
```



FILLMS - MODULE TO FILL IN BLANK FIELDS IN SKELETAL MESSAGES MACY11 30A(1052) 20-SEP-78 17:53 PAGE 20-2  
FILLMS.MAC 15-AUG-78 16:52 CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS ASCII

SEQ 0530

1555 000001 .END

SYMBOL TABLE

ACSR = 000102	CKDNAM 001326R	DT.EVN= 000000	FLDMAX= 000010	KTSTAT= 000020
ACTBIT= 004000	CKDNUM 001406R	DT.EXS= 000060	FLMSAD= 000010	KTXTND= 040000
ADDR22= 001000	CKDPA 001340R	DT.FCH= 000037	FM.EA 000014R	LF = 000012
ADR = 000006	CKDPAS 001373R	DT.FCN= 000036	FM.PA 000012R	LINECN 000002R
APTFER= 000004	CKDRDA 001506R	DT.HMX= 000104	GETAPA 004502R	LPSTAT= 000001
APTPRE= 000200	CKDSB 001436R	DT.KBE= 000024	GETETI 004314R	MAPMSG 001233R
ASB = 000106	CKDSCN 001532R	DT.KBP= 000026	GETNAM 004254R	MAPNAM 001234R
ASSEMB= 000010	CKDSMS 001520R	DT.KBR= 000022	GETPTI 004404R	MAPPCC 001251R
ASTAT = 000104	CKDSNA 001520R	DT.KBU= 000030	GPA = ***** G	MAPSTA= 000200
AUTO = 000010	CKDSRD 001556R	DT.MLS= 000032	HDRADR= 000002	MAPSTT 001266R
AUTOST= 020000	CKDVA 001356R	DT.MTI= 000110	HDRMSG 000342R	MED = 076600
AWAS = 000110	CKDWAS 001452R	DT.OFF= 000070	HDRNAM 000343R	MEMPAS= 040000
BADVCT 001223R	CKDWRA 001467R	DT.PAS= 000074	HDRPA 000355R	MODEXH= 004000
BADVMS 001206R	CKHUNG= ***** G	DT.PC = 000002	HDRPAS 000410R	MODHOL= 002000
BDACNV= ***** G	CKTIM = 100000	DT.PFL= 000062	HDRPC 000373R	MJDSEL= 001000
BIT0 = 000001	CLKPRE= 000001	DT.PSW= 000004	HNGMSG 001275R	MSGADR= 000004
BIT00 = 000001	CONFIG= 000056	DT.PTA= 000064	HNGNAM 001276R	MSGCKD= 000010
BIT01 = 000002	CQOVF = 000001	DT.RCS= 000102	HRDCNT= 000044	MSGCKS= 000011
BIT02 = 000004	CR = 000015	DT.REL= 000040	HRDPAS= 000050	MSGDER= 000005
BIT03 = 000010	CSRA = 000100	DT.SCT= 000066	ICONT = 000036	MSGDRP= 000017
BIT04 = 000020	CSRC = 000102	DT.SMX= 000106	ICDUNT= 000040	MSGGECH= 177777
BIT05 = 000040	CTRLC = 000003	DT.SP = 000006	IDNUM = 000122	MSGGOP= 000013
BIT06 = 000100	CTRL0 = 000017	DT.SSI= 000046	IE = 000100	MSGHDR= 000004
BIT07 = 000200	CTRLU = 000025	DT.ST0= 000010	INDPAR= 000040	MSGHNG= 000022
BIT08 = 000400	DATADD 000520R	DT.ST1= 000012	INHDRP= 040000	MSGHRD= 000007
BIT09 = 001000	DATAMS 000417R	DT.SWR= 000056	INHEPR= 020000	MSGMAP= 000021
BIT1 = 000002	DATAPC 000450R	DT.SYP= 000072	INHREL= 001000	MSGNUL= 177775
BIT10 = 002000	DATCON 000535R	DT.WBU= 000050	INHRR= 000400	MSGPOP= 000002
BIT11 = 004000	DATEND 000601R	DT.WHL= 000054	INIT = 000030	MSGPRM= 177776
BIT12 = 010000	DATERR 000572R	DT.WLL= 000052	INTR = 000120	MSGRES= 000001
BIT13 = 020000	DATNAM 000420R	DVID1 = 000014	IOMOD = 100000	MSGSFT= 000006
BIT14 = 040000	DATNUM 000504R	ECCMEM= 000100	ICMODP= 102000	MSGSKE= 000003
BIT15 = 100000	DATPAS 000465R	ECCSTA= 000010	IOMODR= 112000	MSGSMB= 000015
BIT2 = 000004	DATPC 000432R	ENBEOP= 010000	IOMODX= 110000	MSGSMH= 000014
BIT3 = 000010	DATSTA 000553R	ENBNUL= 000001	JACK = 035060	MSGSMS= 000016
BIT4 = 000020	DATTYP 000473R	ENDLST= 000000	KIPAR0= 172340	MSGSTD= 000000
BIT5 = 000040	DCEVNT= 000011	EOPBIT= 000001	KIPAR1= 172342	MSGSYS= 000012
BIT6 = 000100	DEFRTN= 000400	EOPMSG 000016R	KIPAR2= 172344	MSGVEC= 000020
BIT7 = 000200	DIAGMC= 000000	EOPNAM 000017R	KIPAR3= 172346	NBKMOD= 001000
BIT8 = 000400	DROPMO= 100000	EOPPAS 000037R	KIPAR4= 172350	NCPUOP= 000020
BIT9 = 001000	DRPMSG 001147R	EOPPST 000101R	KIPAR5= 172352	NDAPTY= 000002
BKDEF = 000002	DRPNAM 001150R	EOPTIM 000057R	KIPAR6= 172354	NULL = 000000
BKMOD = 000020	DRPPC 001176R	ERRTYP= 000106	KIPAR7= 172356	OWEN = 024020
BKMODE= 040000	DSEVNT= 000014	EVNTBE= 000200	KIPDR0= 172300	PAERR = 000010
BKSLSH= 000134	DTABLE= 000000	EVNTHD= 000200	KIPDR1= 172302	PARPRE= 002000
BOAC = ***** G	DT.ADD= 000042	EVNTKT= 000203	KIPDR2= 172304	PARSTA= 000100
BOA16 = ***** G	DT.AP = 000100	EVNTPE= 000202	KIPDR3= 172306	PASCNT= 000034
CAPRES= 000004	DT.APK= 000076	EVNTRE= 000201	KIPDR4= 172310	PDPLSI= 020000
CASTAT= 000004	DT.BLS= 000034	EXTEND 001146R	KIPDR5= 172312	PDP60 = 004000
CDERCT= 000146	DT.CFO= 000014	EXTMSG 000602R	KIPDR6= 172314	PDP70 = 010000
CDWDCT= 000144	DT.CF1= 000016	FATERR= 100000	KIPDR7= 172316	PRHMS = ***** G
CKDADD 001422R	DT.ERR= 000020	FILLMS 001600RG	KTERR0= 000040	PRI0 = 000000
CKDMSG 001325R	DT.ESI= 000044	FLDCNT 000000R	KTPRES= 000400	PRI1 = 000040

PRI4 = 000200	RSLTAD= 000004	SYSCNT= 000052	XOFF = 000023	\$NSK5 = 000120
PRI5 = 000240	RSTRT = 000112	SYSERR= 000100	XDN = 000021	\$NSK6 = 000110
PRI6 = 000300	RUBOUT= 000177	TMPEA 000010R	\$BGNLE= 177777	\$SAVLE= 177777
PRI7 = 000340	RUNMOD= 100000	TMPID = 000002	SERFLG= 000400	\$SSKO = 050005
PRPSCN= ***** G	RSVALU= 001740	TMPVA 000006R	SFSAND= 000310	\$TAGLE= 177777
PRO = 000000	SAM = 075464	TMPVA 000004R	SFSBAD= 000401	\$TAGNU= 050002
PR4 = 000200	SAVREG= ***** G	TMP1 000004R	\$FSBLA= 000170	\$TEMP = 000300
PR5 = 000240	SBADR = 000102	TQOVF = 000002	\$FSCAS= 000150	\$TSK0 = 050003
PR6 = 000300	SBKMOD= 000000	TYPCOD= 000002	\$FSDC= 000220	\$TSK1 = 050004
PR7 = 000340	SBKSEL= 010000	UIPAR0= 177640	\$FSDO = 000340	\$TSK10= 050021
PS = 177776	SC.ADR= 000006	UIPAR1= 177642	\$FSFAL= 000405	\$TSK2 = 050005
PSW = 177776	SC.ALC= 000014	UIPAR2= 177644	\$FSGOO= 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.CLD= 000000	UIPAR6= 177654	\$F\$NAM= 000160	\$TSK7 = 050020
RBUFVA= 000124	SC.HLD= 000010	UIPAR7= 177656	\$F\$NO = 000403	\$SARGC= 000006
RDSERV= 000101	SC.SCA= 000012	UIPDR0= 177600	\$F\$OR = 000320	\$S\$BYTE= 000403
RDWHMI= 000022	SENDLS= 177777	UIPDR1= 177602	\$F\$RTI= 000350	\$S\$CASE= 000000
RELERR= 000020	SOFCNT= 000042	UIPDR2= 177604	\$F\$RTN= 000300	\$S\$DST = 000000
RELMOD= 020000	SOFPAS= 000046	UIPDR3= 177606	\$F\$SEL= 000140	\$S\$LOC= 000402
RELTIM= 010000	SPACE = 000040	UIPDR4= 177610	\$F\$THE= 000330	\$S\$ERFL= 000000
RESREG= ***** G	SPOINT= 000032	UIPDR5= 177612	\$F\$TRU= 000404	\$S\$FLAG= 000001
RES1 = 000056	SPVALU= 002200	UIPDR6= 177614	\$F\$UNT= 000130	\$S\$FROM= 000000
RES2 = 000060	SR0 = 177572	UIPDR7= 177616	\$F\$WHI= 000120	\$S\$LOC = 004462R
RICHAR= 031060	SR1 = 177574	UNIPA = ***** G	\$F\$YES= 000402	\$S\$LOCN= 000000
RNBHRD 000240R	SR2 = 177576	UPDTIM= ***** G	\$IFLEV= 177777	\$S\$REG = 177777
RNBMSG 000153R	SR3 = 172516	VA = 000002	\$ISKO = 000001	\$S\$RETU= 000000
RNBNAM 000154R	STAT = 000026	WASADR= 000104	\$ISK1 = 000001	\$S\$RTN1= 050000
RNBPAS 000222R	STATBI= 064757	WBSTAT= 000040	\$ISK2 = 000001	\$S\$RTN2= 050001
RNBPC 000171R	STAT1 = 000027	WBUFEA= 000136	\$LOCTA= 177777	\$S\$SRC = 000000
RNBSFT 000256R	SUSPND= 000001	WBUFPA= 000134	\$LSTIN= 000001	\$S\$TGSV= 000000
RNBSTA 000205R	SVR0 = 000062	WBUFQ= 000140	\$LSTTA= 000001	\$S\$TGS1= 000000
RNHMSG 000114R	SVR1 = 000064	WBUFSZ= 000142	\$NESTL= 177777	\$S\$TGS2= 000000
RNHTIM 000141R	SVR2 = 000066	WDFR = 000116	\$NSKO = 000300	\$S\$TC = 000000
RNSERR 000304R	SVR3 = 000070	WDTO = 000114	\$NSK1 = 000110	\$S\$TAG= 050000
RNSMSG 000264R	SVR4 = 000072	WTINRE= 000352	\$NSK2 = 000120	= 004574R
RNSPWR 000333R	SVR5 = 000074	WTHMI= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:55

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0533

3 COMMON EQUATE MODULE  
552 GETPSW (COMMON DEFINITIONS & REFERENCES)  
555 000000' .PRINT ;SPMAC: VERSION 1.1  
565 GETPSW (CODE)

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550

```
.TITLE GETPSW (GET CALLER'S PS WORD)
.IDENT /V0.0/

;++;
; MODULE NAME:
;     GETPSW
;
; FUNCTIONAL DESCRIPTION:
;     THIS MODULE RETURNS THE CALLER'S PROCESSOR STATUS WORD.
;
; INPUTS:
;     NONE
;
; IMPLICIT INPUTS:
;     NONE
;
; OUTPUTS:
;     PROCESSOR STATUS WORD
;
; IMPLICIT OUTPUTS:
;     NONE
;
; PATHOLOGICAL CONNECTIONS:
;     NONE
;
; SUBORDINATE MODULES CALLED:
;     NONE
;
; FUNCTIONAL SIDE EFFECTS:
;     NONE
;
; CALLING SEQUENCE:
;     CALL GETPSW OUT <AA>
;     WHERE AA = PS WORD
;
; VERSION:
;     0.0
;
;     EDIT      DATE      BY      REASON
;     -----
;
;-----
```

GETPSW (GET CALLER'S PS WORD)  
GETPSW.MAC 28-JUL-78 09:15

MACY11 30A(1052) 20-SEP-78 17:55 PAGE 19-1  
GETPSW (COMMON DEFINITIONS & REFERENCES)

SEQ 0535

```
552 .SBTTL GETPSW (COMMON DEFINITIONS & REFERENCES)
553
554 .MCALL STRUCT
555 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
556 000001 $LSTIN=1
557 000001 $LSTTAG=1
558
559 ;*****
560 ; REFERENCED BY OTHER MODULES
561 ;
562 .GLOBL GETPSW ;MODULE'S ENTRY POINT
563
```

```
565 .SBTTL GETPSW (CODE)
566
567 000000' ROUTINE GETPSW <AA>
(2) 000000' GETPSW:
568
569 ;+
570 ; SAVE VECTOR 10; LOAD ADDRESS TO HANDLE ILLEGAL INSTRUCTION
571 ; TRAP. EXECUTE AN ILLEGAL INSTRUCTION. THE TRAP WILL PUSH PSW
572 ; AND PC+ ON THE STACK. IN THE TRAP HANDLER, GET THE PSW AND
573 ; RETURN TO PC+ BY DOING AN RTI. NOW RESTORE THE ORIGINAL
574 ; TRAP HANDLER AND RETURN.
575 ;-
576
577 000000' PUSH @#10
(2) 000000' 013746 000010 MOV @#10,-(SP)
578 000004' LET @#10 := #1$
(4) 000004' 012737 000022' 000010 MOV #1$,@#10
579 000012' INLINE <#77>
(2) 000012' 000077 #77
580 000014' POP @#10
(2) 000014' 012637 000010 MOV (SP)+,@#10
581 000020' ENDRTN
(3) 000020' 50000$:
(3) 000020' 50001$:
(2) 000020' 000207 RTS PC
582
583 000022' INLINE <1$:>
(2) 000022' 1$:
584 000022' LET AA(R5) := 2(SP)
(4) 000022' 016665 000002 000000 MOV 2(SP),AA(R5)
585 000030' INLINE <RTI> ;RETURN
(2) 000030' 000002 RTI
586
587
588 000001 .END
```

AA = 000000	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ACSR = 000102	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RDSERV= 000101
ACTBIT= 004000	CTRLD = 000017	EVNTBE= 000200	MSGCKD= 000010	RDWHMI= 000022
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RELERR= 000020
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RELMOD= 020000
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RELTIM= 010000
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGGECH= 177777	RES1 = 000056
ASB = 000106	DROPMO= 100000	FATERR= 100000	MSGGEO= 000013	RES2 = 000060
ASSEMB= 000010	DSEVNT= 000014	GETPSW 000000ORG	MSGHDR= 000004	RICHAR= 031060
ASTAT = 000104	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RPTDAT= 002000
AUTO = 000010	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RSTRT = 000112
AUTOST= 020000	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RJBOUT= 000177
AWAS = 000110	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RUNMOD= 100000
BIT0 = 000001	DT.CF0= 000014	IDNUM = 000122	MSGPOP= 000002	R5VALU= 001740
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	SAM = 075464
BIT01 = 000002	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	SBADR = 000102
BIT02 = 000004	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SBKMOD= 000000
BIT03 = 000010	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	SBKSEL= 010000
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	SC.ADR= 000006
BIT05 = 000040	DT.FCH= 000037	INHRR= 000400	MSGSMH= 000014	SC.ALC= 000014
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SC.APC= 000016
BIT07 = 000200	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.CKL= 000002
BIT08 = 000400	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT09 = 001000	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT1 = 000002	DT.KBR= 000022	IOMODR= 112000	NBKM0D= 001000	SC.HLD= 000010
BIT10 = 002000	DT.KBU= 000030	IOMODX= 110000	NCPUOP= 000020	SC.SCA= 000012
BIT11 = 004000	DT.MLS= 000032	JACK = 035060	NDAPTY= 000002	SENDLS= 177777
BIT12 = 010000	DT.MTI= 000110	KIPAR0= 172340	NULL = 000000	SOFCNT= 000042
BIT13 = 020000	DT.OFF= 000070	KIPAR1= 172342	OWEN = 024020	SOFPAS= 000046
BIT14 = 040000	DT.PAS= 000074	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT15 = 100000	DT.PC = 000002	KIPAR3= 172346	PARPRE= 002000	SPOINT= 000032
BIT2 = 000004	DT.PFL= 000062	KIPAR4= 172350	PARSTA= 000100	SPVALU= 002200
BIT3 = 000010	DT.PSW= 000004	KIPAR5= 172352	PASCNT= 000034	SR0 = 177572
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PDPLSI= 020000	SR1 = 177574
BIT5 = 000040	DT.RCS= 000102	KIPAR7= 172356	PDP60 = 004000	SR2 = 177576
BIT6 = 000100	DT.REL= 000040	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
BIT7 = 000200	DT.SCT= 000066	KIPDR1= 172302	PRI0 = 000000	STAT = 000026
BIT8 = 000400	DT.SMX= 000106	KIPDR2= 172304	PRI1 = 000040	STATBI= 064757
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PRI4 = 000200	STAT1 = 000027
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI5 = 000240	SUSPND= 000001
BKMOD = 000020	DT.ST0= 000010	KIPDR5= 172312	PRI6 = 000300	SVR0 = 000062
BKMODE= 040000	DT.ST1= 000012	KIPDR6= 172314	PRI7 = 000340	SVR1 = 000064
BKSLSH= 000134	DT.SWR= 000056	KIPDR7= 172316	PR0 = 000000	SVR2 = 000066
CAPRES= 000004	DT.SYP= 000072	KTERRO= 000040	PR4 = 000200	SVR3 = 000070
CASTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PR5 = 000240	SVR4 = 000072
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR6 = 000300	SVR5 = 000074
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR7 = 000340	SVR6 = 000076
CKTIM = 100000	DVID1 = 000014	LF = 000012	PS = 177776	SYSCNT= 000052
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PSW = 177776	YSERR= 000100
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	RANNUM= 000054	TMPID = 000002
CQOVF = 000001	ENBEO= 010000	MED = 076600	RBUFEA= 000130	TQOVF = 000002
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFFA= 000126	UIPAR0= 177640
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132	UIPAR1= 177642



GETPSW.MAC 28-JUL-78 09:15

SYMBOL TABLE

SEQ 0538

UIPAR2= 177644	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$\$FROM= 000000
UIPAR3= 177646	WBUFQ= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$LOC = 000000
UIPAR4= 177650	WBUFZ= 000142	\$F\$GDO= 000400	\$LSTIN= 000001	\$\$LGCN= 000000
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$\$REG = 177777
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$\$RETN= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$LDO= 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\$DR = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:56  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0539

3 COMMON EQUATE MODULE  
552 COMMON DEFINITIONS AND REFERENCES  
555 000000' .PRINT ;SPMAC: VERSION 1.1  
566 HRDADRCHK ROUTINE

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550

```
.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
;-----
```

```
552 .SBTTL COMMON DEFINITIONS AND REFERENCES
553
554 .MCALL STRUCT
555 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
556 000001 $LSTIN =1
557 000001 $LSTTAG =1
558
559 ;*****
560 ; REFERENCED BY OTHER MODULES
561 ;
562 .GLOBL HRDADR ;MODULE ENTRY POINT
563
564
```

```
566 .SBTTL HRDADRCHK ROUTINE
567
568
569 000000' ROUTINE HRDADRCHK <ADRCHK>
570 (2) 000000' HRDADRCHK:
571
572 ;+
573 ; SAVE REGISTERS AND RETRIEVE ADDRESS
574 ; FROM R5 STACK
575 ;-
576
577 000000' PUSH R0
578 (2) 000000' 010046 MOV R0,-(SP)
579 000002' LET R0 := ADRCHK(R5)
580 (4) 000002' 016500 000000 MOV ADRCHK(R5),R0
581 ;+
582 ; SAVE CURRENT CONTENTS OF LOC. 4 ON STACK
583 ;-
584 000006' PUSH @#4
585 (2) 000006' 013746 000004 MOV @#4,-(SP)
586 ;+
587 ; LOAD LOC. 4 WITH THE TEMPORARY NON-EXISTENT
588 ; MEMORY TRAP ROUTINE AT LABEL 1$
589 ;-
590
591 000012' LET @#4 := #1$
592 (4) 000012' 012737 000034' 000004 MOV #1$,@#4
593 ;+
594 ; CHECK TO SEE IF ADDRESS IS A VALID ADDRESS
595 ;-
596
597 000020' LET (R0) := (R0) + #0
598 (6) 000020' 062710 000000 ADD #0,(R0)
599 ;+
600 ; RESTORE LOC. 4 AND R0, AND RETURN NO ERROR INDICATOR
601 ;-
602
603 000024' POP @#4,R0
604 (2) 000024' 012637 000004 MOV (SP)+,@#4
605 (3) 000030' 012600 MOV (SP)+,R0
606
607 RETURN NO.ERROR
608 (4) 000032' 000410 BR 50000$
609
```

```

608 ;+
609 ; NON-EXISTENT MEMORY TRAP TOOK PLACE,
610 ; THEREFORE ADDRESS TESTED WAS ILLEGAL,
611 ; REPLACE RETURN ADDRESS ON STACK WITH 2$
612 ; AND DO A RTI WHICH WILL SEND YOU TO 2$.
613 ;-
614
615 000034'          INLINE <1$:>
(2) 000034'
616 000034'          LET (SP) := #2$
(4) 000034' 012716 000042'          MOV      #2$, (SP)
617 000040'          INLINE <RTI>
(2) 000040' 000002
618
619 ;+
620 ; RETURN HERE IF TRAP OCCURRED, CLEAN-UP STACK AND RETURN ERROR
621 ;-
622
623 000042'          INLINE <2$:>
(2) 000042'
624
625 000042'          POP      @#4,R0
(2) 000042' 012637 000004          MOV      (SP)+, @#4
(3) 000046' 012600          MOV      (SP)+, R0
626
627 000050'          RETURN ERROR
(2) 000050' 000261          SEC
(4) 000052' 000401          BR      50001$
628
629 ;+
630 ;RETURN TO CALLER
631 ;-
632
633 000054'          ENDRTN
(3) 000054'          50000$:
(2) 000054' 000241          CLC
(3) 000056'          50001$:
(2) 000056' 000207          RTS      PC
634
635          000001          .END
  
```

ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ACTBIT= 004000	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RDSERV= 000101
ADDR22= 001000	CTRLD = 000017	EVNTBE= 000200	MSGCKD= 000010	RDWHMI= 000022
ADR = 000006	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RELERR= 000020
ADRCHK= 000000	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RELMOD= 020000
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RELTIM= 010000
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RES1 = 000056
ASB = 000106	DROPMD= 100000	FATERR= 100000	MSGEOP= 000013	RES2 = 000060
ASSEMB= 000010	DSEVNT= 000014	HRDADR 000000ORG	MSGHDR= 000004	RICHAR= 031060
ASTAT = 000104	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RPTDAT= 002000
AUTO = 000010	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RSTRT = 000112
AUTOST= 020000	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RUBOUT= 000177
AWAS = 000110	DT.BLS= 000034	ICOUNT= 000040	MSGMUL= 177775	RUNMOD= 100000
BIT0 = 000001	DT.CF0= 000014	IDNUM = 000122	MSGPOP= 000002	R5VALU= 001740
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	SAM = 075464
BIT01 = 000002	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	SBADR = 000102
BIT02 = 000004	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SBKMOD= 000000
BIT03 = 000010	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	SBKSEL= 010000
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	SC.ADR= 000006
BIT05 = 000040	DT.FCH= 000037	INHRE= 000400	MSGSMH= 000014	SC.ALC= 000014
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SC.APC= 000016
BIT07 = 000200	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.CKL= 000002
BIT08 = 000400	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT09 = 001000	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT1 = 000002	DT.KBR= 000022	IOADDR= 112000	NBKMOD= 001000	SC.HLD= 000010
BIT10 = 002000	DT.KBU= 000030	IOMODX= 110000	NCPUOP= 000020	SC.SCA= 000012
BIT11 = 004000	DT.MLS= 000032	JACK = 035060	NOAPTY= 000002	SENDLS= 177777
BIT12 = 010000	DT.MTI= 000110	KIPAR0= 172340	NULL = 000000	SOFCNT= 000042
BIT13 = 020000	DT.OFF= 000070	KIPAR1= 172342	OWEN = 024020	SOFPAS= 000046
BIT14 = 040000	DT.PAS= 000074	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT15 = 100000	DT.PC = 000002	KIPAR3= 172346	PARPRE= 002000	SPOINT= 000032
BIT2 = 000004	DT.PFL= 000062	KIPAR4= 172350	PARSTA= 000100	SPVALU= 002200
BIT3 = 000010	DT.PSW= 000004	KIPAR5= 172352	PASCNT= 000034	SR0 = 177572
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PDPLSI= 020000	SR1 = 177574
BIT5 = 000040	DT.RCS= 000102	KIPAR7= 172356	PDP60 = 004000	SR2 = 177576
BIT6 = 000100	DT.REL= 000040	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
BIT7 = 000200	DT.SCT= 000066	KIPDR1= 172302	PRI0 = 000000	STAT = 000026
BIT8 = 000400	DT.SMX= 000106	KIPDR2= 172304	PRI1 = 000040	STATBI= 064757
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PRI4 = 000200	STAT1 = 000027
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI5 = 000240	SUSPND= 000001
BKMOD = 000020	DT.ST0= 000010	KIPDR5= 172312	PRI6 = 000300	SVR0 = 000062
BKMODE= 040000	DT.ST1= 000012	KIPDR6= 172314	PRI7 = 000340	SVR1 = 000064
BKSLSH= 000134	DT.SWR= 000056	KIPDR7= 172316	PRO = 000000	SVR2 = 000066
CAPRES= 000004	DT.SYP= 000072	KTERRO= 000040	PR4 = 000200	SVR3 = 000070
CASTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PR5 = 000240	SVR4 = 000072
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR6 = 000300	SVR5 = 000074
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR7 = 000340	SVR6 = 000076
CKTIM = 100000	DVID1 = 000014	LF = 000012	PS = 177776	SYS CNT= 000052
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PSW = 177776	SYSERR= 000100
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	RANNUM= 000054	TMPID = 000002
CQCVF = 000001	ENBEOP= 010000	MED = 076600	RSUFEA= 000130	TQOVF = 000002
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFPA= 000126	UIPAR0= 177640
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132	UIPAR1= 177642

UIPAR2= 177644	WBUFPA= 000134	\$FSDO = 000340	\$IFLEV= 177777	\$\$FROM= 000000
UIPAR3= 177646	WBUFQ= 000140	\$FSFAL= 000405	\$LOCTA= 177777	\$\$LOC = 000000
UIPAR4= 177650	WBUF SZ= 000142	\$FSGOO= 000400	\$LSTIN= 000001	\$\$LOCN= 000000
UIPAR5= 177652	WDFR = 000116	\$FSIF = 000110	\$LSTTA= 000001	\$\$REG = 177777
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$\$RETU= 000001
UIPAR7= 177656	WTINRE= 000352	\$F\$LDD= 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	XDN = 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	\$\$TD = 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	\$\$SERFL= 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 17:56  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0546

3 COMMON EQUATE MODULE  
552 COMMON DEFINITIONS AND REFERENCES  
555 000000' .PRINT ;SPMAC: VERSION 1.1  
578 HTRAP ROUTINE



```
552          .SBTTL COMMON DEFINITIONS AND REFERENCES
553          ;
554          .MCALL STRUCT
555          STRUCT
556          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
557          000001      $LSTIN = 1
558          000001      $LSTTAG = 1
559          ;
560          ;
561          ;*****
562          ;
563          ;REFERENCED BY OTHER MODULES
564          ;
565          .GLOBL HT.EXT      ;ADDRESS TO EXIT THIS MODULE THROUGH
566          .GLOBL HTBUSS     ;ENTRY POINT FOR BUSS ERROR TRAP
567          .GLOBL HTREIN     ;ENTRY POINT FOR RESERVED INSTRUCTION TRAP
568          .GLOBL HTPAER     ;ENTRY POINT FOR PARITY ERROR TRAP
569          .GLOBL HTKT       ;ENTRY POINT FOR KT TRAP
570          ;
571          ;*****
572          ;
573          ;LOCAL STORAGE:
574          ;
575          000000' 000000 HT.EXT: .WORD 0      ;ADDRESS TO EXIT THIS MODULE THRU
576          ;
```

```
578          .SBTTL HTRAP ROUTINE
579          ;
580
581          ;+
582          ;ENTER HERE ON BUSS ERROR TRAP
583          ;-
584          000002'          INLINE <HTBUSS:>
(2)          000002'          HTBUSS:
585          000002'          PUSH    #EVNTBE
(2)          000002' 012746 000200          MOV    #EVNTBE,-(SP)
586          000006' 000177 177766          JMP    @HT.EXT          ;EXIT
587
588          ;+
589          ;ENTER HERE ON RESERVED INSTR. TRAP
590          ;-
591          000012'          INLINE <HTREIN:>
(2)          000012'          HTREIN:
592          000012'          PUSH    #EVNTRE
(2)          000012' 012746 000201          MOV    #EVNTRE,-(SP)
593          000016' 000177 177756          JMP    @HT.EXT          ;EXIT
594
595          ;+
596          ;ENTER HERE ON PARITY ERROR TRAP
597          ;-
598          000022'          INLINE <HTPAER:>
(2)          000022'          HTPAER:
600          000022'          PUSH    #EVNTPE
(2)          000022' 012746 000202          MOV    #EVNTPE,-(SP)
601          000026' 000177 177746          JMP    @HT.EXT          ;EXIT
602
603          ;+
604          ;ENTER HERE ON MEMORY MANAGEMENT ERROR
605          ;
606
607          ;-
608          000032'          INLINE <HTTKT:>
(2)          000032'          HTTKT:
609          000032'          PUSH    #EVNTKT
(2)          000032' 012746 000203          MOV    #EVNTKT,-(SP)
610          000036' 000177 177736          JMP    @HT.EXT          ;EXIT
611
612          000001          .END
```

ACSR = 000102	CTRLC = 000003	ERRTYP= 000106	MED = 076600	RBUFEA= 000130
ACTBIT= 004000	CTRLD = 000017	EVNTBE= 000200	MEMPAS= 040000	RBUFPA= 000126
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MODEXH= 004000	RBUFSZ= 000132
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MODHOL= 002000	RBUFVA= 000124
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MODSEL= 001000	RDSERV= 000101
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGCKD= 000010	RDWHMI= 000022
ASS = 000106	DROPMO= 100000	FATERR= 100000	MSGCKS= 000011	RELERR= 000020
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGDER= 000005	RELMOD= 020000
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGDRP= 000017	RELTIM= 010000
AUTO = 000010	DT.AP = 000100	HTBUSS 000002RG	MSGGECH= 177777	RES1 = 000056
AUTDST= 020000	DT.APK= 000076	HKT 000032RG	MSGEOP= 000013	RES2 = 000060
AWAS = 000110	DT.BLS= 000034	HTPAER 000022RG	MSGHDR= 000004	RICHAR= 031060
BIT0 = 000001	DT.CFO= 000014	HTREIN 000012RG	MSGHNG= 000022	RPTDAT= 002000
BIT00 = 000001	DT.CF1= 000016	HT.EXT 000000RG	MSGHRD= 000007	RSTRT = 000112
BIT01 = 000002	DT.ERR= 000020	ICONT = 000036	MSGMAP= 000021	RJBOU= 000177
BIT02 = 000004	DT.ESI= 000044	ICOUNT= 000040	MSGNUL= 177775	RUNMOD= 100000
BIT03 = 000010	DT.EVN= 000000	IDNUM = 000122	MSGPOP= 000002	REVALJ= 001740
BIT04 = 000020	DT.EXS= 000060	IE = 000100	MSGPRM= 177776	SAM = 075464
BIT05 = 000040	DT.FCH= 000037	INDPAR= 000040	MSGRES= 000001	SBADR = 000102
BIT06 = 000100	DT.FCN= 000036	INHDRP= 040000	MSGSFT= 000006	SBKMOD= 000000
BIT07 = 000200	DT.HMX= 000104	INHEPR= 020000	MSGSKE= 000003	SBKSEL= 010000
BIT08 = 000400	DT.KBE= 000024	INHREL= 001000	MSGSMB= 000015	SC.ADR= 000006
BIT09 = 001000	DT.KBP= 000026	INHRR= 000400	MSGSMH= 000014	SC.ALC= 000014
BIT1 = 000002	DT.KBR= 000022	INIT = 000030	MSGSMS= 000016	SC.APC= 000016
BIT10 = 002000	DT.KBU= 000030	INTR = 000120	MSGSTD= 000000	SC.CKL= 000002
BIT11 = 004000	DT.MLS= 000032	IOMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT12 = 010000	DT.MTI= 000110	IOMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT13 = 020000	DT.OFF= 000070	IOMODR= 112000	NBKM0D= 001000	SC.HLD= 000010
BIT14 = 040000	DT.PAS= 000074	IOMODX= 110000	NCPUOP= 000020	SC.SCA= 000012
BIT15 = 100000	DT.PC = 000002	JACK = 035060	NOAPTY= 000002	SENDLS= 177777
BIT2 = 000004	DT.PFL= 000062	KIPAR0= 172340	NULL = 000000	SCFCNT= 000042
BIT3 = 000010	DT.PSW= 000004	KIPAR1= 172342	OWEN = 024020	SCFPAS= 000046
BIT4 = 000020	DT.PTA= 000064	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT5 = 000040	DT.RCS= 000102	KIPAR3= 172346	PARPRE= 002000	SPOINT= 000032
BIT6 = 000100	DT.REL= 000040	KIPAR4= 172350	PARSTA= 000100	SPVALU= 002200
BIT7 = 000200	DT.SCT= 000066	KIPAR5= 172352	PASCNT= 000034	SR0 = 177572
BIT8 = 000400	DT.SMX= 000106	KIPAR6= 172354	PDPLSI= 020000	SR1 = 177574
BIT9 = 001000	DT.SP = 000006	KIPAR7= 172356	PDP60 = 004000	SR2 = 177576
BKDEF = 000002	DT.SSI= 000046	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
BKMOD = 000020	DT.STO= 000010	KIPDR1= 172302	PR10 = 000000	STAT = 000026
BKMODE= 040000	DT.ST1= 000012	KIPDR2= 172304	PR11 = 000040	STATBI= 064757
BKSLSH= 000134	DT.SWR= 000056	KIPDR3= 172306	PR14 = 000200	STAT1 = 000027
CAPRES= 000004	DT.SYP= 000072	KIPDR4= 172310	PR15 = 000240	SUSPND= 000001
CASAT= 000004	DT.WBU= 000050	KIPDR5= 172312	PR16 = 000300	SVR0 = 000062
CDERCT= 000146	DT.WHL= 000054	KIPDR6= 172314	PR17 = 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	ECCMEM= 000100	KTPRES= 000400	PR5 = 000240	SVR4 = 000072
CONFIG= 000056	ECCSTA= 000010	KTSTAT= 000020	PR6 = 000300	SVR5 = 000074
COQVF = 000001	ENBEOP= 010000	KTXTND= 040000	PR7 = 000340	SVR6 = 000076
CR = 000015	ENBNUL= 000001	LF = 000012	PS = 177776	SYSNT= 000052
CSRA = 000100	ENDLST= 000000	LPSTAT= 000001	PSW = 177776	SYSERR= 000100
CSRC = 000102	EOPBIT= 000001	MAPSTA= 000200	RANNUM= 000054	TMPID = 000002

TQDVF = 000002	WASADR= 000104	\$FSBLA= 000170	\$FSUNT= 000130	\$\$FLAG= 000000
UIPAR0= 177640	WBSTAT= 000040	\$FSCAS= 000150	\$FSWHI= 000120	\$\$FROM= 000000
UIPAR1= 177642	WBUFEA= 000136	\$F\$DEC= 000220	\$FSYES= 000402	\$\$LOC = 000000
UIPAR2= 177644	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$\$LOCN= 000000
UIPAR3= 177646	WBUFRQ= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$REG = 177777
UIPAR4= 177650	WBUFSZ= 000142	\$F\$G00= 000400	\$LSTIN= 000001	\$\$RETU= 000000
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$\$RTN1= 000000
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$\$RTN2= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$L00= 000200	\$\$SAVLE= 177777	\$\$SRC = 000000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$TAGLE= 177777	\$\$TGSV= 000000
UIPDR1= 177602	XFLAG = 000005	\$F\$ND = 000403	\$TAGNU= 050000	\$\$TGS1= 000000
UIPDR2= 177604	XOFF = 000023	\$F\$OR = 000320	\$\$ARGC= 000000	\$\$TGS2= 000000
UIPDR3= 177606	XDN = 000021	\$F\$RTI= 000350	\$\$BYTE= 000000	\$\$TD = 000000
UIPDR4= 177610	\$BGNLE= 177777	\$F\$RTN= 000300	\$\$CASE= 000000	\$\$TAG= 050000
UIPDR5= 177612	\$ERFLG= 000000	\$F\$SEL= 000140	\$\$DST = 000000	= 000042R
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$\$ELOC= 000000	
UIPDR7= 177616	\$F\$BAD= 000401	\$F\$TRU= 000404	\$\$ERFL= 000000	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:57  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0552

3 COMMON EQUATE MODULE  
562 COMMON DEFINITIONS AND REFERENCES  
565 000000' .PRINT ;SPMAC: VERSION 1.1  
589 INDIRECT ECC CSR ON/OFF ROUTINE

COMMON EQUATE MODULE

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559

```
.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
;--
```



COMMON EQUATE MODULE

561  
562  
563  
564  
565 000000'  
(1) 000000'  
566 000001  
567 000001  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584 152100  
585 152136  
586 170370  
587 170372

.SBTTL COMMON DEFINITIONS AND REFERENCES

.MCALL STRUCT  
STRUCT  
.PRINT ;SPMAC: VERSION 1.1  
\$LSTIN = 1  
\$LSTTAG = 1

\*\*\*\*\*  
; REFERENCED BY OTHER MODULES

.GLOBL ICSR00 ;MODULE ENTRY POINT

\*\*\*\*\*  
; GLOBAL REFERENCES

.GLOBL CCNTRL ;CACHE CONTROL REGISTER

\*\*\*\*\*  
; LOCAL EQUATES

IN.PCSR = 152100 ;START OF PARITY CSRS BY INDIRECT MODE  
IN.PCLS = 152136 ;LAST PARITY CSR BY INDIRECT MODE  
IN.M36 = 170370 ;MAP REG 36 LOWER 16 BITS  
IN.M37 = 170372 ;MAP REG 36 UPPER 6 BITS

```

589
590
591 000000'
592 000000'
593
594
595
596
597
598 000000'
(2) 000000' 010046
(3) 000002' 010146
599 000004'
(4) 000004' 016500 000000
600 000010'
(4) 000010' 016001 000064
601
602
603
604
605
606 000014'
(2) 000014' 017746 000000G
607 000020'
(2) 000020' 013746 177572
608 000024'
(2) 000024' 013746 172516
609 000030'
(2) 000030' 013746 170370
610 000034'
(2) 000034' 013746 170372
611 000040'
(2) 000040' 013746 172354
612
613
614
615
616
617
618 000044'
(6) 000044' 052777 000014 000000G
619 000052'
(4) 000052' 012737 177400 172354
620 000060'
(6) 000060' 052737 000001 177572
621 000066'
(6) 000066' 052737 000060 172516
622 000074'
(4) 000074' 012737 160000 170370
623 000102'
(4) 000102' 012737 000077 170372
624
625
626
627

```

```

INDIRECT ECC CSR ON/OFF ROUTINE
.SBTTL INDIRECT ECC CSR ON/OFF ROUTINE
ROUTINE ICSR00 <DTABLE,FLAG>
ICSROO:
;+
; SAVE REGISTERS AND GET ADDRESS OF PARITY TABLE
;-
PUSH R0,R1
MOV R0,-(SP)
MOV R1,-(SP)
LET R0 := DTABLE(R5)
MOV DTABLE(R5),R0
LET R1 := DT.PTA(R0)
MOV DT.PTA(R0),R1
;+
; SAVE STATUS OF CACHE, KT AND UNIBUS MAP REGISTERS
;-
PUSH @CCNTRL
MOV @CCNTRL,-(SP)
PUSH @#SR0
MOV @#SR0,-(SP)
PUSH @#SR3
MOV @#SR3,-(SP)
PUSH @#IN.M36
MOV @#IN.M36,-(SP)
PUSH @#IN.M37
MOV @#IN.M37,-(SP)
PUSH @#KIPAR6
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
;-
LET @CCNTRL := @CCNTRL SET.BY #14
BIS #14,@CCNTRL
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 @#IN.M36 := #160000
MOV #160000,@#IN.M36
LET @#IN.M37 := #77
MOV #77,@#IN.M37
;+
; WHILE ENTRY IN PARITY TABLE TAKE THE APPROPRIATE ACTION OF TURNING ON/OFF ECC MEMORY.
;-

```

628			WHILE (R1) NE #ENDLST DO		
629	000110'			50002\$:	
(4)	000110'				CMP (R1),#ENDLST
(6)	000110'	021127	000000		BEQ 50003\$
(9)	000114'	001411			
630	000116'		IF FLAG(R5) EQ #0 THEN		TST FLAG(R5)
(6)	000116'	005765	000002		BNE 50004\$
(9)	000122'	001003			
631	000124'		LET @(R1)+ := @(R1)+ SET.BY #3		BIS #3,@(R1)+
(6)	000124'	052731	000003		
632	000130'		ELSE	50004\$:	BR 50005\$
(4)	000130'	000402			
(3)	000132'				
633	000132'		LET @(R1)+ := @(R1)+ CLR.BY #3		BIC #3,@(R1)+
(6)	000132'	042731	000003		
634	000136'		ENDIF	50005\$:	
(4)	000136'				
635					
636	000136'		ENDDO		BR 50002\$
(4)	000136'	000764		50003\$:	
(3)	000140'				
637					
638			;		
639			; NOW RESTORE STATUS TO CACHE, KT AND MAP REGISTERS		
640			;-		
641					
642	000140'		POP @#KIPAR6		MOV (SP)+,@#KIPAR6
(2)	000140'	012637	172354		
643	000144'		POP @#IN.M37		MOV (SP)+,@#IN.M37
(2)	000144'	012637	170372		
644	000150'		POP @#IN.M36		MOV (SP)+,@#IN.M36
(2)	000150'	012637	170370		
645	000154'		POP @#SR3		MOV (SP)+,@#SR3
(2)	000154'	012637	172516		
646	000160'		POP @#SR0		MOV (SP)+,@#SR0
(2)	000160'	012637	177572		
647	000164'		POP @CCNTRL		MOV (SP)+,@CCNTRL
(2)	000164'	012677	000000G		
648					
649			;		
650			; NOW RESTORE REGISTERS AND RETURN		
651			;-		
652					
653	000170'		POP R1,R0		MOV (SP)+,R1
(2)	000170'	012601			MOV (SP)+,R0
(3)	000172'	012600			
654					
655	000174'		ENDRTN	50000\$:	
(3)	000174'			50001\$:	
(3)	000174'				
(2)	000174'	000207			RTS PC
656	000001		.END		

ACSR = 000102	CSRC = 000102	ENDLST= 000000	LF = 000012	PS = 177776
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	LPSTAT= 000001	PSW = 177776
ADDR22= 001000	CTRL0 = 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	MSGGECH= 177777	RES1 = 000056
BIT00 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGGEP= 000013	RES2 = 000060
BIT01 = 000002	DT.CFO= 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	INHRR= 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.CLO= 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	NDAPTY= 000002	SENDLS= 177777
BIT5 = 000040	DT.PSW= 000004	KIPAR0= 172340	NULL = 000000	SCFCNT= 000042
BIT6 = 000100	DT.PTA= 000064	KIPAR1= 172342	OWEN = 024020	SOPPAS= 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.STO= 000010	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
CAPRES= 000004	DT.ST1= 000012	KIPDR1= 172302	PRI0 = 000000	STAT = 000026
CASAT= 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
CQCVF = 000001	ECCSTA= 000010	KTPRES= 000400	PR5 = 000240	SVR4 = 000072
CR = 000015	ENBEOB= 010000	KTSTAT= 000020	PR6 = 000300	SVR5 = 000074
CSRA = 000100	ENBNUL= 000001	KTXTND= 040000	PR7 = 000340	SVR6 = 000076

SYSCNT= 000052	UIPDR7= 177616	\$FSCAS= 000150	\$IFLEV= 177777	\$\$CASE= 000000
SYSERR= 000100	WASADR= 000104	\$F\$DEC= 000220	\$ISKO = 000001	\$\$DST = 000000
TMPID = 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\$GDO= 000400	\$LSTTA= 000001	\$\$FLAG= 000001
UIPAR1= 177642	WBUFRQ= 000140	\$F\$IF = 000110	\$NESTL= 177777	\$\$FROM= 000000
UIPAR2= 177644	WBUFSSZ= 000142	\$F\$INC= 000210	\$NSKO = 000300	\$\$LDC = 000122R
UIPAR3= 177646	WDFR = 000116	\$F\$LOO= 000200	\$NSK1 = 000120	\$\$LOCN= 000000
UIPAR4= 177650	WDT0 = 000114	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$REG = 177777
UIPAR5= 177652	WTINRE= 000352	\$F\$NO = 000403	\$\$SAVLE= 177777	\$\$RETN= 000000
UIPAR6= 177654	WTWHMI= 000222	\$F\$OR = 000320	\$\$SSKO = 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	\$TSKO = 050002	\$\$TGS1= 000000
UIPDR3= 177606	\$ERFLG= 000400	\$F\$TRU= 000404	\$TSK1 = 050003	\$\$TGS2= 000000
UIPDR4= 177610	\$F\$AND= 000310	\$F\$UNT= 000130	\$TSK2 = 050005	\$\$TD = 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: ICSR00, DSKZ: ICSR00=SPMAC/ML, EQUATE, ICSR00  
RUN-TIME: 12 2 .3 SECONDS  
RUN-TIME RATIO: 31/16=1.9  
CORE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:57  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0559

3 COMMON EQUATE MODULE  
566 COMMON DEFINITIONS AND REFERENCES  
569 000000' .PRINT ;SPMAC: VERSION 1.1  
595 INDIRECT ECC MEMORY CSR SIZING ROUTINE

```
508 .TITLE ICRSZ INDIRECT ECC MEMORY CSR SIZING ROUTINE
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; ICRSZ
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE WILL DETERMINE IF ECC MEMORY CSRS ARE HIDDEN ON
517 ; THE MAIN MEMORY BUS OF THE 11/70 SYSTEM.
518 ; THE MANNER IN WHICH THIS IS ACCOMPLISHED IS BY
519 ; THE FOLLOWING:
520 ; - NO NPR ACTIVITY
521 ; - TURN CACHE OFF
522 ; - MAP KIPAR6 TO 177400
523 ; - ENABLE MEMORY MANAGEMENT IN 22 BIT MODE
524 ; - MAP UNIBUS MAP REGISTER 36 TO 17760000
525 ; - TURN ON UNIBUS MAP BOX
526 ; - ACCESS VIRTUAL ADDRESSES 152100 THROUGH 152136
527 ; AS ECC MEMORY CSRS
528 ; IF ECC FOUND THEN THE CSR ADDRESS WILL BE ENTERED INTO THE PARITY TABLE.
529 ;
530 ; INPUTS:
531 ; DTABLE ADDRESS
532 ;
533 ; IMPLICIT INPUTS:
534 ; DT.PTA
535 ; DT.CF0
536 ; DT.ST0
537 ;
538 ; OUTPUTS:
539 ; NONE
540 ;
541 ; IMPLICIT OUTPUTS:
542 ; NONE
543 ;
544 ; PATHOLOGICAL CONNECTIONS:
545 ; NONE
546 ;
547 ; SUBORDINATE ROUTINES CALLED:
548 ; SAVREG - SAVE REGISTERS
549 ; RESREG - RESTORE REGISTERS
550 ; KTSET - SET UP KT REGISTERS
551 ; HRDADRCHK - ILLEGAL ADDRESS CHECK
552 ;
553 ; FUNCTIONAL SIDE EFFECTS:
554 ; NONE
555 ;
556 ; CALLING SEQUENCE:
557 ; CALL ICRSZ IN <DTABLE>
558 ; DTABLE - ADDRESS OF DTABLE
559 ;
560 ; VERSION:
561 ; 0.0
562 ;
563 ; EDIT DATE BY REASON
```

ICRSZ INDIRECT ECC MEMORY CSR SIZING ROUTINE MACY11 30A(1052) 20-SEP-78 17:57 PAGE 19-1  
ICRSZ.MAC 06-SEP-78 15:50 COMMON EQUATE MODULE

SEQ 0561

564

;-



```
566          .SBTTL COMMON DEFINITIONS AND REFERENCES
567
568          .MCALL STRUCT
569 000000'    STRUCT
(1) 000000'    .PRINT ;SPMAC: VERSION 1.1
570          000001    $LSTIN = 1
571          000001    $LSTTAG = 1
572
573
574          ;*****
575          ; REFERENCED BY OTHER MODULES
576          ;
577          .GLOBL ICSRSZ          ;MODULE ENTRY POINT
578
579          ;*****
580          ; GLOBAL REFERENCES
581          ;
582          .GLOBL SAVREG          ;SAVE REGISTERS
583          .GLOBL RESREG          ;RESTORE REGISTERS
584          .GLOBL KTSET          ;SET UP KT
585          .GLOBL HRDADRCHK       ;ILLEGAL ADDRESS CHECK
586          .GLOBL CCNTRL         ;CACHE CONTROL REGISTER
587          ;*****
588          ; LOCAL EQUATES
589          ;
590          152100    IC.PCSR = 152100          ;START OF PARITY CSRS BY INDIRECT MODE
591          152136    IC.PCLS = 152136         ;LAST PARITY CSR BY INDIRECT MODE
592          170370    IC.M36 = 170370         ;MAP REG 36 LOWER 16 BITS
593          170372    IC.M37 = 170372         ;MAP REG 36 UPPER 6 BITS
```

```

595 .SBTTL INDIRECT ECC MEMORY CSR SIZING ROUTINE
596
597 ROUTINE ICSRSZ <DTABLE>
598 000000'
(2) 000000' ICSRSZ:
599
600 ;+
601 ; SAVE REGISTERS AND GET ADDRESS OF DTABLE
602 ; -
603
604 000000' CALL SAVREG
(3) 000000' 004767 000000G JSR PC, SAVREG
605 000004' LET R0 := DTABLE(R5)
(4) 000004' 016500 000000 MOV DTABLE(R5), R0
606
607 ;+
608 ; SAVE CONTENTS OF CACHE CSR, KT STATUS, KIPAR6, AND MAP REGISTER
609 ; -
610
611 000010' PUSH @CCNTRL
(2) 000010' 017746 000000G MOV @CCNTRL, -(SP)
612 000014' PUSH @#SR0
(2) 000014' 013746 177572 MOV @#SR0, -(SP)
613 000020' PUSH @#SR3
(2) 000020' 013746 172516 MOV @#SR3, -(SP)
614 000024' PUSH @#IC.M36
(2) 000024' 013746 170370 MOV @#IC.M36, -(SP)
615 000030' PUSH @#IC.M37
(2) 000030' 013746 170372 MOV @#IC.M37, -(SP)
616 000034' PUSH @#KIPAR6
(2) 000034' 013746 172354 MOV @#KIPAR6, -(SP)
617
618 ;+
619 ; TURN CACHE OFF, SETUP KT, SETUP KIPAR6 TO POINT TO MAP REG 36,
620 ; AND MAP REGISTER 36 TO LOOK AT 128K I/O SHADOW AREA.
621 ; -
622
623 000040' LET @CCNTRL := @CCNTRL SET.BY #14
(6) 000040' 052777 000014 000000G BIS #14, @CCNTRL
624 000046' CALL KTSET IN <R0>
(3) 000046' 010546 MOV R5, -(SP)
(4) 000050' 010045 MOV R0, -(R5)
(3) 000052' 004767 000000G JSR PC, KTSET
(3) 000056' 012605 MOV (SP)+, R5
625 000060' LET @#KIPAR6 := #177400
(4) 000060' 012737 177400 172354 MOV #177400, @#KIPAR6
626 000066' LET @#SR0 := @#SR0 SET.BY #BIT00
(6) 000066' 052737 000001 177572 BIS #BIT00, @#SR0
627 000074' LET @#SR3 := @#SR3 SET.BY #60
(6) 000074' 052737 000060 172516 BIS #60, @#SR3
628 000102' LET R3 := #IC.PCSR
(4) 000102' 012703 152100 MOV #IC.PCSR, R3
629 000106' LET @#IC.M36 := #160000
(4) 000106' 012737 160000 170370 MOV #160000, @#IC.M36
630 000114' LET @#IC.M37 := #77
(4) 000114' 012737 000077 170372 MOV #77, @#IC.M37

```

```

631
632          ;+
633          ; GET ADDRESS OF PARITY TABLE
634          ; -
635
636 000122'          LET R1 := DT.PTA(R0)
(4) 000122' 016001 000064          MOV          DT.PTA(R0),R1
637
638          ;+
639          ; NOW DETERMINE IF ANY PARITY CSRS EXIST INDIRECTLY, IF SO LOAD ADDRESS INTO
640          ; PARITY TABLE AND SET ECC MEMORY PRESENT AND ALSO INDIRECT PARITY REG BIT.
641          ; -
642
643 000126'          WHILE R3 LOS #IC.PCLS DO
(4) 000126'          50002$:
(6) 000126' 020327 152136          CMP          R3,#IC.PCLS
(9) 000132' 101021          BHI          50003$
644          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
645          IF.NO.ERROR THEN
(6) 000146' 103410          BCS          50004$
646          LET (R1)+ := R3
(4) 000150' 010321          MOV          R3,(R1)+
647          LET DT.CF0(R0) := DT.CF0(R0) SET.BY #ECCMEM
(6) 000152' 052760 000100 000014          BIS          #ECCMEM,DT.CF0(R
648          LET DT.CF0(R0) := DT.CF0(R0) SET.BY #INDPAR
(6) 000160' 052760 000040 000014          BIS          #INDPAR,DT.CF0(R
649
650          ;+
651          ; CLEAR OUT CSR, THUS TURNING OFF ECC
652          ; -
653
654          LET (R3) := #0
(4) 000166' 005013          CLR          (R3)
655          ENDIF
(4) 000170'          50004$:
656
657          ;+
658          ; UPDATE CSR ADDRESS AND GO LOOK AGAIN
659          ; -
660
661          LET R3 := R3 + #4
(6) 000170' 062703 000004          ADD          #4,R3
662          ENDDD
(4) 000174' 000754          BR          50002$
(3) 000176'          50003$:
663
664          ;+
665          ; NOW RESTORE CACHE, KT AND MAP REGISTERS
666          ; -
667
668          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 MOV (SP)+,@#IC.M36
(2) 000206' 012637 170370
671 000212' POP @#SR3 MOV (SP)+,@#SR3
(2) 000212' 012637 172516
672 000216' POP @#SR0 MOV (SP)+,@#SR0
(2) 000216' 012637 177572
673 000222' POP @CCNTRL MOV (SP)+,@CCNTRL
(2) 000222' 012677 000000G
674
675 ;+
676 ; RESTORE REGISTERS AND RETURN TO CALLER
677 ; -
678
679 000226' CALL RESREG JSR PC,RESREG
(3) 000226' 004767 000000G
680
681 000232' ENDRTN
(3) 000232'
(3) 000232' 50000S:
(2) 000232' 000207 50001S:
682 000001 .END RTS PC
```

ACSR = 000102	CSRC = 000102	ENDLST= 000000	KTXTND= 040000	PR7 = 000340
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	LF = 000012	PS = 177776
ADDR22= 001000	CTRLD = 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
AUTD = 000010	DTABLE= 000000	HRDADR= ***** G	MSGCKD= 000010	RDWHMI= 000022
AUTDST= 020000	DT.ADD= 000042	HRDCNT= 000044	MSGCKS= 000011	RELERR= 000020
AWAS = 000110	DT.AP = 000100	HRDPAS= 000050	MSGDLS= 000005	RELMOD= 020000
BIT0 = 000001	DT.APK= 000076	ICDNT = 000036	MSGDRP= 000017	RELTIM= 010000
BIT00 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGECH= 177777	RESREG= ***** G
BIT01 = 000002	DT.CFO= 000014	ICRSZ= 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	RVALU= 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	INHRE= 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.STO= 000010	KIPDR0= 172300	PDP60 = 004000	SR0 = 177572
CAPRES= 000004	DT.ST1= 000012	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
CASAT= 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	PRO = 000000	SVR0 = 000062
CQDVF = 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	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= 000001
SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LSTTA= 000001	\$\$FROM= 000000
TMPID = 000002	WBUFEA= 000136	\$F\$GDO= 000400	\$NESTL= 177777	\$\$LOC = 000146R
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NSK0 = 000300	\$\$LOCN= 000000
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	\$NSK1 = 000120	\$\$REG = 177777
UIPAR1= 177642	WBUFSZ= 000142	\$F\$L00= 000200	\$NSK2 = 000110	\$\$RETU= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$\$SAVLE= 177777	\$\$RTN1= 050000
UIPAR3= 177646	WDTO = 000114	\$F\$NO = 000403	\$\$SSK0 = 050003	\$\$RTN2= 050001
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	\$\$TAGLE= 177777	\$\$SRC = 000000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	\$\$TAGNU= 050005	\$\$TGSV= 000000
UIPAR6= 177654	XFLAG = 000005	\$F\$RTN= 000300	\$\$TEMP = 000300	\$\$TGS1= 000000
UIPAR7= 177656	XOFF = 000023	\$F\$SEL= 000140	\$\$TSK0 = 050002	\$\$TGS2= 000000
UIPDR0= 177600	XON = 000021	\$F\$THE= 000330	\$\$TSK1 = 050003	\$\$TD = 000000
UIPDR1= 177602	\$BGNLE= 177777	\$F\$TRU= 000404	\$\$TSK2 = 050004	\$\$\$TAG= 050000
UIPDR2= 177604	\$ERFLG= 000400	\$F\$UNT= 000130	\$\$ARGC= 000002	. = 000234R
UIPDR3= 177606	\$F\$AND= 000310	\$F\$WHI= 000120	\$\$BYTE= 000403	
UIPDR4= 177610	\$F\$BAD= 000401	\$F\$YES= 000402	\$\$CASE= 000000	

. ABS. 000000 000  
000234 001

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

DSKZ:ICRSRZ,DSKZ:ICRSRZ=SPMAC/ML,EQUATE,ICRSRZ  
RUN-TIME: 13 3 .3 SECONDS  
RUN-TIME RATIO: 32/17=1.9  
CORE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:58  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0568

3 COMMON EQUATE MODULE  
564 COMMON DEFINITIONS AND REFERENCES FOR KEXAM  
567 000000' .PRINT ;SPMAC: VERSION 1.1  
611 KEXAM ROUTINE

```
508 .TITLE KEXAM PROCESS THE 'EXAM' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KEXAM
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'EXAM' KEYBOARD COMMAND. IT WILL
517 ; DETERMINE IF AN ARGUMENT IS INCLUDED IN THE COMMAND STRING.
518 ; IF NO ARGUMENTS ARE INCLUDED, THE LAST LOCATION EXAMINED
519 ; WILL BE OUTPUTTED. IF ARGUMENTS ARE INCLUDED, THE MODULE
520 ; NAME AND/OR THE OCTAL ADDRESS WILL BE VERIFIED AND IF
521 ; NO ERRORS, THE APPROPRIATE LOCATION AND
522 ; ITS CONTENTS WILL BE OUTPUTTED.
523 ;
524 ; INPUTS:
525 ; 1. ADDRESS OF DATA TABLE
526 ; 2. DECODE BUFFER POINTER
527 ;
528 ; IMPLICIT INPUTS:
529 ; 1. DT.STO
530 ;
531 ; OUTPUTS:
532 ; NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ; DT.KBRSP
536 ;
537 ; PATHOLOGICAL CONNECTIONS:
538 ; CM.BADNAME,CM.ADR,CM.ODD,CM.ARG
539 ;
540 ; SUBORDINATE ROUTINES CALLED:
541 ; ARGCHK - CHECK FOR ARGUMENTS
542 ; SAVREG
543 ; RESREG
544 ; BOA16
545 ; NUMCHK
546 ; NAMCHK
547 ;
548 ; FUNCTIONAL SIDE EFFECTS:
549 ; NONE
550 ;
551 ; CALLING SEQUENCE:
552 ; CALL KEXAM IN <DTADR,BUFPTR>
553 ; WHERE DTADR = ADDRESS OF DATA TABLE
554 ; BUFPTR = COMMAND DECODE BUFFER POINTER
555 ;
556 ; VERSION:
557 ; 0.0
558 ;
559 ; EDIT DATE BY REASON
560 ;---
561
562
```



```

564          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KEXAM
565
566          .MCALL STRUCT
567 000000'   STRUCT
(1) 000000'   .PRINT ;SPMAC: VERSION 1.1
568
569          000001 $LSTIN=1
570          000001 $LSTTAG=1
571
572          ;
573          ; REFERENCED BY OTHER MODULES:
574          ;
575          .GLOBL KE.LOC          ;INITIALIZED TO 0 IF RUN CMD IS GIVEN
576          .GLOBL KEXAM          ;MODULE ENTRY POINT
577          ;
578          ;*****
579          ;
580          ; GLOBAL REFERENCES:
581          ;
582          .GLOBL SAVREG          ;SAVE REGISTERS
583          .GLOBL RESREG          ;RESTORE REGISTERS
584          .GLOBL BOA16          ;BINARY TO OCTAL ASCII CONV ROUTINE
585          .GLOBL ARGCHK          ;ROUTINE TO CHECK AN ARGUMENT
586          .GLOBL CM.BADNAME      ;"INVALID MODULE NAME" MESSAGE
587          .GLOBL CM.ADR          ;"BAD ADDRESS" MESSAGE
588          .GLOBL NUMCHK          ;VERIFY A NUMBER IS OCTAL, &
589          ;                          ;16 BITS OR LESS IN LENGTH
590          .GLOBL CM.ODD          ;'NOT AN EVEN NUMBER' MSG ADDRESS
591          .GLOBL NAMCHK          ;VERIFY A MODULE NAME
592          .GLOBL CM.ARG          ;"INVALID ARGUMENT" MESSAGE
593          ;
594          ;*****
595          ;
596          ; LOCAL STORAGE:
597          ;
598 000000' 000000 KE.LOC: .WORD 0          ;HOLD LOCATION TO BE EXAMINED
599 000002'   KE.MG1:
600 000002' 000006 KE.MG2: .BLKB 6          ;HOLD ASCII OF ADDRESS TO BE EXAMINED
601 000010' 020057   .ASCII '/'         ;SEPARATOR
602 000012' 000006 KE.MG3: .BLKB 6          ;HOLD ASCII CONTENTS OF ADDRESS
603 000020' 000045   .ASCIZ '/%'
604          .EVEN
605          ;
606          ;*****
607
608
609

```

```

611          .SBTTL  KEXAM ROUTINE
612
613 000022'  ROUTINE KEXAM <DTADR,CMDBUF>
(2) 000022'
614
615          ;+
616          ; SAVE REGISTERS AND SAVE ADDRESS OF DTABLE & ADDRESS OF DECODE BUFFER
617          ; -
618
619 000022'  CALL SAVREG
(3) 000022' 004767 000000G          JSR      PC,SAVREG
620
621 000026'  LET R0 := DTADR(R5)
(4) 000026' 016500 000000          MOV      DTADR(R5),R0
622 000032'  LET R1 := CMDBUF(R5)
(4) 000032' 016501 000002          MOV      CMDBUF(R5),R1
623
624          ;+
625          ; INIT R2 FOR LATER USE
626          ; -
627
628 000036'  LET R2 := #0
(4) 000036' 005002          CLR      R2
629
630          ;+
631          ; GET TO FIRST ARGUMENT
632          ; -
633
634 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
635
636          ;+
637          ; IF THERE IS A CR AS FIRST ARGUMENT - CONVERT LAST EXAMINED ADDRESS
638          ; TO ASCII, OUTPUT IT, AND RETURN
639          ; -
640 000060'  IF.ERROR THEN
(6) 000060' 103003          BCC      50002$
641
642          CALL CONV
643 000062'
(3) 000062' 004767 000264          JSR      PC,CONV
644
645
646
647
648 000066'  ELSE
(4) 000066' 000526          BR       50003$
(3) 000070'
649
650          ;+
651          ; THERE IS AN ARGUMENT - FIND OUT WHETHER ALPHA OR NUMERIC

```



```

684 ;-
685 000166' IF.ERROR THEN
(6) 000166' 103006 BCC 50010$
686
687
688 000170' LET DT.KBRSP(R0) := #CM.ARG
(4) 000170' 012760 000000G 000022 MOV #CM.ARG,DT.KBRSP
689 000176' CALL RESREG
(3) 000176' 004767 000000G JSR PC,RESREG
690 000202' RETURN
(4) 000202' 000462 BR 50000$
691 000204' ENDIF
(4) 000204' ENDIF 50010$:
692 000204' ENDIF 50007$:
(4) 000204' ENDIF 50005$:
693
694 000204' ENDIF
(4) 000204'
695
696
697 ;+
698 ; CHECK OUT THE NUMBER
699 ;-
700
701
702 000204' CALL NUMCHK IN <R0,R1> OUT <R1,R3>
(4) 000204' 162705 000004 SUB #2*2,R5
(3) 000210' 010546 MOV R5,-(SP)
(5) 000212' 010145 MOV R1,-(R5)
(4) 000214' 010045 MOV R0,-(R5)
(3) 000216' 004767 000000G JSR PC,NUMCHK
(3) 000222' 012605 MOV (SP)+,R5
(4) 000224' 012501 MOV (R5)+,R1
(4) 000226' 012503 MOV (R5)+,R3
703
704
705 000230' IF.NO.ERROR THEN
(6) 000230' 103445 BCS 50011$
706
707
708 ;+
709 ; IF VALID VIRTUAL ADDRESS THEN
710 ; SHIFT NUMBER ONE PLACE TO RIGHT. IF THE C BIT SETS IT IS AN ODD
711 ; NUMBER... NOT ALLOWED !!! STUFF ERROR MSG AND RETURN....
712 ; IF THE NUMBER IS OK,RESTORE IT AND CONTINUE.
713 ;-
714
715 000232' LET R3 := R3 ROTATE -1
(7) 000232' 006003 ROR R3
716 000234' IFCOND CS THEN
(6) 000234' 103004 BCC 50012$
717 000236' LET DT.KBRSP(R0) := #CM.ODD
(4) 000236' 012760 000000G 000022 MOV #CM.ODD,DT.KBRSP
718 000244' ELSE
(4) 000244' 000437 BR 50013$
(3) 000246' 50012$:

```

```

719 000246'          LET R3 := R3 ROTATE 1
(7) 000246' 006103          ROL      R3
720
721
722
723
724
725          ;+
726          ; THE NUMBER IS OK...CHECK FOR ANY JUNK ARGUMENTS..
727          ; -
728
729 000250'          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
730
731          ;+
732          ; IF IT'S NOT A CR... STUFF ERROR MSG AND LEAVE
733          ; -
734 000270'          IF.NO.ERROR THEN
(6) 000270' 103404          BCS      50014$
735
736          ; -
737
738 000272'          LET DT.KBRSP(R0) := #CM.ARG
(4) 000272' 012760 000000G 000022          MOV      #CM.ARG,DT.KBRSP
739
740          ;+
741          ; ELSE THERE ARE NO JUNK ARGUMENTS... NOW SEE IF THE NUMBER IS IN
742          ; THE EXERCISER RANGE... AND NOT > THAN 32 K
743          ; -
744
745          ELSE
(4) 000300' 000421          BR      50015$
(3) 000302'          50014$:
746
747
748
749
750          ;+
751          ; FORM THE SUM (MODULE HEADER ADDRESS + NUMBER
752          ; -
753
754 000302'          LET R3 := R3 + R2
(6) 000302' 060203          ADD      R2,R3
755 000304'          IFCOND CS THEN
(6) 000304' 103004          BCC      50016$
756 000306'          LET DT.KBRSP(R0) := #CM.ADR
(4) 000306' 012760 000000G 000022          MOV      #CM.ADR,DT.KBRSP
757 000314'          ELSE
(4) 000314' 000413          BR      50017$
(3) 000316'          50016$:
758          ;+

```

```

759 ; IF THE NUMBER IS TOO BIG... TELL OPERATOR AND LEAVE...
760 ; -
761 000316' IF R3 HI DT.ESIZ(R0) THEN
(6) 000316' 020360 000044 CMP R3,DT.ESIZ(R0)
(9) 000322' 101404 BLOS 50020$
762
763
764 000324' LET DT.KBRSP(R0) := #CM.ADR
(4) 000324' 012760 000000G 000022 MOV #CM.ADR,DT.KBRSP
765
766 ;+
767 ; ELSE THE NUMBER IS OK.... GO OUTPUT IT...
768 ; -
769 000332' ELSE
(4) 000332' 000404 BR 50021$
(3) 000334' 50020$:
770
771
772 000334' LET KE.LOC := R3
(4) 000334' 010367 177440 MOV R3,KE.LOC
773 000340' CALL CONV
(3) 000340' 004767 000006 JSR PC,CONV
774 000344' ENDF
(4) 000344' 50021$:
775 000344' ENDF
(4) 000344' 50017$:
776 000344' ENDF
(4) 000344' 50015$:
777 000344' ENDF
(4) 000344' 50013$:
778 000344' ENDF
(4) 000344' 50011$:
779 000344' ENDF
(4) 000344' 50003$:
780
781 ;+
782 ; CLEAN UP AND LEAVE
783 ; -
784
785 000344' CALL RESREG
(3) 000344' 004767 000000G JSR PC,RESREG
786
787 000350' ENDRTN
(3) 000350' 50000$:
(3) 000350' 50001$:
(2) 000350' 000207 RTS PC
788
789
790 000352' ROUTINE CONV
(2) 000352' CONV:
791
792 ;+
793 ; THIS ROUTINE CONVERTS THE LOCATION TO BE EXAMINED AND ITS CONTENTS
794 ; TO ASCII
795 ; -
796

```

```
797
798 000352'          CALL BOA16 IN <KE.LOC,#KE.MG2>
(3) 000352' 010546
(5) 000354' 012745 000002'
(4) 000360' 016745 177414
(3) 000364' 004767 000000G
(3) 000370' 012605
799 000372'          CALL BOA16 IN <@KE.LOC,#KE.MG3>
(3) 000372' 010546
(5) 000374' 012745 000012'
(4) 000400' 017745 177374
(3) 000404' 004767 000000G
(3) 000410' 012605
800 000412'          LET DT.KBRSP(R0) := #KE.MG1
(4) 000412' 012760 000002' 000022
801 000420'          ENDRTN
(3) 000420'
(3) 000420'
(2) 000420' 000207
802          000001          .END
```

MOV R5,-(SP)  
MOV #KE.MG2,-(R5)  
MOV KE.LOC,-(R5)  
JSR PC,BOA16  
MOV (SP)+,R5  
MOV R5,-(SP)  
MOV #KE.MG3,-(R5)  
MOV @KE.LOC,-(R5)  
JSR PC,BOA16  
MOV (SP)+,R5  
MOV #KE.MG1,DT.KBRSP  
50000S:  
50001S:  
RTS PC

ACSR = 000102	CM.BAD= ***** G	DT.WHL= 000054	KIPDR6= 172314	PRI5 = 000240
ACTBIT= 004000	CM.ODD= ***** G	DT.WLL= 000052	KIPDR7= 172316	PRI6 = 000300
ADDR22= 001000	CONFIG= 000056	DVID1 = 000014	KTERRO= 000040	PRI7 = 000340
ADR = 000006	CONV 000352R	ECCMEM= 000100	KTPRES= 000400	PRC = 000000
APTFER= 000004	CQOVF = 000001	ECCSTA= 000010	KTSTAT= 000020	PR4 = 000200
APTPRE= 000200	CR = 000015	ENBEDP= 010000	KTXTND= 040000	PR5 = 000240
ARGCHK= ***** G	CSRA = 000100	ENBNUL= 000001	LF = 000012	PR6 = 000300
ASB = 000106	CSRC = 000102	ENDLST= 000000	LPSTAT= 000001	PR7 = 000340
ASSEMB= 000010	CTRLC = 000003	EOPBIT= 000001	MAPSTA= 000200	PS = 177776
ASTAT = 000104	CTRLQ = 000017	ERRTYP= 000106	MED = 076600	PSW = 177776
AUTO = 000010	CTRLU = 000025	EVNTBE= 000200	MEMPAS= 040000	RANNUM= 000054
AUTOST= 020000	DCEVNT= 000011	EVNTHD= 000200	MODEXH= 004000	RBUFEA= 000130
AWAS = 000110	DEFRTN= 000400	EVNTKT= 000203	MODHOL= 002000	RBUFPA= 000126
BIT0 = 000001	DIAGMC= 000000	EVNTPE= 000202	MODSEL= 001000	RBUFSZ= 000132
BIT00 = 000001	DROPMO= 100000	EVNTRE= 000201	MSGCKD= 000010	RBUFVA= 000124
BIT01 = 000002	DSEVNT= 000014	FATERR= 100000	MSGCKS= 000011	RDSERV= 000101
BIT02 = 000004	DTADR = 000000	HRDCNT= 000044	MSGDER= 000005	RDWHMI= 000022
BIT03 = 000010	DT.ADD= 000042	HRDPAS= 000050	MSGDRP= 000017	RELERR= 000020
BIT04 = 000020	DT.AP = 000100	ICONT = 000036	MSGECH= 177777	RELMOD= 020000
BIT05 = 000040	DT.APK= 000076	ICOUNT= 000040	MSGEOP= 000013	RELTIM= 010000
BIT06 = 000100	DT.BLS= 000034	IDNUM = 000122	MSGHDR= 000004	RESREG= ***** G
BIT07 = 000200	DT.CFO= 000014	IE = 000100	MSGHNG= 000022	RES1 = 000056
BIT08 = 000400	DT.CF1= 000016	INDPAR= 000040	MSGHRD= 000007	RES2 = 000060
BIT09 = 001000	DT.ERR= 000020	INHDRP= 040000	MSGMAP= 000021	RICHAR= 031060
BIT1 = 000002	DT.ESI= 000044	INHREL= 020000	MSGNUL= 177775	RPTDAT= 002000
BIT10 = 002000	DT.EVN= 000000	INHREL= 001000	MSGPOP= 000002	RSTRT = 000112
BIT11 = 004000	DT.EXS= 000060	INHRRE= 000400	MSGPRM= 177776	RUBOUT= 000177
BIT12 = 010000	DT.FCH= 000037	INIT = 000030	MSGRES= 000001	RUNMOD= 100000
BIT13 = 020000	DT.FCN= 000036	INTR = 000120	MSGSFT= 000006	R5VALU= 001740
BIT14 = 040000	DT.HMX= 000104	IOMOD = 100000	MSGSKE= 000003	SAM = 075464
BIT15 = 100000	DT.KBE= 000024	IOMODP= 102000	MSGSMB= 000015	SAVREG= ***** G
BIT2 = 000004	DT.KBP= 000026	IOMODR= 112000	MSGSMH= 000014	SBADR = 000102
BIT3 = 000010	DT.KBR= 000022	IOMODX= 110000	MSGSMS= 000016	SBKMOD= 000000
BIT4 = 000020	DT.KBU= 000030	JACK = 035060	MSGSTD= 000000	SBKSEL= 010000
BIT5 = 000040	DT.MLS= 000032	KEXAM 000022RG	MSGSYS= 000012	SC.ADR= 000006
BIT6 = 000100	DT.MTI= 000110	KE.LOC 000000RG	MSGVEC= 000020	SC.ALC= 000014
BIT7 = 000200	DT.OFF= 000070	KE.MG1 000002R	NAMCHK= ***** G	SC.APC= 000016
BIT8 = 000400	DT.PAS= 000074	KE.MG2 000002R	NBKMOD= 001000	SC.CKL= 000002
BIT9 = 001000	DT.PC = 000002	KE.MG3 000012R	NPCUOP= 000020	SC.CKP= 000004
BKDEF = 000002	DT.PFL= 000062	KIPAR0= 172340	NOPTY= 000002	SC.CLO= 000000
BKMOD = 000020	DT.PSW= 000004	KIPAR1= 172342	NULL = 000000	SC.HLD= 000010
BKMODE= 040000	DT.PTA= 000064	KIPAR2= 172344	NUMCHK= ***** G	SC.SCA= 000012
BKSLSH= 000134	DT.RCS= 000102	KIPAR3= 172346	OWEN = 024020	SENDLS= 177777
BOA16 = ***** G	DT.REL= 000040	KIPAR4= 172350	PAERR = 000010	SOFCNT= 000042
CAPRES= 000004	DT.SCT= 000066	KIPAR5= 172352	PARPRE= 002000	SOPPAS= 000046
CASAT= 000004	DT.SMX= 000106	KIPAR6= 172354	PARSTA= 000100	SPACE = 000040
CDERCT= 000146	DT.SP = 000006	KIPAR7= 172356	PASCNT= 000034	SPOINT= 000032
CDWDCT= 000144	DT.SSI= 000046	KIPDR0= 172300	PDPPLSI= 020000	SPVALU= 002200
CKTIM = 100000	DT.ST0= 000010	KIPDR1= 172302	PDP60 = 004000	SR0 = 177572
CLKPRE= 000001	DT.ST1= 000012	KIPDR2= 172304	PDP70 = 010000	SR1 = 177574
CMBUF= 000002	DT.SWR= 000056	KIPDR3= 172306	PRI0 = 000000	SR2 = 177576
CM.ADR= ***** G	DT.SYP= 000072	KIPDR4= 172310	PRI1 = 000040	SR3 = 172516
CM.ARG= ***** G	DT.WBU= 000050	KIPDR5= 172312	PRI4 = 000200	STAT = 000026



STATBI= 064757	UIPDR2= 177604	\$FSCAS= 000150	\$ISK4 = 000001	\$\$BYTE= 000403
STAT1 = 000027	UIPDR3= 177606	\$F\$DEC= 000220	\$ISK5 = 000001	\$\$CASE= 000000
SUSPND= 000001	UIPDR4= 177610	\$F\$DO = 000340	\$LOCTA= 177777	\$\$DST = 000000
SVR0 = 000062	UIPDR5= 177612	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$ELOC= 000402
SVR1 = 000064	UIPDR6= 177614	\$F\$G00= 000400	\$LSTTA= 000001	\$\$ERFL= 000000
SVR2 = 000066	UIPDR7= 177616	\$F\$IF = 000110	\$NESTL= 177777	\$\$FLAG= 000001
SVR3 = 000070	WASADR= 000104	\$F\$INC= 000210	\$NSK0 = 000300	\$\$FROM= 000000
SVR4 = 000072	WBSTAT= 000040	\$F\$L00= 000200	\$NSK1 = 000110	\$\$LOC = 000322R
SVR5 = 000074	WBUFEA= 000136	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$LOCN= 000000
SVR6 = 000076	WBUFPA= 000134	\$F\$NO = 000403	\$NSK3 = 000110	\$\$REG = 177777
SYSCNT= 000052	WBUFRQ= 000140	\$F\$OR = 000320	\$NSK4 = 000110	\$\$RETU= 000000
SYERR= 000100	WBUFSZ= 000142	\$F\$RTI= 000350	\$NSK5 = 000110	\$\$RTN1= 050000
TMPI0 = 000002	WDFR = 000116	\$F\$RTN= 000300	\$NSK6 = 000110	\$\$RTN2= 050001
TQOVF = 000002	WDTO = 000114	\$F\$SEL= 000140	\$\$AVLE= 177777	\$\$SRC = 000000
UIPAR0= 177640	WTINRE= 000352	\$F\$THE= 000330	\$TAGLE= 177777	\$\$TGSV= 000000
UIPAR1= 177642	WTWHMI= 000222	\$F\$TRU= 000404	\$TAGNU= 050002	\$\$TGS1= 000000
UIPAR2= 177644	XFLAG = 000005	\$F\$UNT= 000130	\$TEMP = 000300	\$\$TGS2= 000000
UIPAR3= 177646	XOFF = 000023	\$F\$WHI= 000120	\$TSK0 = 050003	\$\$TO = 000002
UIPAR4= 177650	XON = 000021	\$F\$YES= 000402	\$TSK1 = 050011	\$\$STAG= 050000
UIPAR5= 177652	\$BGNLE= 177777	\$IFLEV= 177777	\$TSK2 = 050013	= 000422R
UIPAR6= 177654	SERFLG= 000400	\$ISK0 = 000001	\$TSK3 = 050015	
UIPAR7= 177656	SFSAND= 000310	\$ISK1 = 000001	\$TSK4 = 050017	
UIPDR0= 177600	\$F\$BAD= 000401	\$ISK2 = 000001	\$TSK5 = 050021	
UIPDR1= 177602	\$F\$BLA= 000170	\$ISK3 = 000001	\$ARGC= 000000	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:58  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0579

3 COMMON EQUATE MODULE  
558 COMMON DEFINITIONS AND REFERENCES FOR KFILL  
561 000000' .PRINT ;SPMAC: VERSION 1.1  
594 KFILL ROUTINE

```
508 .TITLE KFILL PROCESS THE 'FILL' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     KFILL
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS ROUTINE PROCESSES THE 'FILL' KEYBOARD COMMAND.
517 ;     IF NO ARGUMENT IS FOUND IN THE DECODE BUFFER, THE CONTENTS
518 ;     OF THE FILL COUNT/FILL CHAR ARE OUTPUTTED.
519 ;
520 ; INPUTS:
521 ;     1. ADDRESS OF DATA TABLE
522 ;     2. DECODE BUFFER POINTER
523 ;
524 ; IMPLICIT INPUTS:
525 ;     NONE
526 ;
527 ; OUTPUTS:
528 ;     NONE
529 ;
530 ; IMPLICIT OUTPUTS:
531 ;     1. DT.KBRSP
532 ;     2. DT.FCNT
533 ;
534 ; PATHOLOGICAL CONNECTIONS:
535 ;     CM.ARG
536 ;
537 ; SUBORDINATE ROUTINES CALLED:
538 ;     1. ARGCHK - VERIFY AN ARGUMENT
539 ;     2. BOA16 - BINARY TO OCTAL ASCII CONVERSION
540 ;     3. SAVREG
541 ;     4. RESREG
542 ;     5. NUMCHK
543 ;
544 ; FUNCTIONAL SIDE EFFECTS:
545 ;     NONE
546 ;
547 ; CALLING SEQUENCE:
548 ;     CALL KFILL IN <DTADR,BUFPTR>
549 ;     WHERE: DTADR = ADDRESS OF DTABLE
550 ;           BUFPTR = CMD DECODE BUFFER POINTER
551 ;
552 ; VERSION:
553 ;     0.0
554 ;
555 ;     EDIT          DATE          BY          REASON
556 ;--
```

```

558 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KFILL
559
560 .MCALL STRUCT
561 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
562
563 000001 $LSTIN=1
564 000001 $LSTTAG=1
565
566 ; REFERENCES BY OTHER MODULES:
567 ;
568 .GLOBL KFILL ;MODULE ENTRY POINT
569 ;
570 ;*****
571 ;
572 ; GLOBAL REFERENCES:
573 ;
574 .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION
575 .GLOBL SAVREG ;SAVE REGISTERS
576 .GLOBL RESREG ;RESTORE REGISTERS
577 .GLOBL ARGCHK ;CHECK AN ARGUMENT
578 .GLOBL NUMCHK ;CHECK AN OCTAL ASCII NUMBER
579 .GLOBL CM.ARG ;"INVALID OR MISSING ARGUMENT" MESSAGE
580 ;
581 ;*****
582 ;*****
583 ;
584 ; LOCAL STORAGE:
585 ;
586 000000' 043045 046111 027514 KF.MG1: .ASCII '%FILL/ ' ;FILL MESSAGE
000006' 040
587 000007' 000006 KF.MG2: .BLKB 6 ;HOLD ASCII DT.FLCNT CONTENTS
588 000015' 045 000 .ASCIZ /%/
589 000020' .EVEN
590 ;*****
591
592

```

```

594 .SBTTL KFILL ROUTINE
595
596 ROUTINE KFILL <DTADR,BUFPTR>
597 000020'
(2) 000020' KFILL:
598
599 ;+
600 ; SAVE REGISTERS AND SAVE DATA TABLE ADDRESS, AND DECODE BUFFER POINTER
601 ; -
602
603 000020' CALL SAVREG
(3) 000020' 004767 000000G JSR PC,SAVREG
604 000024' LET R0 := DTADR(R5) MOV DTADR(R5),R0
(4) 000024' 016500 000000 LET R1 := BUFPTR(R5) MOV BUFPTR(R5),R1
605 000030'
(4) 000030' 016501 000002
606
607
608 ;+
609 ; R3 GETS ADDRESS OF FILL COUNT
610 ; -
611
612 000034' LET R3 := #DT.FCNT + R0
(4) 000034' 012703 000036 MOV #DT.FCNT,R3
(6) 000040' 060003 ADD R0,R3
613
614 ;+
615 ; GET TO FIRST ARGUMENT
616 ; -
617 000042' CALL ARGCHK IN <R1> OUT <R1>
(4) 000042' 162705 000002 SUB #1*2,R5
(3) 000046' 010546 MOV R5,-(SP)
(4) 000050' 010145 MOV R1,-(R5)
(3) 000052' 004767 000000G JSR PC,ARGCHK
(3) 000056' 012605 MOV (SP)+,R5
(4) 000060' 012501 MOV (R5)+,R1
618
619 ;+
620 ; IF IT'S NOT A CARRIAGE RETURN, VERIFY THE NUMBER
621 ; -
622 000062' IF.NO.ERROR THEN
(6) 000062' 103464 BCS 50002$
623
624
625 000064' CALL NUMCHK IN <R0,R1> OUT <R1,R2>
(4) 000064' 162705 000004 SUB #2*2,R5
(3) 000070' 010546 MOV R5,-(SP)
(5) 000072' 010145 MOV R1,-(R5)
(4) 000074' 010045 MOV R0,-(R5)
(3) 000076' 004767 000000G JSR PC,NUMCHK
(3) 000102' 012605 MOV (SP)+,R5
(4) 000104' 012501 MOV (R5)+,R1
(4) 000106' 012502 MOV (R5)+,R2
626
627 ;+
628 ; IF THERE IS AN ERROR IN THE NUMBER, LEAVE

```

```

629          ; -
630 000110'          IF.ERROR THEN
(6) 000110' 103001          BCC      50003$
631
632
633 000112'          INLINE <BR      1$>
(2) 000112' 000463          BR      1$
634
635 000114'          ENDIF
(4) 000114'          50003$:
636
637          ; +
638          ; GET NEXT ARGUMENT
639          ; -
640
641 000114'          CALL ARGCHK IN <R1> OUT <R1>
(4) 000114' 162705 000002          SUB      #1*2,R5
(3) 000120' 010546          MOV      R5,-(SP)
(4) 000122' 010145          MOV      R1,-(R5)
(3) 000124' 004767 000000G          JSR      PC,ARGCHK
(3) 000130' 012605          MOV      (SP)+,R5
(4) 000132' 012501          MOV      (R5)+,R1
642
643          ; +
644          ; IF AN ERROR... HE DIDN'T GIVE ENOUGH ARGUMENTS
645          ; -
646
647 000134'          IF.ERROR THEN
(6) 000134' 103004          BCC      50004$
648
649
650 000136'          LET.DT.KBRSP(R0) := #CM.ARG
(4) 000136' 012760 000000G 000022          MOV      #CM.ARG,DT.KBRSP
651 000144'          INLINE <BR      1$>
(2) 000144' 000446          BR      1$
652
653 000146'          ENDIF
(4) 000146'          50004$:
654
655 000146'          CALL NUMCHK IN <R0,R1> OUT <R1,R4>
(4) 000146' 162705 000004          SUB      #2*2,R5
(3) 000152' 010546          MOV      R5,-(SP)
(5) 000154' 010145          MOV      R1,-(R5)
(4) 000156' 010045          MOV      R0,-(R5)
(3) 000160' 004767 000000G          JSR      PC,NUMCHK
(3) 000164' 012605          MOV      (SP)+,R5
(4) 000166' 012501          MOV      (R5)+,R1
(4) 000170' 012504          MOV      (R5)+,R4
656
657          ; +
658          ; WE GOT THE SECOND PART OF THE ARGUMENT, IF INVALID, LEAVE
659          ; -
660
661 000172'          IF.ERROR THEN
(6) 000172' 103001          BCC      50005$
662

```

```

663 000174'          INLINE <BR      1$>
(2) 000174' 000432          BR      1$
664
665 000176'          ENDIF
(4) 000176'          50005$:
666
667
668          ;+
669          ; NUMBER OK, VERIFY NO JUNK ARGUMENTS EXIST
670          ; -
671
672 000176'          CALL ARGCHK IN <R1> OUT <R1>
(4) 000176' 162705 000002          SUB      #1*2,R5
(3) 000202' 010546          MOV      R5,-(SP)
(4) 000204' 010145          MOV      R1,-(R5)
(3) 000206' 004767 000000G          JSR      PC,ARGCHK
(3) 000212' 012605          MOV      (SP)+,R5
(4) 000214' 012501          MOV      (R5)+,R1
673
674          ;+
675          ; IF THERE ARE MORE JUNK ARGUMENTS - LEAVE, BUT STUFF ERROR MSG FIRST
676          ; -
677 000216'          IF.NO.ERROR THEN
(6) 000216' 103404          BCS      50006$
678
679
680 000220'          LET DT.KBRSP(R0) := #CM.ARG
(4) 000220' 012760 000000G 000022          MOV      #CM.ARG,DT.KBRSP
681 000226'          INLINE <BR      1$>
(2) 000226' 000415          BR      1$
682
683 000230'          ENDIF
(4) 000230'          50006$:
684
685          ;+
686          ; NO JUNK ARGUMENTS - SO REPLACE FILL COUNT/CHAR WITH
687          ; NEW VALUES
688          ; -
689
690 000230'          LET (R3)+ :B= R4
(4) 000230' 110423          MOV      R4,(R3)+
691 000232'          LET (R3) :B= R2
(4) 000232' 110213          MOV      R2,(R3)
692
693
694 000234'          ENDIF
(4) 000234'          50002$:
695
696          ;+
697          ; CONVERT THE NUMBER IN DT.FCNT TO ASCII, STUFF MESSAGE
698          ; AND RETURN
699          ; -
700 000234'          CALL BOA16 IN <DT.FCNT(R0),#KF.MG2>
(3) 000234' 010546          MOV      R5,-(SP)
(5) 000236' 012745 000007'          MOV      #KF.MG2,-(R5)
(4) 000242' 016045 000036          MOV      DT.FCNT(R0),-(R5)

```

```
(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
```



ACSR = 000102	CQOVF = 000001	ECCSTA= 000010	LF = 000012	PR7 = 000340
ACTBIT= 004000	CR = 000015	ENBEOP= 010000	LPSTAT= 000001	PS = 177776
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	MAPSTA= 000200	PSW = 177776
ADR = 000006	CSRC = 000102	ENDLST= 000000	MED = 076600	RANNUM= 000054
APTFER= 000004	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	RBUFEA= 000130
APTPRE= 000200	CTRLD = 000017	ERRTYP= 000106	MODEXH= 004000	RBUFPA= 000126
ARGCHK= ***** G	CTRLU = 000025	EVNTBE= 000200	MODHOL= 002000	RBUFSZ= 000132
ASB = 000106	DCEVNT= 000011	EVNTHD= 000200	MODSEL= 001000	RBUFVA= 000124
ASSEMB= 000010	DEFRTN= 000400	EVNTKT= 000203	MSGCKD= 000010	RDSERV= 000101
ASTAT = 000104	DIAGMC= 000000	EVNTPE= 000202	MSGCKS= 000011	RDWHMI= 000022
AUTO = 000010	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	RELERR= 000020
AUTOST= 020000	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	RELMOD= 020000
AWAS = 000110	DTADR = 000000	HRDCNT= 000044	MSGECH= 177777	RELTIM= 010000
BIT0 = 000001	DT.ADD= 000042	HRDPAS= 000050	MSGEOP= 000013	RESREG= ***** G
BIT00 = 000001	DT.AP = 000100	ICONT = 000036	MSGHDR= 000004	RES1 = 000056
BIT01 = 000002	DT.APK= 000076	ICOUNT= 000040	MSGHNG= 000022	RES2 = 000060
BIT02 = 000004	DT.BLS= 000034	IDNUM = 000122	MSGHRD= 000007	RICHAR= 031060
BIT03 = 000010	DT.CF0= 000014	IE = 000100	MSGMAP= 000021	RPTDAT= 002000
BIT04 = 000020	DT.CF1= 000016	INDPAR= 000040	MSGNUL= 177775	RSTRT = 000112
BIT05 = 000040	DT.ERR= 000020	INHDRP= 040000	MSGPOP= 000002	RUSCUT= 000177
BIT06 = 000100	DT.ESI= 000044	INHPR= 020000	MSGPRM= 177776	RUNMOD= 100000
BIT07 = 000200	DT.EVN= 000000	INHREL= 001000	MSGRES= 000001	R5VALU= 001740
BIT08 = 000400	DT.EXS= 000060	INHRE= 000400	MSGSFT= 000006	SAM = 075464
BIT09 = 001000	DT.FCH= 000037	INIT = 000030	MSGSKE= 000003	SAVREG= ***** G
BIT1 = 000002	DT.FCN= 000036	INTR = 000120	MSGSMB= 000015	SBADR = 000102
BIT10 = 002000	DT.HMX= 000104	IOMOD = 100000	MSGSMH= 000014	SBKMOD= 000000
BIT11 = 004000	DT.KBE= 000024	IOMODP= 102000	MSGSMS= 000016	SBKSEL= 010000
BIT12 = 010000	DT.KBP= 000026	IOMODR= 112000	MSGSTD= 000000	SC.ADR= 000006
BIT13 = 020000	DT.KBR= 000022	IOMODX= 110000	MSGSYS= 000012	SC.ALC= 000014
BIT14 = 040000	DT.KBU= 000030	JACK = 035060	MSGVEC= 000020	SC.APC= 000016
BIT15 = 100000	DT.MLS= 000032	KFILL 000020RG	NBKMOD= 001000	SC.CKL= 000002
BIT2 = 000004	DT.MTI= 000110	KF.MG1 000000R	NCPUOP= 000020	SC.CKP= 000004
BIT3 = 000010	DT.OFF= 000070	KF.MG2 000007R	NOPTY= 000002	SC.CLO= 000000
BIT4 = 000020	DT.PAS= 000074	KIPAR0= 172340	NULL = 000000	SC.HLD= 000010
BIT5 = 000040	DT.PC = 000002	KIPAR1= 172342	NUMCHK= ***** G	SC.SCA= 000012
BIT6 = 000100	DT.PFL= 000062	KIPAR2= 172344	OWEN = 024020	SENDLS= 177777
BIT7 = 000200	DT.PSW= 000004	KIPAR3= 172346	PAERR = 000010	SOFCNT= 000042
BIT8 = 000400	DT.PTA= 000064	KIPAR4= 172350	PARPRE= 002000	SOPPAS= 000046
BIT9 = 001000	DT.RCS= 000102	KIPAR5= 172352	PARSTA= 000100	SPACE = 000040
BKDEF = 000002	DT.REL= 000040	KIPAR6= 172354	PASCNT= 000034	SPOINT= 000032
BKMOD = 000020	DT.SCT= 000066	KIPAR7= 172356	PDPLSI= 020000	SPVALU= 002200
BKMODE= 040000	DT.SMX= 000106	KIPDR0= 172300	PDP60 = 004000	SR0 = 177572
BKSLSH= 000134	DT.SP = 000006	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
BOA16 = ***** G	DT.SSI= 000046	KIPDR2= 172304	PRI0 = 000000	SR2 = 177576
BUFPTR= 000002	DT.ST0= 000010	KIPDR3= 172306	PRI1 = 000040	SR3 = 172516
CAPRES= 000004	DT.ST1= 000012	KIPDR4= 172310	PRI4 = 000200	STAT = 000026
CASTAT= 000004	DT.SWR= 000056	KIPDR5= 172312	PRI5 = 000240	STATBI= 064757
CDERCT= 000146	DT.SYP= 000072	KIPDR6= 172314	PRI6 = 000300	STAT1 = 000027
CDWDCT= 000144	DT.WBU= 000050	KIPDR7= 172316	PRI7 = 000340	SUSPND= 000001
CKTIM = 100000	DT.WHL= 000054	KTERRO= 000040	PRO = 000000	SVR0 = 000062
CLKPRE= 000001	DT.WLL= 000052	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CM.ARG= ***** G	DVID1 = 000014	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CONFIG= 000056	ECCMEM= 000100	KTXTND= 040000	PR6 = 000300	SVR3 = 000070

SVR4 = 000072	UIPDR5= 177612	\$FSBLA= 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 = 000216R
TMPID = 000002	WBUFEA= 000136	\$F\$GDO= 000400	\$LSTTA= 000001	\$\$LOCN= 000000
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NESTL= 177777	\$\$REG = 177777
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	SNSK0 = 000300	\$\$RETU= 000000
UIPAR1= 177642	WBUF SZ= 000142	\$F\$LDO= 000200	SNSK1 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	SNSK2 = 000110	\$\$RTN2= 050001
UIPAR3= 177646	WDTQ = 000114	\$F\$NO = 000403	SSAVLE= 177777	\$\$SRC = 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	STAGLE= 177777	\$\$TGSV= 000000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	STAGNU= 050007	\$\$TGS1= 000000
UIPAR6= 177654	XFLAG = 000005	\$F\$RTN= 000300	STEMP = 000300	\$\$TGS2= 000000
UIPAR7= 177656	XOFF = 000023	\$F\$SEL= 000140	STSK0 = 050002	\$\$TO = 000000
UIPDR0= 177600	XON = 000021	\$F\$THE= 000330	STSK1 = 050006	\$\$TAG= 050000
UIPDR1= 177602	\$BGNLE= 177777	\$F\$TRU= 000404	\$\$ARGC= 000004	= 000270R
UIPDR2= 177604	\$ERFLG= 000400	\$F\$UNT= 000130	\$\$BYTE= 000000	
UIPDR3= 177606	\$F\$AND= 000310	\$F\$WHI= 000120	\$\$CASE= 000000	
UIPDR4= 177610	\$F\$BAD= 000401	\$F\$YES= 000402	\$\$DST = 000000	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 17:59  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0588

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES FOR KMAP  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
598 KMAP ROUTINE

```
508 .TITLE KMAP PROCESS THE 'MAP' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KMAP
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'MAP' KEYBOARD COMMAND. IT WILL
517 ; SEARCH THE MODULE LIST AND ENQUEUE THE MAP MESSAGE FOR
518 ; EACH MODULE. IF A MODULE NAME IS SPECIFIED, A MAP
519 ; MESSAGE IS OUTPUTTED FOR THAT MODULE ONLY.
520 ;
521 ; INPUTS:
522 ; 1. ADDRESS OF DATA TABLE
523 ; 2. COMMAND DECODE BUFFER POINTER
524 ;
525 ; IMPLICIT INPUTS:
526 ; DT.MLST
527 ; DT.STO
528 ;
529 ; OUTPUTS:
530 ; NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ; DT.KBRSP
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ; CM.ARG,CM.BADNAME
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ; 1. ARGCHK - CHECK FOR JUNK ARGUMENTS
540 ; 2. ENQTQ - ENQUEUE A MESSAGE
541 ; 3. NAMCHK - VERIFY A MODULE NAME
542 ; 4. SAVREG - SAVE REGISTERS
543 ; 5. RESREG - RESTORE REGISTERS
544 ;
545 ; FUNCTIONAL SIDE EFFECTS:
546 ; NONE
547 ;
548 ; CALLING SEQUENCE:
549 ; CALL KMAP IN <DTADR,BUFPTR>
550 ; WHERE DTADR = ADDRESS OF DATA TABLE
551 ; BUFPTR = COMMAND DECODE BUFFER POINTER
552 ;
553 ; VERSION:
554 ; 0.0
555 ;
556 ; EDIT DATE BY REASON
557 ;--
558
559
```

```
561 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KMAP
562
563 .MCALL STRUCT
564 000000' STRUCT
565 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
566 000001 $LSTTAG=1
567 000001 $LSTIN=1
568
569
570 ;*****
571 ;
572 ; REFERENCED BY OTHER MODULES:
573 ;
574 .GLOBL KMAP ;MODULE ENTRY POINT
575 ;
576 ;*****
577 ;
578 ; GLOBAL REFERENCES:
579 ;
580 .GLOBL ARGCHK ;CHECK AN ARGUMENT
581 .GLOBL SAVREG ;SAVE REGS 0-4
582 .GLOBL RESREG ;RESTORE REGS 0-4
583 .GLOBL ENQTQ ;ENQUEUE A MESSAGE
584 .GLOBL CM.ARG ;'ILLEGAL ARGUMENT' ERROR MSG
585 .GLOBL NAMCHK ;CHECK A MODULE NAME
586 .GLOBL CM.BADNAME ;'INVALID MODULE NAME' ERROR MESSAGE
587 ;
588 ;*****
589 ;
590 ; LOCAL STORAGE:
591 ;
592 000000' 000000 KM.HLD: .WORD 0 ;HOLD ADDRESS OF MESSAGE
593 000002' 041445 042115 000076 KM.CMD: .ASCIZ /%CMD>/
594 000010' 041045 054523 000076 KM.RUN: .ASCIZ /%BSY>/
595 .EVEN
596
```

```

598 .SBTTL KMAP ROUTINE
599
600 000016' ROUTINE KMAP <DTADR,BUFPTR>
(2) 000016' KMAP:
601
602 ;+
603 ; INITIALIZE AND SAVE DTABLE ADDRESS AND DECODE BUFFER PTR
604 ; -
605
606 000016' CALL SAVREG
(3) 000016' 004767 000000G JSR PC,SAVREG
607 000022' LET R0 := DTADR(R5)
(4) 000022' 016500 000000 MOV DTADR(R5),R0
608 000026' LET R1 := BUFPTR(R5)
(4) 000026' 016501 000002 MOV BUFPTR(R5),R1
609
610
611
612
613
614 ;+
615 ; IF IN RUNMODE , STUFF FAKE COMMAND PROMPT SINCE
616 ; THIS COMMAND USES ENQTQ, IT WILL O/P THE NORMAL
617 ; PROMPT BEFORE ANY OF THE MAP MESSAGES WILL BE
618 ; PRINTED - THEN THIS DUMMY PROMPT WILL BE
619 ; OUTPUTTED.....
620 ; -
621 000032' IF #RUNMODE SETIN DT.STO(R0) THEN
(6) 000032' 032760 100000 000010 BIT #RUNMODE,DT.STO(
(9) 000040' 001404 BEQ 50002$
622 000042' LET KM.HLD := #KM.RUN
(4) 000042' 012767 000010' 177730 MOV #KM.RUN,KM.HLD
623 000050' ELSE
(4) 000050' 000403 BR 50003$
(3) 000052' 50002$:
624 000052' LET KM.HLD := #KM.CMD
(4) 000052' 012767 000002' 177720 MOV #KM.CMD,KM.HLD
625 000060' ENDIF
(4) 000060' 50003$:
626
627 ;+
628 ; SEE IF ANY ARGUMENTS ARE INCLUDED IN DECODE BUFFER
629 ; -
630
631 000060' CALL ARGCHK IN <R1> OUT <R1>
(4) 000060' 162705 000002 SUB #1*2,R5
(3) 000064' 010546 MOV R5,-(SP)
(4) 000066' 010145 MOV R1,-(R5)
(3) 000070' 004767 000000G JSR PC,ARGCHK
(3) 000074' 012605 MOV (SP)+,R5
(4) 000076' 012501 MOV (R5)+,R1
632 ;+
633 ; IF THERE ARE NO ARGUMENTS DO A FULL MAP (ALL MODULES). GET ADDRESS
634 ; OF MODULE LIST. THEN, WHILE
635 ; THE ENTRY IN THE MODULE LIST IS NOT 0, ENQUEUE THE MAP MESSAGE
636 ; FOR THE MODULES.

```

```

637          ; -
638
639
640 000100'          IF.ERROR THEN
(6) 000100' 103023          BCC          50004$
641
642 000102'          LET R2 := DT.MLST(R0)
(4) 000102' 016002 000032          MOV          DT.MLST(R0),R2
643 000106'          WHILE (R2) NE #ENDLST DO
(4) 000106'          50005$:
(6) 000106' 021227 000000          CMP          (R2),#ENDLST
(9) 000112' 001415          BEQ          50006$
644 000114'          CALL ENQTQ IN <R0,#MSGMAP,#0,(R2)+,#0>
(3) 000114' 010546          MOV          R5,-(SP)
(8) 000116' 012745 000000          MOV          #0,-(R5)
(7) 000122' 012245          MOV          (R2)+,-(R5)
(6) 000124' 012745 000000          MOV          #0,-(R5)
(5) 000130' 012745 000021          MOV          #MSGMAP,-(R5)
(4) 000134' 010045          MOV          R0,-(R5)
(3) 000136' 004767 000000G        JSR          PC,ENQTQ
(3) 000142' 012605          MOV          (SP)+,R5
645 000144'          ENDDO
(4) 000144' 000760          BR          50005$
(3) 000146'          50006$:
646
647 000146'          ELSE
(4) 000146' 000450          BR          50007$
(3) 000150'          50004$:
648
649          ; +
650          ; THERE IS AN ARGUMENT - GO CHECK IT OUT
651          ; -
652
653 000150'          CALL NAMCHK IN <R0,R1> OUT <R2,R1>
(4) 000150' 162705 000004          SUB          #2*2,R5
(3) 000154' 010546          MOV          R5,-(SP)
(5) 000156' 010145          MOV          R1,-(R5)
(4) 000160' 010045          MOV          R0,-(R5)
(3) 000162' 004767 000000G        JSR          PC,NAMCHK
(3) 000166' 012605          MOV          (SP)+,R5
(4) 000170' 012502          MOV          (R5)+,R2
(4) 000172' 012501          MOV          (R5)+,R1
654          ; +
655          ; IF MODULE NAME IS O.K. - JUST VERIFY THERE IS NO JUNK ARGUMENT
656          ; -
657 000174'          IF.NO.ERROR THEN
(6) 000174' 103432          BCS          50010$
658
659
660          CALL ARGCHK IN <R1> OUT <R1>
(4) 000176' 162705 000002          SUB          #1*2,R5
(3) 000202' 010546          MOV          R5,-(SP)
(4) 000204' 010145          MOV          R1,-(R5)
(3) 000206' 004767 000000G        JSR          PC,ARGCHK
(3) 000212' 012605          MOV          (SP)+,R5
(4) 000214' 012501          MOV          (R5)+,R1

```

```

661          ;+
662          ; IF NO JUNK ARGUMENTS.... ENQUEUE THE MESSAGE FOR THIS MODULE
663          ;-
664 000216'          IF.ERROR THEN
665 (6) 000216' 103015          BCC          50011$
666
667 000220'          CALL ENQTQ IN <R0,#MSGMAP,#0,R2,#0>
668 (3) 000220' 010546          MOV          R5,-(SP)
669 (8) 000222' 012745 000000          MOV          #0,-(R5)
670 (7) 000226' 010245          MOV          R2,-(R5)
671 (6) 000230' 012745 000000          MOV          #0,-(R5)
672 (5) 000234' 012745 000021          MOV          #MSGMAP,-(R5)
673 (4) 000240' 010045          MOV          R0,-(R5)
674 (3) 000242' 004767 000000G          JSR          PC,ENQTQ
675 (3) 000246' 012605          MOV          (SP)+,R5
676
677          ELSE
678 (4) 000250' 000403          BR          50012$
679 (3) 000252'          50011$:
680
681          ;+
682          ; THERE IS A JUNK ARGUMENT IN DECODE BUFFER... STUFF ERROR MSG...
683          ;-
684 000252'          LET DT.KBRSP(R0) := #CM.ARG
685 (4) 000252' 012760 000000G 000022          MOV          #CM.ARG,DT.KBRSP
686
687          ENDIF
688 (4) 000260'          50012$:
689
690          ELSE
691 (4) 000260' 000403          BR          50013$
692 (3) 000262'          50010$:
693
694          ;+
695          ; BAD MODULE NAME....STUFF ERROR MSG....
696          ;-
697 000262'          LET DT.KBRSP(R0) := #CM.BADNAME
698 (4) 000262' 012760 000000G 000022          MOV          #CM.BADNAME,DT.K
699
700          ENDIF
701 (4) 000270'          50013$:
702
703          ENDIF
704 (4) 000270'          50007$:
705
706          ;+
707          ; IF NO ERRORS, ALSO ENQUEUE A PROMPT
708          ;-
709
710          IF DT.KBRSP(R0) EQ #0 THEN
711 (6) 000270' 005760 000022          TST          DT.KBRSP(R0)
712 (9) 000274' 001015          BNE          50014$

```



```
697 000276'          CALL ENQTQ IN <R0,#MSGSTD,KM.HLD,#0,#0>
(3) 000276' 010546
(8) 000300' 012745 000000
(7) 000304' 012745 000000
(6) 000310' 016745 177464
(5) 000314' 012745 000000
(4) 000320' 010045
(3) 000322' 004767 000000G
(3) 000326' 012605
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	CQOVF = 000001	ECCSTA= 000010	KTSTAT= 000020	PR5 = 000240
ACTBIT= 004000	CR = 000015	ENBEOB= 010000	KTXTND= 040000	PR6 = 000300
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	LF = 000012	PR7 = 000340
ADR = 000006	CSRC = 000102	ENLST= 000000	LPSTAT= 000001	PS = 177776
APTFER= 000004	CTRLC = 000003	ENQTQ = ***** G	MAPSTA= 000200	PSW = 177776
APTPRE= 000200	CTRLQ = 000017	EOPBIT= 000001	MED = 076600	RANNUM= 000054
ARGCHK= ***** G	CTRLU = 000025	ERRTYP= 000106	MEMPAS= 040000	RBUFEA= 000130
ASB = 000106	DCEVNT= 000011	EVNTBE= 000200	MODEXH= 004000	RBUFPA= 000126
ASSEMB= 000010	DEFRTN= 000400	EVNTHD= 000200	MODHOL= 002000	RBUFSZ= 000132
ASTAT = 000104	DIAGMC= 000000	EVNTKT= 000203	MODSEL= 001000	RBUFVA= 000124
AUTO = 000010	DROPMO= 100000	EVNTPE= 000202	MSGCKD= 000010	RDSERV= 000101
AUTQST= 020000	DSEVNT= 000014	EVNTRE= 000201	MSGCKS= 000011	RDWHMI= 000022
AWAS = 000110	DTADR = 000000	FATERR= 100000	MSGDER= 000005	RELERR= 000020
BIT0 = 000001	DT.ADD= 000042	HRDCNT= 000044	MSGDRP= 000017	RELMOD= 020000
BIT00 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGECH= 177777	RELTIM= 010000
BIT01 = 000002	DT.APK= 000076	ICONT = 000036	MSGEOP= 000013	RESREG= ***** G
BIT02 = 000004	DT.BLS= 000034	ICOUNT= 000040	MSGHDR= 000004	RES1 = 000056
BIT03 = 000010	DT.CF0= 000014	IDNUM = 000122	MSGHNG= 000022	RES2 = 000060
BIT04 = 000020	DT.CF1= 000016	IE = 000100	MSGHRD= 000007	RICHAR= 031060
BIT05 = 000040	DT.ERR= 000020	INDPAR= 000040	MSGMAP= 000021	RPTDAT= 002000
BIT06 = 000100	DT.ESI= 000044	INHDRP= 040000	MSGNUL= 177775	RSTRT = 000112
BIT07 = 000200	DT.EVN= 000000	INHEPR= 020000	MSGPOP= 000002	RUBCUT= 000177
BIT08 = 000400	DT.EXS= 000060	INHREL= 001000	MSGPRM= 177776	RUNMOD= 100000
BIT09 = 001000	DT.FCH= 000037	INHRRR= 000400	MSGRES= 000001	R5VALU= 001740
BIT1 = 000002	DT.FCN= 000036	INIT = 000030	MSGSFT= 000006	SAM = 075464
BIT10 = 002000	DT.HMX= 000104	INTR = 000120	MSGSKE= 000003	SAVREG= ***** G
BIT11 = 004000	DT.KBE= 000024	IOMOD = 100000	MSGSMB= 000015	SBADR = 000102
BIT12 = 010000	DT.KBP= 000026	IOMODP= 102000	MSGSMH= 000014	SBKMOD= 000000
BIT13 = 020000	DT.KBR= 000022	IOMODR= 112000	MSGSMS= 000016	SBKSEL= 010000
BIT14 = 040000	DT.KBU= 000030	IOMODX= 110000	MSGSTD= 000000	SC.ADR= 000006
BIT15 = 100000	DT.MLS= 000032	JACK = 035060	MSGSYS= 000012	SC.ALC= 000014
BIT2 = 000004	DT.MTI= 000110	KIPAR0= 172340	MSGVEC= 000020	SC.APC= 000016
BIT3 = 000010	DT.OFF= 000070	KIPAR1= 172342	NAMCHK= ***** G	SC.CKL= 000002
BIT4 = 000020	DT.PAS= 000074	KIPAR2= 172344	NBKMOD= 001000	SC.CKP= 000004
BIT5 = 000040	DT.PC = 000002	KIPAR3= 172346	NCPJOP= 000020	SC.CLO= 000000
BIT6 = 000100	DT.PFL= 000062	KIPAR4= 172350	NOAPTY= 000002	SC.HLD= 000010
BIT7 = 000200	DT.PSW= 000004	KIPAR5= 172352	NULL = 000000	SC.SCA= 000012
BIT8 = 000400	DT.PTA= 000064	KIPAR6= 172354	OWEN = 024020	SENDLS= 177777
BIT9 = 001000	DT.RCS= 000102	KIPAR7= 172356	PAERR = 000010	SOFCNT= 000042
BKDEF = 000002	DT.REL= 000040	KIPDR0= 172300	PARPRE= 002000	SOFPAS= 000046
BKMOD = 000020	DT.SCT= 000066	KIPDR1= 172302	PARSTA= 000100	SPACE = 000040
BKMODE= 040000	DT.SMX= 000106	KIPDR2= 172304	PASCNT= 000034	SPOINT= 000032
BKSLSH= 000134	DT.SP = 000006	KIPDR3= 172306	PDPLSI= 020000	SPVALU= 002200
BUFPTR= 000002	DT.SSI= 000046	KIPDR4= 172310	PDP60 = 004000	SR0 = 177572
CAPRES= 000004	DT.STO= 000010	KIPDR5= 172312	PDP70 = 010000	SR1 = 177574
CASAT= 000004	DT.ST1= 000012	KIPDR6= 172314	PR10 = 000000	SR2 = 177576
CDERCT= 000146	DT.SWR= 000056	KIPDR7= 172316	PRI1 = 000040	SR3 = 172516
CDWDCT= 000144	DT.SYP= 000072	KMAP 000016RG	PRI4 = 000200	STAT = 000026
CKTIM = 100000	DT.WBU= 000050	KM.CMD 000002R	PRI5 = 000240	STATBI= 064757
CLKPRE= 000001	DT.WHL= 000054	KM.HLD 000000R	PRI6 = 000300	STAT1 = 000027
CM.ARG= ***** G	DT.WLL= 000052	KM.RUN 000010R	PRI7 = 000340	SUSPND= 000001
CM.BAD= ***** G	DVID1 = 000014	KTERRO= 000040	PRO = 000000	SVRO = 000062
CONFIG= 000056	ECCMEM= 000100	KTPRES= 000400	PR4 = 000200	SVR1 = 000064

SYMBOL TABLE

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	\$\$ELOD= 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\$GOD= 000400	\$LSTTA= 000001	\$\$FROM= 000000
SYSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$NESTL= 177777	\$\$LOC = 000274R
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSKO = 000300	\$\$LOCN= 000000
TQOVF = 000002	WBUFRQ= 000140	\$F\$LQD= 000200	\$NSK1 = 000110	\$\$REG = 177777
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$RETU= 000000
UIPAR1= 177642	WDFR = 000116	\$F\$ND = 000403	\$NSK3 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDT0 = 000114	\$F\$OR = 000320	\$\$AVLE= 177777	\$\$RTN2= 050001
UIPAR3= 177646	WTINRE= 000352	\$F\$RTI= 000350	\$SSKO = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:00

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0597

3 COMMON EQUATE MODULE  
573 COMMON DEFINITIONS AND REFERENCES FOR KMOD  
576 000000' .PRINT ;SPMAC: VERSION 1.1  
628 KMOD ROUTINE



564  
565  
566  
567  
568  
569  
570  
571

```
;  
; VERSION:  
; 0.0  
;  
; EDIT DATE BY REASON  
;---
```

```
573 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KMOD
574
575 .MCALL STRUCT
576 000000' STRUCT
577 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
578 000001 $LSTTAG=1
579 000001 $LSTIN=1
580
581 ;*****
582 ;
583 ; REFERENCED BY OTHER MODULES:
584 ;
585 .GLOBL KMOD ;MODULE ENTRY POINT
586 ;
587 ;*****
588 ;
589 ; GLOBAL REFERENCES:
590 ;
591 .GLOBL CM.ARG ;'ILLEGAL ARGUMENT'
592 .GLOBL MSGDEQ ;MESSAGE DEQUEUER
593 .GLOBL CMDCPY ;COPY KEYBOARD BUF TO DECODE BUF
594 .GLOBL CM.ADR ;'ILLEGAL ADDRESS'
595 .GLOBL CM.RUN ;'ILLEGAL COMMAND IN RUN MODE
596 .GLOBL NAMCHK ;CHECK AN OPTION MODULE NAME
597 .GLOBL CM.BADNAME ;'INVALID MODULE MODULE NAME'
598 .GLOBL ENQTQ ;ENQUEUE A MESSAGE
599 .GLOBL NUMCHK ;CHECK A NUMBER
600 .GLOBL ARGCHK ;CHECK AN ARGUMENT
601 .GLOBL HRDADRCHK ;CHECK AN ADDRESS ROUTINE
602 .GLOBL CM.ODD ;'NUMBER MUST BE EVEN
603 .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION
604 .GLOBL SAVREG ;SAVE REGISTERS
605 .GLOBL KBINI ;KEYBOARD INIT ROUTINE
606 .GLOBL RESREG ;RESTORE REGISTERS
607 .GLOBL DX.KFL ;CARRIAGE RETURN FLAG
608 ;
609 ;*****
610 ;
611 ; LOCAL STORAGE:
612 ;
613 000000' 000000 KM.LOC: .WORD 0 ;HOLD LOCATION TO BE MODIFIED
614 000002' 000006 KM.MG1: .BLKB 6 ;HOLD ASCII LOCATION
615 000010' 020057 .ASCII '/' ;SEPARATOR
616 000012' 000006 KM.MG2: .BLKB 6 ;HOLD ASCII CONTENTS OF LOCATION
617 000020' 020040 .ASCII / / ;SPACES
618 000022' 000 .BYTE 0 ;TERMINATOR (NO CR DESIRED)
619 000023' 000 KM.ABT: .BYTE 0 ;ABORT FLAG
620 000024' 000 KM.FLG: .BYTE 0 ;KBINT FLAG
621 ;
622 000026' .EVEN
623 ;*****
624
625
626
```

```
628 .SBTTL KMOD ROUTINE
629
630 000026' ROUTINE KMOD <DTADR,BUFPTR>
(2) 000026' KMOD:
631
632 ;+
633 ;SAVE REGISTERS AND SAVE DTABLE ADDRESS + BUFFER POINTER
634 ;-
635
636 000026' CALL SAVREG
(3) 000026' 004767 000000G JSR PC,SAVREG
637 000032' LET R0 := DTADR(R5) MOV DTADR(R5),R0
(4) 000032' 016500 000000 MOV BUFPTR(R5),R1
638 000036' LET R1 := BUFPTR(R5)
(4) 000036' 016501 000002
639 ;+
640 ; TYPE OUT THE CR AT END OF MOD COMMAND
641 ;-
642
643 000042' CALL MSGDEQ IN <R0>
(3) 000042' 010546 MOV R5,-(SP)
(4) 000044' 010045 MOV R0,-(R5)
(3) 000046' 004767 000000G JSR PC,MSGDEQ
(3) 000052' 012605 MOV (SP)+,R5
644
645 ;+
646 ; INIT ABORT FLAG
647 ;-
648
649 000054' LET KM.ABT :B= #0
(4) 000054' 105067 177743 CLR KM.ABT
650
651
652
653 ;+
654 ; IF IN RUNMODE, LOAD MESSAGE AND LEAVE
655 ;-
656
657 000060' IF #RUNMODE SETIN DT.STO(R0) THEN
(6) 000060' 032760 100000 000010 BIT #RUNMODE,DT.STO(
(9) 000066' 001405 BEQ 50002$
658
659 000070' LET DT.KBRSP(R0) := #CM.RUN
(4) 000070' 012760 000000G 000022 MOV #CM.RUN,DT.KBRSP
660 000076' INLINE <JMP 1$>
(2) 000076' 000167 001014 JMP 1$
661
662 000102' ENDIF
(4) 000102' 50002$:
663
664 ;+
665 ; INIT R2 TO 0. WILL BE MODULE HEADER ADDRESS LATER IF SPECIFIED IN
666 ; ARGUMENT
667 ;-
668 000102' LET R2 := #0
(4) 000102' 005002 CLR R2
```



```

669
670          ;+
671          ; GET THE FIRST ARGUMENT
672          ; -
673
674          CALL ARGCHK IN <R1> OUT <R1>
(4) 000104' 162705 000002          SUB      #1*2,R5
(3) 000110' 010546          MOV      R5,-(SP)
(4) 000112' 010145          MOV      R1,-(R5)
(3) 000114' 004767 000000G    JSR      PC,ARGCHK
(3) 000120' 012605          MOV      (SP)+,R5
(4) 000122' 012501          MOV      (R5)+,R1
675
676          ;+
677          ; IF IT'S NOT A CR, SEE IF ALPHABETIC CHAR
678          ; -
679          IF.NO.ERROR THEN
(6) 000124' 103532          BCS      50003$
680
681          IFB (R1) LT #'0 ORB (R1) GT #'9 THEN
(6) 000126' 121127 000060          CMPB    (R1),#'0
(8) 000132' 002403          BLT     50004$
(6) 000134' 121127 000071          CMPB    (R1),#'9
(9) 000140' 003436          BLE     50005$
(6) 000142'
682
683          ;+
684          ; IT'S ALPHABETIC - GO VERIFY MODULE NAME....
685          ; -
686
687          CALL NAMCHK IN <R0,R1> OUT <R2,R1>
688          CALL NAMCHK IN <R0,R1> OUT <R2,R1>
(4) 000142' 162705 000004          SUB      #2*2,R5
(3) 000146' 010546          MOV      R5,-(SP)
(5) 000150' 010145          MOV      R1,-(R5)
(4) 000152' 010045          MOV      R0,-(R5)
(3) 000154' 004767 000000G    JSR      PC,NAMCHK
(3) 000160' 012605          MOV      (SP)+,R5
(4) 000162' 012502          MOV      (R5)+,R2
(4) 000164' 012501          MOV      (R5)+,R1
689
690          ;+
691          ; IF MODULE NAME BAD - STUFF ERROR MSG AND RETURN
692          ; -
693          IF.ERROR THEN
(6) 000166' 103005          BCC     50006$
694
695          LET DT.KBRSP(R0) := #CM.BADNAME
696          LET DT.KBRSP(R0) := #CM.BADNAME
(4) 000170' 012760 000000G 000022          MOV      #CM.BADNAME,DT.K
697
698          INLINE <JMP      1$>
(2) 000176' 000167 000714          JMP      1$
699
700          ENDIF
(4) 000202'
(4) 000202'          50006$:
    
```

```

701
702          ;+
703          ; NAME O.K. ... CHECK NEXT ARGUMENT
704          ; -
705
706          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
707
708          ;+
709          ; IF THE ARGUMENT IS A CR.... WHEN A MODULE NAME IS SPECIFIED, A NUMBER
710          ; MUST FOLLOW...BUST COMMAND!!!... STUFF ERROR MSG
711          ; -
712          IF.ERROR THEN
(6) 000222' 103005          BCC      50007$
713
714
715          LET DT.KBRSP(R0) := #CM.ARG
(4) 000224' 012760 000000G 000022    MOV      #CM.ARG,DT.KBRSP
716          INLINE <JMP      1$>
(2) 000232' 000167 000660          JMP      1$
717
718          ENDIF
(4) 000236'          50007$:
719
720          ENDIF
(4) 000236'          50005$:
721
722          ;+
723          ; THERE WAS ANOTHER ARGUMENT... GO CHECK IT.. SHOULD BE A NUMBER...
724          ; -
725
726          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
727
728          ;+
729          ; IF NUMBER WAS TOO LARGE... OR NON OCTAL... BUST COMMAND...
730          ; -
731          IF.ERROR THEN
(6) 000262' 103002          BCC      50010$
732
733
734          INLINE <JMP      1$>
(2) 000264' 000167 000626          JMP      1$
735

```

```

736 000270'                ENDIF
(4) 000270'                50010$:
737
738
739 ;+
740 ; NOW MAKE SURE IT'S AN EVEN NUMBER
741 ; -
742 000270'                LET R3 := R3 ROTATE -1
(7) 000270' 006003                ROR      R3
743
744 000272'                IFCOND CS THEN
(6) 000272' 103005                BCC     50011$
745
746 ;+
747 ; UH OH... AN ODD ADDRESS... BUST COMMAND.... STUFF ERROR AND LEAVE..
748 ; -
749
750 000274'                LET DT.KBRSP(R0) := #CM.ODD
(4) 000274' 012760 000000G 000022    MOV     #CM.ODD,DT.KBRSP
751 000302'                INLINE <JMP      1$>
(2) 000302' 000167 000610                JMP     1$
752
753 000306'                ENDIF
(4) 000306'                50011$:
754
755 ;+
756 ; RESTORE NUMBER
757 ; -
758
759 000306'                LET R3 := R3 ROTATE 1
(7) 000306' 006103                ROL     R3
760
761
762 ;+
763 ; NOW... JUST CHECK FOR ANY JUNK ARGUMENTS....
764 ; -
765
766 000310'                CALL ARGCHK IN <R1> OUT <R1>
(4) 000310' 162705 000002                SUB     #1*2,R5
(3) 000314' 010546                MOV     R5,-(SP)
(4) 000316' 010145                MOV     R1,-(R5)
(3) 000320' 004767 000000G    JSR     PC,ARGCHK
(3) 000324' 012605                MOV     (SP)+,R5
(4) 000326' 012501                MOV     (R5)+,R1
767
768 ;+
769 ; IF JUNK ARGUMENTS FOUND ! SHOULD HAVE BEEN CR ! BUST COMMAND....
770 ; STUFF ERROR MSG & RETURN....
771 ; -
772 000330'                IF.NO.ERROR THEN
(6) 000330' 103405                BCS     50012$
773
774
775 000332'                LET DT.KBRSP(R0) := #CM.ARG
(4) 000332' 012760 000000G 000022    MOV     #CM.ARG,DT.KBRSP
776 000340'                INLINE <JMP      1$>

```

```

(2) 000340' 000167 000552                                JMP      1$
777
778 000344'                                ENDIF
(4) 000344'                                50012$:
779
780
781 ;+
782 ; FORM FINAL ADDRESS - ADD MODULE HEADER ADDRESS(IF ANY) TO NUMBER
783 ; IN R3 AND STORE IN R3
784 ;-
785 000344'                                LET R3 := R3 + R2
(6) 000344' 060203                                ADD      R2,R3
786
787
788 ;+
789 ; SEE IF ADDRESS IS > 32K.... NEVER ALLOWED....
790 ;-
791
792 000346'                                IFCOND CS THEN
(6) 000346' 103005                                BCC      50013$
793
794 ;+
795 ; NUMBER > 32K... SORRY... NOT ALLOWED....
796 ;-
797 000350'                                LET DT.KBRSP(R0) := #CM.ADR
(4) 000350' 012760 000000G 000022                MOV      #CM.ADR,DT.KBRSP
798 000356'                                INLINE <JMP      1$>
(2) 000356' 000167 000534                                JMP      1$
799
800 000362'                                ENDIF
(4) 000362'                                50013$:
801
802
803 ;+
804 ; BUT DOES IT REALLY EXIST....
805 ;-
806
807 000362'                                CALL HRDADR IN <R3>
(3) 000362' 010546                                MOV      R5,-(SP)
(4) 000364' 010345                                MOV      R3,-(R5)
(3) 000366' 004767 000000G                        JSR      PC,HRDADR
(3) 000372' 012605                                MOV      (SP)+,R5
808
809 ;+
810 ; IF NO SUCH MEMORY ADDRESS....STUFF ERROR.... AND SO LONG....
811 ;-
812 000374'                                IF.ERROR THEN
(6) 000374' 103005                                BCC      50014$
813
814
815 000376'                                LET DT.KBRSP(R0) := #CM.ADR
(4) 000376' 012760 000000G 000022                MOV      #CM.ADR,DT.KBRSP
816 000404'                                INLINE <JMP      1$>
(2) 000404' 000167 000506                                JMP      1$
817
818 000410'                                ENDIF
    
```



```

(4) 000452' 000402
(3) 000454'
860
861
862
863 ;+
864 ; SAVE R3 IN KM.LOC
865 ; -
866
866 000454' LET KM.LOC := R3
(4) 000454' 010367 177320 MOV R3, KM.LOC
867
868 000460' ENDIF
(4) 000460' 50017S:
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 ;+
884 ; AND PRINT IT
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
    
```

```

890          ; IS STRUCK, INTERRUPT THE REPEAT-LOOP
891
892          ;-
893          000564'          LET DX.KFL := #0
894          (4) 000564' 005067 000000G          CLR          DX.KFL
895
896          000570'          CALL KBINI IN <R0>
897          (3) 000570' 010546          MOV          R5,-(SP)
898          (4) 000572' 010045          MOV          R0,-(R5)
899          (3) 000574' 004767 000000G          JSR          PC,KBINI
900          (3) 000600' 012605          MOV          (SP)+,R5
901
902          REPEAT
903          (3) 000602'          50020$:
904          000602'          CALL MSGDEQ IN <R0>
905          (3) 000602' 010546          MOV          R5,-(SP)
906          (4) 000604' 010045          MOV          R0,-(R5)
907          (3) 000606' 004767 000000G          JSR          PC,MSGDEQ
908          (3) 000612' 012605          MOV          (SP)+,R5
909
910          UNTILB DX.KFL NE #0
911          (3) 000614' 105767 000000G          TSTB         DX.KFL
912          (6) 000620' 001770          BEQ          50020$
913
914
915          000622'          INLINE <2$:>
916          (2) 000622'          2$:
917
918          ;+
919          ; INIT KM.FLG
920          ;-
921          000622'          LET KM.FLG :B= #0
922          (4) 000622' 105067 177176          CLRB         KM.FLG
923
924          ;+
925          ; IF FIRST CHAR WAS A CR... JUST LEAVE....
926          ;-
927          000626'          LET R1 := DT.KBUF(R0)
928          (4) 000626' 016001 000030          MOV          DT.KBUF(R0),R1
929          000632'          IFB (R1) EQ #CR THEN
930          (6) 000632' 121127 000015          CMPB         (R1),#CR
931          (9) 000636' 001002          SNE          50021$
932          000640'          INLINE <JMP          1$>
933          (2) 000640' 000167 000252          JMP          1$
934          000644'          ENDF
935          (4) 000644'          50021$:
936
937          ;+
938          ; GET ORIGINAL DECODE BUFFER POINTER... COPY KEYBOARD BUFFER INTO
939          ; DECODE BUFFER.....

```

```

927          ; -
928 000644'          LET R1 := BUFPTR(R5)
(4) 000644' 016501 000002          MOV      BUFPTR(R5),R1
929
930
931 000650'          CALL CMDCPY IN <R0,R1> OUT <R4>
(4) 000650' 162705 000002          SUB      #1*2,R5
(3) 000654' 010546          MOV      R5,-(SP)
(5) 000656' 010145          MOV      R1,-(R5)
(4) 000660' 010045          MOV      R0,-(R5)
(3) 000662' 004767 000000G      JSR     PC,CMDCPY
(3) 000666' 012605          MOV      (SP)+,R5
(4) 000670' 012504          MOV      (R5)+,R4
932
933          ; +
934          ; GET FIRST ARGUMENT
935          ; -
936
937
938 000672'          CALL ARGCHK IN <R1> OUT <R1>
(4) 000672' 162705 000002          SUB      #1*2,R5
(3) 000676' 010546          MOV      R5,-(SP)
(4) 000700' 010145          MOV      R1,-(R5)
(3) 000702' 004767 000000G      JSR     PC,ARGCHK
(3) 000706' 012605          MOV      (SP)+,R5
(4) 000710' 012501          MOV      (R5)+,R1
939
940          ; +
941          ; IF THE FIRST CHAR IS NOT A TERMINATOR (CR,LF,^U)... GO CHECK NUMBER...
942          ; -
943 000712'          IF R4 EQ #0 ANDB (R1) NE #LF ANDB (R1) NE #CR THEN
(6) 000712' 005704          TST     R4
(9) 000714' 001054          BNE     50022$
(6) 000716' 121127 000012          CMPB   (R1),#LF
(9) 000722' 001451          BEQ     50022$
(6) 000724' 121127 000015          CMPB   (R1),#CR
(9) 000730' 001446          BEQ     50022$
944
945
946 000732'          CALL NUMCHK IN <R0,R1> OUT <R1,R2>
(4) 000732' 162705 000004          SUB      #2*2,R5
(3) 000736' 010546          MOV      R5,-(SP)
(5) 000740' 010145          MOV      R1,-(R5)
(4) 000742' 010045          MOV      R0,-(R5)
(3) 000744' 004767 000000G      JSR     PC,NUMCHK
(3) 000750' 012605          MOV      (SP)+,R5
(4) 000752' 012501          MOV      (R5)+,R1
(4) 000754' 012502          MOV      (R5)+,R2
947
948
949 000756'          IF.NO.ERROR THEN
(6) 000756' 103430          BCS     50023$
950
951          ; +
952          ; NUMBER O.K. - GET NEXT ARGUMENT
953          ; -
    
```



```

954
955 000760' CALL ARGCHK IN <R1> OUT <R1>
(4) 000760' 162705 000002 SUB #1*2,R5
(3) 000764' 010546 MCV R5,-(SP)
(4) 000766' 010145 MOV R1,-(R5)
(3) 000770' 004767 000000G JSR PC,ARGCHK
(3) 000774' 012605 MOV (SP)+,R5
(4) 000776' 012501 MCV (R5)+,R1
956
957 ;+
958 ; IF NOT A TERMINATOR (CR OR LF) STUFF ERROR MSG
959 ;-
960 001000' IFB (R1) NE #LF ANDB (R1) NE #CR THEN
(6) 001000' 121127 000012 CMPB (R1),#LF
(9) 001004' 001412 BEQ 50024$
(6) 001006' 121127 000015 CMPB (R1),#CR
(9) 001012' 001407 BEQ 50024$
961
962
963 001014' LET DT.KBRSP(R0) := #CM.ARG
(4) 001014' 012760 000000G 000022 MOV #CM.ARG,DT.KBRSP
964 001022' LET KM.ABT :B= #1
(4) 001022' 112767 000001 176773 MOVB #1,KM.ABT
965
966 ELSE
(4) 001030' 000402 BR 50025$
(3) 001032' 50024$:
967
968 ;+
969 ; IT WAS A TERMINATOR... CHANGE CONTENTS OF LOCATION(IN LOCAL STORAGE)
970 ;-
971
972 001032' LET @KM.LOC := R2
(4) 001032' 010277 176742 MOV R2,@KM.LOC
973
974 ENDIF
(4) 001036' 50025$:
(4) 001036'
975
976 ELSE
(4) 001036' 000403 BR 50026$
(3) 001040' 50023$:
977
978 ;+
979 ; BAD NUMBER.... SET ABORT FLAG
980 ;-
981
982 001040' LET KM.ABT :B= #1
(4) 001040' 112767 000001 176755 MOVB #1,KM.ABT
983
984 ENDIF
(4) 001046' 50026$:
(4) 001046'
985
986 ENDIF
(4) 001046' 50022$:
987
988
    
```

```

989          ;+
990          ; SEE IF TERMINATOR IS LF AND ABORT FLAG = 0 (NO ABORT)
991          ; -
992
993          001046'          IFB (R1) EQ #LF ANDB KM.ABT EQ #0 AND R4 EQ #0 THEN          CMPB      (R1),#LF
(6) 001046' 121127 000012          BNE      50027$
(9) 001052' 001007          TSTB     KM.ABT
(6) 001054' 105767 176743          BNE      50027$
(9) 001060' 001004          TST      R4
(6) 001062' 005704          BNE      50027$
(9) 001064' 001002
994
995          001066'          LET R3 := R3 + #2          ADD      #2,R3
(6) 001066' 062703 000002
996
997          001072'          ENDIF          50027$:
(4) 001072'
998          ;+
999          ; UNTIL TERMINATOR = CR AND ABORT FLAG NOT SET (= 1)
1000         ; -
1001
1002
1003
1004         001072'          INLINE <CMPB (R1),#CR>          CMPB (R1),#CR
(2) 001072' 121127 000015          INLINE <BEQ 6$>          BEQ 6$
1005         001076'          BEQ 6$
(2) 001076' 001402          INLINE <JMP 5$>          JMP 5$
1006         001100'          JMP 5$
(2) 001100' 000167 177312          INLINE <6$:>          6$:
1007         001104'          TSTB KM.ABT          TSTB KM.ABT
(2) 001104'          BEQ 7$
1008         001104'          BEQ 7$
(2) 001104' 105767 176713          INLINE <BEQ 7$>          BEQ 7$
1009         001110'          JMP 5$
(2) 001110' 001402          JMP 5$
1010         001112'          JMP 5$
(2) 001112' 000167 177300          INLINE <7$:>          7$:
1011         001116'          7$:
(2) 001116'
1012
1013
1014
1015
1016         001116'          INLINE <1$:>          1$:
(2) 001116'          CALL RESREG
1017         001116'          JSR      PC,RESREG
(3) 001116' 004767 000000G
1018
1019         001122'          ENDRTN          50000$:
(3) 001122'          50001$:
(3) 001122'          RTS      PC
(2) 001122' 000207
1020
1021         000001          .END
    
```

ACSR = 000102	CM.ARG= ***** G	DT.WBU= 000050	KIPDR6= 172314	PASCNT= 000034
ACTBIT= 004000	CM.BAD= ***** G	DT.WHL= 000054	KIPDR7= 172316	PDPLSI= 020000
ADDR22= 001000	CM.ODD= ***** G	DT.WLL= 000052	KMOD 000026RG	PDP60 = 004000
ADR = 000006	CM.RUN= ***** G	DVID1 = 000014	KM.ABT 000023R	PDP70 = 010000
APTFER= 000004	CONFIG= 000056	DX.KFL= ***** G	KM.FLG 000024R	PRI0 = 000000
APTPRE= 000200	CQOVF = 000001	ECCMEM= 000100	KM.LOC 000000R	PRI1 = 000040
ARGCHK= ***** G	CR = 000015	ECCSTA= 000010	KM.MG1 000002R	PRI4 = 000200
ASB = 000106	CSRA = 000100	ENBEOB= 010000	KM.MG2 000012R	PRI5 = 000240
ASSEMB= 000010	CSRC = 000102	ENBNUL= 000001	KTERRD= 000040	PRI6 = 000300
ASTAT = 000104	CTRLC = 000003	ENDLST= 000000	KTPRES= 000400	PRI7 = 000340
AUTO = 000010	CTRLD = 000017	ENQTQ = ***** G	KTSTAT= 000020	PRC = 000000
AUTOST= 020000	CTRLU = 000025	EOPBIT= 000001	KTXTND= 040000	PR4 = 000200
AWAS = 000110	DCEVNT= 000011	ERRTYP= 000106	LF = 000012	PR5 = 000240
BIT0 = 000001	DEFRTN= 000400	EVNTBE= 000200	LPSTAT= 000001	PR6 = 000300
BIT00 = 000001	DIAGMC= 000000	EVNTHD= 000200	MAPSTA= 000200	PR7 = 000340
BIT01 = 000002	DROPMO= 100000	EVNTKT= 000203	MED = 076600	PS = 177776
BIT02 = 000004	DSEVNT= 000014	EVNTPE= 000202	MEMPAS= 040000	PSW = 177776
BIT03 = 000010	DTADR = 000000	EVNTRE= 000201	MODEXH= 004000	RANNUM= 000054
BIT04 = 000020	DT.ADD= 000042	FATERR= 100000	MODHOL= 002000	RBUFEA= 000130
BIT05 = 000040	DT.AP = 000100	HRDADR= ***** G	MODSEL= 001000	RBUFPA= 000126
BIT06 = 000100	DT.APK= 000076	HRDCNT= 000044	MSGCKD= 000010	RSUFSZ= 000132
BIT07 = 000200	DT.BLS= 000034	HRDPAS= 000050	MSGCKS= 000011	RBUFVA= 000124
BIT08 = 000400	DT.CFO= 000014	ICONT = 000036	MSGDEQ= ***** G	RDSEV= 000101
BIT09 = 001000	DT.CF1= 000016	ICOUNT= 000040	MSGDER= 000005	RDWHMI= 000022
BIT1 = 000002	DT.ERR= 000020	IDNUM = 000122	MSGDRP= 000017	RELERR= 000020
BIT10 = 002000	DT.ESI= 000044	IE = 000100	MSGECH= 177777	RELMOD= 020000
BIT11 = 004000	DT.EVN= 000000	INDPAR= 000040	MSGEOP= 000013	RELTIM= 010000
BIT12 = 010000	DT.EXS= 000060	INHDRP= 040000	MSGHDR= 000004	RESREG= ***** G
BIT13 = 020000	DT.FCH= 000037	INHEPR= 020000	MSGHNG= 000022	RES1 = 000056
BIT14 = 040000	DT.FCN= 000036	INHREL= 001000	MSGHRD= 000007	RES2 = 000060
BIT15 = 100000	DT.HMX= 000104	INHRR= 000400	MSGMAP= 000021	RICHAR= 031060
BIT2 = 000004	DT.KBE= 000024	INIT = 000030	MSGNUL= 177775	RPTDAT= 002000
BIT3 = 000010	DT.KBP= 000026	INTR = 000120	MSGPDP= 000002	RSTRT = 000112
BIT4 = 000020	DT.KBR= 000022	IOMOD = 100000	MSGPRM= 177776	RUBOUT= 000177
BIT5 = 000040	DT.KBU= 000030	IOMODP= 102000	MSGRES= 000001	RUNMOD= 100000
BIT6 = 000100	DT.MLS= 000032	IOMODR= 112000	MSGSFT= 000006	RSVALU= 001740
BIT7 = 000200	DT.MTI= 000110	IOMODX= 110000	MSGSKE= 000003	SAM = 075464
BIT8 = 000400	DT.OFF= 000070	JACK = 035060	MSGSMB= 000015	SAVREG= ***** G
BIT9 = 001000	DT.PAS= 000074	KBINI = ***** G	MSGSMH= 000014	SBADR = 000102
BKDEF = 000002	DT.PC = 000002	KIPAR0= 172340	MSGSMS= 000016	SBKMOD= 000000
BKMOD = 000020	DT.PFL= 000062	KIPAR1= 172342	MSGSTD= 000000	SBKSEL= 010000
BKMODE= 040000	DT.PSW= 000004	KIPAR2= 172344	MSGSYS= 000012	SC.ADR= 000006
BKSLSH= 000134	DT.PTA= 000064	KIPAR3= 172346	MSGVEC= 000020	SC.ALC= 000014
BDA16 = ***** G	DT.RCS= 000102	KIPAR4= 172350	NAMCHK= ***** G	SC.APC= 000016
BUFPTR= 000002	DT.REL= 000040	KIPAR5= 172352	NBKM0D= 001000	SC.CKL= 000002
CAPRES= 000004	DT.SCT= 000066	KIPAR6= 172354	NCPUOP= 000020	SC.CKP= 000004
CASTAT= 000004	DT.SMX= 000106	KIPAR7= 172356	NDAPTY= 000002	SC.CLO= 000000
CDERCT= 000146	DT.SP = 000006	KIPDR0= 172300	NULL = 000000	SC.HLD= 000010
CDWDCT= 000144	DT.SSI= 000046	KIPDR1= 172302	NUMCHK= ***** G	SC.SCA= 000012
CKTIM = 100000	DT.STO= 000010	KIPDR2= 172304	OWEN = 024020	SENDLS= 177777
CLKPRE= 000001	DT.ST1= 000012	KIPDR3= 172306	PAERR = 000010	SQFCNT= 000042
CMDCPY= ***** G	DT.SWR= 000056	KIPDR4= 172310	PARPRE= 002000	SQFPAS= 000046
CM.ADR= ***** G	DT.SYP= 000072	KIPDR5= 172312	PARSTA= 000100	SPACE = 000040

SPOINT= 000032	UIPAR2= 177644	WTWHMI= 000222	\$FSTHE= 000330	\$TSK2 = 050025
SPVALU= 002200	UIPAR3= 177646	XFLAG = 000005	\$FSTRU= 000404	\$\$ARGC= 000004
SR0 = 177572	UIPAR4= 177650	XOFF = 000023	\$FSUNT= 000130	\$\$BYTE= 000403
SR1 = 177574	UIPAR5= 177652	XCN = 000021	\$FSWHI= 000120	\$\$CASE= 000000
SR2 = 177576	UIPAR6= 177654	\$BGNLE= 177777	\$FSYES= 000402	\$\$DST = 000000
SR3 = 172516	UIPAR7= 177656	\$ERFLG= 000400	\$IFLEV= 177777	\$\$ELDC= 000402
STAT = 000026	UIPDR0= 177600	\$FSAND= 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 = 001064R
SVR0 = 000062	UIPDR4= 177610	\$F\$DEC= 000220	\$LSTIN= 000001	\$\$LOCN= 000000
SVR1 = 000064	UIPDR5= 177612	\$F\$DD = 000340	\$LSTTA= 000001	\$\$REG = 177777
SVR2 = 000066	UIPDR6= 177614	\$F\$FAL= 000405	\$NESTL= 177777	\$\$RETU= 000000
SVR3 = 000070	UIPDR7= 177616	\$F\$GOD= 000400	\$NSK0 = 000300	\$\$RTN1= 050000
SVR4 = 000072	WASADR= 000104	\$F\$IF = 000110	\$NSK1 = 000110	\$\$RTN2= 050001
SVR5 = 000074	WESTAT= 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	\$\$AVLE= 177777	\$\$TGS1= 000000
YSERR= 000100	WBUFQR= 000140	\$F\$NO = 000403	\$TAGLE= 177777	\$\$TGS2= 000000
TMPID = 000002	WBUFSZ= 000142	\$F\$DR = 000320	\$TAGNU= 050030	\$\$TO = 000000
TQOVF = 000002	WDFR = 000116	\$F\$RTI= 000350	\$TEMP = 000300	\$\$STAG= 050000
UIPAR0= 177640	WDTO = 000114	\$F\$RTN= 000300	\$TSK0 = 050027	. = 001124R
UIPAR1= 177642	WTINRE= 000352	\$F\$SEL= 000140	\$TSK1 = 050026	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:01  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0614

3 COMMON EQUATE MODULE  
526 KMON PROCESS MON KEYBOARD COMMAND  
578 KMOFF PROCESS MOFF KEYBOARD COMMAND  
631 COMMON DEFINITIONS AND REFERENCES FOR KMON  
634 000000' .PRINT ;SPMAC: VERSION 1.1  
664 KMON ROUTINE

```
508 .TITLE KMONOF PROCESS MON, MOFF KEYBOARD COMMANDS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; KMONOF
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
517 ; 1. KMON - TURN THE UNIBUS MAP HARDWARE ON
518 ; 2. KMOFF - TURN THE UNIBUS MAP HARDWARE OFF
519 ;
520 ; VERSION:
521 ; 0.0
522 ;
523 ; EDIT DATE BY REASON
524 ;---
```

```
526 .SBTTL KMON PROCESS MON KEYBOARD COMMAND
527 .IDENT /V0.0/
528
529 ;++
530 ; MODULE NAME:
531 ; KMON
532 ;
533 ; FUNCTIONAL DESCRIPTION:
534 ; THIS ROUTINE PROCESSES THE "MON" KEYBOARD COMMAND.
535 ; IT WILL TEST TO SEE IF 22 BIT ADDRESSING IS AVAILABLE, AND THAT THE
536 ; EXERCISER IS NOT IN RUN MODE. IT WILL
537 ; ENABLE THE UNIBUS MAP HARDWARE.
538 ; IF IN RUN MODE OR NO 22 BIT ADDRESSING AVAILABLE, THE APPROPRIATE
539 ; ERROR MESSAGE IS OUTPUTTED.
540 ;
541 ; INPUTS:
542 ; 1. ADDRESS OF DATA TABLE
543 ; 2. COMMAND DECODE BUFFER POINTER
544 ;
545 ; IMPLICIT INPUTS:
546 ; DT.STO
547 ; DT.CFO
548 ;
549 ; OUTPUTS:
550 ; NONE
551 ;
552 ; IMPLICIT OUTPUTS:
553 ; DT.STO
554 ; DT.KBRSP
555 ;
556 ; PATHOLOGICAL CONNECTIONS:
557 ; CM.ARG,CM.RUN
558 ;
559 ; SUBORDINATE ROUTINES CALLED:
560 ; ARGCHK
561 ;
562 ; FUNCTIONAL SIDE EFFECTS:
563 ; NONE
564 ;
565 ; CALLING SEQUENCE:
566 ; CALL KMON IN <DTADR,BUFPTR>
567 ; WHERE DTADR = ADDRESS OF DATA TABLE
568 ; BUFPTR = COMMAND DECODE BUFFER POINTER
569 ;
570 ; VERSION:
571 ; 0.0
572 ;
573 ; EDIT DATE BY REASON
574 ;--
575
```

```
578 .SBTTL KMOFF PROCESS MOFF KEYBOARD COMMAND
579 .IDENT /V0.0/
580
581 ;++
582 ; MODULE NAME:
583 ; KMOFF
584 ;
585 ; FUNCTIONAL DESCRIPTION:
586 ; THIS ROUTINE WILL PROCESS THE "MOFF" KEYBOARD COMMAND. IT
587 ; WILL TEST FOR THE PRESENCE OF 22 BIT ADDRESSING. IF NO 22 BIT ADDRESSING,
588 ; IT WILL STUFF A NO UNIBUS MAP MESSAGE. IF UNIBUS MAP IS AVAILABLE, A CHECK
589 ; IS MADE TO SEE IF THE EXERCISER IS IN RUN MODE. IF IN RUN
590 ; MODE, THE ERROR MESSAGE IS STUFFED. IF NOT IN RUN MODE,
591 ; THE STATUS WORD WILL BE UPDATED(THE UNIBUS MAP BIT CLEARED)
592 ; AND THE UNIBUS MAP TURNED OFF.
593 ; LOCATION.
594 ;
595 ;
596 ; INPUTS:
597 ; ADDRESS OF DATA TABLE
598 ; COMMAND DECODE BUFFER POINTER
599 ;
600 ; IMPLICIT INPUTS:
601 ; DT.CFO
602 ; DT.STO
603 ;
604 ; OUTPUTS:
605 ; NONE
606 ;
607 ; IMPLICIT OUTPUTS:
608 ; DT.STO
609 ; DT.KBRSP
610 ;
611 ; PATHOLOGICAL CONNECTIONS:
612 ; CM.ARG,CM.RUN
613 ;
614 ; SUBORDINATE ROUTINES CALLED:
615 ; ARGCHK
616 ;
617 ; FUNCTIONAL SIDE EFFECTS:
618 ; NONE
619 ;
620 ; CALLING SEQUENCE:
621 ; CALL KMOFF IN <DTADR,CMDBUF>
622 ; WHERE DTADR = ADDRESS OF DATA TABLE
623 ; CMDBUF = COMMAND DECODE BUFFER POINTER
624 ;
625 ; VERSION:
626 ; 0.0
627 ;
628 ; EDIT DATE BY REASON
629 ;--
```



```

631          .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KMON
632
633          .MCALL  STRUCT
634 000000'   STRUCT
(1) 000000'   .PRINT  ;SPMAC: VERSION 1.1
635          000001   $LSTIN=1
636          000001   $LSTTAG=1
637
638
639          ;*****
640          ;
641          ; REFERENCED BY OTHER MODULES:
642          ;
643          .GLOBL  KMOFF          ;MODULE ENTRY POINT
644          .GLOBL  KMON          ;MODULE ENTRY POINT
645          ;
646          ;*****
647          ;
648          ; GLOBAL REFERENCES:
649          ;
650          .GLOBL  ARGCHK          ;CHECK FOR EXISTENCE OF AN ARGUMENT
651          .GLOBL  CM.ARG          ;'ILLEGAL OR INVALID ARG' MESSAGE
652          .GLOBL  CM.RUN          ;"NOT A VALID COMMAND IN RUN MODE" MESSAGE
653          ;*****
654          ;
655          ; LOCAL STORAGE:
656          ;
657 000000' 047516 052440 044516 MB.NOMB:      .ASCIZ  /NO UNIBUS MAP%/
000006' 052502 020123 040515
000014' 022520      000
658 000017'      125 044516 052502 MB.ON:      .ASCIZ  /UNIBUS MAP ON%/
000024' 020123 040515 020120
000032' 047117 000045
659 000036' 047125 041111 051525 MB.OFF:     .ASCIZ  /UNIBUS MAP OFF%/
000044' 046440 050101 047440
000052' 043106 000045
660 000056'      000          MB.FLG:     .BYTE   0          ;UNIBUS MAP ON/OFF FLAG
661          000060'          .EVEN
662

```

```

664          .SBTTL KMON  ROUTINE
665
666 000060'  ROUTINE KMON <DTADR,BUFPTR>
667          (2) 000060'
668          ;+
669          ; SET ON FLAG
670          ; -
671
672 000060'  LET MB.FLG :B= #0
673 (4) 000060' 105067 177772          CLRB  MB.FLG
674 000064'  INLINE <BR MBROU>          BR MBROU
675 (2) 000064' 000404
676 000066'  ENDRTN
677 (3) 000066'
678 (3) 000066'
679 (2) 000066' 000207          50000$:
680          50001$:
681          RTS  PC
682
683 000070'  ROUTINE KMOFF <DTADR,BUFPTR>
684 (2) 000070'
685          ;+
686          ; SET THE OFF FLAG
687          ; -
688
689 000070'  LET MB.FLG :B= #1
690 (4) 000070' 112767 000001 177760          MOVB  #1,MB.FLG
691
692          ;+
693          ; SAVE REGISTERS AND DATA TABLE ADDRESS
694          ; -
695
696 000076'  INLINE <MBROU:>
697 (2) 000076'
698 000076'  PUSH R0,R1          MBRU:
699 (2) 000076' 010046          MOV  R0,-(SP)
700 (3) 000100' 010146          MOV  R1,-(SP)
701 000102'  LET R0 := DTADR(R5)          MOV  DTADR(R5),R0
702 (4) 000102' 016500 000000          LET R1 := BUFPTR(R5)
703 000106'  LET R1 := BUFPTR(R5)          MOV  BUFPTR(R5),R1
704 (4) 000106' 016501 000002
705
706          ;+
707          ; IF NOT IN RUN MODE CONTINUE
708          ; -
709
710 000112'  IF #RUNMODE NOTSETIN DT.STO(R0) THEN
711 (6) 000112' 032760 100000 000010          BIT  #RUNMODE,DT.STO(
712 (9) 000120' 001054          BNE  50002$
713
714          ;+
715          ; SEE IF ANY ARGUMENTS EXIST & IF
716          ; THEY DO , THIS IS A BAD COMMAND
717          ; -
718
719 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.STO OTHERWISE CLEAR IT.
710          ;+
711          ;-          IF.ERROR THEN
(6) 000142' 103037          BCC      50003$
712
713
714          IF #ADDR22 SETIN DT.CF0(R0) THEN
715 000144'          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          ;-          IFB MB.FLG EQ #0 THEN
720 000154'          TSTB     MB.FLG
(6) 000154' 105767 177676          BNE     50005$
(9) 000160' 001012          LET DT.STO(R0) := DT.STO(R0) SET.BY #MAPSTAT
721 000162'          BIS     #MAPSTAT,DT.STO(
(6) 000162' 052760 000200 000010  LET @#SR3 := @#SR3 SET.BY #BIT05
722 000170'          BIS     #BIT05,@#SR3
(6) 000170' 052737 000040 172516
723          ;+
724          ; LOAD UNIBUS MAP ON MESSAGE
725          ;+
726          ;-          LET DT.KBRSP(R0) := #MB.ON
727 000176'          MOV      #MB.ON,DT.KBRSP(
(4) 000176' 012760 000017' 000022  ELSE
728 000204'          BR      50006$
(4) 000204' 000411
(3) 000206'          50005$:
729
730          ;+
731          ; TURN IT OFF
732          ;+
733          ;-          LET DT.STO(R0) := DT.STO(R0) CLR.BY #MAPSTAT
734 000206'          BIC     #MAPSTAT,DT.STO(
(6) 000206' 042760 000200 000010  LET @#SR3 := @#SR3 CLR.BY #BIT05
735 000214'          BIC     #BIT05,@#SR3
(6) 000214' 042737 000040 172516
736          ;+
737          ; LOAD THE UNIBUS MAP OFF MESSAGE
738          ;+
739          ;-          LET DT.KBRSP(R0) := #MB.OFF
740 000222'          MOV      #MB.OFF,DT.KBRSP
(4) 000222' 012760 000036' 000022
741 000230'          ENDIF

```

```
(4) 000230'
742 000230' ELSE
(4) 000230' 000403
(3) 000232'
743 ;+
744 ; LOAD THE NO UNIBUS MAP MESSAGE
745 ; -
746
747 000232' LET DT.KBRSP(RO) := #MB.NOMB
(4) 000232' 012760 000000' 000022 MOV #MB.NOMB,DT.KBRSP
748 000240' ENDIF
(4) 000240'
749
750 000240' ELSE
(4) 000240' 000403
(3) 000242'
751 ;+
752 ; LOAD THE ILLEGAL ARGUMENT MESSAGE
753 ; -
754
755 000242' LET DT.KBRSP(RO) := #CM.ARG
(4) 000242' 012760 000000G 000022 MOV #CM.ARG,DT.KBRSP
756 000250' ENDIF
(4) 000250'
757 000250' ELSE
(4) 000250' 000403
(3) 000252'
758
759 ;+
760 ; LOAD THE RUN MODE MESSAGE
761 ; -
762
763 000252' LET DT.KBRSP(RO) := #CM.RUN
(4) 000252' 012760 000000G 000022 MOV #CM.RUN,DT.KBRSP
764 000260' ENDIF
(4) 000260'
765
766 ;+
767 ; CLEAN UP
768 ; -
769
770 000260' POP R1,RO
(2) 000260' 012601 MOV (SP)+,R1
(3) 000262' 012600 MOV (SP)+,RO
771
772 000264' ENDRTN
(3) 000264'
(3) 000264'
(2) 000264' 000207
773
774 000001 .END
```

```
50006$:
BR 50007$
50004$:
MOV #MB.NOMB,DT.KBRSP
50007$:
BR 50010$
50003$:
MOV #CM.ARG,DT.KBRSP
50010$:
BR 50011$
50002$:
MOV #CM.RUN,DT.KBRSP
50011$:
MOV (SP)+,R1
MOV (SP)+,RO
50000$:
50001$:
RTS PC
```

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

SVR3 = 000070	UIPDR5= 177612	\$FSCAS= 000150	\$ISK1 = 000001	\$\$BYTE= 000402
SVR4 = 000072	UIPDR6= 177614	\$F\$DEC= 000220	\$ISK2 = 000001	\$\$CASE= 000000
SVR5 = 000074	UIPDR7= 177616	\$F\$DD = 000340	\$ISK3 = 000001	\$\$DST = 000000
SVR6 = 000076	WASADR= 000104	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$ELOC= 000402
SYSCNT= 000052	WBSTAT= 000040	\$F\$GDD= 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\$LDD= 000200	\$NSK0 = 000300	\$\$LOC = 000160R
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$LOCN= 000000
UIPAR1= 177642	WDFR = 000116	\$F\$ND = 000403	\$NSK2 = 000110	\$\$REG = 177777
UIPAR2= 177644	WDT0 = 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	\$\$STAG= 050000
UIPDR3= 177606	\$F\$BAD= 000401	\$IFLEV= 177777	\$TSK3 = 050006	. = 000266R
UIPDR4= 177610	\$F\$BLA= 000170	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:01

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0624

3 COMMON EQUATE MODULE  
586 COMMON DEFINITIONS AND REFERENCES FOR KRUN  
589 000000' .PRINT ;SPMAC: VERSION 1.1  
644 KRUN ROUTINE

```
508 .TITLE KRUN PROCESS THE 'RUN' COMMAND & 'RUNL' COMMAND
509 .IDENT /VO.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KRUN
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'RUN' AND THE 'RUNL' KEYBOARD
517 ; COMMANDS. IF AN ADDRESS IS INCLUDED, IT WILL BE VERIFIED.
518 ; ALL OPTION MODULES HEADERS WILL BE INITIALIZED. THE SYS-
519 ; TEM ERROR COUNT, POWER FAIL COUNT, AND RELOCATION FLAG
520 ; WILL BE INITIALIZED. THE KT WILL BE MAPPED TO THE NEW
521 ; LOCATION (IF ANY SPECIFIED). THE WRITE BUFFER LIMITS
522 ; WILL BE SET UP.
523 ;
524 ; INPUTS:
525 ; 1. DTABLE ADDRESS
526 ; 2. DECODE BUFFER POINTER
527 ;
528 ; IMPLICIT INPUTS:
529 ; 1. DT.MLST
530 ; 2. DT.STO
531 ; 3. DT.SSIZ
532 ; 4. DT.CFO
533 ; 5. DT.ADDR
534 ;
535 ; OUTPUTS:
536 ; NONE
537 ;
538 ; IMPLICIT OUTPUTS:
539 ; 1. DT.STO
540 ; 2. DT.KBRSP
541 ; 3. DT.ESIZ
542 ; 4. DT.SYP
543 ; 5. DT.EXS
544 ; 6. DT.PFL
545 ;
546 ; PATHOLOGICAL CONNECTIONS:
547 ; 1. BE.MCNT - NUMBER OF MONITOR SYS ERRORS
548 ; 2. PR.RL1 - RELOCATION FLAG
549 ; 3. CM.ADDR
550 ; 4. MD.COD
551 ; 5. CM.RUN
552 ; 6. CM.NBAD
553 ; 7. CM.NUM
554 ; 8. CM.ARG
555 ; 9. PA.CNT
556 ;
557 ; SUBORDINATE ROUTINES CALLED:
558 ; 1. ARGCHK
559 ; 2. CLRTIM
560 ; 3. SAVREG
561 ; 4. RESREG
562 ; 5. WBFLIM
563 ; 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
```

```

586          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KRUN
587
588          .MCALL STRUCT
589 000000'   STRUCT
(1) 000000'   .PRINT ;SPMAC: VERSION 1.1
590
591          000001 $LSTTAG=1
592          000001 $LSTIN=1
593
594          ;*****
595          ;
596          ; REFERENCED BY OTHER MODULES:
597          ;
598          .GLOBL KRUN          ;MODULE ENTRY POINT
599          .GLOBL KRUNL        ;MODULE ENTRY POINT
600          ;
601          ;*****
602          ;
603          ; GLOBAL REFERENCES:
604          ;
605          .GLOBL CLKCHK        ;SEE IF A CLOCK IS HERE
606          .GLOBL APTSEL        ;APT SELECT ROUTINE
607          .GLOBL MD.COD        ;CODE
608          .GLOBL CM.ADR        ;'ILLEGAL ADDRESS'
609          .GLOBL PRRLOC        ;RELOCATE THE CODE
610          .GLOBL MSGDEQ        ;MESSAGE DEQUE
611          .GLOBL SETVEC        ;SET UP UNUSED VECTORS
612          .GLOBL UNIMAP        ;UNIBUS MAPPING
613          .GLOBL CM.RUN        ;'ILLEGAL COMMAND IN RUN MODE'
614          .GLOBL CM.NSAD       ;'NOT AN OCTAL #'
615          .GLOBL CM.NUM        ;'NUMBER TOO LARGE'
616          .GLOBL CM.ARG        ;'ILLEGAL OR MISSING ARGUMENT'
617          .GLOBL CLRTIM        ;CLEAR TIME TABLE
618          .GLOBL ARGCHK        ;CHECK AN ARGUMENT
619          .GLOBL SAVREG        ;SAVE REGISTERS
620          .GLOBL RESREG        ;RESTORE REGISTERS
621          .GLOBL PR.RL1        ;RELOCATION FLAG
622          .GLOBL PA.CNT        ;NUMBER OF PARITY ERRORS
623          .GLOBL BE.MCNT       ;MONITOR SYSTEM ERROR COUNT
624          .GLOBL WBFLIM        ;ESTABLISH WRITE BUFFER LIMITS
625          .GLOBL WSTBUS        ;WRITE WORST CASE UNIBUS PATTERN
626          ;
627          ;*****
628          ;
629          ; LOCAL STORAGE:
630          ;
631 000000' 000000 KR.FLG: .WORD 0 ;'1' IF CMD IS RUNL - '0' IF CMD IS RUN
632 000002' 000000 KR.MOV: .WORD 0 ;'1' IF MOVING CODE - '0' IF NOT
633 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 D.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
```

```
644 .SBTTL KRUN ROUTINE
645
646 000222' ROUTINE KRUN <DTADR,BUFPTR>
(2) 000222' KRUN:
647
648 ;+
649 ; INDICATE 'RUN' COMMAND
650 ; -
651
652 000222' LET KR.FLG := #0
(4) 000222' 005067 177552 CLR KR.FLG
653 000226' INLINE <BR RUNSUB> BR RUNSUB
(2) 000226' 000404
654 000230' ENDRTN 50000S:
(3) 000230' 50001S:
(3) 000230' (2) 000230' 000207 RTS PC
655
656 000232' ROUTINE KRUNL <DTADR,BUFPTR>
(2) 000232' KRUNL:
657
658 ;+
659 ; INDICATE 'RUNL' COMMAND
660 ; -
661
662 000232' LET KR.FLG := #1
(4) 000232' 012767 000001 177540 MOV #1,KR.FLG
663
664
```

```

666 000240'          INLINE <RUNSUB:>
(2) 000240'
667
668
669          ;+
670          ;INIT FLAG,SAVE REGISTERS,DTABLE AND BUF PTR
671          ;+
672
673 000240'          LET KR.MOV := #0
(4) 000240' 005067 177536          CLR      KR.MOV
674 000244'          CALL SAVREG
(3) 000244' 004767 000000G          JSR      PC,SAVREG
675 000250'          LET R0 := DTADR(R5)
(4) 000250' 016500 000000          MOV      DTADR(R5),R0
676 000254'          LET R1 := BUFPTR(R5)
(4) 000254' 016501 000002          MOV      BUFPTR(R5),R1
677          ;+
678          ; IF IN AUTO OR APTPRES MODE, DON'T CHECK ANY ARGUMENTS - JUST GO TO WORK
679          ;+
680
681 000260'          IF #AUTO SETIN DT.CF0(R0) OR #APTPRES SETIN DT.CF0(R0) THEN
(6) 000260' 032760 000010 000014          BIT      #AUTO,DT.CF0(R0)
(8) 000266' 001004          BNE      50002$
(6) 000270' 032760 000200 000014          BIT      #APTPRES,DT.CF0(
(9) 000276' 001402          BEQ      50003$
(6) 000300'          50002$:
682 000300'          INLINE <JMP      339$>
(2) 000300' 000167 000504          JMP      339$
683 000304'          ENDIF
(4) 000304'          50003$:
684
685
686          ;+
687          ; SEE IF IN RUNMODE - IF SO STUFF ERROR AND SCRAM
688          ;+
689
690 000304'          IF #RUNMODE SETIN DT.ST0(R0) THEN
(6) 000304' 032760 100000 000010          BIT      #RUNMODE,DT.ST0(
(9) 000312' 001405          BEQ      50004$
691 000314'          LET DT.KBRSP(R0) := #CM.RUN
(4) 000314' 012760 000000G 000022          MOV      #CM.RUN,DT.KBRSP
692 000322'          INLINE <JMP      1$>
(2) 000322' 000167 001166          JMP      1$
693
694 000326'          ENDIF
(4) 000326'          50004$:
695
696
697          ;+
698          ; WE ARE NOT IN RUNMODE....ALL IS OK, SAVE DT.ADDR.....GET AN ARGUMENT IF ANY.....
699          ;+
700
701
702          ;+
703          ; SAVE DT.ADDR
704          ;+
    
```

```

705
706 000326' LET R3 := DT.ADDR(R0)
(4) 000326' 016003 000042 MOV DT.ADDR(R0),R3
707
708 000332' CALL ARGCHK IN <R1> OUT <R1>
(4) 000332' 162705 000002 SUB #1*2,R5
(3) 000336' 010546 MOV R5,-(SP)
(4) 000340' 010145 MOV R1,-(R5)
(3) 000342' 004767 000000G JSR PC,ARGCHK
(3) 000346' 012605 MOV (SP)+,R5
(4) 000350' 012501 MOV (R5)+,R1
709 ;+
710 ; IF NO ARGS, JUMP AHEAD
711 ; -
712
713
714 000352' IF.ERROR THEN
(6) 000352' 103002 BCC 50005$
715
716 000354' INLINE <JMP 339$>
(2) 000354' 000167 000430 JMP 339$
717
718 000360' ENDIF
(4) 000360' 50005$:
719 ;+
720 ; NOT A CARRIAGE RETURN..SET FLAG.. GET BY ANY LEADING 0'S AND VERIFY NUMBER....
721 ; -
722
723
724
725 000360' LET KR.MOV := #1
(4) 000360' 012767 000001 177414 MOV #1,KR.MOV
726
727 000366' WHILEB (R1) EQ #'0 DO
(4) 000366' 50006$:
(6) 000366' 121127 000060 CMPB (R1),#'0
(9) 000372' 001002 BNE 50007$
728 000374' LET R1 := R1 + #1
(6) 000374' 005201 INC R1
729 000376' ENDDO
(4) 000376' 000773 BR 50006$
(3) 000400' 50007$:
730
731
732 ;+
733 ; INIT C-BIT AND REGISTERS
734 ; -
735
736 000400' LET CARRY := 0
(4) 000400' 000241 CLC
737 000402' LET R4 := #0
(4) 000402' 005004 CLR R4
738 000404' LET R3 := #0
(4) 000404' 005003 CLR R3
739 000406' LET R2 := #0
(4) 000406' 005002 CLR R2

```

```
740
741
742 ;+
743 ; NUMBER VERIFICATION CONSISTS OF CHECKING UNTIL SOMETHING OTHER
744 ; THAN A '0' THRU '7' IS DETECTED IN THE DECODE BUFFER.....THE OCTAL
745 ; ASCII CHARACTERS WILL BE DE-ASCIIED, SHIFTED INTO R3 AND ANY
746 ; OVERFLOW SHIFTED INTO R4....A COUNT OF THE NUMBER OF CHARACTERS IS
747 ; MAINTAINED.....
748 ;-
749
750 000410'          WHILEB (R1) GE #'0 ANDB (R1) LE #'7 DO
751 (4) 000410'
752 (6) 000410' 121127 000060
753 (9) 000414' 002420
754 (6) 000416' 121127 000067
755 (9) 000422' 003015
756
757
758 ;+
759 ; GET A BYTE
760 ;-
761 000424'          LET R0 :B= (R1)
762 (4) 000424' 111100
763
764 ;+
765 ; DE-ASCII IT
766 ;-
767 000426'          LET R0 := R0 - #'0
768 (6) 000426' 162700 000060
769
770
771 ;+
772 ; SHIFT OLD STUFF(LO ORDER) TO RIGHT
773 ;-
774 000432'          LET R3 := R3 SHIFT 1
775 (7) 000432' 006303
776
777 ;+
778 ; ROTATE HIGH ORDER
779 ;-
780 000434'          LET R4 := R4 ROTATE 1
781 (7) 000434' 006104
782 000436'          LET R3 := R3 SHIFT 1
783 (7) 000436' 006303
784 000440'          LET R4 := R4 ROTATE 1
785 (7) 000440' 006104
786 000442'          LET R3 := R3 SHIFT 1
787 (7) 000442' 006303
788 000444'          LET R4 := R4 ROTATE 1
789 (7) 000444' 006104
790
791 ;+
792 ; ADD THE NEW CHAR
793 ;-
794 000446'          LET R3 := R3 + R0
795 (6) 000446' 060003
796
```

50010\$:	CMPB	(R1),#'0
	BLT	50011\$
	CMPB	(R1),#'7
	BGT	50011\$
	MOVB	(R1),R0
	SUB	#'0,R0
	ASL	R3
	ROL	R4
	ASL	R3
	ROL	R4
	ASL	R3
	ROL	R4
	ADD	R0,R3

```
782
783
784 ;+
785 ; INCREMENT CHAR COUNTER
786 ;-
786 000450' LET R2 := R2 + #1
(6) 000450' 005202 INC R2
787
788 ;+
789 ; GET NEXT BYTE
790 ;-
791
792 000452' LET R1 := R1 + #1
(6) 000452' 005201 INC R1
793
794 000454' ENDDO
(4) 000454' 000755 BR 50010$
(3) 000456' 50011$:
795
796
797 ;+
798 ; RESTORE DTABLE TO R0
799 ;-
800
801 000456' LET R0 := DTADR(R5)
(4) 000456' 016500 000000 MOV DTADR(R5),R0
802
803 ;+
804 ; R1 IS NOW POINTING TO FIRST NON-OCTAL-ASCII-CHAR....IT MUST BE
805 ; CR OR SPACE...IF NOT - STUFF ERROR MSG AND SCRAM....
806 ;-
807
808 000462' IFB (R1) NE #CR ANDB (R1) NE #SPACE THEN
(6) 000462' 121127 000015 CMPB (R1),#CR
(9) 000466' 001410 BEQ 50012$
(6) 000470' 121127 000040 CMPB (R1),#SPACE
(9) 000474' 001405 BEQ 50012$
809
810 000476' LET DT.KBRSP(R0) := #CM.NBAD
(4) 000476' 012760 000000G 000022 MOV #CM.NBAD,DT.KBRSP
811 000504' INLINE <JMP 1$>
(2) 000504' 000167 001004 JMP 1$
812 000510' ENDDIF
(4) 000510' 50012$:
813
814 ;+
815 ; IF THE ADDRESS GIVEN WAS MORE THAN 8 CHARS...(NOT COUNTING LEADING 0'S)..STUFF
816 ; ERROR MSG AND RETURN...
817 ;-
818
819 000510' IF R2 GT #10 THEN
(6) 000510' 020227 000010 CMP R2,#10
(9) 000514' 003405 BLE 50013$
820 000516' LET DT.KBRSP(R0) := #CM.ADR
(4) 000516' 012760 000000G 000022 MOV #CM.ADR,DT.KBRSP
821 000524' INLINE <JMP 1$>
(2) 000524' 000167 000764 JMP 1$
```



```
822 000530'          ENDIF                                50013$:
(4) 000530'
823
824
825                ;+
826                ; CONVERT THE NUMBER TO PAR FORMAT...TO DD, SHIFT THE HIGH AND LOW
827                ; ADDRESS BITS TO THE RIGHT 6 TIMES.....
828                ;-
829
830
831                ;+
832                ; INIT COUNTER
833                ;-
834 000530'          LET R2 := #0
(4) 000530' 005002                                CLR      R2
835
836 000532'          WHILE R2 NE #6 DO
(4) 000532'
(6) 000532' 020227 000006                                50014$:  CMP      R2,#6
(9) 000536' 001404                                BEQ      50015$
837
838 000540'          LET R4 := R4 SHIFT -1
(7) 000540' 006204                                ASR      R4
839 000542'          LET R3 := R3 ROTATE -1
(7) 000542' 006003                                ROR      R3
840 000544'          LET R2 := R2 + #1
(6) 000544' 005202                                INC      R2
841
842 000546'          ENDDO
(4) 000546' 000771                                BR       50014$
(3) 000550'          50015$:
843
844
845                ;+
846                ; R3 IS NOW PAR FORMAT. IF R4, THE HIGH ORDER BITS, IS NOT ZERO,
847                ; THE ADDRESS WAS TOO BIG....STUFF ERROR AND SCRAM...
848                ;-
849
850 000550'          IF R4 NE #0 THEN
(6) 000550' 005704                                TST      R4
(9) 000552' 001405                                BEQ      50016$
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$:
855                ;+
856                ; ALSO BUST COMMAND IF THE LOW ORDER WORD IS LESS THAN 4K
857                ;-
858
859
860 000566'          IF R3 LOS #177 THEN
(6) 000566' 020327 000177                                CMP      R3,#177
(9) 000572' 101005                                BHI      50017$
```

```

861
862 000574'          LET DT.KBRSP(R0) := #KR.SML
(4) 000574' 012760 000004' 000022
863 000602'          MOV          #KR.SML,DT.KBRSP
(2) 000602' 000167 000706          JMP          1$
864
865 000606'          ENDIF
(4) 000606'          50017$:
866
867
868
869 ;+
870 ; NEXT, SEE IF THE EXERCISER CAN POSSIBLY FIT INTO THE NEW ADDRESS SPACE
871 ; AND SEE IF ADDRESS EXISTS AT ALL....GET THE EXERCISER SIZE, CONVERT
872 ; TO PAR FORMAT, IF THE NEW ADDRESS DESIRED IS GREATER THAN THE SYSTEM
873 ; SIZE....STUFF ERROR MSG AND RETURN...IF NOT, SUBTRACT THE NEW ADDRESS
874 ; FROM THE SYSTEM SIZE....SEE IF THE EXERCISER WILL FIT....IF NOT
875 ; STUFF ERROR MSG AND RETURN.....
876 ;-
876 000606'          IF R3 HI DT.SSIZ(R0) THEN
(6) 000606' 020360 000046          CMP          R3,DT.SSIZ(R0)
(9) 000612' 101405          BLOS         50020$
877 000614'          LET DT.KBRSP(R0) := #CM.ADR
(4) 000614' 012760 000000G 000022  MOV          #CM.ADR,DT.KBRSP
878 000622'          INLINE <JMP      1$>
(2) 000622' 000167 000666          JMP          1$
879 000626'          ENDIF
(4) 000626'          50020$:
880
881
882 000626'          LET R2 := DT.ESIZ(R0)
(4) 000626' 016002 000044          MOV          DT.ESIZ(R0),R2
883 000632'          LET R4 := #0
(4) 000632' 005004          CLR          R4
884 000634'          WHILE R4 NE #6 DO
(4) 000634'          50021$:
(6) 000634' 020427 000006          CMP          R4,#6
(9) 000640' 001404          BEQ          50022$
885 000642'          LET CARRY := 0
(4) 000642' 000241          CLC
886 000644'          LET R2 := R2 ROTATE -1
(7) 000644' 006002          ROR          R2
887 000646'          LET R4 := R4 + #1
(6) 000646' 005204          INC          R4
888 000650'          ENDDO
(4) 000650' 000771          BR          50021$
(3) 000652'          50022$:
889
890 ;+
891 ; R2 NOW HAS EXERCISER SIZE IN PAR FORMAT....SEE IF ROOM EXISTS FOR IT...
892 ; FIRST SUBTRACT 4K FROM IT (THE UN MOVABLE PORTION) AND SUBTRACT NEW
893 ; ADDR FROM SYS SIZE....
894 ;-
895
896 000652'          LET R2 := R2 - #177
(6) 000652' 162702 000177          SUB          #177,R2
897 000656'          LET R4 := DT.SSIZ(R0) - R3

```

```

(4) 000656' 016004 000046          MOV    DT.SSIZ(R0),R4
(6) 000662' 160304          SUB    R3,R4
898
899
900          ;+
901          ; IF R4 (THE DIFFERENCE BETWEEN THE TOP OF MEMORY AND RELOCATION
902          ; ADDRESS) IS LESS THAN THE EXERCISER SIZE, WE OBVIOUSLY DON'T HAVE ROOM.
903          ; STUFF ERROR MSG AND RETURN.
904          ;-
905          IF R4 LE R2 THEN
(6) 000664' 020402          CMP    R4,R2
(9) 000666' 003005          BGT    50023$
906          LET DT.KBRSP(R0) := #KR.BIG
(4) 000670' 012760 000061' 000022    MOV    #KR.BIG,DT.KBRSP
907          000676'          INLINE <JUMP    1$>
(2) 000676' 000167 000612          JMP    1$
908          000702'          ENDIF
(4) 000702'          50023$:
909
910          ;+
911          ; AT THIS POINT, THE NUMBER HAS BEEN VERIFIED.... FINALLY CHECK FOR
912          ; JUNK ARGUMENTS
913          ;-
914          CALL ARGCHK IN <R1> OUT <R1>
915          000702'          SUB    #1*2,R5
(4) 000702' 162705 000002          MOV    R5,-(SP)
(3) 000706' 010546          MOV    R1,-(R5)
(4) 000710' 010145          JSR    PC,ARGCHK
(3) 000712' 004767 000000G        MOV    (SP)+,R5
(3) 000716' 012605          MOV    (R5)+,R1
(4) 000720' 012501
916
917          IF.NO.ERROR THEN
(6) 000722' 103405          BCS    50024$
918          000724'          LET DT.KBRSP(R0) := #CM.ARG
(4) 000724' 012760 000000G 000022    MOV    #CM.ARG,DT.KBRSP
919          000732'          INLINE <JUMP    1$>
(2) 000732' 000167 000556          JMP    1$
920          000736'          ENDIF
(4) 000736'          50024$:
921
922          ;+
923          ; JUST CHECK TO MAKE SURE KT IS ENABLED....
924          ;-
925
926          IF #KTSTAT NOTSETIN DT.ST0(R0) THEN
(6) 000736' 032760 000020 000010    BIT    #KTSTAT,DT.ST0(R
(9) 000744' 001005          BNE    50025$
928          000746'          LET DT.KBRSP(R0) := #KR.KT
(4) 000746' 012760 000127' 000022    MOV    #KR.KT,DT.KBRSP(
929          000754'          INLINE <JUMP    1$>
(2) 000754' 000167 000534          JMP    1$
930          000760'          ENDIF
(4) 000760'          50025$:
931

```

```

932          ;+
933          ; IF R3 IS BIGGER THAN OR EQUAL TO 6000 AND MAPPING IS NOT ON, BUST
934          ; COMMAND....THAT IS , YOU CANNOT RUN ABOVE 96K (ASSUMES AN EXERCISER SIZE
935          ; OF 28K) BECAUSE YOU MAY OVERLAP SOME OF THE EXERCISER ABOVE
936          ; 124K AND WITH MAP BOX OFF, YOU WILL DIE.....
937          ; -
938
939          IF R3 HIS #6000 AND #MAPSTAT NOTSETIN DT.STO(R0) THEN
(6) 000760' 020327 006000          CMP      R3,#6000
(9) 000764' 103411          BLO     50026$
(6) 000766' 032760 000200 000010          BIT     #MAPSTAT,DT.STO(
(9) 000774' 001005          BNE     50026$
940          LET DT.KBRSP(R0) := #KR.MAP
(4) 000776' 012760 000175' 000022          MOV     #KR.MAP,DT.KBRSP
941          INLINE <JMP      1$>
(2) 001004' 000167 000504          JMP     1$
942          ENDIF
(4) 001010'          50026$:
943
944
945
946          001010'          339$:
947          ;+
948          ; IF UNDER APT, USE APT'S TABLE TO SELECT THE MODULES
949          ; AND SET UP DEVICE COUNT
950          ; -
951
952          IF #APTPRES SETIN DT.CFO(R0) THEN
(6) 001010' 032760 000200 000014          BIT     #APTPRES,DT.CFO(
(9) 001016' 001405          BEQ     50027$
953          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
954          ENDIF
(4) 001032'          50027$:
955
956          LET R1 := DT.MLST(R0) ;GET ADDRESS OF MODULE LIST
(4) 001032' 016001 000032          MOV     DT.MLST(R0),R1
957
958          LET R4 := #0
(4) 001036' 005004          CLR     R4
959          ;+
960          ; BEFORE HONORING COMMAND, MAKE SURE AT LEAST ONE MODULE IS SELECTED
961          ; -
962
963          WHILE (R1) NE #ENDLST DO
(4) 001040'          50030$:
(6) 001040' 021127 000000          CMP     (R1),#ENDLST
(9) 001044' 001410          BEQ     50031$
964
965          ;+
966          ; GET A MODULE ADDRESS
967          ; -
968          LET R2 := (R1)+
  
```

```

(4) 001046' 012102
969 001050'
(6) 001050' 032762 040000 000026
(9) 001056' 001402
970 001060'
(4) 001060' 012704 000001
971 001064'
(4) 001064'
972 001064'
(4) 001064' 000765
(3) 001066'
973
974
975
976
977
978
979
980 001066'
(6) 001066' 005704
(9) 001070' 001005
981 001072'
(4) 001072' 012760 000150' 000022
982 001100'
(2) 001100' 000167 000410
983 001104'
(4) 001104'
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000 001104'
(3) 001104'
1001 001104'
(3) 001104' 010546
(4) 001106' 010045
(3) 001110' 004767 000000G
(3) 001114' 012605
1002 001116'
(3) 001116' 026727 000000G 177775
(6) 001124' 001367
1003
1004
1005

```

```

IF #BIT14 SET IN STAT(R2) THEN
    LET R4 := #1
ENDIF
ENDDDO

```

```

;+
; IF R4 IS ZERO...NO MODULES SELECTED.....STUFF ERROR AND GOODBYE...
;-
IF R4 EQ #0 THEN
    LET DT.KBRSP(R0) := #KR.MOD
    INLINE <JMP 1$>
ENDIF

```

```

;+
; !!!!! 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
    CALL MSGDEQ IN <R0>
UNTIL MD.COD EQ #MSGNUL

```

```

MOV (R1)+,R2
BIT #BIT14,STAT(R2)
BEQ 50032$
MOV #1,R4
50032$:
BR 50030$
50031$:
TST R4
BNE 50033$
MOV #KR.MOD,DT.KBRSP
JMP 1$
50033$:
50034$:
MOV R5,-(SP)
MOV R0,-(R5)
JSR PC,MSGDEQ
MOV (SP)+,R5
CMP MD.COD,#MSGNUL
ENE 50034$

```

```

;+

```

```

1006 ;IF THE CLOCK IS PRESENT, CLEAR CLOCK TABLE....SEE IF IT IS...
1007 ;-
1008
1009 001126' CALL CLKCHK IN <R0>
(3) 001126' 010546 MOV R5,-(SP)
(4) 001130' 010045 MOV R0,-(R5)
(3) 001132' 004767 000000G JSR PC,CLKCHK
(3) 001136' 012605 MOV (SP)+,R5
1010
1011 001140' IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 001140' 032760 000001 000014 BIT #CLKPRES,DT.CF0(
(9) 001146' 001405 BEQ 50035$
1012
1013 001150' CALL CLRTIM IN <R0>
(3) 001150' 010546 MOV R5,-(SP)
(4) 001152' 010045 MOV R0,-(R5)
(3) 001154' 004767 000000G JSR PC,CLRTIM
(3) 001160' 012605 MOV (SP)+,R5
1014
1015 001162' ENDIF
(4) 001162' 50035$:
1016
1017
1018 ;+
1019 ; GET MODULE LIST
1020 ;-
1021 001162' LET R1 := DT.MLST(R0)
(4) 001162' 016001 000032 MOV DT.MLST(R0),R1
1022
1023 001166' WHILE (R1) NE #ENDLST DO
(4) 001166' 50036$:
(6) 001166' 021127 000000 CMP (R1),#ENDLST
(9) 001172' 001450 BEQ 50037$
1024
1025
1026
1027 ;+
1028 ; CLEAR OUT HEADER STUFF AND COPY CONFIG WORD TO RES1 IN HEADER
1029 ;-
1030 001174' LET R2 := (R1)+
(4) 001174' 012102 MOV (R1)+,R2
1031
1032
1033 001176' LET PASCNT(R2) := #0
(4) 001176' 005062 000034 CLR PASCNT(R2)
1034 001202' LET ICOUNT(R2) := #0
(4) 001202' 005062 000040 CLR ICOUNT(R2)
1035 001206' LET SOFCNT(R2) := #0
(4) 001206' 005062 000042 CLR SOFCNT(R2)
1036 001212' LET HRDCNT(R2) := #0
(4) 001212' 005062 000044 CLR HRDCNT(R2)
1037 001216' LET SOFPAS(R2) := #0
(4) 001216' 005062 000046 CLR SOFPAS(R2)
1038 001222' LET HRDPAS(R2) := #0
(4) 001222' 005062 000050 CLR HRDPAS(R2)
1039 001226' LET SYSCNT(R2) := #0

```

```

(4) 001226' 005062 000052          CLR   SYSCNT(R2)
1040 001232'          LET XFLAG(R2) := #0
(4) 001232' 105062 000005          CLRB  XFLAG(R2)
1041 001236'          LET RES1(R2) := DT.CF0(R0)
(4) 001236' 016062 000014 000056  MOV   DT.CF0(R0),RES1(
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
(6) 001244' 042762 024000 000026  BIC   #BIT13!ACTBIT,ST
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$:
1060
1061          ;+
1062          ; THIS MODULE NEEDS X - TRA WORK - CLEAR ERROR COUNTS
1063          ; -
1064
1065 001302'          LET CDERCT(R2) := #0
(4) 001302' 005062 000146          CLR   CDERCT(R2)
1066 001306'          LET CDWDCT(R2) := #0
(4) 001306' 005062 000144          CLR   CDWDCT(R2)
1067 001312'          ENDIF
(4) 001312'          50041$:
1068
1069 001312'          ENDDO
(4) 001312' 000725          BR    50036$
(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 ; CLEAR # OF POWER FAILS
1085 ;-
1086 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 ; STUFF UNUSED VECTORS
1111 ;-
1112 001346' CALL SETVEC
(3) 001346' 004767 000000G JSR PC,SETVEC
1116
1117 ;+
1118 ; WRITE WORST CASE UNIBUS PATTERN
1119 ;-
1120 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

```



```

1125 ; MOVE THE CODE IF NECESSARY
1126 ; -
1127
1128 001364' IF KR.MOV EQ #1 THEN
(6) 001364' 026727 176412 000001 CMP KR.MOV,#1
(9) 001372' 001010 BNE 50042$
1129
1130 001374' CALL PRRLOC IN <R0,R3>
(3) 001374' 010546 MOV R5,-(SP)
(5) 001376' 010345 MOV R3,-(R5)
(4) 001400' 010045 MOV R0,-(R5)
(3) 001402' 004767 000000G JSR PC,PRRLOC
(3) 001406' 012605 MOV (SP)+,R5
1131
1132
1133
1134 001410' LET DT.ADDR(R0) := R3
(4) 001410' 010360 000042 MOV R3,DT.ADDR(R0)
1135 001414' ENDIF
(4) 001414' 50042$:
1136
1137 ;+
1138 ; IF THIS IS LOCK MODE OR KT OFF, CLEAR THE RELMODE BIT
1139 ; -
1140
1141 001414' IF KR.FLG EQ #1 OR #KTSTAT NOTSETIN DT.ST0(R0) THEN
(6) 001414' 026727 176360 000001 CMP KR.FLG,#1
(8) 001422' 001404 BEQ 50043$
(6) 001424' 032760 000020 000010 BIT #KTSTAT,DT.ST0(R
(9) 001432' 001004 BNE 50044$
(6) 001434' 50043$:
1142
1143 001434' LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RELMODE
(6) 001434' 042760 020000 000010 BIC #RELMODE,DT.ST0(
1144
1145 001442' ELSE
(4) 001442' 000403 BR 50045$
(3) 001444' 50044$:
1146
1147
1148 ;+
1149 ; NOT LOCK MODE
1150 ; -
1151 001444' LET DT.ST0(R0) := DT.ST0(R0) SET.BY #RELMODE
(6) 001444' 052760 020000 000010 BIS #RELMODE,DT.ST0(
1152
1153 001452' ENDIF
(4) 001452' 50045$:
1154
1155
1156 ;+
1157 ; SETUP THE WRITE BUFFER LIMITS
1158 ; -
1159
1160 001452' CALL WBFLIM IN <R0>
(3) 001452' 010546 MOV R5,-(SP)

```

```
(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
(6) 001464' 032760 001000 000014                          BIT      #ADDR22,DT.CF0(R
(9) 001472' 001405                          BEQ      50046$
1167
1168 001474'                                CALL UNIMAP IN <R0>
(3) 001474' 010546                                MOV      R5,-(SP)
(4) 001476' 010045                                MOV      R0,-(R5)
(3) 001500' 004767 000000G                          JSR      PC,UNIMAP
(3) 001504' 012605                                MOV      (SP)+,R5
1169
1170 001506'                                ENDIF
(4) 001506'                                50046$:
1171
1172 ;+
1173 ; SET RUNMODE BIT.... AND DONE
1174 ; -
1175
1176
1177 001506'                                LET DT.ST0(R0) := DT.ST0(R0) SET.BY #RUNMODE
(6) 001506' 052760 100000 000010                          BIS      #RUNMODE,DT.ST0(
1178
1179
1180 001514'                                INLINE <1$:>
(2) 001514'                                1$:
1181 001514'                                CALL RESREG
(3) 001514' 004767 000000G                          JSR      PC,RESREG
1182
1183 001520'                                ENDRTN
(3) 001520'                                50000$:
(3) 001520'                                50001$:
(2) 001520' 000207                                RTS      PC
1184
1185 000001                                .END
```

SYMBOL TABLE

ACSR = 000102	CLRTIM= ***** G	DT.SWR= 000056	KRUN 000222RG	PASCNT= 000034
ACTBIT= 004000	CM.ADR= ***** G	DT.SYP= 000072	KRUNL 000232RG	PA.CNT= ***** G
ADDR22= 001000	CM.ARG= ***** G	DT.WBU= 000050	KR.BIG 000061R	PDFLSI= 020000
ADR = 000006	CM.NBA= ***** G	DT.WHL= 000054	KR.FLG 000000R	PDP60 = 004000
APTFER= 000004	CM.NUM= ***** G	DT.WLL= 000052	KR.KT 000127R	PDP70 = 010000
APTPRE= 000200	CM.RUN= ***** G	DVID1 = 000014	KR.MAP 000175R	PRI0 = 000000
APTSEL= ***** G	CONFIG= 000056	ECCMEM= 000100	KR.MOD 000150R	PRI1 = 000040
ARGCHK= ***** G	CQOVF = 000001	ECCSTA= 000010	KR.MOV 000002R	PRI4 = 000200
ASB = 000106	CR = 000015	ENBEOB= 010000	KR.SML 000004R	PRI5 = 000240
ASSEMB= 000010	CSRA = 000100	ENBNUL= 000001	KTERRO= 000040	PRI6 = 000300
ASTAT = 000104	CSRC = 000102	ENDLST= 000000	KTPRES= 000400	PRI7 = 000340
AUTO = 000010	CTRLC = 000003	EOPBIT= 000001	KTSTAT= 000020	PRRLOC= ***** G
AUTOST= 020000	CTRLD = 000017	ERRTYP= 000106	KTXTND= 040000	PR.RL1= ***** G
AWAS = 000110	CTRLU = 000025	EVNTBE= 000200	LF = 000012	PRO = 000000
BE.MCN= ***** G	DCEVNT= 000011	EVNTHD= 000200	LPSTAT= 000001	PR4 = 000200
BIT0 = 000001	DEFRTN= 000400	EVNTKT= 000203	MAPSTA= 000200	PR5 = 000240
BIT00 = 000001	DIAGMC= 000000	EVNTPE= 000202	MD.COD= ***** G	PR6 = 000300
BIT01 = 000002	DROPMO= 100000	EVNTRE= 000201	MED = 076600	PR7 = 000340
BIT02 = 000004	DSEVNT= 000014	FATERR= 100000	MEMPAS= 040000	PS = 177776
BIT03 = 000010	DTADR = 000000	HRDCNT= 000044	MODEXH= 004000	PSW = 177776
BIT04 = 000020	DT.ADD= 000042	HRDPAS= 000050	MODHOL= 002000	RANNUM= 000054
BIT05 = 000040	DT.AP = 000100	ICONT = 000036	MODSEL= 001000	RBUFEA= 000130
BIT06 = 000100	DT.APK= 000076	ICOUNT= 000040	MSGCKD= 000010	RBUFPA= 000126
BIT07 = 000200	DT.BLS= 000034	IDNUM = 000122	MSGCKS= 000011	RBUFSZ= 000132
BIT08 = 000400	DT.CFO= 000014	IE = 000100	MSGDEQ= ***** G	RBUFVA= 000124
BIT09 = 000100	DT.CF1= 000016	INDPAR= 000040	MSGDER= 000005	RDSERV= 000101
BIT1 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGDRP= 000017	RDWHMI= 000022
BIT10 = 002000	DT.ESI= 000044	INHEPR= 020000	MSGECH= 177777	RELERR= 000020
BIT11 = 004000	DT.EVN= 000000	INHREL= 001000	MSGEOP= 000013	RELMOD= 020000
BIT12 = 010000	DT.EXS= 000060	INHRR= 000400	MSGHDR= 000004	RELTIM= 010000
BIT13 = 020000	DT.FCH= 000037	INIT = 000030	MSGHNG= 000022	RESREG= ***** G
BIT14 = 040000	DT.FCN= 000036	INTR = 000120	MSGHRD= 000007	RES1 = 000056
BIT15 = 100000	DT.HMX= 000104	IOMOD = 100000	MSGMAP= 000021	RES2 = 000060
BIT2 = 000004	DT.KBE= 000024	IOMODP= 102000	MSGNUL= 177775	RICHAR= 031060
BIT3 = 000010	DT.KBP= 000026	IOMODR= 112000	MSGPOP= 000002	RPTDAT= 002000
BIT4 = 000020	DT.KBR= 000022	IOMODX= 110000	MSGPRM= 177776	RSTRT = 000112
BIT5 = 000040	DT.KBU= 000030	JACK = 035060	MSGRES= 000001	RUBOUT= 000177
BIT6 = 000100	DT.MLS= 000032	KIPAR0= 172340	MSGSFT= 000006	RUNMOD= 100000
BIT7 = 000200	DT.MTI= 000110	KIPAR1= 172342	MSGSKE= 000003	RUNSUB 000240R
BIT8 = 000400	DT.OFF= 000070	KIPAR2= 172344	MSGSMB= 000015	RSVALU= 001740
BIT9 = 001000	DT.PAS= 000074	KIPAR3= 172346	MSGSMH= 000014	SAM = 075464
BKDEF = 000002	DT.PC = 000002	KIPAR4= 172350	MSGSMS= 000016	SAVREG= ***** G
BKMOD = 000020	DT.PFL= 000062	KIPAR5= 172352	MSGSTD= 000000	SBADR = 000102
BKMODE= 040000	DT.PSW= 000004	KIPAR6= 172354	MSGSYS= 000012	SBKMOD= 000000
BKSLSH= 000134	DT.PTA= 000064	KIPAR7= 172356	MSGVEC= 000020	SBKSEL= 010000
BUFPtr= 000002	DT.RCS= 000102	KIPDR0= 172300	NBKMOD= 001000	SC.ADR= 000006
CAPRES= 000004	DT.REL= 000040	KIPDR1= 172302	NCPUOP= 000020	SC.ALC= 000014
CASTAT= 000004	DT.SCT= 000066	KIPDR2= 172304	NOAPTY= 000002	SC.APC= 000016
CDERCT= 000146	DT.SMX= 000106	KIPDR3= 172306	NULL = 000000	SC.CKL= 000002
CDWDCT= 000144	DT.SP = 000006	KIPDR4= 172310	OWEN = 024020	SC.CKP= 000004
CKTIM = 100000	DT.SSI= 000046	KIPDR5= 172312	PAERR = 000010	SC.CLO= 000000
CLKCHK= ***** G	DT.ST0= 000010	KIPDR6= 172314	PARPRE= 002000	SC.HLD= 000010
CLKPRE= 000001	DT.ST1= 000012	KIPDR7= 172316	PARSTA= 000100	SC.SCA= 000012

SENDLS= 177777	TMP10 = 000002	WBUFRQ= 000140	\$FSNO = 000403	\$TSKO = 050046
SETVEC= ***** G	TQOVF = 000002	WBUFSZ= 000142	\$FSOR = 000320	\$TSK1 = 050037
SOFcnt= 000042	UIPAR0= 177640	WDFR = 000116	\$FSRTI= 000350	\$TSK2 = 050041
SOFPAS= 000046	UIPAR1= 177642	WDT0 = 000114	\$FSRTN= 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	\$\$RETN= 000000
SVR0 = 000062	UIPDR5= 177612	\$F\$CAS= 000150	\$NESTL= 177777	\$\$RTN1= 050000
SVR1 = 000064	UIPDR6= 177614	\$F\$DEC= 000220	\$NSKO = 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\$G00= 000400	\$\$AVLE= 177777	\$\$TGS1= 000000
SVR5 = 000074	WBFLIM= ***** G	\$F\$IF = 000110	\$\$SKO = 050037	\$\$TGS2= 000000
SVR6 = 000076	WBSTAT= 000040	\$F\$INC= 000210	\$TAGLE= 177777	\$\$TD = 000000
SYSCNT= 000052	WBUFEA= 000136	\$F\$L00= 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  
CCRE USED: 14K (27 PAGES)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:03  
EQUATE.MAC 13-SEP-78 16:13 . TABLE OF CONTENTS

SEQ 0646

3 COMMON EQUATE MODULE  
576 COMMON DEFINITIONS AND REFERENCES FOR KRUN  
579 000000' .PRINT ;SPMAC: VERSION 1.1  
625 KRUN ROUTINE

```
508 .TITLE KRUNS PROCESS THE 'RUN' COMMAND FOR SMALL MONITORS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KRUNS
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE IS FOR NON-KT MONITORS ONLY.
517 ; THIS ROUTINE PROCESSES THE 'RUN' KEYBOARD
518 ; COMMAND. IF AN ARGUMENT IS INCLUDED, AN ERROR MSG. IS ISSUED.
519 ; ALL OPTION MODULES HEADERS WILL BE INITIALIZED. THE SYS-
520 ; TEM ERROR COUNT AND POWER FAIL COUNT
521 ; WILL BE INITIALIZED. THE WRITE BUFFER LIMITS
522 ; WILL BE SET UP.
523 ;
524 ; INPUTS:
525 ; 1. DTABLE ADDRESS
526 ; 2. DECODE BUFFER POINTER
527 ;
528 ; IMPLICIT INPUTS:
529 ; 1. DT.MLST
530 ; 2. DT.STO
531 ; 3. DT.CFO
532 ;
533 ; OUTPUTS:
534 ; NONE
535 ;
536 ; IMPLICIT OUTPUTS:
537 ; 1. DT.STO
538 ; 2. DT.KBRSP
539 ; 3. DT.ESIZ
540 ; 4. DT.PFL
541 ; 5. DT.EXS
542 ;
543 ; PATHOLOGICAL CONNECTIONS:
544 ; 1. BE.MCNT
545 ; 2. MD.COD
546 ; 3. PA.CNT
547 ; 4. CM.RUN
548 ; 5. CM.ARG
549 ;
550 ; SUBORDINATE ROUTINES CALLED:
551 ; 1. ARGCHK
552 ; 2. CLRTIM
553 ; 3. SAVREG
554 ; 4. RESREG
555 ; 5. WBFLIM
556 ; 6. WSTBUS
557 ; 7. CLKCHK
558 ; 8. APTSEL
559 ;
560 ; FUNCTIONAL SIDE EFFECTS:
561 ; NONE
562 ;
563 ; CALLING SEQUENCE:
```

```
564 ; CALL KRUN IN <DTADR,BUFPTR>  
565 ; WHERE DTADR = DTABLE ADDRESS  
566 ; BUFPTR = COMMAND DECODE BUFFER POINTER  
567 ;  
568 ; VERSION:  
569 ; 0.0  
570 ;  
571 ; EDIT DATE BY REASON  
572 ;--  
573  
574
```

```

576 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KRUN
577
578 .MCALL STRUCT
579 000000'
(1) 000000'
580
581 000001 $LSTTAG=1
582 000001 $LSTIN=1
583 ;*****
584 ;
585 ;
586 ; REFERENCED BY OTHER MODULES:
587 ;
588 .GLOBL KRUN ;MODULE ENTRY POINT
589 ;
590 ;*****
591 ;
592 ; GLOBAL REFERENCES:
593 ;
594 .GLOBL CLKCHK ;SEE IF A CLOCK IS HERE
595 .GLOBL APTSEL ;APT SELECT ROUTINE
596 .GLOBL MD.COD ;CODE
597 .GLOBL MSGDEQ ;MESSAGE DEQUE
598 .GLOBL CM.RUN ;'ILLEGAL COMMAND IN RUN MODE'
599 .GLOBL CM.ARG ;'ILLEGAL OR MISSING ARGUMENT'
600 .GLOBL CLRTIM ;CLEAR TIME TABLE
601 .GLOBL ARGCHK ;CHECK AN ARGUMENT
602 .GLOBL SAVREG ;SAVE REGISTERS
603 .GLOBL RESREG ;RESTORE REGISTERS
604 .GLOBL PA.CNT ;NUMBER OF PARITY ERRORS
605 .GLOBL BE.MCNT ;MONITOR SYSTEM ERROR COUNT
606 .GLOBL WBFLIM ;ESTABLISH WRITE BUFFER LIMITS
607 .GLOBL WSTBUS ;WRITE WORST CASE UNIBUS PATTERN.
608 ;
609 ;*****
610 ;
611 ; GLOBAL EQUATES:
612 ;
613 ;
614 ;*****
615 ;
616 ; LOCAL STORAGE:
617 ;
618 000000' 047045 020117 047515 KR.MOD: .ASCIZ /%NO MODULES SELECTED%/
000006' 052504 042514 020123
000014' 042523 042514 052103
000022' 042105 000045
619 .EVEN
620 ;
621 ;*****
622
623

```



```

625 .SBTTL KRUN ROUTINE
626
627 000026' ROUTINE KRUN <DTADR,BUFPTR>
(2) 000026' KRUN:
628
629
630
631
632 ;+
633 ; SAVE REGISTERS AND SAVE DTABLE ADDR AND BUFFER POINTER
634 ;-
635
636 000026' CALL SAVREG
(3) 000026' 004767 000000G JSR PC,SAVREG
637 000032' LET R0 := DTADR(R5)
(4) 000032' 016500 000000 MOV DTADR(R5),R0
638 000036' LET R1 := BUFPTR(R5)
(4) 000036' 016501 000002 MOV BUFPTR(R5),R1
639 ;+
640 ; IF IN AUTO OR APTPRES MODE, DON'T CHECK ANY ARGUMENTS - JUST GO TO WORK
641 ;-
642
643 000042' IF #AUTO SETIN DT.CF0(R0) OR #APTPRES SETIN DT.CF0(R0) THEN
(6) 000042' 032760 000010 000014 BIT #AUTO,DT.CF0(R0)
(8) 000050' 001004 BNE 50002$
(6) 000052' 032760 000200 000014 BIT #APTPRES,DT.CF0(
(9) 000060' 001401 BEQ 50003$
(6) 000062' 50002$:
644 000062' INLINE <BR 339$>
(2) 000062' 000431 BR 339$
645 000064' ENDF
(4) 000064' 50003$:
646
647
648 ;+
649 ; SEE IF IN RUNMODE - IF SO STUFF ERROR AND SCRAM
650 ;-
651
652 000064' IF #RUNMODE SETIN DT.ST0(R0) THEN
(6) 000064' 032760 100000 000010 BIT #RUNMODE,DT.ST0(
(9) 000072' 001405 BEQ 50004$
653 000074' LET DT.KBRSP(R0) := #CM.RUN
(4) 000074' 012760 000000G 000022 MOV #CM.RUN,DT.KBRSP
654 000102' INLINE <JMP 1$>
(2) 000102' 000167 000424 JMP 1$
655
656 000106' ENDF
(4) 000106' 50004$:
657
658
659 ;+
660 ; WE ARE NOT IN RUNMODE....ALL IS OK....GET AN ARGUMENT IF ANY.....
661 ;-
662
663
664
    
```

```

665
666 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
667          ;+
668          ; IF NO ARGS, JUMP AHEAD
669          ; -
670
671
672 000126'          IF.ERROR THEN
(6) 000126' 103002          BCC     50005$
673
674 000130'          INLINE <BR      339$>
(2) 000130' 000406          BR      339$
675
676 000132'          ELSE
(4) 000132' 000405          BR      50006$
(3) 000134'          50005$:
677
678          ;+
679          ; THERE ARE ARGUMENTS.....TOO BAD - NOT ALLOWED IN THIS MONITOR....
680          ; STUFF ERROR MSG AND SEE YA.....NEXT TIME TYPE 'RUN' ONLY
681          ; -
682 000134'          LET DT.KBRSP(R0) := #CM.ARG
(4) 000134' 012760 000000G 000022      MOV      #CM.ARG,DT.KBRSP
683 000142'          INLINE <JMP      1$>
(2) 000142' 000167 000364          JMP     1$
684 000146'          ENDIF
(4) 000146'          50006$:
685
686 000146'          INLINE <339$:>
(2) 000146'          339$:
687
688          ;+
689          ; IF UNDER APT, USE APT'S TABLE TO SELECT THE MODULES
690          ; AND SET UP DEVICE COUNT
691          ; -
692 000146'          IF #APTPRES SETIN DT.CF0(R0) THEN
(6) 000146' 032760 000200 000014      BIT     #APTPRES,DT.CF0(
(9) 000154' 001405          BEQ     50007$
693 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
694 000170'          ENDIF
(4) 000170'          50007$:
695
696
697          ;+
698          ; GET ADDR OF MODULE LIST
699          ; -
    
```

```

700
701 000170'          LET R1 := DT.MLST(R0)
(4) 000170' 016001 000032          MOV      DT.MLST(R0),R1
702
703
704          ;+
705          ; FLAG
706          ; -
707 000174'          LET R4 := #0
(4) 000174' 005004          CLR      R4
708
709
710          ;+
711          ; WHILE NOT AT END OF MODULE LIST...SEE IF A MODULE IS SELECTED...
712          ; IF SO, INCREMENT A COUNT
713          ; -
714 000176'          WHILE (R1) NE #0 DO
(4) 000176'
(6) 000176' 005711          50010$: TST      (R1)
(9) 000200' 001407          BEQ      50011$
715 000202'          LET R2 := (R1)+
(4) 000202' 012102          MOV      (R1)+,R2
716 000204'          IF #BIT14 SETIN STAT(R2) THEN
(6) 000204' 032762 040000 000026          BIT      #BIT14,STAT(R2)
(9) 000212' 001401          BEQ      50012$
717 000214'          LET R4 := R4 + #1
(6) 000214' 005204          INC      R4
718 000216'          ENDIF
(4) 000216'          50012$:
719 000216'          ENDDO
(4) 000216' 000767          BR      50010$
(3) 000220'          50011$:
720
721
722
723          ;+
724          ; IF R4 IS ZERO...NO MODULES SELECTED.....STUFF ERROR AND GOODBYE...
725          ; -
726
727 000220'          IF R4 EQ #0 THEN
(6) 000220' 005704          TST      R4
(9) 000222' 001005          BNE      50013$
728 000224'          LET DT.KBRSP(R0) := #KR.MOD
(4) 000224' 012760 000000' 000022          MOV      #KR.MOD,DT.KBRSP
729 000232'          INLINE <JMP      1$>
(2) 000232' 000167 000274          JMP      1$
730 000236'          ENDIF
(4) 000236'          50013$:
731
732
733
734          ;+
735          ; !!!!! GO TO WORK !!!!!!   GET THE MODULE HEADERS CLEANED UP !
736          ; BUT FIRST PRINT THE CR - IT TAKES SO LONG TO WRITE WSTBUS
737          ; PATTERN THAT THE CR WOULD NOT BE ECHOED FOR A LONG TIME
738          ; AND IT MAY CAUSE WORRY - HENCE THE CALL TO MSGDEQ WILL PRINT

```

```

739 ; IT RIGHT AWAY....
740 ; -
741
742
743
744 000236' REPEAT
(3) 000236' 50014$:
745 000236' CALL MSGDEQ IN <R0>
(3) 000236' 010546 MOV R5,-(SP)
(4) 000240' 010045 MOV R0,-(R5)
(3) 000242' 004767 000000G JSR PC,MSGDEQ
(3) 000246' 012605 MOV (SP)+,R5
746 000250' UNTIL MD.COD EQ #MSGNUL
(3) 000250' 026727 000000G 177775 CMP MD.COD,#MSGNUL
(6) 000256' 001367 BNE 50014$
747
748
749 ;+
750 ; IF THIS IS THE SCRAWNY MONITOR... DON'T EVEN CONSIDER THE CLOCK
751 ; STUFF
752 ; -
753
754 000260' IF #NCPUOP NOTSETIN DT.CF0(R0) THEN
(6) 000260' 032760 000020 000014 BIT #NCPUOP,DT.CF0(R
(9) 000266' 001016 BNE 50015$
755
756 ;+
757 ; IF THE CLOCK IS PRESENT, CLEAR CLOCK TABLE....SEE IF IT IS...
758 ; -
759 000270' CALL CLKCHK IN <R0>
(3) 000270' 010546 MOV R5,-(SP)
(4) 000272' 010045 MOV R0,-(R5)
(3) 000274' 004767 000000G JSR PC,CLKCHK
(3) 000300' 012605 MOV (SP)+,R5
760
761 000302' IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 000302' 032760 000001 000014 BIT #CLKPRES,DT.CF0(
(9) 000310' 001405 BEQ 50016$
762
763 000312' CALL CLRTIM IN <R0>
(3) 000312' 010546 MOV R5,-(SP)
(4) 000314' 010045 MOV R0,-(R5)
(3) 000316' 004767 000000G JSR PC,CLRTIM
(3) 000322' 012605 MOV (SP)+,R5
764
765 000324' ENDIF
(4) 000324' 50016$:
766 000324' ENDIF
(4) 000324' 50015$:
767
768
769 ;+
770 ; GET MODULE LIST
771 ; -
772 000324' LET R1 := DT.MLST(R0)
(4) 000324' 016001 000032 MOV DT.MLST(R0),R1
    
```

```

773
774
775          ;+
776          ; WHILE NOT AT END OF MODULE LIST, CLR OUT MODULE HEADER LOCATIONS
777          ; AND COPY CONFIG WORD 0 INTO RES1.
778          ;-
779
780          000330'          WHILE (R1) NE #0 DO
(4)          000330'
(6)          000330' 005711          50017$:          TST          (R1)
(9)          000332' 001450          BEQ          50020$
781
782          000334'          LET R2 := (R1)+
(4)          000334' 012102          MOV          (R1)+,R2
783
784
785
786          000336'          LET PASCNT(R2) := #0          CLR          PASCNT(R2)
(4)          000336' 005062 000034
787          000342'          LET ICOUNT(R2) := #0          CLR          ICOUNT(R2)
(4)          000342' 005062 000040
788          000346'          LET SOFCNT(R2) := #0          CLR          SOFCNT(R2)
(4)          000346' 005062 000042
789          000352'          LET HRDCNT(R2) := #0          CLR          HRDCNT(R2)
(4)          000352' 005062 000044
790          000356'          LET SOFPAS(R2) := #0          CLR          SOFPAS(R2)
(4)          000356' 005062 000046
791          000362'          LET HRDPAS(R2) := #0          CLR          HRDPAS(R2)
(4)          000362' 005062 000050
792          000366'          LET SYSCNT(R2) := #0          CLR          SYSCNT(R2)
(4)          000366' 005062 000052
793          000372'          LET XFLAG(R2) :B= #0          CLRB         XFLAG(R2)
(4)          000372' 105062 000005
794          000376'          LET RES1(R2) := DT.CF0(R0)
(4)          000376' 016062 000014 000056          MOV          DT.CF0(R0),RES1(
795
796          ;+
797          ; CLEAR ACTIVE AND DROPPED BIT IN STAT WORD OF MODULE
798          ;-
799
800          000404'          LET STAT(R2) := STAT(R2) CLR.BY #BIT13!ACTBIT
(6)          000404' 042762 024000 000026          BIC          #BIT13!ACTBIT,ST
801
802
803
804
805
806
807
808          ;+
809          ; IF AN IOMODR(P)(X) CLEAR CDATA ERROR AND WORD COUNTS
810          ;-
811
812          000412'          IF #IOMODR SETIN STAT(R2) OR #IOMODP SETIN STAT(R2) OR #IOMODX SETIN STAT(R2) TH
(6)          000412' 032762 112000 000026          BIT          #IOMODR,STAT(R2)
(8)          000420' 001010          BNE          50021$
    
```

```

(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'          5000C$:
(3) 000536'          50001$:
(2) 000536' 000207          RTS      PC
862
863          000001          .END
```

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



SUSPND= 000001	UIPDR2= 177604	SERFLG= 000400	\$F\$YES= 000402	\$\$DST = 000000
SVR0 = 000062	UIPDR3= 177606	\$F\$AND= 000310	\$IFLEV= 177777	\$\$ELOC= 000402
SVR1 = 000064	UIPDR4= 177610	\$F\$BAD= 000401	\$ISK0 = 000001	\$\$ERFL= 000000
SVR2 = 000066	UIPDR5= 177612	\$F\$BLA= 000170	\$ISK1 = 000001	\$\$FLAG= 000001
SVR3 = 000070	UIPDR6= 177614	\$F\$CAS= 000150	\$LOCTA= 177777	\$\$FROM= 000000
SVR4 = 000072	UIPDR7= 177616	\$F\$DEC= 000220	\$LSTIN= 000001	\$\$LCC = 000440R
SVR5 = 000074	WASADR= 000104	\$F\$DD = 000340	\$LSTTA= 000001	\$\$LCCN= 000000
SVR6 = 000076	WBFLIM= ***** G	\$F\$FAL= 000405	\$NESTL= 177777	\$\$REG = 177777
SYSCNT= 000052	WBSTAT= 000040	\$F\$GDD= 000400	\$NSKO = 000300	\$\$RETU= 000000
SYSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$NSK1 = 000120	\$\$RTN1= 050000
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSK2 = 000110	\$\$RTN2= 050001
TQOVF = 000002	WBUFRQ= 000140	\$F\$LDD= 000200	\$\$SAVLE= 177777	\$\$SRC = 000000
UIPAR0= 177640	WBUFSZ= 000142	\$F\$NAM= 000160	\$\$SKO = 050020	\$\$TGSV= 000000
UIPAR1= 177642	WDFR = 000116	\$F\$NO = 000403	STAGLE= 177777	\$\$TGS1= 000000
UIPAR2= 177644	WDTO = 000114	\$F\$OR = 000320	STAGNU= 050023	\$\$TGS2= 000000
UIPAR3= 177646	WSTBUS= ***** G	\$F\$RTI= 000350	\$TEMP = 000300	\$\$TO = 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$RTN= 000300	\$TSKO = 050017	\$\$TAG= 050000
UIPAR5= 177652	WTWHMI= 000222	\$F\$SEL= 000140	\$TSK1 = 050020	= 000540R
UIPAR6= 177654	XFLAG = 000005	\$F\$THE= 000330	\$TSK2 = 050022	
UIPAR7= 177656	XOFF = 000023	\$F\$TRU= 000404	\$\$ARGC= 000004	
UIPDR0= 177600	XON = 000021	\$F\$UNT= 000130	\$\$BYTE= 000403	
UIPDR1= 177602	\$BGNLE= 177777	\$F\$WHI= 000120	\$\$CASE= 000000	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:03

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0659

3 COMMON EQUATE MODULE  
563 COMMON DEFINITIONS AND REFERENCES FOR KSUM  
566 000000' .PRINT ;SPMAC: VERSION 1.1  
602 KSUM ROUTINE

```
508 .TITLE KSUM PROCESS THE 'SUM' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KSUM
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'SUM' KEYBOARD COMMAND. IT WILL
517 ; SEARCH THE MODULE LIST AND ENQUEUE THE SUM MESSAGE FOR
518 ; EACH MODULE. IF A MODULE NAME IS SPECIFIED, A SUMMARY
519 ; MESSAGE IS OUTPUTTED FOR THAT
520 ; MODULE ONLY AND NO HEADER OR TRAILER MESSAGES
521 ; ARE OUTPUTTED.
522 ;
523 ;
524 ; INPUTS:
525 ; 1. ADDRESS OF DATA TABLE
526 ; 2. COMMAND DECODE BUFFER POINTER
527 ;
528 ; IMPLICIT INPUTS:
529 ; DT.MLST, DT.STO
530 ;
531 ; OUTPUTS:
532 ; NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ; DT.KBRSP
536 ;
537 ; PATHOLOGICAL CONNECTIONS:
538 ; CM.ARG, CM.BADNAME
539 ;
540 ; SUBORDINATE ROUTINES CALLED:
541 ; 1. ARGCHK - CHECK FOR JUNK ARGUMENTS
542 ; 2. ENQTQ - ENQUEUE A MESSAGE
543 ; 3. NAMCHK - VERIFY A MODULE NAME
544 ; 4. SAVREG - SAVE REGISTERS
545 ; 5. RESREG - RESTORE REGISTERS
546 ;
547 ; FUNCTIONAL SIDE EFFECTS:
548 ; NONE
549 ;
550 ; CALLING SEQUENCE:
551 ; CALL KSUM IN <DTADR,BUFPTR>
552 ; WHERE DTADR = ADDRESS OF DATA TABLE
553 ; BUFPTR = COMMAND DECODE BUFFER POINTER
554 ;
555 ; VERSION:
556 ; 0.0
557 ;
558 ; EDIT DATE BY REASON
559 ;
560 ;
561 ;
```

```

563          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KSUM
564
565          .MCALL STRUCT
566 000000'   STRUCT
(1) 000000'   .PRINT ;SPMAC: VERSION 1.1
567
568          000001   $LSTTAG=1
569          000001   $LSTIN=1
570
571
572          ;*****
573          ;
574          ; REFERENCED BY OTHER MODULES
575          ;
576          .GLOBL KSUM          ;MODULE ENTRY POINT
577          ;
578          ;*****
579          ;
580          ; GLOBAL REFERENCES:
581          ;
582          .GLOBL SAVREG        ;SAVE REGISTERS
583          .GLOBL RESREG        ;RESTORE REGISTERS
584          .GLOBL ARGCHK        ;CHECK AN ARGUMENT
585          .GLOBL ENQTQ         ;ENQUEUE A MESSAGE
586          .GLOBL CM.ARG        ;'ILLEGAL ARGUMENT' ERROR MSG
587          .GLOBL NAMCHK        ;CHECK A MODULE NAME
588          .GLOBL CM.BADNAME    ;'INVALID MODULE NAME' ERROR MESSAGE
589          ;
590          ;*****
591          ;
592          ; LOCAL STORAGE:
593          ;
594 000000' 000000   KS.HLD:      .WORD 0          ;HOLD ADDRESS OF PROMPT
595 000002' 041445 042115 000076   KS.CMD: .ASCIZ /%CMD>/
596 000010' 041045 054523 000076   KS.RUN: .ASCIZ /%BSY>/
597          .EVEN
598          ;
599
600

```

```

602          .SBTTL  KSUM ROUTINE
603
604 000016'  ROUTINE KSUM <DTADR,BUFPTR>
605          (2) 000016'
606
607          ;+
608          ; SAVE REGISTERS, DTABLE ADDRESS, AND DECODE BUFFER PTR
609          ; -
610 000016'  CALL SAVREG
611 (3) 000016' 004767 000000G          JSR      PC,SAVREG
612 000022'  LET R0 := DTADR(R5)          MOV      DTADR(R5),R0
613 (4) 000022' 016500 000000          MOV      BUFPTR(R5),R1
614 000026'  LET R1 := BUFPTR(R5)
615 (4) 000026' 016501 000002
616
617          ;+
618          ; IF IN RUN MODE - STUFF A PROMPT TO BE ENQUEUED LATER
619          ; IN THIS MODULE - IT IS USED BECAUSE THE NORMAL PROMPT
620          ; IS OUTPUTTED BEFORE ANY SUM MESSAGES ARE OUTPUTTED -
621          ; THIS ONE WILL FOLLOW THE SUM MESSAGES....
622          ; -
623 000032'  IF #RUNMODE SETIN DT.STO(R0) THEN
624 (6) 000032' 032760 100000 000010          BIT      #RUNMODE,DT.STO(
625 (9) 000040' 001404          BEQ      50002$
626          LET KS.HLD := #KS.RUN          MOV      #KS.RUN,KS.HLD
627 (4) 000042' 012767 000010' 177730          BR      50003$
628 000050'  ELSE
629 (4) 000050' 000403          LET KS.HLD := #KS.CMD          MOV      #KS.CMD,KS.HLD
630 (3) 000052'          ENDIF
631 000052'  LET KS.HLD := #KS.CMD          MOV      #KS.CMD,KS.HLD
632 (4) 000052' 012767 000002' 177720          ENDIF
633 000060'  CALL ARGCHK IN <R1> OUT <R1>
634 (4) 000060' 162705 000002          SUB      #1*2,R5
635 (3) 000064' 010546          MOV      R5,-(SP)
636 (4) 000066' 010145          MOV      R1,-(R5)
637 (3) 000070' 004767 000000G          JSR      PC,ARGCHK
638 (3) 000074' 012605          MOV      (SP)+,R5
639 (4) 000076' 012501          MOV      (R5)+,R1
640
641          ;+
642          ; IF THERE ARE NO ARGUMENTS DO A FULL SUMMARY (ALL MODULES). GET ADDRESS
643          ; OF MODULE LIST, ENQUEUE THE RUN SUMMARY HEADER MESSAGE, THEN, WHILE
644          ; THE ENTRY IN THE MODULE LIST IS NOT 0, ENQUEUE THE SUMMARY MESSAGE
645          ; FOR THE MODULES. WHEN MODULE LIST ENTRY = 0, ENQUEUE THE TRAILING MESSAGE..
646          ; -
    
```

```

641
642 000100'          IF.ERROR THEN
(6) 000100' 103026          BCC      50004$
643
644          LET R1 := #MSGSMH
(4) 000102' 012701 000014          MOV      #MSGSMH,R1
645          LET R2 := #0
(4) 000106' 005002          CLR      R2
646          LET R3 := #0
(4) 000110' 005003          CLR      R3
647          CALL NQ
(3) 000112' 004767 000176          JSR      PC,NQ
648
649          LET R1 := #MSGSMB
(4) 000116' 012701 000015          MOV      #MSGSMB,R1
650          LET R4 := DT.MLST(R0)
(4) 000122' 016004 000032          MOV      DT.MLST(R0),R4
651          WHILE (R4) NE #0 DO
(4) 000126'
(6) 000126' 005714          50005$: TST      (R4)
(9) 000130' 001404          BEQ      50006$
652          LET R3 := (R4)+
(4) 000132' 012403          MOV      (R4)+,R3
653          CALL NQ
(3) 000134' 004767 000154          JSR      PC,NQ
654          ENDDO
(4) 000140' 000772          BR      50005$
(3) 000142'          50006$:
655
656          LET R1 := #MSGSMS
(4) 000142' 012701 000016          MOV      #MSGSMS,R1
657          LET R2 := #0
(4) 000146' 005002          CLR      R2
658          CALL NQ
(3) 000150' 004767 000140          JSR      PC,NQ
659
660          ELSE
(4) 000154' 000442          BR      50007$
(3) 000156'          50004$:
661
662          ;+
663          ; THERE IS AN ARGUMENT - GO CHECK IT OUT
664          ;+
665
666          CALL NAMCHK IN <R0,R1> OUT <R2,R1>
(4) 000156' 162705 000004          SUB      #2*2,R5
(3) 000162' 010546          MOV      R5,-(SP)
(5) 000164' 010145          MOV      R1,-(R5)
(4) 000166' 010045          MOV      R0,-(R5)
(3) 000170' 004767 000000G          JSR      PC,NAMCHK
(3) 000174' 012605          MOV      (SP)+,R5
(4) 000176' 012502          MOV      (R5)+,R2
(4) 000200' 012501          MOV      (R5)+,R1
667          IF.NO.ERROR THEN
(6) 000202' 103424          BCS      50010$
668

```

```
669          ;+
670          ; MODULE NAME IS O.K. - NOW JUST VERIFY THERE IS NO JUNK ARGUMENT
671          ; -
672
673          CALL ARGCHK IN <R1> OUT <R1>
674          (4) 000204' 162705 000002          SUB      #1*2,R5
675          (3) 000210' 010546          MOV      R5,-(SP)
676          (4) 000212' 010145          MOV      R1,-(R5)
677          (3) 000214' 004767 000000G      JSR      PC,ARGCHK
678          (3) 000220' 012605          MOV      (SP)+,R5
679          (4) 000222' 012501          MOV      (R5)+,R1
680          IF.ERROR THEN
681          (6) 000224' 103007          BCC      50011$
682
683          ;+
684          ; NO JUNK ARGUMENTS.... ENQUEUE THE MESSAGE FOR THIS MODULE
685          ; -
686
687          LET R1 := #MSG SMB
688          (4) 000226' 012701 000015          MOV      #MSG SMB,R1
689          (3) 000232' 010203          LET R3 := R2
690          (4) 000234' 005002          MOV      R2,R3
691          (3) 000236' 004767 000052          LET R2 := #0
692          (4) 000238' 005002          CLR      R2
693          (3) 000240' 004767 000052          CALL NQ
694          (4) 000242' 000403          JSR      PC,NQ
695          (3) 000244'
696          ELSE
697          (4) 000246' 000403          BR      50012$
698          (3) 000248'          50011$:
699
700          ;+
701          ; THERE IS A JUNK ARGUMENT IN DECODE BUFFER... STUFF ERROR MSG...
702          ; -
703          LET DT.KBRSP(R0) := #CM.ARG
704          (4) 000244' 012760 000000G 000022          MOV      #CM.ARG,DT.KBRSP
705          (3) 000250'
706          ENDF
707          (4) 000252'          50012$:
708          (3) 000254'
709          ELSE
710          (4) 000256' 000403          BR      50013$
711          (3) 000258'          50010$:
712
713          ;+
714          ; BAD MODULE NAME....STUFF ERROR MSG....
715          ; -
716          LET DT.KBRSP(R0) := #CM.BADNAME
717          (4) 000254' 012760 000000G 000022          MOV      #CM.BADNAME,DT.K
718          (3) 000260'
719          ENDF
720          (4) 000262'          50013$:
721          (3) 000264'
722          ENDF
723          (4) 000266'          50007$:
```

```

705
706          ;+
707          ; IF NO ERRORS - ENQUEUE A PROMPT
708          ; -
709
710          IF DT.KBRSP(R0) EQ #0 THEN
711          (6) 000262' 005760 000022          TST      DT.KBRSP(R0)
712          (9) 000266' 001007          BNE      50014$
713          LET R1 := #MSGSTD
714          (4) 000270' 012701 000000          MOV      #MSGSTD,R1
715          (4) 000274' 016702 177500          LET R2 := KS.HLD
716          (4) 000274' 016702 177500          MOV      KS.HLD,R2
717          (4) 000300' 005003          LET R3 := #0
718          (4) 000300' 005003          CLR      R3
719          (4) 000302'          CALL NQ
720          (3) 000302' 004767 000006          JSR      PC,NQ
721          (4) 000306'          ENDIF
722          (4) 000306'          50014$:
723
724          ;+
725          ; CLEAN UP AND GOODBYE
726          ; -
727
728          CALL RESREG
729          (3) 000306' 004767 000000G          JSR      PC,RESREG
730
731          ENDRTN
732          (3) 000312'          50000$:
733          (3) 000312'          50001$:
734          (2) 000312' 000207          RTS      PC
735
736          ;+
737          ; THIS IS THE ACTUAL CALL TO THE TYPE-QUEUE ENQUEUEING ROUTINE.
738          ; -
739
740          ROUTINE NQ
741          (2) 000314'          NQ:
742
743          CALL ENQTQ IN <R0,R1,R2,R3,#0>
744          (3) 000314'          MOV      R5,-(SP)
745          (8) 000316' 010546          MOV      #0,-(R5)
746          (7) 000322' 010345          MOV      R3,-(R5)
747          (6) 000324' 010245          MOV      R2,-(R5)
748          (5) 000326' 010145          MOV      R1,-(R5)
749          (4) 000330' 010045          MOV      R0,-(R5)
750          (3) 000332' 004767 000000G          JSR      PC,ENQTQ
751          (3) 000336' 012605          MOV      (SP)+,R5
752
753          ENDRTN
754          (3) 000340'          50000$:
755          (3) 000340'          50001$:
756          (2) 000340' 000207          RTS      PC
757          (2) 000340' 000001          .END

```



ACSR = 000102	CQOVF = 000001	ECCSTA= 000010	KTSTAT= 000020	PR4 = 000200
ACTBIT= 004000	CR = 000015	ENBEOP= 010000	KTXTND= 040000	PR5 = 000240
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	LF = 000012	PR6 = 000300
ADR = 000006	CSRC = 000102	ENDLST= 000000	LPSTAT= 000001	PR7 = 000340
APTFER= 000004	CTRLC = 000003	ENQTQ = ***** G	MAPSTA= 000200	PS = 177776
APTPRE= 000200	CTRLD = 000017	EOPBIT= 000001	MED = 076600	PSW = 177776
ARGCHK= ***** G	CTRLU = 000025	ERRTYP= 000106	MEMPAS= 040000	RANNUM= 000054
ASB = 000106	DCEVNT= 000011	EVNTBE= 000200	MODEXH= 004000	RBUFEA= 000130
ASSEMB= 000010	DEFRTN= 000400	EVNTHD= 000200	MODHOL= 002000	RBUFPA= 000126
ASTAT = 000104	DIAGMC= 000000	EVNTKT= 000203	MODSEL= 001000	RBUFSZ= 000132
AUTO = 000010	DROPMO= 100000	EVNTPE= 000202	MSGCKD= 000010	RBUFVA= 000124
AUTOST= 020000	DSEVNT= 000014	EVNTRE= 000201	MSGCKS= 000011	RDSERV= 000101
AWAS = 000110	DTADR = 000000	FATERR= 100000	MSGDER= 000005	RDWHMI= 000022
BIT0 = 000001	DT.ADD= 000042	HRDCNT= 000044	MSGDRP= 000017	RELERR= 000020
BIT00 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGECH= 177777	RELMOD= 020000
BIT01 = 000002	DT.APK= 000076	ICGNT = 000036	MSGEOP= 000013	RELTIM= 010000
BIT02 = 000004	DT.BLS= 000034	ICOUNT= 000040	MSGHRD= 000004	RESREG= ***** G
BIT03 = 000010	DT.CF0= 000014	IDNUM = 000122	MSGHNG= 000022	RES1 = 000056
BIT04 = 000020	DT.CF1= 000016	IE = 000100	MSGHRD= 000007	RES2 = 000060
BIT05 = 000040	DT.ERR= 000020	INDPAR= 000040	MSGMAP= 000021	RICHAR= 031060
BIT06 = 000100	DT.ESI= 000044	INHDRP= 040000	MSGNL= 177775	RPTDAT= 002000
BIT07 = 000200	DT.EVN= 000000	INHEPR= 020000	MSGPOP= 000002	RSTRT = 000112
BIT08 = 000400	DT.EXS= 000060	INHREL= 001000	MSGPRM= 177776	RUBOUT= 000177
BIT09 = 001000	DT.FCH= 000037	INHRE= 000400	MSGRES= 000001	RUNMOD= 100000
BIT1 = 000002	DT.FCN= 000036	INIT = 000030	MSGSFT= 000006	R5VALU= 001740
BIT10 = 002000	DT.HMX= 000104	INTR = 000120	MSGSK= 000003	SAM = 075464
BIT11 = 004000	DT.KBE= 000024	IOMOD = 100000	MSGSM3= 000015	SAVREG= ***** G
BIT12 = 010000	DT.KBP= 000026	IOMODP= 102000	MSGSMH= 000014	SBADR = 000102
BIT13 = 020000	DT.KBR= 000022	IOMODR= 112000	MSGSMS= 000016	SBKMOD= 000000
BIT14 = 040000	DT.KBU= 000030	IOMODX= 110000	MSGSTD= 000000	SBKSEL= 010000
BIT15 = 100000	DT.MLS= 000032	JACK = 035060	MSGSYS= 000012	SC.ADR= 000006
BIT2 = 000004	DT.MTI= 000110	KIPAR0= 172340	MSGVEC= 000020	SC.ALC= 000014
BIT3 = 000010	DT.OFF= 000070	KIPAR1= 172342	NAMCHK= ***** G	SC.APC= 000016
BIT4 = 000020	DT.PAS= 000074	KIPAR2= 172344	NBKM0D= 001000	SC.CKL= 000002
BIT5 = 000040	DT.PC = 000002	KIPAR3= 172346	NCPUOP= 000020	SC.CKP= 000004
BIT6 = 000100	DT.PFL= 000062	KIPAR4= 172350	NOPTY= 000002	SC.CLO= 000000
BIT7 = 000200	DT.PSW= 000004	KIPAR5= 172352	NQ = 000314R	SC.HLD= 000010
BIT8 = 000400	DT.PTA= 000064	KIPAR6= 172354	NULL = 000000	SC.SCA= 000012
BIT9 = 001000	DT.RCS= 000102	KIPAR7= 172356	OWEN = 024020	SENDLS= 177777
BKDEF = 000002	DT.REL= 000040	KIPDR0= 172300	PAERR = 000010	SQFCNT= 000042
BKMOD = 000020	DT.SCT= 000066	KIPDR1= 172302	PARPRE= 002000	SQFPAS= 000046
BKMODE= 040000	DT.SMX= 000106	KIPDR2= 172304	PARSTA= 000100	SPACE = 000040
BKSLSH= 000134	DT.SP = 000006	KIPDR3= 172306	PASCNT= 000034	SPOINT= 000032
BUFPtr= 000002	DT.SSI= 000046	KIPDR4= 172310	PDPLSI= 020000	SPVALU= 002200
CAPRES= 000004	DT.STO= 000010	KIPDR5= 172312	PDP60 = 004000	SRC = 177572
CASSTAT= 000004	DT.ST1= 000012	KIPDR6= 172314	PDP70 = 010000	SR1 = 177574
CDERCT= 000146	DT.SWR= 000056	KIPDR7= 172316	PRI0 = 000000	SR2 = 177576
CDWDCT= 000144	DT.SYP= 000072	KSUM 000016RG	PRI1 = 000040	SR3 = 172516
CKTIM = 100000	DT.WBU= 000050	KS.CMD 000002R	PR14 = 000200	STAT = 000026
CLKPRE= 000001	DT.WHL= 000054	KS.HLD 000000R	PR15 = 000240	STATBI= 064757
CM.ARG= ***** G	DT.WLL= 000052	KS.RUN 000010R	PR16 = 000300	STAT1 = 000027
CM.BAD= ***** G	DVID1 = 000014	KTERRO= 000040	PR17 = 000340	SUSPND= 000001
CONFIG= 000056	ECCMEM= 000100	KTPRES= 000400	PRO = 000000	SVRC = 000062

SVR1 = 000064	UIPDR3= 177606	\$FSBAD= 000401	\$IFLEV= 177777	\$\$BYTE= 000403
SVR2 = 000066	UIPDR4= 177610	\$FSBLA= 000170	\$ISK0 = 000001	\$\$CASE= 000000
SVR3 = 000070	UIPDR5= 177612	\$FSCAS= 000150	\$ISK1 = 000001	\$\$DST = 000000
SVR4 = 000072	UIPDR6= 177614	\$FSDEC= 000220	\$ISK2 = 000001	\$\$ELOC= 000402
SVR5 = 000074	UIPDR7= 177616	\$FSDO = 000340	\$LOCTA= 177777	\$\$ERFL= 000000
SVR6 = 000076	WASADR= 000104	\$FSFAL= 000405	\$LSTIN= 000001	\$\$FLAG= 000001
SYSCNT= 000052	WBSTAT= 000040	\$FSGDD= 000400	\$LSTTA= 000001	\$\$FROM= 000000
SYSERR= 000100	WBUFEA= 000136	\$FSIF = 000110	\$NESTL= 177777	\$\$LOC = 000266R
TMPID = 000002	WBUFPA= 000134	\$FSINC= 000210	\$NSK0 = 000300	\$\$LOCN= 000000
TQOVF = 000002	WBUFRQ= 000140	\$FSLOD= 000200	\$NSK1 = 000110	\$\$REG = 177777
UIPAR0= 177640	WBUFSZ= 000142	\$FSNAM= 000160	\$NSK2 = 000110	\$\$RETU= 000000
UIPAR1= 177642	WDFR = 000116	\$FSNO = 000403	\$NSK3 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDTO = 000114	\$FSOR = 000320	\$\$AVLE= 177777	\$\$RTN2= 050001
UIPAR3= 177646	WTINRE= 000352	\$FSRTI= 000350	\$\$SK0 = 050006	\$\$SRC = 000000
UIPAR4= 177650	WTWHMI= 000222	\$FSRTN= 000300	STAGLE= 177777	\$\$TGSV= 000000
UIPAR5= 177652	XFLAG = 000005	\$FSSEL= 000140	STAGNU= 050002	\$\$TGS1= 000000
UIPAR6= 177654	XOFF = 000023	\$FSTHE= 000330	STEMP = 000300	\$\$TGS2= 000000
UIPAR7= 177656	XON = 000021	\$FSTRU= 000404	STSK0 = 050014	\$\$TD = 000005
UIPDR0= 177600	\$BGNLE= 177777	\$FSUNT= 000130	STSK1 = 050013	\$\$TAG= 050000
UIPDR1= 177602	\$ERFLG= 000400	\$FSWHI= 000120	STSK2 = 050012	. = 000342R
UIPDR2= 177604	\$FSAND= 000310	\$FSYES= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:04  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0668

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES FOR KSWR  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
598 KSWR ROUTINE

```
508 .TITLE KSWR PROCESS THE 'SWR' KEYBOARD COMMAND
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; KSWR
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'SWR' KEYBOARD COMMAND.
517 ; IF NO ARGUMENT IS FOUND IN THE DECODE BUFFER, THE CONTENTS
518 ; OF THE SOFTWARE SWITCH REGISTER ARE OUTPUTTED. THE NUMBER
519 ; IS VERIFIED TO BE OCTAL AND 16 BITS OR LESS IN LENGTH. IF
520 ; THE NUMBER IS VALID, IT WILL REPLACE THE CONTENTS OF THE
521 ; SOFTWARE SWITCH REGISTER AND THE NEW CONTENTS WILL BE OUTPUTTED.
522 ;
523 ; INPUTS:
524 ; 1. ADDRESS OF DATA TABLE
525 ; 2. DECODE BUFFER POINTER
526 ;
527 ; IMPLICIT INPUTS:
528 ; NONE
529 ;
530 ; OUTPUTS:
531 ; NONE
532 ;
533 ; IMPLICIT OUTPUTS:
534 ; 1. DT.KBRSP
535 ; 2. DT.SWR
536 ;
537 ; PATHOLOGICAL CONNECTIONS:
538 ; CM.ARG
539 ;
540 ; SUBORDINATE ROUTINES CALLED:
541 ; 1. ARGCHK - VERIFY AN ARGUMENT
542 ; 2. BOA16 - BINARY TO OCTAL ASCII CONVERSION
543 ; 3. SAVREG - SAVE REGISTERS
544 ; 4. RESREG - RESTORE REGISTERS
545 ; 5. NUMCHK - CHECK FOR VALID NUMBER
546 ;
547 ; FUNCTIONAL SIDE EFFECTS:
548 ; NONE
549 ;
550 ; CALLING SEQUENCE:
551 ; CALL KSWR IN <DTADR,BUFPTR>
552 ; WHERE: DTADR = ADDRESS OF DTABLE
553 ; BUFPTR = CMD DECODE BUFFER POINTER
554 ;
555 ; VERSION:
556 ; 0.0
557 ;
558 ; EDIT DATE BY REASON
559 ;--
```

```
561 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR KSWR
562
563 .MCALL STRUCT
564 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
565
566 000001 $LSTIN=1
567 000001 $LSTTAG=1
568
569 ;*****
570 ;
571 ; REFERENCED BY OTHER MODULES:
572 ;
573 .GLOBL KSWR ;MODULE ENTRY POINT
574 ;
575 ;*****
576 ;
577 ; GLOBAL REFERENCES:
578 ;
579 .GLOBL SAVREG ;SAVE REGISTERS
580 .GLOBL RESREG ;RESTORE REGISTERS
581 .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION
582 .GLOBL ARGCHK ;CHECK AN ARGUMENT
583 .GLOBL NUMCHK ;CHECK AN OCTAL ASCII NUMBER
584 .GLOBL CM.ARG ;"INVALID OR MISSING ARGUMENT" MESSAGE
585 ;
586 ;*****
587 ;
588 ; LOCAL STORAGE:
589 ;
590 000000' 053523 027522 040 KS.MG1: .ASCII 'SWR/ ' ;SWR MESSAGE
591 000005' 000006 KS.MG2: .BLKB 6 ;HOLD ASCII DT.SWR CONTENTS
592 000013' 045 000 .ASCIZ /%/
593 000016' .EVEN
594 ;*****
595
596
```

```

598 .SBTTL KSWR ROUTINE
599
600
601 000016' ROUTINE KSWR <DTADR,BUFPTR>
(2) 000016' KSWR:
602
603 ;+
604 ; SAVE REGISTERS, DATA TABLE ADDRESS, AND DECODE BUFFER POINTER
605 ; -
606
607 000016' CALL SAVREG JSR PC,SAVREG
(3) 000016' 004767 000000G
608 000022' LET R0 := DTADR(R5) MOV DTADR(R5),R0
(4) 000022' 016500 000000
609 000026' LET R1 := BUFPTR(R5) MOV BUFPTR(R5),R1
(4) 000026' 016501 000002
610 ;+
611 ; GET TO FIRST ARGUMENT
612 ; -
613
614 000032' CALL ARGCHK IN <R1> OUT <R1>
(4) 000032' 162705 000002 SUB #2*2,R5
(3) 000036' 010546 MOV R5,-(SP)
(4) 000040' 010145 MOV R1,-(R5)
(3) 000042' 004767 000000G JSR PC,ARGCHK
(3) 000046' 012605 MOV (SP)+,R5
(4) 000050' 012501 MOV (R5)+,R1
615
616
617 ;+
618 ; IF IT'S NOT A CARRIAGE RETURN, VERIFY THAT IT IS NUMBER
619 ; -
620
621 000052' IF.NO.ERROR THEN
(6) 000052' 103435 BCS 50002$
622 000054' CALL NUMCHK IN <R0,R1> OUT <R1,R2>
(4) 000054' 162705 000004 SUB #2*2,R5
(3) 000060' 010546 MOV R5,-(SP)
(5) 000062' 010145 MOV R1,-(R5)
(4) 000064' 010045 MOV R0,-(R5)
(3) 000066' 004767 000000G JSR PC,NUMCHK
(3) 000072' 012605 MOV (SP)+,R5
(4) 000074' 012501 MOV (R5)+,R1
(4) 000076' 012502 MOV (R5)+,R2
623
624
625 ;+
626 ; IF THERE IS AN ERROR IN THE NUMBER, LEAVE
627 ; -
628
629 000100' IF.ERROR THEN
(6) 000100' 103002 BCC 50003$
630 000102' INLINE <BR 1$>
(2) 000102' 000434 BR 1$
631
632 000104' ELSE
    
```

```

(4) 000104' 000420
(3) 000106'
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 MCV #CM.ARG,DT.KBRSP
(4) 000130' 012760 000000G 000022 INLINE <BR 1$> BR 1$
647 000136'
(2) 000136' 000416
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 50004$:
(4) 000146'
660
661 000146' ENDIF 50002$:
(4) 000146'
662
663 ;+
664 ; CONVERT THE NUMBER IN DT.SWR TO ASCII, STUFF MESSAGE
665 ; AND RETURN
666 ; -
667 000146' CALL BDA16 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,BDA16
    
```

```
(3) 000164' 012605 MOV (SP)+,R5
668
669 000166' LET DT.KBRSP(R0) := #KS.MG1
(4) 000165' 012760 000000' 000022 MOV #KS.MG1,DT.KBRSP
670
671 ;+
672 ; RESTORE REGISTERS AND RETURN.
673 ; -
674
675 000174' INLINE <1$:>
(2) 000174' 1$:
676 000174' CALL RESREG
(3) 000174' 004767 000000G JSR PC,RESREG
677
678 000200' ENDRTN
(3) 000200'
(3) 000200'
(2) 000200' 000207
679 000001 .END
```

```
50000$:
50001$:
RTS PC
```



ACSR = 000102	CQOVF = 000001	ECCSTA= 000010	LF = 000012	PR7 = 000340
ACTBIT= 004000	CR = 000015	ENBEOP= 010000	LPSTAT= 000001	PS = 177776
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	MAPSTA= 000200	PSW = 177776
ADR = 000006	CSRC = 000102	ENDLST= 000000	MED = 076600	RANNUM= 000054
APTFER= 000004	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	RBUFEA= 000130
APTPRE= 000200	CTRLD = 000017	ERRTYP= 000106	MODEXH= 004000	RBUFPA= 000126
ARGCHK= ***** G	CTRLU = 000025	EVNTBE= 000200	MODHOL= 002000	RBUFSZ= 000132
ASB = 000106	DCEVNT= 000011	EVNTHD= 000200	MODSEL= 001000	RBUFVA= 000124
ASSEMB= 000010	DEFRTN= 000400	EVNTKT= 000203	MSGCKD= 000010	RDSERV= 000101
ASTAT = 000104	DIAGMC= 000000	EVNTPE= 000202	MSGCKS= 000011	RDWHMI= 000022
AUTO = 000010	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	RELERR= 000020
AUTDST= 020000	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	RELMOD= 020000
AWAS = 000110	DTADR = 000000	HRDCNT= 000044	MSGECH= 177777	RELTIM= 010000
BIT0 = 000001	DT.ADD= 000042	HRDPAS= 000050	MSGEOP= 000013	RESREG= ***** G
BIT00 = 000001	DT.AP = 000100	ICONT = 000036	MSGHDR= 000004	RES1 = 000056
BIT01 = 000002	DT.APK= 000076	ICOUNT= 000040	MSGHNG= 000022	RES2 = 000060
BIT02 = 000004	DT.BLS= 000034	IDNUM = 000122	MSGHRD= 000007	RICHAR= 031060
BIT03 = 000010	DT.CF0= 000014	IE = 000100	MSGMAP= 000021	RPTDAT= 002000
BIT04 = 000020	DT.CF1= 000016	INDPAR= 000040	MSGMUL= 177775	RSTRT = 000112
BIT05 = 000040	DT.ERR= 000020	INHDRP= 040000	MSGPOP= 000002	RUBOUT= 000177
BIT06 = 000100	DT.ESI= 000044	INHEPR= 020000	MSGPRM= 177776	RUNMOD= 100000
BIT07 = 000200	DT.EVN= 000000	INHREL= 001000	MSGRES= 000001	RVALU = 001740
BIT08 = 000400	DT.EXS= 000060	INHRR= 000400	MSGSFT= 000006	SAM = 075464
BIT09 = 001000	DT.FCH= 000037	INIT = 000030	MSGSKE= 000003	SAVREG= ***** G
BIT1 = 000002	DT.FCN= 000036	INTR = 000120	MSGSMB= 000015	SBADR = 000102
BIT10 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSMH= 000014	SBKMOD= 000000
BIT11 = 004000	DT.KBE= 000024	IOMODP= 102000	MSGSMS= 000016	SBKSEL= 010000
BIT12 = 010000	DT.KBP= 000026	IOMODR= 112000	MSGSTD= 000000	SC.ADR= 000006
BIT13 = 020000	DT.KBR= 000022	IOMODX= 110000	MSGSYS= 000012	SC.ALC= 000014
BIT14 = 040000	DT.KBU= 000030	JACK = 035060	MSGVEC= 000020	SC.APC= 000016
BIT15 = 100000	DT.MLS= 000032	KIPAR0= 172340	NBKMOD= 001000	SC.CKL= 000002
BIT2 = 000004	DT.MTI= 000110	KIPAR1= 172342	NCPUOP= 000020	SC.CKP= 000004
BIT3 = 000010	DT.OFF= 000070	KIPAR2= 172344	NDAPTY= 000002	SC.CLO= 000000
BIT4 = 000020	DT.PAS= 000074	KIPAR3= 172346	NULL = 000000	SC.HLD= 000010
BIT5 = 000040	DT.PC = 000002	KIPAR4= 172350	NUMCHK= ***** G	SC.SCA= 000012
BIT6 = 000100	DT.PFL= 000062	KIPAR5= 172352	OWEN = 024020	SENDSL= 177777
BIT7 = 000200	DT.PSW= 000004	KIPAR6= 172354	PAERR = 000010	SOFcnt= 000042
BIT8 = 000400	DT.PTA= 000054	KIPAR7= 172356	PARPRE= 002000	SOPPAS= 000046
BIT9 = 001000	DT.RCS= 000102	KIPDR0= 172300	PARSTA= 000100	SPACE = 000040
BKDEF = 000002	DT.REL= 000040	KIPDR1= 172302	PASCNT= 000034	SPOINT= 000032
BKMOD = 000020	DT.SCT= 000066	KIPDR2= 172304	PDPLSI= 020000	SPVALU= 002200
BKMODE= 040000	DT.SMX= 000106	KIPDR3= 172306	PDP60 = 004000	SRO = 177572
BKSLSH= 000134	DT.SP = 000006	KIPDR4= 172310	PDP70 = 010000	SR1 = 177574
BOA16 = ***** G	DT.SSI= 000046	KIPDR5= 172312	PRI0 = 000000	SR2 = 177576
BUFPtr= 000002	DT.ST0= 000010	KIPDR6= 172314	PRI1 = 000040	SR3 = 172516
CAPRES= 000004	DT.ST1= 000012	KIPDR7= 172316	PRI4 = 000200	STAT = 000026
CASAT= 000004	DT.SWR= 000056	KSWR 000016RG	PRI5 = 000240	STATBI= 064757
CDERCT= 000146	DT.SYP= 000072	KS.MG1 000000R	PRI6 = 000300	STAT1 = 000027
CDWDCT= 000144	DT.WBU= 000050	KS.MG2 000005R	PRI7 = 000340	SUSPND= 000001
CKTIM = 100000	DT.WHL= 000054	KTERRO= 000040	PRO = 000000	SVRO = 000062
CLKPRE= 000001	DT.WLL= 000052	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CM.ARG= ***** G	DVID1 = 000014	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CONFIG= 000056	ECCMEM= 000100	KTXTND= 040000	PR6 = 000300	SVR3 = 000070

SVR4 = 000072	UIPDR5= 177612	\$F\$BLA= 000170	\$IFLEV= 177777	\$S\$BYTE= 000000
SVR5 = 000074	UIPDR6= 177614	\$F\$CAS= 000150	\$ISK0 = 000001	\$S\$CASE= 000000
SVR6 = 000076	UIPDR7= 177616	\$F\$DEC= 000220	\$ISK1 = 000001	\$S\$DST = 000000
SYSCNT= 000052	WASADR= 000104	\$F\$DD = 000340	\$ISK2 = 000001	\$S\$ELOC= 000402
SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$S\$ERFL= 000000
TMPIO = 000002	WBUFEA= 000136	\$F\$GDO= 000400	\$LSTIN= 000001	\$S\$FLAG= 000001
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$LSTTA= 000001	\$S\$FROM= 000000
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	\$NESTL= 177777	\$S\$LOC = 000126R
UIPAR1= 177642	WBUFSZ= 000142	\$F\$LDO= 000200	\$NSK0 = 000300	\$S\$LOCN= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK1 = 000110	\$S\$REG = 177777
UIPAR3= 177646	WDT0 = 000114	\$F\$ND = 000403	\$NSK2 = 000110	\$S\$RETU= 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	\$NSK3 = 000110	\$S\$RTN1= 050000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	\$SAVLE= 177777	\$S\$RTN2= 050001
UIPAR6= 177654	XFLAG = 000005	\$F\$RTN= 000300	\$TAGLE= 177777	\$S\$SRC = 000000
UIPAR7= 177656	XOFF = 000023	\$F\$SEL= 000140	\$TAGNU= 050007	\$S\$TGSV= 000000
UIPDR0= 177600	XON = 000021	\$F\$THE= 000330	\$TEMP = 000300	\$S\$TGS1= 000000
UIPDR1= 177602	\$BGNLE= 177777	\$F\$TRU= 000404	\$TSK0 = 050002	\$S\$TGS2= 000000
UIPDR2= 177604	\$ERFLG= 000400	\$F\$UNT= 000130	\$TSK1 = 050004	\$S\$TO = 000000
UIPDR3= 177606	\$F\$AND= 000310	\$F\$WHI= 000120	\$TSK2 = 050006	\$S\$TAG= 050000
UIPDR4= 177610	\$F\$BAD= 000401	\$F\$YES= 000402	\$SARGC= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:05  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0676

3 COMMON EQUATE MODULE  
526 KKTON PROCESS KTON KEYBOARD COMMAND  
579 KKTOFF PROCESS KTOFF KEYBOARD COMMAND  
633 COMMON DEFINITIONS AND REFERENCES FOR KKTON  
636 000000' .PRINT ;SPMAC: VERSION 1.1  
667 KKTON ROUTINE

```
508 .TITLE KTONOF PROCESS KTON, KTOFF KEYBOARD COMMANDS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; KTONOF
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
517 ; 1. KKTON - TURN THE MEMORY MANAGEMENT HARDWARE ON
518 ; 2. KKTOFF - TURN THE MEMORY MANAGEMENT OFF
519 ;
520 ; VERSION:
521 ; 0.0
522 ;
523 ; EDIT DATE BY REASON
524 ;
```

```
526 .SBTTL KKTON PROCESS KTON KEYBOARD COMMAND
527 .IDENT /V0.0/
528
529 ;++
530 ; MODULE NAME:
531 ; KKTON
532 ;
533 ; FUNCTIONAL DESCRIPTION:
534 ; THIS ROUTINE PROCESSES THE "KTON" KEYBOARD COMMAND.
535 ; IT WILL TEST TO SEE IF KT IS AVAILABLE, AND THAT THE
536 ; EXERCISER IS NOT IN RUN MODE. IT WILL SET THE KT STATUS
537 ; BIT IN STATUS WORD 0 AND ENABLE THE KT HARDWARE.
538 ; IF IN RUN MODE OR NO KT AVAILABLE, THE APPROPRIATE
539 ; ERROR MESSAGE IS STUFFED INTO THE DATA TABLE'S ADDRESS
540 ; OF KB CMD RESPONSE.
541 ;
542 ; INPUTS:
543 ; 1. ADDRESS OF DATA TABLE
544 ; 2. COMMAND DECODE BUFFER POINTER
545 ;
546 ; IMPLICIT INPUTS:
547 ; DT.CFO
548 ; DT.STO
549 ;
550 ; OUTPUTS:
551 ; NONE
552 ;
553 ; IMPLICIT OUTPUTS:
554 ; DT.STO
555 ; DT.KBRSP
556 ;
557 ; PATHOLOGICAL CONNECTIONS:
558 ; CM.ARG
559 ; CM.RUN
560 ;
561 ; SUBORDINATE ROUTINES CALLED:
562 ; ARGCHK - CHECK FOR EXISTENCE OF AN ARGUMENT
563 ;
564 ; FUNCTIONAL SIDE EFFECTS:
565 ; NONE
566 ;
567 ; CALLING SEQUENCE:
568 ; CALL KKTON IN <DTADR,BUFPTR>
569 ; WHERE DTADR = ADDRESS OF DATA TABLE
570 ; BUFPTR = COMMAND DECODE BUFFER POINTER
571 ;
572 ; VERSION:
573 ; 0.0
574 ;
575 ; EDIT DATE BY REASON
576 ;---
577
```

```
579 .SBTTL KKTOFF PROCESS KTOFF KEYBOARD COMMAND
580 .IDENT /V0.0/
581
582 ;++
583 ; MODULE NAME:
584 ; KKTOFF
585 ;
586 ; FUNCTIONAL DESCRIPTION:
587 ; THIS ROUTINE WILL PROCESS THE KTOFF KEYBOARD COMMAND. IT
588 ; WILL TEST FOR THE PRESENCE OF KT. IF NO KT AVAILABLE, IT
589 ; WILL STUFF A NO KT MESSAGE. IF KT IS AVAILABLE, A CHECK
590 ; IS MADE TO SEE IF THE EXERCISER IS IN RUN MODE. IF IN RUN
591 ; MODE, THE ERROR MESSAGE IS STUFFED. IF NOT IN RUN MODE,
592 ; THE STATUS WORD WILL BE UPDATED(THE KT BIT CLEARED)
593 ; AND THE KT TURNED OFF. NOTE: THE MESSAGES ARE STUFFED
594 ; INTO THE DATA TABLE'S ADDRESS OF KB CMD RESPONSE
595 ; LOCATION.
596 ;
597 ;
598 ; INPUTS:
599 ; ADDRESS OF DATA TABLE
600 ; COMMAND DECODE BUFFER POINTER
601 ;
602 ; IMPLICIT INPUTS:
603 ; DT.STO
604 ; DT.CFO
605 ;
606 ; OUTPUTS:
607 ; NONE
608 ;
609 ; IMPLICIT OUTPUTS:
610 ; DT.STO
611 ; DT.KBRSP
612 ;
613 ; PATHOLOGICAL CONNECTIONS:
614 ; NONE
615 ;
616 ; SUBORDINATE ROUTINES CALLED:
617 ; ARGCHK
618 ;
619 ; FUNCTIONAL SIDE EFFECTS:
620 ; NONE
621 ;
622 ; CALLING SEQUENCE:
623 ; CALL KKTOFF IN <DTADR,CMDBUF>
624 ; WHERE DTADR = ADDRESS OF DATA TABLE
625 ; CMDBUF = COMMAND DECODE BUFFER POINTER
626 ;
627 ; VERSION:
628 ; 0.0
629 ;
630 ; EDIT DATE BY REASON
631 ;
```

```

633          .SBTTL  COMMON DEFINITIONS AND REFERENCES FOR KKTON
634
635          .MCALL  STRUCT
636 000000'   STRUCT
(1) 000000'   .PRINT  ;SPMAC: VERSION 1.1
637          000001   $LSTIN=1
638          000001   $LSTTAG=1
639
640
641          ;*****
642          ;
643          ; REFERENCED BY OTHER MODULES:
644          ;
645          .GLOBL  KKTOFF          ;MODULE ENTRY POINT
646          .GLOBL  KKTON          ;MODULE ENTRY POINT
647          ;
648          ;*****
649          ;
650          ; GLOBAL REFERENCES:
651          ;
652          .GLOBL  ARGCHK          ;CHECK FOR EXISTENCE OF AN ARGUMENT
653          .GLOBL  CM.ARG          ;'ILLEGAL OR INVALID ARG' MESSAGE
654          .GLOBL  CM.RUN          ;"NOT A VALID COMMAND IN RUN MODE" MESSAGE
655          ;
656          ;*****
657          ;
658          ; LOCAL STORAGE:
659          ;
660 000000' 047516 045440 022524  KT.NOKT:      .ASCIZ  /NO KT%/
000006'      000
661 000007'      113 020124 047117  KT.ON:      .ASCIZ  /KT ON%/
000014' 000045
662 000016' 052113 047440 043106  KT.OFF:      .ASCIZ  /KT OFF%/
000024' 000045
663 000026'      000          KT.FLG: .BYTE  0          ;KT ON/OFF FLAG
664          000030'          .EVEN
665
    
```

```

667          .SBTTL KKTON  ROUTINE
668
669 000030'          ROUTINE KKTON <DTADR,BUFPTR>
(2) 000030'
670
671          ;+
672          ; SET UP THE ON FLAG
673          ; -
674 000030'          LET KT.FLG :B= #0
(4) 000030' 105067 177772          CLRB  KT.FLG
675 000034'          INLINE <BR KTROU>
(2) 000034' 000404          BR  KTROU
676 000036'          ENDRTN
(3) 000036'
(3) 000036'
(2) 000036' 000207          50000$:
50001$:
RTS  PC
677
678 000040'          ROUTINE KKTOFF <DTADR,BUFPTR>
(2) 000040'
679
680          ;+
681          ; SET UP THE OFF FLAG
682          ; -
683 000040'          LET KT.FLG :B= #1
(4) 000040' 112767 000001 177760          MOVB  #1,KT.FLG
684
685          ;+
686          ; SAVE REGISTERS AND DATA TABLE ADDRESS
687          ; -
688
689 000046'          INLINE <KTROU:>
(2) 000046'
690 000046'          PUSH R0,R1
(2) 000046' 010046          MOV   R0,-(SP)
(3) 000050' 010146          MOV   R1,-(SP)
691 000052'          LET R0 := DTADR(R5)
(4) 000052' 016500 000000          MOV   DTADR(R5),R0
692 000056'          LET R1 := BUFPTR(R5)
(4) 000056' 016501 000002          MCV  BUFPTR(R5),R1
693
694          ;+
695          ; IF NOT IN RUNMODE THEN CONTINUE
696          ; -
697
698 000062'          IF #RUNMODE NOTSETIN DT.STO(R0) THEN
(6) 000062' 032760 100000 000010          BIT  #RUNMODE,DT.STO(
(9) 000070' 001071          BNE  50002$
699
700          ;+
701          ; SEE IF ANY ARGUMENTS EXIST & IF
702          ; THEY DO , THIS IS A BAD COMMAND
703          ; -
704
705 000072'          CALL ARGCHK IN <R1> OUT <R1>
(4) 000072' 162705 000002          SUB  #1*2,R5
(3) 000076' 010546          MOV  R5,-(SP)
    
```



```
(4) 000100' 010145
(3) 000102' 004767 000000G
(3) 000106' 012605
(4) 000110' 012501
706
707
708 ;+
709 ; THE ARGUMENT IS A CR SO ALL IS D.K.
710 ; SEE IF KT IS PRESENT AND IF IT IS,
711 ; IF KT FLAG IS 0 THEN SET THE KTSTAT BIT IN DT.STO OTHERWISE CLEAR IT
712 ;-
713 000112' IF.ERROR THEN
(6) 000112' 103054 BCC 50003$
714 000114' IF #KTPRES SETIN DT.CFO(R0) THEN
(6) 000114' 032760 000400 000014 BIT #KTPRES,DT.CFO(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.STO(R0) := DT.STO(R0) SET.BY #KTSTAT
(6) 000132' 052760 000020 000010 BIS #KTSTAT,DT.STO(R
717 000140' LET @#SR0 := #1
(4) 000140' 012737 000001 177572 MOV #1,@#SR0
718 000146' IF #ADDR22 SETIN DT.CFO(R0) THEN
(6) 000146' 032760 001000 000014 BIT #ADDR22,DT.CFO(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$:
727 000174' LET DT.STO(R0) := DT.STO(R0) CLR.BY #KTSTAT
(6) 000174' 042760 000020 000010 BIC #KTSTAT,DT.STO(R
728 000202' LET @#SR0 := #0
(4) 000202' 005037 177572 CLR @#SR0
729 000206' IF #ADDR22 SETIN DT.CFO(R0) THEN
(6) 000206' 032760 001000 000014 BIT #ADDR22,DT.CFO(R
(9) 000214' 001403 BEQ 50010$
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'
738 000232'                               ELSE
(4) 000232' 000403
(3) 000234'
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
745 000242'                               ENDIF
(4) 000242'
746
747 000242'                               ELSE
(4) 000242' 000403
(3) 000244'
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
754 000252'                               ENDIF
(4) 000252'
755 000252'                               ELSE
(4) 000252' 000403
(3) 000254'
756
757 ;+
758 ; LOAD THE RUN MODE MESSAGE
759 ; -
760
761 000254'                               LET DT.KBRSP(R0) := #CM.RUN
(4) 000254' 012760 000000G 000022
762 000262'                               ENDIF
(4) 000262'
763
764 ;+
765 ; CLEAN UP
766 ; -
767
768 000262'                               POP R1,R0
(2) 000262' 012601
(3) 000264' 012600
769
770 000266'                               ENDRTN
(3) 000266'
(3) 000266'
(2) 000266' 000207
771
772
773 000001                               .END
```

50007\$:  
BR 50011\$  
50004\$:  
MOV #KT.NOKT,DT.KBRS  
50011\$:  
BR 50012\$  
50003\$:  
MOV #CM.ARG,DT.KBRSP  
50012\$:  
BR 50013\$  
50002\$:  
MOV #CM.RUN,DT.KBRSP  
50013\$:  
MOV (SP)+,R1  
MOV (SP)+,R0  
50000\$:  
50001\$:  
RTS PC

ACSR = 000102	CQOVF = 000001	ECCSTA= 000010	KT.FLG 000026R	PR4 = 000200
ACTBIT= 004000	CR = 000015	ENBEDP= 010000	KT.NOK 000000R	PR5 = 000240
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	KT.OFF 000016R	PR6 = 000300
ADR = 000006	CSRC = 000102	ENDLST= 000000	KT.ON 000007R	PR7 = 000340
APTFER= 000004	CTRLC = 000003	EOPBIT= 000001	LF = 000012	PS = 177776
APTPRE= 000200	CTRLD = 000017	ERRTYP= 000106	LPSTAT= 000001	PSW = 177776
ARGCHK= ***** G	CTRLU = 000025	EVNTBE= 000200	MAPSTA= 000200	RANNUM= 000054
ASB = 000106	DCEVNT= 000011	EVNTHD= 000200	MED = 076600	RBUFEA= 000130
ASSEMB= 000010	DEFRTN= 000400	EVNTKT= 000203	MEMPAS= 040000	RBUFPA= 000126
ASTAT = 000104	DIAGMC= 000000	EVNTPE= 000202	MODEXH= 004000	RBUFSZ= 000132
AUTO = 000010	DROPMO= 100000	EVNTRE= 000201	MODHOL= 002000	RBUFVA= 000124
AUTOST= 020000	DSEVNT= 000014	FATERR= 100000	MODSEL= 001000	RDSERV= 000101
AWAS = 000110	DTADR = 000000	HRDCNT= 000044	MSGCKD= 000010	RJWHMI= 000022
BIT0 = 000001	DT.ADD= 000042	HRDPAS= 000050	MSGCKS= 000011	RELERR= 000020
BIT00 = 000001	DT.AP = 000100	ICONT = 000036	MSGDER= 000005	RELMOD= 020000
BIT01 = 000002	DT.APK= 000076	ICOUNT= 000040	MSGDRP= 000017	RELTIM= 010000
BIT02 = 000004	DT.BLS= 000034	IDNUM = 000122	MSGECH= 177777	RES1 = 000056
BIT03 = 000010	DT.CFO= 000014	IE = 000100	MSGEOP= 000013	RES2 = 000060
BIT04 = 000020	DT.CF1= 000016	INDPAR= 000040	MSGHDR= 000004	RICHAR= 031060
BIT05 = 000040	DT.ERR= 000020	INHDRP= 040000	MSGHNG= 000022	RPTDAT= 002000
BIT06 = 000100	DT.ESI= 000044	INHPR= 020000	MSGHRD= 000007	RSTRT = 000112
BIT07 = 000200	DT.EVN= 000000	INHREL= 001000	MSGMAP= 000021	RUBOUT= 000177
BIT08 = 000400	DT.EXS= 000060	INHRE= 000400	MSGNUL= 177775	RUNMOD= 100000
BIT09 = 001000	DT.FCH= 000037	INIT = 000030	MSGPOP= 000002	RVALU= 001740
BIT1 = 000002	DT.FCN= 000036	INTR = 000120	MSGPRM= 177776	SAM = 075464
BIT10 = 002000	DT.HMX= 000104	IOMOD = 100000	MSGRES= 000001	SBADR = 000102
BIT11 = 004000	DT.KBE= 000024	IOMODP= 102000	MSGSFT= 000006	SBKMOD= 000000
BIT12 = 010000	DT.KBP= 000026	IOMODR= 112000	MSGSKE= 000003	SBKSEL= 010000
BIT13 = 020000	DT.KBR= 000022	IOMODX= 110000	MSGSMB= 000015	SC.ADR= 000006
BIT14 = 040000	DT.KBU= 000030	JACK = 035060	MSGSMH= 000014	SC.ALC= 000014
BIT15 = 100000	DT.MLS= 000032	KIPAR0= 172340	MSGSMS= 000016	SC.APC= 000016
BIT2 = 000004	DT.MTI= 000110	KIPAR1= 172342	MSGSTD= 000000	SC.CKL= 000002
BIT3 = 000010	DT.DFF= 000070	KIPAR2= 172344	MSGSYS= 000012	SC.CKP= 000004
BIT4 = 000020	DT.PAS= 000074	KIPAR3= 172346	MSGVEC= 000020	SC.CLD= 000000
BIT5 = 000040	DT.PC = 000002	KIPAR4= 172350	NBKMOD= 001000	SC.HLD= 000010
BIT6 = 000100	DT.PFL= 000062	KIPAR5= 172352	NCPUOP= 000020	SC.SCA= 000012
BIT7 = 000200	DT.PSW= 000004	KIPAR6= 172354	NOPTY= 000002	SENDLS= 177777
BIT8 = 000400	DT.PTA= 000064	KIPAR7= 172356	NULL = 000000	SOFCNT= 000042
BIT9 = 001000	DT.RCS= 000102	KIPDR0= 172300	OWEN = 024020	SQFPAS= 000046
BKDEF = 000002	DT.REL= 000040	KIPDR1= 172302	PAERR = 000010	SPACE = 000040
BKMOD = 000020	DT.SCT= 000066	KIPDR2= 172304	PARPRE= 002000	SPOINT= 000032
BKMODE= 040000	DT.SMX= 000106	KIPDR3= 172306	PARSTA= 000100	SPVALU= 002200
BKSLSH= 000134	DT.SP = 000006	KIPDR4= 172310	PASCNT= 000034	SRO = 177572
BUFPtr= 000002	DT.SSI= 000046	KIPDR5= 172312	PDPLSI= 020000	SR1 = 177574
CAPRES= 000004	DT.ST0= 000010	KIPDR6= 172314	PDP60 = 004000	SR2 = 177576
CASTAT= 000004	DT.ST1= 000012	KIPDR7= 172316	PDP70 = 010000	SR3 = 172516
CDERCT= 000146	DT.SWR= 000056	KKTOFF 000040RG	PRI0 = 000000	STAT = 000026
CDWDCT= 000144	DT.SYP= 000072	KKTON 000030RG	PRI1 = 000040	STATBI= 064757
CKTIM = 100000	DT.WBU= 000050	KTERRO= 000040	PRI4 = 000200	STAT1 = 000027
CLKPRE= 000001	DT.WHL= 000054	KTPRES= 000400	PRI5 = 000240	SUSPND= 000001
CM.ARG= ***** G	DT.WLL= 000052	KTROU 000046R	PRI6 = 000300	SVR0 = 000062
CM.RUN= ***** G	DVID1 = 000014	KTSTAT= 000020	PRI7 = 000340	SVR1 = 000064
CONFIG= 000056	ECCMEM= 000100	KTXTND= 040000	PRO = 000000	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\$GDO= 000400	\$LSTIN= 000001	\$\$ELOC= 000402
SVR6 = 000076	WBSTAT= 000040	\$F\$IF = 000110	SLSTTA= 000001	\$\$ERFL= 000000
SYSCNT= 000052	WBUFEA= 000136	\$F\$INC= 000210	\$NESTL= 177777	\$\$FLAG= 000001
YSERR= 000100	WBUFPA= 000134	\$F\$LDO= 000200	SNSK0 = 000300	\$\$FROM= 000001
TMPI0 = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	SNSK1 = 000110	\$\$LOC = 000214R
TQOVF = 000002	WBUFSZ= 000142	\$F\$NO = 000403	SNSK2 = 000110	\$\$LOCN= 000000
UIPAR0= 177640	WDFR = 000116	\$F\$OR = 000320	SNSK3 = 000110	\$\$REG = 177777
UIPAR1= 177642	WDTO = 000114	\$F\$RTI= 000350	SNSK4 = 000110	\$\$RETU= 000000
UIPAR2= 177644	WTINRE= 000352	\$F\$RTN= 000300	SNSK5 = 000110	\$\$RTN1= 050000
UIPAR3= 177646	WTWHMI= 000222	\$F\$SEL= 000140	SSAVLE= 177777	\$\$RTN2= 050001
UIPAR4= 177650	XFLAG = 000005	\$F\$THE= 000330	STAGLE= 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	\$\$STAG= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:06  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0686

3 COMMON EQUATE MODULE  
529 COMMON DEFINITIONS AND REFERENCES FOR LPONOF  
532 000000' .PRINT ;SPMAC: VERSION 1.1  
562 KLPON PROCESS THE 'LPON' KEYBOARD COMMAND  
613 KLPOFF PROCESS THE 'ROTOF' KEYBOARD COMMAND  
663 KLPON ROUTINE

```
508 .TITLE LPONOF PROCESS THE 'LPON' AND 'LPOFF' KEYBOARD COMMANDS
509 .IDENT /V0.0/
510
511
512 ;++
513 ; MODULE PACKAGE NAME:
514 ; LPONOF
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ; THIS ROUTINE CONSISTS OF TWO MODULES:
518 ; 1. KL PON - PROCESS THE KEYBOARD COMMAND 'LPON'
519 ; 2. KLPOFF - PROCESS THE KEYBOARD COMMAND 'LPOFF'
520 ;
521 ; VERSION:
522 ; 0.0
523 ;
524 ; EDIT BY DATE REASON
525 ;--
526
527
```

```
529 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR LPONOF
530
531 .MCALL STRUCT
532 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
533 000001 $LSTIN=1
534 000001 $LSTTAG = 1
535
536 ;*****
537 ;
538 ; REFERENCED BY OTHER MODULES:
539 ;
540 .GLOBL KLPON ;'LPON' MODULE ENTRY POINT
541 .GLOBL KLPOFF ;'LPOFF' MODULE ENTRY POINT
542 ;
543 ;*****
544 ;
545 ; GLOBAL REFERENCES:
546 ;
547 .GLOBL CM.ARG ;ILLEGAL ARGUMENT MESSAGE
548 .GLOBL ARGCHK ;CHECK AN ARGUMENT
549 .GLOBL LPINT ;LP INTERRUPT ROUTINE
550 ;
551 ;*****
552 ;
553 ; LOCAL STORAGE:
554 ;
555 000000' 050114 047440 022516 LP.MGN: .ASCIZ /LP ON%/
000006' 000
556 000007' 114 020120 043117 LP.MGF: .ASCIZ /LP OFF%/
000014' 022506 000
557 000017' 000 LP.FLG: .BYTE 0 ;ON/OFF FLAG BYTE
558 .EVEN
559
560
```

```
562 .SBTTL KLPON PROCESS THE 'LPON' KEYBOARD COMMAND
563 .IDENT /V0.0/
564
565 ;++
566 ; MODULE NAME:
567 ; KLPON
568 ;
569 ; FUNCTIONAL DESCRIPTION:
570 ; THIS ROUTINE PROCESSES THE 'LPON' KEYBOARD ROUTINE.
571 ; IT WILL FIRST VERIFY THAT NO ARGUMENT APPEARS
572 ; IN THE COMMAND DECODE BUFFER. IF AN ARGUMENT IS IN THE
573 ; BUFFER, AN ERROR MESSAGE IS ISSUED. OTHERWISE,
574 ; THE LP ON/OFF STATUS BIT IN THE STATUS
575 ; INDICATOR IS SET.
576 ;
577 ; INPUTS:
578 ; 1. ADDRESS OF DTABLE
579 ; 2. COMMAND DECODE BUFFER POINTER
580 ;
581 ; IMPLICIT INPUTS:
582 ; NONE
583 ;
584 ; OUTPUTS:
585 ; NONE
586 ;
587 ; IMPLICIT OUTPUTS:
588 ; DT.KBRSP
589 ; DT.STO
590 ;
591 ; PATHOLOGICAL CONNECTIONS:
592 ; LPINT ROUTINE, LOC. 200 AND 202
593 ; CM.ARG
594 ;
595 ; SUBORDINATE ROUTINES CALLED:
596 ; ARGCHK - CHECK FOR UNWANTED ARGUMENTS
597 ;
598 ; FUNCTIONAL SIDE EFFECTS:
599 ; NONE
600 ;
601 ; CALLING SEQUENCE:
602 ; CALL KLPON IN <DTADR,BUFPTR>
603 ; WHERE: DTADR = ADDRESS OF DTABLE
604 ; BUFPTR = COMMAND DECODE BUFFER POINTER
605 ;
606 ; VERSION:
607 ; 0.0
608 ;
609 ; EDIT DATE BY REASON
610 ;--
611
```



```
613 .SBTTL KLPOFF PROCESS THE 'ROTOF' KEYBOARD COMMAND
614 .IDENT /V0.0/
615
616 ;++
617 ; MODULE NAME:
618 ;     KLPOFF
619 ;
620 ; FUNCTIONAL DESCRIPTION:
621 ;     THIS ROUTINE PROCESSES THE 'LPOFF' KEYBOARD ROUTINE.
622 ;     IT WILL FIRST VERIFY THAT NO ARGUMENT APPEARS
623 ;     IN THE COMMAND DECODE BUFFER. IF AN ARGUMENT IS IN THE
624 ;     BUFFER, AN ERROR MESSAGE IS ISSUED. OTHERWISE,
625 ;     THE LP ON/OFF STATUS BIT IN THE STATUS
626 ;     INDICATOR IS CLEARED.
627 ;
628 ; INPUTS:
629 ;     1. ADDRESS OF DTABLE
630 ;     2. COMMAND DECODE BUFFER POINTER
631 ;     3. ADDRESS OF COMMAND RESPONSE BUFFER
632 ;
633 ; IMPLICIT INPUTS:
634 ;     NONE
635 ;
636 ; OUTPUTS:
637 ;     NONE
638 ;
639 ; IMPLICIT OUTPUTS:
640 ;     DT.KBRSP
641 ;     DT.STO
642 ;
643 ; PATHOLOGICAL CONNECTIONS:
644 ;     CM.ARG
645 ;
646 ; SUBORDINATE ROUTINES CALLED:
647 ;     ARGCHK - CHECK FOR UNWANTED ARGUMENTS
648 ;
649 ; FUNCTIONAL SIDE EFFECTS:
650 ;     NONE
651 ;
652 ; CALLING SEQUENCE:
653 ;     CALL KLPOFF IN <DTADR,BUFPTR>
654 ;     WHERE:  DTADR = ADDRESS OF DTABLE
655 ;            BUFPTR = COMMAND DECODE BUFFER POINTER
656 ;
657 ; VERSION:
658 ;     0.0
659 ;
660 ;     EDIT          DATE          BY          REASON
661 ;---
```

```

663          .SBTTL KLPON ROUTINE
664
665
666 000020'          ROUTINE KLPON <DTADR,BUFPTR>
667 (2) 000020'
668
669          ;+
670          ; SET ON FLAG
671          ; -
672 000020'          LET LP.FLG :B= #0
673 (4) 000020' 105067 177773
674 000024'          INLINE <BR LPROU>
675 (2) 000024' 000404
676 000026'          ENDRTN
677 (3) 000026'
678 (3) 000026'
679 (2) 000026' 000207
680
681          ROUTINE KLPOFF <DTADR,BUFPTR>
682 (2) 000030'
683
684          ;+
685          ; SET OFF FLAG
686          ; -
687 000030'          LET LP.FLG :B= #1
688 (4) 000030' 112767 000001 177761
689
690          ;+
691          ; INITIALIZE AND SAVE DTABLE ADDRESS AND DECODE BUFFER POINTER
692          ; -
693 000036'          INLINE <LPROU:>
694 (2) 000036'
695 000036'          PUSH R0,R1
696 (2) 000036' 010046
697 (3) 000040' 010146
698 000042'          LET R0 := DTADR(R5)
699 (4) 000042' 016500 000000
700 000046'          LET R1 := BUFPTR(R5)
701 (4) 000046' 016501 000002
702
703          ;+
704          ; MAKE SURE NO ADDITIONAL ARGUMENTS ARE IN DECODE BUFFER
705          ; -
706 000052'          CALL ARGCHK IN <R1> OUT <R1>
707 (4) 000052' 162705 000002
708 (3) 000056' 010546
709 (4) 000060' 010145
710 (3) 000062' 004767 000000G
711 (3) 000066' 012605
712 (4) 000070' 012501

```

KLPON:

CLRB LP.FLG  
 BR LPROU

50000S:  
 50001S:  
 RTS PC

KLPOFF:

MOVB #1,LP.FLG

LPROU:  
 MOV R0,-(SP)  
 MOV R1,-(SP)  
 MOV DTADR(R5),R0  
 MOV BUFPTR(R5),R1

SUB #1\*2,R5  
 MOV R5,-(SP)  
 MOV R1,-(R5)  
 JSR PC,ARGCHK  
 MOV (SP)+,R5  
 MOV (R5)+,R1

```

700
701
702          ;+
703          ; THERE IS NO <CR> YET SO IT'S A BAD COMMAND...STUFF ERROR
704          ; MESSAGE AND BEAT IT....
705          ; -
706
707          IF.NO.ERROR THEN
708          (6) 000072' 103404          BCS          50002$
709          (4) 000074' 012760 000000G 000022          MOV          #CM.ARG,DT.KBRSP
710          ELSE
711          (4) 000102' 000426          BR          50003$
712          (3) 000104'          50002$:
713          ;+
714          ; NOW DETERMINE IF LPON OR LPOFF COMMAND
715          ; -
716          IFB LP.FLG EQ #0 THEN
717          (6) 000104' 105767 177707          TSTB          LP.FLG
718          (9) 000110' 001015          BNE          50004$
719          ;+
720          ; SET THE LPSTAT BIT
721          ; IN DT.STO...AND LOAD UP INTERRUPT ROUTINE
722          ; -
723          LET DT.STO(R0) := DT.STO(R0) SET.BY #LPSTAT          BIS          #LPSTAT,DT.STO(R
724          LET @#200 := #LPINT          MOV          #LPINT,@#200
725          (4) 000120' 012737 000000G 000200          MOV          #LPINT,@#200
726          LET @#202 := #PRI1          MOV          #PRI1,@#202
727          (4) 000126' 012737 000040 000202          MOV          #PRI1,@#202
728          LET DT.KBRSP(R0) := #LP.MGN          MOV          #LP.MGN,DT.KBRSP
729          (4) 000134' 012760 000000' 000022          MOV          #LP.MGN,DT.KBRSP
730          ELSE
731          (4) 000142' 000406          BR          50005$
732          (3) 000144'          50004$:
733          ;+
734          ; CLEAR THE LPSTAT BIT IN DT.STO.....
735          ; -
736          LET DT.STO(R0) := DT.STO(R0) CLR.BY #LPSTAT          BIC          #LPSTAT,DT.STO(R
737          LET DT.KBRSP(R0) := #LP.MGF          MOV          #LP.MGF,DT.KBRSP
738          (4) 000152' 012760 000007' 000022          MOV          #LP.MGF,DT.KBRSP
739          ENDIF
740          (4) 000160'          50005$:
741          ENDIF
742          (4) 000160'          50003$:
743          ;+
744          ; CLEAN UP AND LEAVE
745          ; -

```

```
740 ;-
741
742 000160' POP R1,R0
(2) 000160' 012601 MOV (SP)+,R1
(3) 000162' 012600 MOV (SP)+,R0
743
744
745 000164' ENDRTN
(3) 000164'
(3) 000164' 50000S:
(2) 000164' 000207 50001S:
746 000001 .END RTS PC
```

SYMBOL TABLE

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

SVR4 = 000072	UIPDR5= 177612	\$FSBLA= 000170	\$IFLEV= 177777	\$SELOC= 000402
SVR5 = 000074	UIPDR6= 177614	\$FSCAS= 000150	\$ISK0 = 000001	\$\$ERFL= 000000
SVR6 = 000076	UIPDR7= 177616	\$FSDEC= 000220	\$ISK1 = 000001	\$\$FLAG= 000001
SYSCNT= 000052	WASADR= 000104	\$FSDD = 000340	\$LOCTA= 177777	\$\$FROM= 000001
YSERR= 000100	WBSTAT= 000040	\$FSFAL= 000405	\$LSTIN= 000001	\$\$LOC = 000110R
TMPIO = 000002	WBUFEA= 000136	\$FSGDO= 000400	\$LSTTA= 000001	\$\$LOCN= 000000
TQOVF = 000002	WBUFPA= 000134	\$FSIF = 000110	\$NESTL= 177777	\$\$REG = 177777
UIPAR0= 177640	WBUFRQ= 000140	\$FSINC= 000210	\$NSK0 = 000300	\$\$RETU= 000000
UIPAR1= 177642	WBFSZ = 000142	\$FL00= 000200	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDFR = 000116	\$FSNAM= 000160	\$NSK2 = 000110	\$\$RTN2= 050001
UIPAR3= 177646	WDT0 = 000114	\$FSNO = 000403	\$\$SAVLE= 177777	\$\$SRC = 000000
UIPAR4= 177650	WTINRE= 000352	\$FSOR = 000320	\$TAGLE= 177777	\$\$TGSV= 000000
UIPAR5= 177652	WTWHMI= 000222	\$FSRTI= 000350	\$TAGNU= 050006	\$\$TGS1= 000000
UIPAR6= 177654	XFLAG = 000005	\$FSRTN= 000300	STEMP = 000300	\$\$TGS2= 000000
UIPAR7= 177656	XOFF = 000023	\$FSSEL= 000140	\$TSK0 = 050003	\$\$TO = 000001
UIPDR0= 177600	XON = 000021	\$FSTHE= 000330	\$TSK1 = 050005	\$\$\$TAG= 050000
UIPDR1= 177602	\$BGNLE= 177777	\$FSTRU= 000404	\$\$ARGC= 000004	. = 000166R
UIPDR2= 177604	\$ERFLG= 000400	\$FSUNT= 000130	\$\$BYTE= 000402	
UIPDR3= 177606	\$FSAND= 000310	\$FSWHI= 000120	\$\$CASE= 000000	
UIPDR4= 177610	\$FSBAD= 000401	\$FSYES= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:07  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0696

3 COMMON EQUATE MODULE  
528 COMMON DEFINITIONS AND REFERENCES FOR LSTCHK  
531 000000' .PRINT ;SPMAC: VERSION 1.1  
553 CHKEOP CHECK THAT ALL MODULES HAVE DONE END OF PASS  
603 CHKEOP ROUTINE  
703 CHKACT CHECK FOR ACTIVE MODULES  
747 CHKACT ROUTINE

```
508 .TITLE LSTCHK CHECK MODULES TO DETERMINE IF ALL HAVE MADE END OF PASS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; LSTCHK
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE PACKAGE CONTAINS TWO ROUTINES:
517 ; 1. CHKEOP - CHECK FOR END-OF-PASS
518 ; 2. CHKACT - CHECK ALL MODULES INACTIVE
519 ;
520 ; VERSION:
521 ; 0.0
522 ;
523 ; EDIT DATE BY REASON
524 ;--
525
526
```



```
528          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR LSTCHK
529
530          .MCALL STRUCT
531          STRUCT
532          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
533          000001 $LSTTAG=1
534          000001 $LSTIN=1
535          ;*****
536          ;
537          ; REFERENCED BY OTHER MODULES:
538          ;
539          .GLOBL CHKEOP          ;MODULE ENTRY POINT
540          .GLOBL CHKACT          ;MCDULE ENTRY POINT
541          ;
542          ;*****
543          ;
544          ; GLOBAL REFERENCES:
545          ;
546          .GLOBL SAVREG
547          .GLOBL RESREG
548          ;
549          ;*****
550
551
```

```
553 .SBTTL CHKEOP CHECK THAT ALL MODULES HAVE DONE END OF PASS
554 .IDENT /V0.0/
555
556 ;++
557 ; MODULE NAME:
558 ; CHKEOP
559 ;
560 ; FUNCTIONAL DESCRIPTION:
561 ; THIS MODULE IS CALLED TO CHECK END OF PASS CONDITION
562 ; OF ALL MODULES. IF ALL MODULES HAVE MADE END OF PASS
563 ; AT THIS RELOCATION, A RETURN NO ERROR OCCURS. IF
564 ; ONLY BACKGROUND MODULES ARE CONFIGURED, ONLY
565 ; ONE MUST COMPLETE END OF PASS FOR THE RETURN NO ERROR
566 ; CONDITION
567 ;
568 ; INPUTS:
569 ; DTABLE
570 ;
571 ; IMPLICIT INPUTS:
572 ; DT.MLST
573 ; DT.BLST
574 ;
575 ; OUTPUTS:
576 ; NONE
577 ;
578 ; IMPLICIT OUTPUTS:
579 ; NONE
580 ;
581 ; PATHOLOGICAL CONNECTIONS:
582 ; NONE
583 ;
584 ; SUBORDINATE ROUTINES CALLED:
585 ; SAVREG
586 ; RESREG
587 ;
588 ; FUNCTIONAL SIDE EFFECTS:
589 ; NONE
590 ;
591 ; CALLING SEQUENCE:
592 ; CALL CHKEOP IN <DTADR>
593 ; WHERE DTADR IS DATA TABLE ADDRESS
594 ;
595 ; VERSION:
596 ; 0.0
597 ;
598 ; EDIT DATE BY REASON
599 ;--
600
601
```

```

603          .SBTTL  CHKEOP ROUTINE
604
605 000000'          ROUTINE CHKEOP <DTADR>
606          (2) 000000'
607
608          ;+
609          ; SAVE REGISTERS AND GET DTABLE ADDR AND MODULE LIST
610          ; -
611 000000'          CALL SAVREG
612          (3) 000000' 004767 000000G          JSR      PC,SAVREG
613          000004'          LET R0 := DTADR(R5)          MOV      DTADR(R5),R0
614          (4) 000004' 016500 000000          MOV      DT.MLST(R0),R1
615          000010'          LET R1 := DT.MLST(R0)
616          (4) 000010' 016001 000032
617
618          ;+
619          ; INIT COUNTERS
620          ; -
621          LET R3 := #0          CLR      R3
622          (4) 000014' 005003
623          LET R4 := #0          CLR      R4
624          (4) 000016' 005004
625
626          ;+
627          ; WHILE NOT AT END OF LIST AND NOT AT END OF IOMOD LIST,
628          ; GET A MODULE HEADER ADDRESS. IF THE MODULE IS SELECTED AND NOT
629          ; DROPPED, INCREMENT COUNT IN R4. IF, IN ADDITION, IT IS
630          ; ACTIVE, INCREMENT R3 COUNT. IF ACTIVE, CHECK EOP BIT - IF SET
631          ; CONTINUE TO NEXT MODULE. IF NOT SET - RETURN ERROR - CAN'T RELOCATE.
632          ; -
633          WHILE (R1) NE #ENDLST AND R1 NE DT.BLST(R0) DO          50002$:
634          (4) 000020'          CMP      (R1),#ENDLST
635          (6) 000020' 021127 000000          BEQ      50003$
636          (9) 000024' 001430          CMP      R1,DT.BLST(R0)
637          (6) 000026' 020160 000034          BEQ      50003$
638          (9) 000032' 001425          LET R2 := (R1)+          MOV      (R1)+,R2
639          000034'          IF #BIT14 SETIN STAT(R2) AND #BIT13 NOTSETIN STAT(R2) THEN          BIT      #BIT14,STAT(R2)
640          (4) 000034' 012102          BIT      #BIT14,STAT(R2)          BEQ      50004$
641          000036'          BIT      #BIT13,STAT(R2)          BNE      50004$
642          (6) 000036' 032762 040000 000026          INC      R4
643          (9) 000044' 001417          IF #ACTBIT SETIN STAT(R2) THEN          BIT      #ACTBIT,STAT(R2)
644          (6) 000046' 032762 020000 000026          BEQ      50005$
645          (9) 000054' 001013          LET R4 := R4 + #1          INC      R4
646          000056'          IF #ACTBIT SETIN STAT(R2) THEN          BIT      #ACTBIT,STAT(R2)
647          (4) 000056' 005204          BEQ      50005$
648          000060'          LET R3 := R3 + #1          INC      R3
649          (6) 000060' 032762 004000 000026          IFB #EOPBIT NOTSETIN XFLAG(R2) THEN          BITB     #EOPBIT,XFLAG(R2)
650          (9) 000066' 001406          BEQ      50006$
651          000070'          LET R3 := R3 + #1          INC      R3
652          (6) 000070' 005203          IFB #EOPBIT NOTSETIN XFLAG(R2) THEN          BITB     #EOPBIT,XFLAG(R2)
653          000072'          BEQ      50006$
654          (6) 000072' 132762 000001 000005          BITB     #EOPBIT,XFLAG(R2)
655          (9) 000100' 001001          BNE      50006$

```



```

666 ; THERE WAS AT LEAST ONE IOMOD ACTIVE....SINCE WE GOT THIS
667 ; FAR, THE EOP BIT MUST HAVE BEEN SET IN ALL- SO RELOCATE.....
668 ;--
669
670 000166' CALL RESREG
(3) 000166' 004767 000000G JSR PC,RESREG
671 000172' RETURN NO.ERROR
(4) 000172' 000413 BR 50000$
672 000174' ENDIF
(4) 000174' 50014$:
673
674
675 ;+
676 ; IF WE EVER GET HERE.... !!!!!!! ..... THE SITUATION COULD
677 ; BE THIS: THE NUMBER OF ACTIVE IOMODS IS ZERO.
678 ; SO WE TRIED TO USE BACKGROUND EOP BITS AS RELOCATION CRITERIA AND
679 ; COULDN'T - IF THERE ARE NO BACKGROUND MODULES SELECTED AND NOT DROPPED,
680 ; (OR MAYBE NONE AT ALL (R3 WOULD BE ZERO))THERE ARE IOMODS THAT
681 ; ARE NOT CURRENTLY ACTIVE BUT COULD RUN SOMEPLACE (R4 NE #0)
682 ; THEN WE MAY AS WELL RELOCATE OVER AND OVER AGAIN IN HOPES OF FINDING
683 ; A PLACE WHERE THEY CAN RUN.....
684 ;--
685
686 000174' IF R3 EQ #0 AND R4 NE #0 THEN
(6) 000174' 005703 TST R3
(9) 000176' 001005 BNE 50015$
(6) 000200' 005704 TST R4
(9) 000202' 001403 BEQ 50015$
687 000204' CALL RESREG
(3) 000204' 004767 000000G JSR PC,RESREG
688 000210' RETURN NO.ERROR
(4) 000210' 000404 BR 50000$
689 000212' ENDIF
(4) 000212' 50015$:
690
691 ;+
692 ; IF WE GET HERE - WE SIMPLY ARE WAITING FOR A BACKGROUND TO EOP
693 ;--
694
695 000212' INLINE <1$:>
(2) 000212' 1$:
696 000212' CALL RESREG
(3) 000212' 004767 000000G JSR PC,RESREG
697 000216' RETURN ERROR
(2) 000216' 000261 SEC
(4) 000220' 000401 BR 50001$
698
699 000222' ENDRTN
(3) 000222' 50000$:
(2) 000222' 000241 CLC
(3) 000224' 50001$:
(2) 000224' 000207 RTS PC
700
701

```

```
703 .SBTTL CHKACT CHECK FOR ACTIVE MODULES
704 .IDENT /VO.0/
705
706 ;++
707 ; MODULE NAME:
708 ;     CHKACT
709 ;
710 ; FUNCTIONAL DESCRIPTION:
711 ;     THIS ROUTINE SIMPLY CHECKS ALL OPTION MODULES TO
712 ;     SEE IF THEY ARE ACTIVE
713 ;
714 ; INPUTS:
715 ;     ADDRESS OF DTABLE
716 ;
717 ; IMPLICIT INPUTS:
718 ;     DT.MLST
719 ;
720 ; OUTPUTS:
721 ;     NONE
722 ;
723 ; IMPLICIT OUTPUTS:
724 ;     NONE
725 ;
726 ; PATHOLOGICAL CONNECTIONS:
727 ;     NONE
728 ;
729 ; SUBORDINATE ROUTINES CALLED:
730 ;     1. SAVREG
731 ;     2. RESREG
732 ;
733 ; FUNCTIONAL SIDE EFFECTS:
734 ;     NONE
735 ;
736 ; CALLING SEQUENCE:
737 ;     CALL CHKACT IN <DT>
738 ;     WHERE DT = DTABLE ADDRESS
739 ;
740 ; VERSION:
741 ;     C.C
742 ;
743 ;     EDIT             DATE             BY             REASON
744 ;--
745
```

```

747          .SBTTL  CHKACT ROUTINE
748
749 000226'          ROUTINE CHKACT <DTADR>
750          (2) 000226'          CHKACT:
751
752          ;+
753          ; SAVE REGISTERS AND GET DTABLE ADDR. AND ADDR. OF MODULE LIST
754          ; -
755
756 000226'          CALL SAVREG
757 (3) 000226' 004767 000000G          JSR      PC, SAVREG
758 000232'          LET R0 := DTADR(R5)
759 (4) 000232' 016500 000000          MOV      DTADR(R5), R0
760 000236'          LET R1 := DT.MLST(R0)
761 (4) 000236' 016001 000032          MOV      DT.MLST(R0), R1
762
763          ;+
764          ; SEARCH LIST OF MODULES UNTIL END OR UNTIL AN ACTIVE
765          ; BIT IS SET IN STAT WORD
766          ; -
767
768 000242'          WHILE (R1) NE #ENDLST DO
769 (4) 000242'          50002$:
770 (6) 000242' 021127 000000          CMP      (R1), #ENDLST
771 (9) 000246' 001412          BEQ      50003$
772          LET R2 := (R1)+
773 (4) 000250' 012102          MOV      (R1)+, R2
774 000252'          IF #ACTBIT SETIN STAT(R2) THEN
775 (6) 000252' 032762 004000 000026          BIT      #ACTBIT, STAT(R2)
776 (9) 000260' 001404          BEQ      50004$
777          CALL RESREG
778 (3) 000262' 004767 000000G          JSR      PC, RESREG
779          RETURN ERROR
780 (2) 000266' 000261          SEC
781 (4) 000270' 000405          BR      50001$
782          ENDIF
783 (4) 000272'          50004$:
784 000272'          ENDDG
785 (4) 000272' 000763          BR      50002$
786 (3) 000274'          50003$:
787
788          ;+
789          ; WE WENT ALL THE WAY THRU .... GUESS NONE ACTIVE
790          ; -
791
792 000274'          CALL RESREG
793 (3) 000274' 004767 000000G          JSR      PC, RESREG
794 000300'          RETURN NO.ERROR
795 (4) 000300' 000400          BR      50000$
796
797          ENDRTN
798 (3) 000302'          50000$:
799 (2) 000302' 000241          CLC
800 (3) 000304'          50001$:
801 (2) 000304' 000207          RTS      PC

```

LSTCHK CHECK MODULES TO DETERMINE IF ALL HAVE MADE END OF PASS MACY11 30A(1052) 20-SEP-78 18:07 PAGE 19-8  
LSTCHK.MAC 28-JUL-78 09:22 CHKACT ROUTINE

SEQ 0705

781  
782           000001           .END



ACSR = 000102	CSRA = 000100	ENBNUL= 000001	MODEXH= 004000	RSUFVSZ= 000132
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	MODHOL= 002000	RBUFVA= 000124
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RDSERV= 000101
ADR = 000006	CTRLD = 000017	ERRTYP= 000106	MSGCKD= 000010	RDWHMI= 000022
APTFER= 000004	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RELERR= 000020
APTPRE= 000200	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RELMOD= 020000
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELTIM= 010000
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RESREG= ***** G
ASTAT = 000104	DROPMO= 100000	EVNTRE= 000201	MSGEOP= 000013	RES1 = 000056
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
AUTOST= 020000	DTADR = 000000	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AWAS = 000110	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
BIT0 = 000001	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
BIT00 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUBOUT= 000177
BIT01 = 000002	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT02 = 000004	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	RVALU= 001740
BIT03 = 000010	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT04 = 000020	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SAVREG= ***** G
BIT05 = 000040	DT.ESI= 000044	INHEPR= 020000	MSGSKE= 000003	SBADR = 000102
BIT06 = 000100	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBKMOD= 000000
BIT07 = 000200	DT.EXS= 000060	INHRR= 000400	MSGSMH= 000014	SBKSEL= 010000
BIT08 = 000400	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.ADR= 000006
BIT09 = 001000	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.ALC= 000014
BIT1 = 000002	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.APC= 000016
BIT10 = 002000	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CKL= 000002
BIT11 = 004000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CKP= 000004
BIT12 = 010000	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.CLO= 000000
BIT13 = 020000	DT.KBU= 000030	JACK = 035060	NDAPTY= 000002	SC.HLD= 000010
BIT14 = 040000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT15 = 100000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SENDLS= 177777
BIT2 = 000004	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOFCNT= 000042
BIT3 = 000010	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SOFPAS= 000048
BIT4 = 000020	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPACE = 000040
BIT5 = 000040	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPOINT= 000032
BIT6 = 000100	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SPVALU= 002200
BIT7 = 000200	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR0 = 177572
BIT8 = 000400	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR1 = 177574
BIT9 = 001000	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	SR2 = 177576
BKDEF = 000002	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	SR3 = 172516
BKMOD = 000020	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STAT = 000026
BKMODE= 040000	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	STATBI= 064757
BKSLSH= 000134	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	STAT1 = 000027
CAPRES= 000004	DT.ST0= 000010	KIPDR6= 172314	PRI7 = 000340	SUSPND= 000001
CASAT= 000004	DT.ST1= 000012	KIPDR7= 172316	PRO = 000000	SVR0 = 000062
CDERCT= 000146	DT.SWR= 000056	KTERR0= 000040	PR4 = 000200	SVR1 = 000064
CDWDCT= 000144	DT.SYP= 000072	KTPRES= 000400	PR5 = 000240	SVR2 = 000066
CHKACT 000226RG	DT.WBU= 000050	KTSTAT= 000020	PR6 = 000300	SVR3 = 000070
CHKEOP 000000RG	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
CONFIG= 000056	ECCMEM= 000100	MAPSTA= 000200	RANNUM= 000054	SYSCNT= 000052
CQOVF = 000001	ECCSTA= 000010	MED = 076600	RBUFEA= 000130	YSERR= 000100
CR = 000015	ENBEDP= 010000	MEMPAS= 040000	RBUFPA= 000126	TMPIO = 000002

TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$LSTTA= 000001	\$\$DST = 000000
UIPAR0= 177640	WBUFQR= 000140	\$F\$INC= 000210	\$NESTL= 177777	\$\$ELOC= 000403
UIPAR1= 177642	WBUFSS= 000142	\$F\$L00= 000200	\$NSK0 = 000300	\$\$SERFL= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK1 = 000120	\$\$FLAG= 000001
UIPAR3= 177646	WDT0 = 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	\$\$AVLE= 177777	\$\$REG = 177777
UIPAR7= 177656	XOFF = 000023	\$F\$SEL= 000140	\$\$SK0 = 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	\$\$T0 = 000000
UIPDR7= 177616	\$F\$DEC= 000220	\$ISK1 = 000001	\$TSK4 = 050013	\$\$STAG= 050000
WASADR= 000104	\$F\$DO = 000340	\$ISK2 = 000001	\$\$ARGC= 000002	. = 000306R
WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$BYTE= 000403	
WBUFEA= 000136	\$F\$G00= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:08  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0708

3 COMMON EQUATE MODULE  
563 COMMON DEFINITIONS AND REFERENCES  
569 000000' .PRINT ;SPMAC: VERSION 1.1  
618 PROCESS NEXT MESSAGE  
722 PERFORM CLEAN-UP FOR THE LAST MESSAGE  
770 RESTORE THE INTERRUPTED MODULE'S R5 BEFORE RETURNING

```
508 .TITLE MSGDEQ - DE-QUEUE NEXT MESSAGE
509 .IDENT /V0.0/
510
511
512
513 ;++
514 ; MODULE NAME:
515 ; MSGDEQ
516 ;
517 ; FUNCTIONAL DESCRIPTION:
518 ; THIS MODULE CONTAINS THE TWO ROUTINES MSGDEQ AND MSGDRET.
519 ; THE MSGDEQ ROUTINE DE-QUEUES THE NEXT MESSAGE TO BE PROCESSED,
520 ; WHICH CONSISTS OF PERFORMING THE FOLLOWING:
521 ; 1. DEQUEUEING NEXT MESSAGE
522 ; 2. ENABLING/INHIBITTING PRINTING OF MESSAGE BASED
523 ; ON THE SOFTWARE SWITCH REGISTER SETTING
524 ; 3. HOOKING MESSAGE TO APPROPRIATE OUTPUT DRIVER
525 ;
526 ; THE MSGDRET ROUTINE PERFORMS THE NECESSARY CLEAN-UP ASSOCIATED
527 ; WITH THE PROCESSING OF THE LAST MESSAGE. THIS CONSISTS OF:
528 ; 1. ACTIVATING REQUESTOR MODULE, I.E., THE MODULE THAT
529 ; JUST HAD ITS MESSAGE PROCESSED
530 ; 2. RESETTING THE CONTROL-O FLAG
531 ;
532 ; INPUTS:
533 ; 1. ADDRESS OF DATA TABLE
534 ;
535 ; IMPLICIT INPUTS:
536 ; DT.SWR
537 ;
538 ; OUTPUTS:
539 ; NONE
540 ;
541 ; IMPLICIT OUTPUTS:
542 ; NONE
543 ;
544 ; PATHOLOGICAL CONNECTIONS:
545 ; 1. CTRLDF ;CONTROL-O FLAG
546 ; 2. DX.KFL ;CARRIAGE RETURN FLAG
547 ;
548 ; SUBORDINATE MODULES CALLED:
549 ; 1. ENQCQ MAKE ENTRY IN CONTROL QUEUE
550 ; 2. KBINI INITIALIZE KEYBOARD HANDLER ROUTINE
551 ; 3. MSGDGET GET NEXT MESSAGE MODULE
552 ; 4. MSGDHOOK HOOK MESSAGE TO OUTPUT DRIVER
553 ;
554 ; CALLING SEQUENCE:
555 ; CALL MSGDEQ IN <DTABLE>
556 ;
557 ; VERSION:
558 ; 0.0
559 ;
560 ; EDIT BY DATE REASON
561 ;--
```

```

563          .SBTTL COMMON DEFINITIONS AND REFERENCES
564
565
566
567
568          .MCALL STRUCT
569          STRUCT
570          (1) 000000' .PRINT ;SPMAC: VERSION 1.1
571          000001 $LSTIN=1
572          000001 $LSTTAG=1
573
574
575
576          ;
577          ;*****
578          ;
579          ; REFERENCED BY OTHER MODULES
580          ;
581          .GLOBL MSGDEQ          ;MODULE'S ENTRY POINT
582          .GLOBL MD.BSY         ;BUSY FLAG
583          .GLOBL MD.COD         ;MESSAGE TYPE INDICATOR
584          ;
585          ;*****
586          ;
587          ; GLOBAL REFERENCES
588          ;
589          .GLOBL ENQCQ           ;CONTROL QUEUE EN-QUEING MODULE
590          .GLOBL BA.STAT
591          .GLOBL KBINI           ;KB I/O INITIATOR MODULE
592          .GLOBL DX.KFL         ;CARRIAGE RETURN FLAG
593          .GLOBL MSGDGET        ;MESSAGE DE-QUEING MODULE
594          .GLOBL MSGDHOCK       ;OUTPUT MESSAGE TO DRIVER MODULE
595          .GLOBL CTRLQF         ;CONTROL-O FLAG
596          ;
597          ;*****
598          ;
599          ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
600          ;
601          000000' 000000 MD.COD: .WORD 0          ;MESSAGE TYPE CODE
602          000002' 000000 MD.MSG: .WORD 0          ;ADDRESS OF MESSAGE
603          000004' 000000 MD.HDR: .WORD 0          ;ADDRESS OF MODULE'S HEADER
604          000006' 000000 MD.RET: .WORD 0          ;ADDRESS AT WHICH MODULE IS TO BE RESUMED
605          000010' 000000 MD.DTBL: .WORD 0         ;ADDRESS OF DATA TABLE
606          000012' 000000 MD.BSY: .WORD 0          ;MESSAGE-QUEUE BUSY INDICATOR
607          000005          .REPT ^D<5>          ;SAVE 5 WORDS FOR R5 STACK
608          .WORD 0
609          .ENDR
610          MD.R5:                ;LOCAL R5 STACK
611          000026' 000050 MD.KBECH: .BLKW ^D40      ;KEYBOARD ECHO BUFFER
612          000146' 001 000 MD.NUL: .BYTE 1,0      ;NULL CHAR MESSAGE
613          ;
614          ;*****
615          .EVEN
616          ;

```

```

618 .SBTTL PROCESS NEXT MESSAGE
619
620 000150' ROUTINE MSGDEQ <DTABLE>
(2) 000150' MSGDEQ:
621
622 ;+
623 ; MAKE SURE THE MESSAGE QUEUE MECHANISM IS NOT BUSY. IF
624 ; IT IS, SIMPLY RETURN TO THE CALLER
625 ;-
626
627 000150' IF MD.BSY EQ #1 THEN
(6) 000150' 026727 177636 000001 CMP MD.BSY,#1
(9) 000156' 001001 BNE 50002$
628 000160' RETURN BR 50000$
(4) 000160' 000561
629 000162' ENDIF
(4) 000162' 50002$:
630
631 ;+
632 ; SAVE REGISTERS, THEN GET THE DATA TABLE ADDRESS AND SAVE IT IN
633 ; LOCAL STORAGE.
634 ;-
635
636 000162' PUSH R0,R1
(2) 000162' 010046 MOV R0,-(SP)
(3) 000164' 010146 MOV R1,-(SP)
637 000166' LET R0 := DTABLE(R5)
(4) 000166' 016500 000000 MOV DTABLE(R5),R0
638 000172' LET MD.DTBL := R0
(4) 000172' 010067 177612 MOV R0,MD.DTBL
639
640 ;+
641 ; GET THE NEXT MESSAGE TO BE PRINTED
642 ;-
643
644 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
645
646 ;+
647 ; IF THIS IS THE NULL CHARACTER MESSAGE AND IF OUTPUTTING THE
648 ; NULL IS ENABLED, THEN HOOK UP THE MESSAGE. OTHERWISE,
649 ; JUST RETURN. IN EITHER CASE, DON'T BOTHER MAKING ANY OF THE OTHER TESTS.
650 ;-
651
652 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'                                ENDF
(4) 000272'                                50005$:
658 000272'                                ENDF
(4) 000272'                                50003$:
659
660
661
662 ;+
663 ; IF THIS IS AN ERROR MESSAGE OR A MESSAGE COMING DIRECTLY
664 ; FROM A MODULE, AND IF THE PRINTING OF THESE TYPES OF MESSAGES
665 ; IS INHIBITED, DON'T PRINT THE MESSAGE. INSTEAD, JUST ENTER THE MODULE IN
666 ; THE CONTROL QUEUE AND RETURN.
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'                                ENDF
(4) 000356'                                50010$:
673 000356'                                ENDF
(4) 000356'                                50007$:
674
675
676 ;+
677 ; IF THIS IS AN 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 #ENBEOPT NOTSETIN DT.SWR(R0) THEN
(6) 000356' 026727 177416 000013
(9) 000364' 001030
(6) 000366' 032760 010000 000056
(9) 000374' 001024
685 000376' LET R1 := MD.HDR
(4) 000376' 016701 177402
686 000402' IF #BKMOD SETIN STAT(R1) THEN
(6) 000402' 032761 000020 000026
(9) 000410' 001404
687 000412' LET BA.STAT := BA.STAT CLR.BY #BKDEF
(6) 000412' 042767 000002 000000G
688 000420' ELSE
(4) 000420' 000411
(3) 000422'
689 000422' CALL ENQCQ IN <R0,MD.HDR,MD.RET>
(3) 000422' 010546
(6) 000424' 016745 177356
(5) 000430' 016745 177350
(4) 000434' 010045
(3) 000436' 004767 000000G
(3) 000442' 012605
690 000444' ENDF
(4) 000444'
691 000444' INLINE <BR 2$>
(2) 000444' 000425
692 000446' ENDF
(4) 000446'
693
694
695 ; +
696 ; SET THE MESSAGE-QUEUE BUSY FLAG.
697 ; -
698
699 000446' INLINE <1$:>
(2) 000446'
700 000446' LET MD.BSY := #1
(4) 000446' 012767 000001 177336
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
(9) 000462' 001014
709 000464' CALL MSGDHOOK IN <R0,MD.COD,MD.MSG,#MSGDRET>
(3) 000464' 010546
(7) 000466' 012745 000526'
(6) 000472' 016745 177304
(5) 000476' 016745 177276
(4) 000502' 010045

```

```

CMP MD.COD,#MSGEOP
BNE 50011$
BIT #ENBEOPT,DT.SWR(R
BNE 50011$
MOV MD.HDR,R1
BIT #BKMOD,STAT(R1)
BEQ 50012$
BIC #BKDEF,BA.STAT
BR 50013$
50012$:
MOV R5,-(SP)
MOV MD.RET,-(R5)
MOV MD.HDR,-(R5)
MOV R0,-(R5)
JSR PC,ENQCQ
MOV (SP)+,R5
50013$:
BR 2$
50011$:
1$:
MOV #1,MD.BSY
BIT #NOAPTY,DT.CF0(R
BNE 50014$
MOV R5,-(SP)
MOV #MSGDRET,-(R5)
MOV MD.MSG,-(R5)
MOV MD.COD,-(R5)
MOV R0,-(R5)

```



(3) 000504' 004767 000000G  
(3) 000510' 012605  
710 000512'  
(4) 000512' 000402  
(3) 000514'  
711 000514'  
(3) 000514' 004767 000006  
712 000520'  
(4) 000520'  
713  
714  
715  
716  
717  
718 000520'  
(2) 000520'  
719 000520'  
(2) 000520' 012601  
(3) 000522' 012600  
720 000524'  
(3) 000524'  
(3) 000524'  
(2) 000524' 000207

ELSE  
  
CALL MSGDRET  
  
ENDIF  
  
;+  
; RETURN TO CALLER  
;-  
  
INLINE <2\$:>  
  
POP R1,R0  
  
ENDRTN

JSR PC,MSGDHOOK  
MOV (SP)+,R5  
  
BR 50015\$  
50014\$:  
  
JSR PC,MSGDRET  
50015\$:  
  
2\$:  
  
MOV (SP)+,R1  
MOV (SP)+,R0  
  
50000\$:  
50001\$:  
RTS PC

```

722          .SBTTL PERFORM CLEAN-UP FOR THE LAST MESSAGE
723
724
725          ;+
726          ; SET THE MESSAGE QUEUE AS BEING NOT BUSY, AND RESET THE
727          ; CONTROL-O FLAG
728          ; -
729
730          MSGDRET: LET MD.BSY := #0
731          (4) 000526' 005067 177260          CLR      MD.BSY
732          (4) 000532' 005067 000000G          LET CTRLQF := #0
733          (4) 000532' 005067 000000G          CLR      CTRLQF
734
735          ;+
736          ; SWITCH TO A LOCAL R5 STACK
737          ; -
738          PUSH R5,R0,R1
739          (2) 000536' 010546          MOV      R5,-(SP)
740          (3) 000540' 010046          MOV      R0,-(SP)
741          (4) 000542' 010146          MOV      R1,-(SP)
742          (4) 000544' 016701 177240          LET R1 := MD.DTBL
743          (4) 000544' 016701 177240          MOV      MD.DTBL,R1
744          (4) 000550' 012705 000026'          LET R5 := #MD.R5S
745          (4) 000550' 012705 000026'          MOV      #MD.R5S,R5
746
747          ;+
748          ; DETERMINE IF THE LAST MESSAGE WAS ISSUED BY AN OPTION MODULE
749          ; OR MONITOR (I.E. MD.HDR IS NOT 0).
750          ; AND IF IT WAS, RESUME ITS EXECUTION BY MAKING AN ENTRY FOR IT
751          ; IN THE CONTROL QUEUE
752          ; -
753          IF MD.HDR NE #0 THEN
754          (6) 000554' 005767 177224          TST      MD.HDR
755          (9) 000560' 001422          BEQ      50016$
756          (6) 000562' 026727 177212 000000          IF MD.COD EQ #MSGSTD OR MD.COD GE #MSGSKEL THEN
757          (8) 000570' 001404          CMP      MD.COD,#MSGSTD
758          (6) 000572' 026727 177202 000003          BEQ      50017$
759          (9) 000600' 002412          CMP      MD.COD,#MSGSKEL
760          (6) 000602'          BLT      50020$
761          (6) 000602'          50017$:
762          (3) 000602' 010546          CALL ENQCQ IN <MD.DTBL,MD.HDR,MD.RET>
763          (6) 000604' 016745 177176          MOV      R5,-(SP)
764          (5) 000610' 016745 177170          MOV      MD.RET,-(R5)
765          (4) 000614' 016745 177170          MOV      MD.HDR,-(R5)
766          (3) 000620' 004767 000000G          MCV      MD.DTBL,-(R5)
767          (3) 000624' 012605          JSR      PC,ENQCQ
768          (3) 000624' 012605          MOV      (SP)+,R5
769          (4) 000626'          ENDIF
770          (4) 000626'          50020$:
771          (4) 000626'          ENDIF          50016$:
772          (4) 000626'
773
774          ;+
775

```

MSGDEQ - DE-QUEUE NEXT MESSAGE MACY11 30A(1052) 20-SEP-78 18:08 PAGE 22-1  
MSGDEQ.MAC 28-JUL-78 09:22 PERFORM CLEAN-UP FOR THE LAST MESSAGE

SEQ 0716

```
756 ; IF THE MESSAGE WAS ONE OF THE KEYBOARD PROMPTS, THEN UNLOCK THE
757 ; KEYBOARD AND ALLOW THE NEXT COMMAND TO BE ENTERED
758 ;-
759
760 000626' IF MD.COD EQ #MSGPRM THEN
(6) 000626' 026727 177146 177776 CMP MD.COD,#MSGPRM
(9) 000634' 001006 BNE 50021$
761 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
762 000652' ENDIF
(4) 000652' 50021$:
763
764
765
```

767  
 768  
 769  
 770  
 771  
 772  
 773  
 774  
 775  
 776  
 777  
 778  
 779  
 780  
 781  
 782  
 (6)  
 (9)  
 783  
 (2)  
 (3)  
 (4)  
 784  
 (2)  
 785  
 (4)  
 (3)  
 786  
 (2)  
 (3)  
 (4)  
 787  
 (2)  
 788  
 (4)  
 789  
 790

```

.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'
000652' 032761 000002 000014
000660' 001005
000662'
      POP R1,R0,R5
000662' 012601
000664' 012600
000666' 012605
      INLINE <RTI>
000670'
(2) 000670' 000002
      ELSE
000672'
(4) 000672' 000404
000674'
      POP R1,R0,R5
000674'
(2) 000674' 012601
000676' 012600
000700' 012605
      INLINE <RTS      PC>
(2) 000702' 000207
      ENDIF
000704'
(4) 000704'
000001
      .END
  
```

```

      BIT      #NOAPTY,DT.CF0(R
      BNE      50022$
      MOV      (SP)+,R1
      MOV      (SP)+,R0
      MOV      (SP)+,R5
      RTI
      BR       50023$
50022$:
      MOV      (SP)+,R1
      MOV      (SP)+,R0
      MOV      (SP)+,R5
      RTS      PC
50023$:
  
```

ACSR = 000102	CSRC = 000102	ENSEOP= 010000	MAPSTA= 000200	PRIC = 000000
ACTBIT= 004000	CTRLC = 000003	ENBNUL= 000001	MD.BSY 000012RG	PRI1 = 000040
ADDR22= 001000	CTRLD = 000017	ENDLST= 000000	MD.COD 000000RG	PRI4 = 000200
ADR = 000006	CTRLOF= ***** G	ENQCQ = ***** G	MD.DTB 000010R	PRI5 = 000240
APTFER= 000004	CTRLU = 000025	EOPBIT= 000001	MD.HDR 000004R	PRI6 = 0C0300
APTPRE= 000200	DCEVNT= 000011	ERRTYP= 000106	MD.KBE 000026R	PRI7 = 000340
ASB = 000106	DEFRTN= 000400	EVNTBE= 000200	MD.MSG 000002R	PRO = 000000
ASSEMB= 000010	DIAGMC= 000000	EVNTHD= 000200	MD.NUL 000146R	PR4 = 000200
ASTAT = 000104	DROPMO= 100000	EVNTKT= 000203	MD.RET 000006R	PR5 = 000240
AUTO = 000010	DSEVNT= 000014	EVNTPE= 000202	MD.RSS 000026R	PR6 = 000300
AUTOST= 020000	DTABLE= 000000	EVNTRE= 000201	MED = 076600	PR7 = 000340
AWAS = 000110	DT.ADD= 000042	FATERR= 100000	MEMPAS= 040000	PS = 177776
BA.STA= ***** G	DT.AP = 000100	HRDCNT= 000044	MODEXH= 004000	PSW = 177776
BIT0 = 000001	DT.APK= 000076	HRDPAS= 000050	MODHOL= 002000	RANNUM= 000054
BIT00 = 000001	DT.BLS= 000034	ICONT = 000036	MODSEL= 001000	RBUFEA= 000130
BIT01 = 000002	DT.CF0= 000014	ICOUNT= 000040	MSGCKD= 000010	RBUFPA= 000126
BIT02 = 000004	DT.CF1= 000016	IDNUM = 000122	MSGCKS= 000011	RBUFSZ= 000132
BIT03 = 000010	DT.ERR= 000020	IE = 000100	MSGDEQ 000150RG	RBUFVA= 000124
BIT04 = 000020	DT.ESI= 000044	INDPAR= 000040	MSGDER= 000005	RDSERV= 000101
BIT05 = 000040	DT.EVN= 000000	INHDRP= 040000	MSGDGE= ***** G	RDWHMI= 000022
BIT06 = 000100	DT.EXS= 000060	INHEPR= 020000	MSGDHO= ***** G	RELERR= 000020
BIT07 = 000200	DT.FCH= 000037	INHREL= 001000	MSGDRE 000526R	RELMOD= 020000
BIT08 = 000400	DT.FCN= 000036	INHRR= 000400	MSGDRP= 000017	RELTIM= 010000
BIT09 = 001000	DT.HMX= 000104	INIT = 000030	MSGECH= 177777	RES1 = 000056
BIT1 = 000002	DT.KBE= 000024	INTR = 000120	MSGEOP= 000013	RES2 = 000060
BIT10 = 002000	DT.KBP= 000026	IOMOD = 100000	MSGHDR= 000004	RICHAR= 031060
BIT11 = 004000	DT.KBR= 000022	IOMODP= 102000	MSGHNG= 000022	RPTDAT= 002000
BIT12 = 010000	DT.KBU= 000030	IOMODR= 112000	MSGHRD= 000007	RSTRT = 000112
BIT13 = 020000	DT.MLS= 000032	IOMODX= 110000	MSGMAP= 000021	RUBOUT= 000177
BIT14 = 040000	DT.MTI= 000110	JACK = 035060	MSGNUL= 177775	RUNMOD= 100000
BIT15 = 100000	DT.OFF= 000070	KBINI = ***** G	MSGPOP= 000002	RVALU = 001740
BIT2 = 000004	DT.PAS= 000074	KIPAR0= 172340	MSGPRM= 177776	SAM = 075464
BIT3 = 000010	DT.PC = 000002	KIPAR1= 172342	MSGRES= 000001	SBADR = 000102
BIT4 = 000020	DT.PFL= 000062	KIPAR2= 172344	MSGSFT= 000006	SBKMOD= 000000
BIT5 = 000040	DT.PSW= 000004	KIPAR3= 172346	MSGSKE= 000003	SBKSEL= 010000
BIT6 = 000100	DT.PTA= 000064	KIPAR4= 172350	MSGSMB= 000015	SC.ADR= 000006
BIT7 = 000200	DT.RCS= 000102	KIPAR5= 172352	MSGSMH= 000014	SC.ALC= 000014
BIT8 = 000400	DT.REL= 000040	KIPAR6= 172354	MSGSMS= 000016	SC.APC= 000016
BIT9 = 001000	DT.SCT= 000066	KIPAR7= 172356	MSGSTD= 000000	SC.CKL= 000002
BKDEF = 000002	DT.SMX= 000106	KIPDR0= 172300	MSGSYS= 000012	SC.CKP= 000004
BKMOD = 000020	DT.SP = 000006	KIPDR1= 172302	MSGVEC= 000020	SC.CLO= 000000
BKMODE= 040000	DT.SSI= 000046	KIPDR2= 172304	NBKMOD= 001000	SC.HLD= 000010
BKSLSH= 000134	DT.ST0= 000010	KIPDR3= 172306	NCPUOP= 000020	SC.SCA= 000012
CAPRES= 000004	DT.ST1= 000012	KIPDR4= 172310	NOPTY= 000002	SENDLS= 177777
CASTAT= 000004	DT.SWR= 000056	KIPDR5= 172312	NULL = 000000	SOFCNT= 000042
CDERCT= 000146	DT.SYP= 000072	KIPDR6= 172314	OWEN = 024020	SOFPAS= 000046
CDWDCT= 000144	DT.WBU= 000050	KIPDR7= 172316	PAERR = 000010	SPACE = 000040
CKTIM = 100000	DT.WHL= 000054	KTERR0= 000040	PARPRE= 002000	SPOINT= 000032
CLKPRE= 000001	DT.WLL= 000052	KTPRES= 000400	PARSTA= 000100	SPVALU= 002200
CONFIG= 000056	DVID1 = 000014	KTSTAT= 000020	PASCNT= 000034	SRO = 177572
CQOVF = 000001	DX.KFL= ***** G	KTXTND= 040000	PDPLSI= 020000	SR1 = 177574
CR = 000015	ECCMEM= 000100	LF = 000012	PDP60 = 004000	SR2 = 177576
CSRA = 000100	ECCSTA= 000010	LPSTAT= 000001	PDP70 = 010000	SR3 = 172516

STAT = 000026	UIPAR6= 177654	XOFF = 000023	\$FSTHE= 000330	\$\$ARGC= 000002
STATBI= 064757	UIPAR7= 177656	XON = 000021	\$FSTRU= 000404	\$\$BYTE= 000403
STAT1 = 000027	UIPDR0= 177600	\$BGNLE= 177777	\$FSUNT= 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\$DD = 000340	\$LSTIN= 000001	\$\$LOCN= 000000
SVR6 = 000076	WASADR= 000104	\$F\$FAL= 000405	\$LSTTA= 000001	\$\$REG = 177777
SYSCNT= 000052	WBSTAT= 000040	\$F\$GOD= 000400	\$NESTL= 177777	\$\$RETU= 000000
SYSERR= 000100	WBUFEA= 000136	\$F\$IF = 000110	\$NSKO = 000110	\$\$RTN1= 050000
TMPID = 000002	WBUFPA= 000134	\$F\$INC= 000210	\$NSK1 = 000110	\$\$RTN2= 050001
TQOVF = 000002	WBUFRQ= 000140	\$F\$LDD= 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	WDT0 = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:09  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0720

3 COMMON EQUATE MODULE  
570 COMMON DEFINITIONS AND REFERENCES  
574 000000' .PRINT ;SPMAC: VERSION 1.1  
602 DE-QUEUE NEXT MESSAGE

```
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
```



564  
565  
566  
567  
568

```
;  
; VERSION:      0.0  
;  
;      EDIT      BY      DATE      REASON  
;---
```

```
570 .SBTTL COMMON DEFINITIONS AND REFERENCES
571
572
573 .MCALL STRUCT
574 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
575 000001 $LSTIN = 1
576 000001 $LSTTAG = 1
577
578
579
580 ;
581 ;*****
582 ;
583 ; REFERENCED BY OTHER MODULES
584 ;
585 .GLOBL MSGDGET ;MODULE'S ENTRY POINT
586 ;
587 ;*****
588 ;
589 ; GLOBAL REFERENCES
590 ;
591 .GLOBL DEQTQ ;TYPE QUEUE DE-QUEUING MODULE
592 .GLOBL FILLMSG ;FILL IN SKELETAL MESSAGE MODULE
593 .GLOBL KB.INP ;KEYBOARD'S INPUT BUFFER POINTER
594 .GLOBL SAVREG ;SAVE REGISTER ROUTINE
595 .GLOBL RESREG ;RESTORE REGISTER ROUTINE
596 ;
597 ;*****
598 ;
599
600
```

```

602          .SBTTL DE-QUEUE NEXT MESSAGE
603
604 000000'          ROUTINE MSGDGET <DTABLE,KBECH,DFTMSG,MSGTYPE,MSGADDR,HDRADDR,RETADDR>
(2) 000000'          MSGDGET:
605
606          ;+
607          ; SAVE REGISTERS AND GET DTABLE ADDRESS AND KEYBOARD ECHO POINTER
608          ; -
609
610 000000'          CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC,SAVREG
611 000004'          LET R0 := DTABLE(R5)
(4) 000004' 016500 000000          MOV      DTABLE(R5),R0
612 000010'          LET R1 := KBECH(R5)
(4) 000010' 016501 000002          MOV      KBECH(R5),R1
613
614          ;+
615          ; SEE IF ANY OPERATOR INPUT IS AWAITING ECHOING, AND IF SO, ZERO THE KEYBOARD
616          ; ECHO BUFFER, COPY THE CHARACTERS TO BE ECHOED INTO IT AND USE THIS AS THE
617          ; FIRST MESSAGE
618          ; -
619
620 000014'          IF DT.KBECH(R0) LD KB.INP THEN
(6) 000014' 026067 000024 000000G          CMP      DT.KBECH(R0),KB.
(9) 000022' 103031          BHIS     50002$
621 000024'          LET R2 := #^D<40>
(4) 000024' 012702 000050          MOV      #^D<40>,R2
622 000030'          WHILE R2 GT #0 DO
(4) 000030'          50003$:
(6) 000030' 005702          TST      R2
(9) 000032' 003403          BLE      50004$
623 000034'          LET (R1)+ := #0
(4) 000034' 005021          CLR      (R1)+
624 000036'          LET R2 := R2 - #1
(6) 000036' 005302          DEC      R2
625 000040'          ENDDO
(4) 000040' 000773          BR      50003$
(3) 000042'          50004$:
626
627          ;+
628          ; LOAD UP ECHO BUFFER FROM INPUT BUFFER
629          ; -
630
631 000042'          LET R1 := KBECH(R5)
(4) 000042' 016501 000002          MOV      KBECH(R5),R1
632 000046'          WHILE DT.KBECH(R0) LT KB.INP DO
(4) 000046'          50005$:
(6) 000046' 026067 000024 000000G          CMP      DT.KBECH(R0),KB.
(9) 000054' 002005          BGE      50006$
633 000056'          LET (R1)+ :B= @DT.KBECH(R0)
(4) 000056' 117021 000024          MOVEB   @DT.KBECH(R0),(R
634 000062'          LET DT.KBECH(R0) := DT.KBECH(R0) + #1
(6) 000062' 005260 000024          INC      DT.KBECH(R0)
635 000066'          ENDDO
(4) 000066' 000767          SR      50005$
(3) 000070'          50006$:

```

```

636
637
638 ;+
639 ; SET UP OUTPUT ARGUMENTS FOR ECHO TYPE MESSAGE TYPE
640 ; -
641 000070' LET R1 := #MSGECH MOV #MSGECH,R1
(4) 000070' 012701 177777
642 000074' LET R2 := KBECH(R5) MOV KBECH(R5),R2
(4) 000074' 016502 000002
643 000100' LET R3 := #0 CLR R3
(4) 000100' 005003
644 000102' LET R4 := #0 CLR R4
(4) 000102' 005004
645 000104' ELSE BR 50007$
(4) 000104' 000466
(3) 000106' 50002$:
646
647 ;+
648 ; SEE IF THERE IS A KEYBOARD COMMAND RESPONSE WAITING TO BE OUTPUT,
649 ; IF SO, TAKE IT NEXT, AND THEN CLEAR KEYBOARD RESPONSE INDICATOR
650 ; -
651
652 000106' IF DT.KBRSP(R0) NE #0 THEN TST DT.KBRSP(R0)
(6) 000106' 005760 000022 BEQ 50010$
(9) 000112' 001411
653 000114' LET R1 := #MSGRES MOV #MSGRES,R1
(4) 000114' 012701 000001
654 000120' LET R2 := DT.KBRSP(R0) MOV DT.KBRSP(R0),R2
(4) 000120' 016002 000022
655 000124' LET R3 := #0 CLR R3
(4) 000124' 005003
656 000126' LET R4 := #0 CLR R4
(4) 000126' 005004
657 000130' LET DT.KBRSP(R0) := #0 CLR DT.KBRSP(R0)
(4) 000130' 005060 000022
658 000134' ELSE BR 50011$
(4) 000134' 000452
(3) 000136' 50010$:
659
660 ;+
661 ; SEE IF THERE IS A KEYBOARD PROMPT AWAITING OUTPUT, IF SO, TAKE
662 ; IT, AND CLEAR THE KEYBOARD PROMPT INDICATOR
663 ; -
664
665 000136' IF DT.KBPRM(R0) NE #0 THEN TST DT.KBPRM(R0)
(6) 000136' 005760 000026 BEQ 50012$
(9) 000142' 001411
666 000144' LET R1 := #MSGPRM MOV #MSGPRM,R1
(4) 000144' 012701 177776
667 000150' LET R2 := DT.KBPRM(R0) MOV DT.KBPRM(R0),R2
(4) 000150' 016002 000026
668 000154' LET R3 := #0 CLR R3
(4) 000154' 005003
669 000156' LET R4 := #0 CLR R4
(4) 000156' 005004
670 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'          50011S:
698 000262'          ENDIF
(4) 000262'          50007S:
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'          50000S:
(3) 000306'          50001S:
(2) 000306' 000207          RTS      PC
715
716                .END
000001
```

SYMBOL TABLE

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

SYMBOL TABLE

SVR2 = 000066	UIPDR5= 177612	\$F\$DEC= 000220	\$ISK3 = 000001	\$\$ARGC= 000016
SVR3 = 000070	UIPDR6= 177614	\$F\$DD = 000340	\$ISK4 = 000001	\$\$BYTE= 000403
SVR4 = 000072	UIPDR7= 177616	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$CASE= 000000
SVR5 = 000074	WASADR= 000104	\$F\$GDD= 000400	\$LSTIN= 000001	\$\$DST = 000000
SVR6 = 000076	WBSTAT= 000040	\$F\$IF = 000110	\$LSTTA= 000001	\$\$ELDC= 000402
SYSCNT= 000052	WBUFEA= 000136	\$F\$INC= 000210	\$NESTL= 177777	\$\$ERFL= 000000
SYSERR= 000100	WBUFPA= 000134	\$F\$LDD= 000200	\$NSKO = 000300	\$\$FLAG= 000001
TMPIO = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$FROM= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$ND = 000403	\$NSK2 = 000110	\$\$LOC = 000214R
UIPAR0= 177640	WDFR = 000116	\$F\$DR = 000320	\$NSK3 = 000110	\$\$LOCN= 000000
UIPAR1= 177642	WDT0 = 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	\$\$AVLE= 177777	\$\$RTN1= 050000
UIPAR4= 177650	XFLAG = 000005	\$F\$THE= 000330	\$\$SK0 = 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	\$\$TD = 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:11  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0730

3 COMMON EQUATE MODULE  
565 COMMON DEFINITIONS AND REFERENCES  
570 000000' .PRINT ;SPMAC: VERSION 1.1  
598 MSGDHOOK ROUTINE

```
508 .TITLE MSGDHOOK - HOOK MESSAGE TO OUTPUT DRIVER
509 .IDENT /V0.0/
510
511
512
513 ;++
514 ; MODULE NAME:
515 ; MSGDHOOK
516 ;
517 ; FUNCTIONAL DESCRIPTION:
518 ; THIS ROUTINE HOOKS THE ADDRESS OF THE NEXT MESSAGE
519 ; TO BE PRINTED TO THE APPROPRIATE OUTPUT DRIVER
520 ; AND THEN INITIATES THE PRINTING OF THE MESSAGE
521 ; BY CALLING THE DRIVER AT ITS I/O INITIATOR ENTRY POINT.
522 ;
523 ; INPUTS:
524 ; 1. ADDRESS OF DATA TABLE
525 ; 2. MESSAGE TYPE CODE
526 ; 3. ADDRESS OF MESSAGE
527 ; 4. RETURN ADDRESS
528 ;
529 ; IMPLICIT INPUTS:
530 ; 1. DT.STO
531 ; 2. DT.CFO
532 ;
533 ; OUTPUTS:
534 ; NONE
535 ;
536 ; IMPLICIT OUTPUTS:
537 ; NONE
538 ;
539 ; PATHOLOGICAL CONNECTIONS:
540 ; NONE
541 ;
542 ; SUBORDINATE ROUTINES CALLED:
543 ; TTINI
544 ; LPINI
545 ; SAVREG
546 ; RESREG
547 ;
548 ; FUNCTIONAL SIDE EFFECTS:
549 ; NONE
550 ;
551 ; CALLING SEQUENCE:
552 ; CALL MSGDHOOK IN <DTABLE,MSGCOD,MSGADDR,RETADR>
553 ; WHERE DTABLE = ADDRESS OF DATA TABLE
554 ; MSGCOD = MESSAGE TYPE CODE
555 ; MSGADDR = MESSAGE ADDRESS
556 ; RETADDR = RETURN ADDRESS
557 ;
558 ; VERSION:
559 ; 0.0
560 ;
561 ; EDIT BY DATE REASON
562 ;---
```

```
564  
565  
566  
567  
568  
569  
570 000000'  
(1) 000000'  
571 000001  
572 000001  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595 000000' 000000  
596
```

```
.SBTTL COMMON DEFINITIONS AND REFERENCES  
  
      .MCALL STRUCT  
      STRUCT  
.PRINT ;SPMAC: VERSION 1.1  
      $LSTIN = 1  
      $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:  
;  
MS.EXT: .WORD 0 ;EXIT THROUGH HERE IF NO TTY(UNDER APT)  
;AND ITS A MSGPOP
```

```

598 .SBTTL MSGDHOOK ROUTINE
599
600
601 000002' ROUTINE MSGDHOOK <DTABLE,MSGCODE,MSGADDR,RETADR>
(2) 000002' MSGDHOOK:
602
603
604 ;+
605 ; DO THE NECESSARY SET-UP STUFF
606 ; -
607
608 000002' CALL SAVREG
(3) 000002' 004767 000000G JSR PC, SAVREG
609 000006' LET R1 := MSGCODE(R5) MOV MSGCODE(R5), R1
(4) 000006' 016501 000002 MOV MSGADDR(R5), R2
610 000012' LET R2 := MSGADDR(R5) MOV MSGADDR(R5), R2
(4) 000012' 016502 000004 MOV RETADR(R5), R3
611 000016' LET R3 := RETADR(R5) MOV RETADR(R5), R3
(4) 000016' 016503 000006 MOV R3, MS.EXT
612 000022' LET MS.EXT := R3 MOV R3, MS.EXT
(4) 000022' 010367 177752 MOV DTABLE(R5), R0
613 000026' LET R0 := DTABLE(R5) MOV DTABLE(R5), R0
(4) 000026' 016500 000000
614
615
616
617
618 ;+
619 ; IF THERE IS NO TTY (UNDER APT) WE WILL BE EXITTING DIRECTLY (BELOW)
620 ; -
621
622 000032' IF #NOAPTY NOTSETIN DT.CF0(R0) THEN
(6) 000032' 032760 000002 000014 BIT #NOAPTY, DT.CF0(R
(9) 000040' 001026 BNE 50002$
623
624 ;+
625 ; IS THE "LPON" KEYBOARD COMMAND IN EFFECT? IF IT IS,
626 ; THIS MESSAGE IS TO GO TO THE LINE PRINTER
627 ; -
628 000042' IF #LPSTAT SETIN DT.ST0(R0) THEN
(6) 000042' 032760 000001 000010 BIT #LPSTAT, DT.ST0(R
(9) 000050' 001411 BEQ 50003$
629 000052' CALL LPINI IN <R0,R1,R2,R3>
(3) 000052' 010546 MOV R5, -(SP)
(7) 000054' 010345 MOV R3, -(R5)
(6) 000056' 010245 MOV R2, -(R5)
(5) 000060' 010145 MOV R1, -(R5)
(4) 000062' 010045 MOV R0, -(R5)
(3) 000064' 004767 000000G JSR PC, LPINI
(3) 000070' 012605 MOV (SP)+, R5
630 000072' ELSE BR 50004$
(4) 000072' 000410
(3) 000074' 50003$:
631
632 ;+
633 ; IF NO ONE ELSE WANTS IT, PASS IT TO THE TERMINAL OUTPUT DRIVER

```

```
634 ;-
635 000074'          CALL TTINI IN <R0,R1,R2,R3>
(3) 000074' 010546
(7) 000076' 010345
(6) 000100' 010245
(5) 000102' 010145
(4) 000104' 010045
(3) 000106' 004767 000000G
(3) 000112' 012605
636 000114'          ENDIF
(4) 000114'          50004$:
637 000114'          ELSE
(4) 000114' 000406
(3) 000116'          50002$:
638 ;+
639 ; IF WE ARE UNDER APT WITH NO TTY,
640 ; CLEAN UP IN AN UNNATURAL WAY (SIMULATE THE NORMAL RETURN
641 ; THAT IS, RESTORE REGS, DO ONE FINAL POP, WHICH KILLS
642 ; THE RTS PC. RESTORE R5 THEN JUMP THRU MS.EXT)
643 ;-
644 000116'          CALL RESREG
(3) 000116' 004767 000000G
645
646 000122'          INLINE <TST (SP)+>
(2) 000122' 005726
647 000124'          INLINE <MOV (SP)+,R5>
(2) 000124' 012605
648 000126'          INLINE <JMP @MS.EXT>
(2) 000126' 000177 177646
649 000132'          ENDIF
(4) 000132'          50005$:
650
651
652 ;+
653 ; CHCW BABY!!!!!! !!
654 ;-
655
656 000132'          CALL RESREG
(3) 000132' 004767 000000G
657 000136'          ENDRTN
(3) 000136'
(3) 000136'
(2) 000136' 000207
658
659
660 000001          .END
RTS PC
```

SYMBOL TABLE

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODHOL= 002000	RANNUM= 000054
ACTBIT= 004000	CTRLD = 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	DROPMD= 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	INHRE= 000400	MSGRES= 000001	RUNMOD= 100000
BIT06 = 000100	DT.FCH= 000037	INIT = 000030	MSGSFT= 000006	RSVALU= 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	SRO = 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
CASSTAT= 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	PRO = 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	\$FSCAS= 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\$GOD= 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	\$\$REU= 000000
UIPAR1= 177642	WBUFSZ= 000142	\$F\$LOD= 000200	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$RTN2= 050001
UIPAR3= 177646	WDTO = 000114	\$F\$NO = 000403	\$\$AVLE= 177777	\$\$SRC = 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	STAGLE= 177777	\$\$TGSV= 000000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	STAGNU= 050006	\$\$TGS1= 000000
UIPAR6= 177654	XFLAG = 000005	\$F\$RTN= 000300	STEMP = 000300	\$\$TGS2= 000000
UIPAR7= 177656	XOFF = 000023	\$F\$SEL= 000140	\$TSK0 = 050005	\$\$TO = 000000
UIPDR0= 177600	XDN = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:12  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0737

3 COMMON EQUATE MODULE  
565 COMMON DEFINITIONS AND REFERENCES FOR NAMCHK  
568 000000' .PRINT ;SPMAC: VERSION 1.1  
590 NAMCHK ROUTINE



```
508 .TITLE NAMCHK VERIFY A MODULE NAME
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; NAMCHK
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE IS CALLED TO VERIFY A MODULE NAME. IT
517 ; RECEIVES THE ADDRESS OF THE FIRST CHARACTER OF THE MODULE
518 ; NAME AND SEARCHES THE MODULE LIST FOR A FIVE CHARACTER MATCH.
519 ; IF THE MODULE NAME IS FOUND, THE HEADER ADDRESS OF THE
520 ; MODULE IS RETURNED. IF IT IS NOT FOUND, A RETURN WITH
521 ; ERROR OCCURS. THE UPDATED COMMAND DECODE BUFFER IS
522 ; RETURNED WHETHER OR NOT THERE WAS AN ERROR.
523 ; A RETURN WITH ERROR OCCURS IF THE MODULE NAME IS
524 ; MORE THAN 5 CHARACTERS IN LENGTH.
525 ;
526 ; INPUTS:
527 ; 1. ADDRESS OF DATA TABLE
528 ; 2. COMMAND DECODE BUFFER POINTER
529 ;
530 ; IMPLICIT INPUTS:
531 ; DT.MLST
532 ;
533 ; OUTPUTS:
534 ; 1. HEADER ADDRESS OF MODULE
535 ; 2. UPDATED COMMAND DECODE BUFFER POINTER
536 ;
537 ; IMPLICIT OUTPUTS:
538 ; NONE
539 ;
540 ; PATHOLOGICAL CONNECTIONS:
541 ; NONE
542 ;
543 ; SUBORDINATE ROUTINES CALLED:
544 ; SAVREG
545 ; RESREG
546 ;
547 ; FUNCTIONAL SIDE EFFECTS:
548 ; NONE
549 ;
550 ; CALLING SEQUENCE:
551 ; CALL NAMCHK IN <DTADR,MODBUF> OUT <HEAD,NEWBUF>
552 ; WHERE DTADR = ADDRESS OF DATA TABLE
553 ; MODBUF = ADDRESS OF 5 BYTE BUFFER CONTAINING MODULE NAME
554 ; HEAD = HEADER ADDRESS OF VERIFIED MODULE
555 ; NEWBUF = UPDATED DECODE BUFFER POINTER
556 ;
557 ; VERSION:
558 ; 0.0
559 ;
560 ; EDIT DATE BY REASON
561 ;--
562
563
```

NAMCHK VERIFY A MODULE NAME  
NAMCHK.MAC 28-JUL-78 09:23

MACY11 30A(1052) 20-SEP-78 18:12 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES FOR NAMCHK

SEQ 0739

```
565 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR NAMCHK
566
567 .MCALL STRUCT
568 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
569
570 000001 $LSTIN=1
571 000001 $LSTTAG=1
572
573 ;*****
574 ;
575 ; REFERENCED BY OTHER MODULES:
576 ;
577 .GLOBL NAMCHK ;ROUTINE ENTRY POINT
578 ;
579 ;*****
580 ;
581 ; GLOBAL REFERENCES:
582 ;
583 .GLOBL SAVREG
584 .GLOBL RESREG
585 ;
586 ;*****
587
588
```

```

590 .SBTTL NAMCHK ROUTINE
591
592 000000' ROUTINE NAMCHK <DTADR,MODBUF,HEAD,NEWBUF>
(2) 000000' NAMCHK:
593
594
595 ;+
596 ; SAVE REGISTERS, UPDATE CMD DECODE BUFFER POINTER, AND SEE IF IT'S
597 ; A SPACE OR <CR>.... IF NOT, POP R1 AND RETURN ERROR BECAUSE
598 ; MODULE NAME MAY NOT EXCEED 5 CHARACTERS.....
599 ;-
600
601 000000' CALL SAVREG
(3) 000000' 004767 000000G JSR PC, SAVREG
602 000004' LET R1 := MODBUF(R5)
(4) 000004' 016501 000002 MOV MODBUF(R5), R1
603 000010' LET NEWBUF(R5) := R1 + #5
(4) 000010' 010165 000006 MOV R1, NEWBUF(R5)
(6) 000014' 062765 000005 000006 ADD #5, NEWBUF(R5)
604 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$
605 000042' INLINE <BR 1$>
(2) 000042' 000433 BR 1$
606 000044' ENDF
(4) 000044' 50002$:
607
608 ;+
609 ; SAVE DTABLE ADR, THEN MODULE LIST
610 ;-
611
612 000044' LET R0 := DTADR(R5)
(4) 000044' 016500 000000 MOV DTADR(R5), R0
613 000050' LET R0 := DT.MLST(R0)
(4) 000050' 016000 000032 MOV DT.MLST(R0), R0
614
615 ;+
616 ; SAVE FIRST MODULE ADDR IN R2
617 ;-
618
619 000054' LET R2 := (R0)
(4) 000054' 011002 MOV (R0), R2
620
621 ;+
622 ; WHILE MODULE LIST IS NOT EXHAUSTED (=0).....
623 ;-
624
625 000056' WHILE (R0) NE #ENDLST DO
(4) 000056' 50003$:
(6) 000056' 021027 000000 CMP (R0), #ENDLST
(9) 000062' 001423 BEQ 50004$
626
627 ;+
628 ; INIT COUNT - WHILE CHARACTER IN MODULE HEADER = CHARACTER IN

```

```

629 ; BUFFER.....
630 ; -
631
632 000064' LET R3 := #5
(4) 000064' 012703 000005 MOV #5,R3
633 000070' WHILEB (R2)+ EQ (R1)+ DO
(4) 000070' 50005$:
(6) 000070' 122221 CMPB (R2)+,(R1)+
(9) 000072' 001011 BNE 50006$
634
635 ;+
636 ; ADVANCE TO NEXT CHARACTER, DECREMENT COUNT, SEE IF COUNT = 0.
637 ; IF IT IS ZERO, SAVE CURRENT HEADER ADDRESS (R0) AND
638 ; RETURN WITH NO ERROR.
639 ; -
640
641 000074' LET R3 := R3 - #1
(6) 000074' 005303 DEC R3
642 000076' IF R3 EQ #0 THEN
(6) 000076' 005703 TST R3
(9) 000100' 001005 BNE 50007$
643 000102' LET HEAD(R5) := (R0)
(4) 000102' 011065 000004 MOV (R0),HEAD(R5)
644 000106' CALL RESREG
(3) 000106' 004767 000000G JSR PC,RESREG
645 000112' RETURN NO.ERROR
(4) 000112' 000413 BR 50000$
646 000114' ENDIF
(4) 000114' 50007$:
647 000114' ENDDO
(4) 000114' 000765 BR 50005$
(3) 000116' 50006$:
648
649 ;+
650 ; GET TO THE NEXT TABLE ADDRESS
651 ; -
652 000116' LET R0 := R0 + #2
(6) 000116' 062700 000002 ADD #2,R0
653
654
655 ;+
656 ; SAVE NEW MODULE ADDRESS IN R2
657 ; -
658
659
660 000122' LET R2 := (R0)
(4) 000122' 011002 MOV (R0),R2
661
662 ;+
663 ; RESTORE MODULE BUFFER POINTER
664 ; -
665
666 000124' LET R1 := MODBUF(R5)
(4) 000124' 016501 000002 MOV MODBUF(R5),R1
667 000130' ENDDO
(4) 000130' 000752 BR 50003$

```

```
(3) 000132'                               50004$:  
668  
669  
670      ;+  
671      ; CLEAN UP AND RETURN WITH ERROR BECAUSE MODULE NAME NOT FOUND  
672      ;-  
673 000132'      INLINE <1$:>  
(2) 000132'                               1$:  
674 000132'      CALL RESREG                               JSR      PC,RESREG  
(3) 000132' 004767 000000G  
675  
676 000136'      RETURN ERROR                               SEC  
(2) 000136' 000261  
(4) 000140' 000401                               BR      50001$  
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	CTRLC = 000003	EOPBIT= 000001	MODEXH= 004000	RBUFEA= 000130
ACTBIT= 004000	CTRL0 = 000017	ERRTYP= 000106	MODHOL= 002000	RBUFPA= 000126
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MODSEL= 001000	RBUFSZ= 000132
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MSGCKD= 000010	RBUFVA= 000124
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGCKS= 000011	RDSERV= 000101
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGDER= 000005	RDWHMI= 000022
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGDRP= 000017	RELERR= 000020
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGECH= 177777	RELMOD= 020000
ASTAT = 000104	DTADR = 000000	HEAD = 000004	MSGEOP= 000013	RELTIM= 010000
AUTO = 000010	DT.ADD= 000042	HRDCNT= 000044	MSGHDR= 000004	RESREG= ***** G
AUTOST= 020000	DT.AP = 000100	HRDPAS= 000050	MSGHNG= 000022	RES1 = 000056
AWAS = 000110	DT.APK= 000076	ICONT = 000036	MSGHRD= 000007	RES2 = 000060
BIT0 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGMAP= 000021	RICHAR= 031060
BIT00 = 000001	DT.CF0= 000014	IDNUM = 000122	MSGNUL= 177775	RPTDAT= 002000
BIT01 = 000002	DT.CF1= 000016	IE = 000100	MSGPDP= 000002	RSTRT = 000112
BIT02 = 000004	DT.ERR= 000020	INDPAR= 000040	MSGPRM= 177776	RUBOUT= 000177
BIT03 = 000010	DT.ESI= 000044	INHDRP= 040000	MSGRES= 000001	RUNMOD= 100000
BIT04 = 000020	DT.EVN= 000000	INHEPR= 020000	MSGSFT= 000006	R5VALU= 001740
BIT05 = 000040	DT.EXS= 000060	INHREL= 001000	MSGSKE= 000003	SAM = 075464
BIT06 = 000100	DT.FCH= 000037	INHRR= 000400	MSGSMB= 000015	SAVREG= ***** G
BIT07 = 000200	DT.FCN= 000036	INIT = 000030	MSGSMH= 000014	SBADR = 000102
BIT08 = 000400	DT.HMX= 000104	INTR = 000120	MSGSMS= 000016	SBKMOD= 000000
BIT09 = 001000	DT.KBE= 000024	IOMOD = 100000	MSGSTD= 000000	SBKSEL= 010000
BIT1 = 000002	DT.KBP= 000026	IOMQDP= 102000	MSGSYS= 000012	SC.ADR= 000006
BIT10 = 002000	DT.KBR= 000022	IOMODR= 112000	MSGVEC= 000020	SC.ALC= 000014
BIT11 = 004000	DT.KBU= 000030	IOMODX= 110000	NAMCHK 000000RG	SC.APC= 000016
BIT12 = 010000	DT.MLS= 000032	JACK = 035060	NBKMOD= 001000	SC.CKL= 000002
BIT13 = 020000	DT.MTI= 000110	KIPAR0= 172340	NCPUDP= 000020	SC.CKP= 000004
BIT14 = 040000	DT.OFF= 000070	KIPAR1= 172342	NEWBUF= 000006	SC.CLO= 000000
BIT15 = 100000	DT.PAS= 000074	KIPAR2= 172344	NDAPTY= 000002	SC.HLD= 000010
BIT2 = 000004	DT.PC = 000002	KIPAR3= 172346	NULL = 000000	SC.SCA= 000012
BIT3 = 000010	DT.PFL= 000062	KIPAR4= 172350	QWEN = 024020	SENDLS= 177777
BIT4 = 000020	DT.PSW= 000004	KIPAR5= 172352	PAERR = 000010	SOFCNT= 000042
BIT5 = 000040	DT.PTA= 000064	KIPAR6= 172354	PARPRE= 002000	SQFPAS= 000046
BIT6 = 000100	DT.RCS= 000102	KIPAR7= 172356	PARSTA= 000100	SPACE = 000040
BIT7 = 000200	DT.REL= 000040	KIPDR0= 172300	PASCNT= 000034	SPOINT= 000032
BIT8 = 000400	DT.SCT= 000066	KIPDR1= 172302	PDPLSI= 020000	SPVALU= 002200
BIT9 = 001000	DT.SMX= 000106	KIPDR2= 172304	PDP60 = 004000	SRO = 177572
BKDEF = 000002	DT.SP = 000006	KIPDR3= 172306	PDP70 = 010000	SR1 = 177574
BKMOD = 000020	DT.SSI= 000046	KIPDR4= 172310	PRI0 = 000000	SR2 = 177576
BKMODE= 040000	DT.ST0= 000010	KIPDR5= 172312	PRI1 = 000040	SR3 = 172516
BKSLSH= 000134	DT.ST1= 000012	KIPDR6= 172314	PRI4 = 000200	STAT = 000026
CAPRES= 000004	DT.SWR= 000056	KIPDR7= 172316	PRI5 = 000240	STATBI= 064757
CASTAT= 000004	DT.SYP= 000072	KTERRO= 000040	PRI6 = 000300	STAT1 = 000027
CDERCT= 000146	DT.WBU= 000050	KTPRES= 000400	PRI7 = 000340	SUSPND= 000001
CDWDCT= 000144	DT.WHL= 000054	KTSTAT= 000020	PRO = 000000	SVR0 = 000062
CKTIM = 100000	DT.WLL= 000052	KTXTND= 040000	PR4 = 000200	SVR1 = 000064
CLKPRE= 000001	DVID1 = 000014	LF = 000012	PR5 = 000240	SVR2 = 000066
CONFIG= 000056	ECCMEM= 000100	LPSTAT= 000001	PR6 = 000300	SVR3 = 000070
CQOVF = 000001	ECCSTA= 000010	MAPSTA= 000200	PR7 = 000340	SVR4 = 000072
CR = 000015	ENBEOP= 010000	MED = 076600	PS = 177776	SVR5 = 000074
CSRA = 000100	ENBNUL= 000001	MEMPAS= 040000	PSW = 177776	SVR6 = 000076
CSRC = 000102	ENDLST= 000000	MODBUF= 000002	RANUM= 000054	SYSCNT= 000052

SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	SLSTTA= 000001	\$SELOC= 000403
TMPID = 000002	WBUFEA= 000136	\$F\$G00= 000400	SNESTL= 177777	\$SERFL= 000000
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	SNSK0 = 000300	\$FLAG= 000001
UIPAR0= 177640	WBUFQR= 000140	\$F\$INC= 000210	SNSK1 = 000120	\$FROM= 000000
UIPAR1= 177642	WBUFSS= 000142	\$F\$L00= 000200	SNSK2 = 000120	\$SLOC = 000100R
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	SNSK3 = 000110	\$SLOCN= 000000
UIPAR3= 177646	WDT0 = 000114	\$F\$ND = 000403	\$SAVLE= 177777	\$REG = 177777
UIPAR4= 177650	WTINRE= 000352	\$F\$DR = 000320	\$SSK0 = 050004	\$RETI= 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	\$TD = 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	\$BITE= 000403	
UIPDR7= 177616	\$F\$DEC= 000220	\$LOCTA= 177777	\$CASE= 000000	
WASADR= 000104	\$F\$DD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:14  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0745

3 COMMON EQUATE MODULE  
558 COMMON DEFENITIONS AND REFERENCES  
561 000000' .PRINT ;SPMAC: VERSION 1.1  
603 NEWBA ROUTINE



```
508 .TITLE NEWBA - ESTABLISH NEW BASE ADDRESS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     NEWBA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CALCULATES A NEW RELOCATION BASE ADDRESS BY USING
517 ;     EITHER THE CONSTANT OFFSET OR A RANDOMLY GENERATED
518 ;     OFFSET VALUE
519 ;
520 ; INPUTS:
521 ;     DATA TABLE ADDRESS, CURRENT ADDRESS OFFSET,
522 ;     FIRST RELOCATION FLAG
523 ;
524 ; IMPLICIT INPUTS:
525 ;     DT.REL, DT.ESIZ, DT.SSIZ, DT.SWR, DT.STO
526 ;
527 ; OUTPUTS:
528 ;     NEW ADDRESS OFFSET
529 ;
530 ; IMPLICIT OUTPUTS:
531 ;     NONE
532 ;
533 ; PATHOLOGICAL CONNECTIONS:
534 ;     NONE
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ;     RANDOM                ;RANDOM NUMBER GENERATOR
538 ;     SAVREG                ;SAVE REGISTERS
539 ;     RESREG                ;RESTORE REGISTERS
540 ;
541 ; FUNCTIONAL SIDE EFFECTS
542 ;     NONE
543 ;
544 ; CALLING SEQUENCE:
545 ;     CALL NEWBA IN <A,B,C> OUT <D>
546 ;           A=ADDRESS OF DATA TABLE
547 ;           B=CURRENT ADDRESS OFFSET
548 ;           C=FIRST RELOCATION FLAG
549 ;           D=NEW ADDRESS OFFSET
550 ;
551 ; VERSION:
552 ;     0.0
553 ;
554 ;     EDIT                BY                DATE                REASON
555 ;
556 ;--
```

```
558 .SBTTL COMMON DEFINITIONS AND REFERENCES
559
560 .MCALL STRUCT
561 000000' STRUCT
562 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
563 000001 $LSTIN=1
564 000001 $LSTTAG=1
565
566 ;
567 ;*****
568 ;
569 ; REFERENCED BY OTHER MODULES
570 ;
571 .GLOBL NEWBA ;MODULE ENTRY POINT
572 ;
573 ;*****
574 ;
575 ; GLOBAL REFERENCES
576 ;
577 .GLOBL RANDOM ;RANDOM NUMBER GENERATOR
578 .GLOBL SAVREG ;SAVE REGISTERS
579 .GLOBL RESREG ;RESTORE REGISTERS
580 ;
581 ;*****
582 ;
583 ; LOCAL STORAGE
584 ;
585 000000' 000000 NB.CNT: .WORD 0 ;NO. OF RANDOM RELOCATIONS SINCE LAST
586 ; MOVE TO BOTTOM OF MEMORY
587 000002' 000000 NB.CYCLNG: .WORD 0 ;CYCLING FLAG - INDICATES, WHEN SET, RELOCATING
588 ; BY CONSTANT OFFSET (CYCLING)
589 000004' 000000 NB.MAX: .WORD 0 ;MAX. NO. OF RANDOM RELOCATIONS ALLOWED
590 ; BEFORE RELOCATING BACK TO BOTTOM
591 000006' 000000 NB.NBR: .WORD 0 ;STORAGE FOR RANDOM NUMBER
592 000010' 000000 NB.STADDR: .WORD 0 ;STORAGE FOR EXERCISER'S STARTING ADDRESS
593 ; OFFSET VALUE
594 000012' 000000 NB.OFLAG: .WORD 0 ;O-FLAG - INDICATES, WHEN SET, THAT THERE
595 ; HAS BEEN A RELOCATION TO BOTTOM
596 000014' 000000 NB.TOP: .WORD 0 ;OFFSET NEEDED TO MOVE THE EXERCISER
597 ; TO HIGHEST MEMORY
598 000016' 000000 NB.BIT: .WORD 0 ;BIT PATTERN FOR USE WITH RANDOM NUMBER
599 ;
600 ;*****
601 ;
```

```

603          .SBTTL NEWBA ROUTINE
604
605 000020'  ROUTINE NEWBA <TABL,OLDPAR,RL1,NEWPAR>
606          NEWBA:
607
608          ;+
609          ;SAVE REGISTERS
610          ;-
611
612 000020'  CALL SAVREG
613 (3) 000020' 004767 000000G          JSR      PC,SAVREG
614
615          ;+
616          ;SET R0 TO START OF DATA TABLE
617          ;-
618
619 000024'  LET R0 := TABL(R5)
620 (4) 000024' 016500 000000          MOV      TABL(R5),R0
621
622          ;+
623          ;IF THIS IS THE FIRST RELOCATION, DO THE NECESSARY INITIALIZATION
624          ;-
625
626 000030'  IF RL1(R5) EQ #1 THEN
627 (6) 000030' 026527 000004 000001          CMP      RL1(R5),#1
628 (9) 000036' 001067          BNE      50002$
629
630          ;+
631          ;INITIALIZE FLAGS AND NB.CNT
632          ;-
633          LET NB.CYCLNG := #1
634 (4) 000040' 012767 000001 177734          MOV      #1,NB.CYCLNG
635          LET NB.OFLAG := #0
636 (4) 000046' 005067 177740          CLR      NB.OFLAG
637          LET NB.CNT := #0
638 (4) 000052' 005067 177722          CLR      NB.CNT
639
640          ;+
641          ;SAVE THE INITIAL ADDRESS OFFSET
642          ;-
643          LET NB.STADDR := OLDPAR(R5)
644 (4) 000056' 016567 000002 177724          MOV      OLDPAR(R5),NB.ST
645
646          ;+
647          ;CALCULATE NB.MAX BY DIVIDING BY 2 THE NUMBER OF K'S OF MEMORY.
648          ;THIS IS DONE BY GETTING THE CONTENTS OF DT.SSIZ AND SHIFTING
649          ;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
```

```

659
660
661 ;+
662 ;CALCULATE THE ADDRESS OFFSET NEEDED TO MOVE THE EXERCISER TO THE
663 ;TOP OF CORE. THIS IS DONE USING THE FOLLOWING ALGORITHM:
664 ; 1. GET THE SIZE OF THE MOVABLE PORTION OF THE EXERCISER
665 ; 2. SHIFT OUT THE SIX LEAST SIGNIFICANT BITS
666 ; 3. ADD ONE TO THE RESULT TO ALLOW ROOM FOR THE BLOCKS JUST
667 ; SHIFTED OUT
668 ; 4. SUBTRACT THIS FROM THE SYSTEM SIZE. IF, HOWEVER, THE
669 ; UNIBUS MAP IS SHUT OFF, (IT MAY NOT
670 ; EVEN EXIST), THEN DON'T ALLOW RELOCATION ABOVE 124K.
671 ;-
672
673 000122' LET R2 := DT.ESIZ(R0) - #20000
674 (4) 000122' 016002 000044
675 (6) 000126' 162702 020000
676
677 LET R2 := R2 SHIFT #-6
678 (7) 000132' 006202
679 (7) 000134' 006202
680 (7) 000136' 006202
681 (7) 000140' 006202
682 (7) 000142' 006202
683 (7) 000144' 006202
684
685 LET R2 := R2 CLR.BY #176000
686 (6) 000146' 042702 176000
687
688 LET R2 := R2 + #1
689 (6) 000152' 005202
690
691
692 IF #MAPSTAT NOTSETIN DT.STO(R0) AND DT.SSIZ(R0) HIS #7600 THEN
693 (6) 000154' 032760 000200 000010
694 (9) 000162' 001007
695 (6) 000164' 026027 000046 007600
696 (9) 000172' 103403
697
698 LET R1 := #7600
699 (4) 000174' 012701 007600
700
701 ELSE
702 (4) 000200' 000402
703 (3) 000202'
704
705 LET R1 := DT.SSIZ(R0)
706 (4) 000202' 016001 000046
707
708 ENDIF
709 (4) 000206'
710
711 LET NB.TOP := R1 - R2
712 (4) 000206' 010167 177602
713 (6) 000212' 160267 177576
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
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
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

693 ;THE CYCLING FLAG IS SET OR IF BIT 8 OF THE SOFTWARE SWITCH
694 ;REGISTER IS SET.
695 ;-
696
697 000216' IF NB.CYCLNG EQ #1 DR #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$:
698
699
700 ;+
701 ;RELOCATING BY CONSTANT OFFSET (CYCLING)
702 ;-
703
704 ;+
705 ;ADD THE OFFSET CONSTANT TO THE CURRENT ADDRESS OFFSET TO FIND
706 ;A NEW BASE
707 ;-
708
709 000236' LET NEWPAR(R5) := OLDPAR(R5) + DT.REL(R0)
(4) 000236' 016565 000002 000006 MOV OLDPAR(R5),NEWPA
(6) 000244' 066065 000040 000006 ADD DT.REL(R0),NEWPA
710
711
712 ;+
713 ;SEE IF THIS NEW BASE VALUE IS TOO LARGE FOR THE SYSTEM.
714 ;-
715
716 000252' IF NEWPAR(R5) HI NB.TOP THEN
(6) 000252' 026567 000006 177534 CMP NEWPAR(R5),NB.TO
(9) 000260' 101406 BLOS 50010$
717
718
719 ;+
720 ;THE NEW ADDRESS WILL BE TOO HIGH - WE'VE HIT THE TOP OF CORE, SO
721 ;IT'S TIME TO RELOCATE TO THE BOTTOM. THE NEW OFFSET VALUE WILL BE 200.
722 ;THIS WILL PUT THE EXERCISER BACK TOGETHER AGAIN AT THE BOTTOM OF
723 ;MEMORY. SET NB.OFLAG TO INDICATE WE'VE GONE TO THE BOTTOM, AND RETURN.
724 ;-
725
726 000262' LET NEWPAR(R5) := #200
(4) 000262' 012765 000200 000006 MOV #200,NEWPAR(R5)
727 000270' LET NB.OFLAG := #1
(4) 000270' 012767 000001 177514 MOV #1,NB.OFLAG
728
729
730 000276' ENDIF
(4) 000276' 50010$:
731
732
733 ;+
734 ;NEW ADDRESS IS OK. BUT BEFORE WE LEAVE, WE HAVE TO SEE IF WE HAVE
735 ;FINISHED CYCLING. THIS IS DONE BY CHECKING IF THE NEW BASE IS
736 ;GREATER THAN OR EQUAL TO THE STARTING ADDRESS OFFSET, AND IF NB.OFLAG

```

```
737 ;IS SET
738 ;+
739 ;-
740 000276' IF NEWPAR(R5) HIS NB.STADDR AND NB.OFLAG 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.OFLAG,#1
(9) 000314' 001004 BNE 50011$
741
742
743 ;+
744 ;CYCLING IS COMPLETED. CLEAR THE CYCLING FLAG.
745 ;ALSO CLEAR NB.OFLAG.
746 ;-
747
748 000316' LET NB.CYCLNG := #0
(4) 000316' 005067 177460 CLR NB.CYCLNG
749 000322' LET NB.OFLAG := #0
(4) 000322' 005067 177464 CLR NB.OFLAG
750
751 000326' ENDIF
(4) 000326' 50011$:
752
753
754 000326' ELSE
(4) 000326' 000463 BR 50012$
(3) 000330' 50007$:
755
756
757 ;+
758 ;RELOCATING BY RANDOM OFFSET VALUES
759 ;-
760
761 ;+
762 ;IF THE LAST RELOCATION WAS BACK TO THE BOTTOM OF MEMORY, THEN
763 ;WE MUST NOW RELOCATE TO THE TOP-MOST PIECE OF MEMORY.
764 ;-
765
766 000330' IF NB.OFLAG EQ #1 THEN
(6) 000330' 026727 177456 000001 CMP NB.OFLAG,#1
(9) 000336' 001006 BNE 50013$
767
768
769 ;+
770 ;GET THE PREVIOUSLY CALCULATED OFFSET NEEDED TO GET US TO THE TOP
771 ;OF MEMORY. ALSO CLEAR NB.OFLAG.
772 ;-
773 000340' LET NEWPAR(R5) := NB.TOP
(4) 000340' 016765 177450 000006 MOV NB.TOP,NEWPAR(R5)
774 000346' LET NB.OFLAG := #0
(4) 000346' 005067 177440 CLR NB.OFLAG
775
776
777 000352' ELSE
(4) 000352' 000451 BR 50014$
(3) 000354' 50013$:
```

778  
 779  
 780  
 781  
 782  
 783  
 784  
 785  
 786  
 (6)  
 (9)  
 787  
 (4)  
 788  
 (4)  
 789  
 (4)  
 790  
 791  
 (4)  
 (3)  
 792  
 793  
 794  
 795  
 796  
 797  
 798  
 799  
 800  
 801  
 802  
 803  
 804  
 805  
 (3)  
 806  
 (4)  
 807  
 (4)  
 (3)  
 (4)  
 808  
 (4)  
 (6)  
 (9)  
 809  
 (6)  
 810  
 (7)  
 811  
 (4)  
 (3)  
 812  
 (3)  
 (6)

```

;+
;IF THE NUMBER OF PREVIOUS RANDOM RELOCATIONS EQUALS THE MAXIMUM,
;THEN RELOCATE TO THE BOTTOM OF CORE, SET NB.OFLAG, AND RESET THE
;RELOCATION COUNTER
;-

IF NB.CNT EQ NB.MAX THEN
    CMP     NB.CNT,NB.MAX
    BNE     50015$
    LET     NEWPAR(R5) := #200
    MOV     #200,NEWPAR(R5)
    LET     NB.OFLAG := #1
    MOV     #1,NB.OFLAG
    LET     NB.CNT := #0
    CLR     NB.CNT
ELSE
    BR     50016$
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.
;-

```

```

REPEAT
    50017$:
    LET     NB.BIT := #100000
    MOV     #100000,NB.BIT
    CALL    RANDOM OUT <NB.NBR>
    SUB     #1*2,R5
    JSR     PC,RANDOM
    MOV     (R5)+,NB.NBR
    WHILE  NB.NBR HI NB.TOP DO
        50020$:
        CMP     NB.NBR,NB.TOP
        BLOS    50021$
        LET     NB.NBR := NB.NBR CLR.BY NB.BIT
        BIC     NB.BIT,NB.NBR
        LET     NB.BIT := NB.BIT SHIFT #-1
        ASR     NB.BIT
    ENDDO
    BR     50020$
    50021$:
    UNTIL  NB.NBR HIS #200
    CMP     NB.NBR,#200
    BLO     50017$

```



```
813 000464'          LET NEWPAR(R5) := NB.NBR
(4) 000464' 016765 177316 000006          MCV      NB.NBR,NEWPAR(R5)
814 000472'          LET NB.CNT := NB.CNT + #1          INC      NB.CNT
(6) 000472' 005267 177302
815
816 000476'          ENDIF
(4) 000476'          50016S:
817 000476'          ENDIF
(4) 000476'          50014S:
818 000476'          ENDIF
(4) 000476'          50012S:
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'          50000S:
(3) 000502'          50001S:
(2) 000502' 000207          RTS      PC
828
829          000001          .END
```

ACSR = 000102	CTRLC = 000003	ERRTYP= 000105	MSGCKD= 000010	PR5 = 000240
ACTBIT= 004000	CTRL0 = 000017	EVNTBE= 000200	MSGCKS= 000011	PR6 = 000300
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	PR7 = 000340
ADR = 000006	DCEVNT= 000011	EVNKT= 000203	MSGDRP= 000017	PS = 177776
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	PSW = 177776
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGEOP= 000013	RANDOM= ***** G
ASB = 000106	DROPMO= 100000	FATERR= 100000	MSGHDR= 000004	RANNUM= 000054
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RSUFEA= 000130
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RBUFPA= 000126
AUTO = 000010	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RBUFSZ= 000132
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RBUFVA= 000124
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RDSERV= 000101
BIT0 = 000001	DT.CF0= 000014	IE = 000100	MSGPRM= 177776	RDWHMI= 000022
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	RELERR= 000020
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	RELMOD= 020000
BIT02 = 000004	DT.ESI= 000044	INHPR= 020000	MSGSKE= 000003	RELTIM= 010000
BIT03 = 000010	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	RESREG= ***** G
BIT04 = 000020	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	RES1 = 000056
BIT05 = 000040	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	RES2 = 000060
BIT06 = 000100	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	RICHAR= 031060
BIT07 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	RL1 = 000004
BIT08 = 000400	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	RPTDAT= 002000
BIT09 = 001000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	RSTRT = 000112
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NB.BIT 000016R	RUBOUT= 000177
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NB.CNT 000000R	RUNMOD= 100000
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NB.CYC 000002R	RVALU= 001740
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	NB.MAX 000004R	SAM = 075464
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	NB.NBR 000006R	SAVREG= ***** G
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	NB.STA 000010R	SBADR = 000102
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	NB.TOP 000014R	SBKMOD= 000000
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	NB.OFL 000012R	SBKSEL= 010000
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	NCFUOP= 000020	SC.ADR= 000006
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	NEWBA 000020RG	SC.ALC= 000014
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	NEWPAR= 000006	SC.APC= 000016
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	NOPTY= 000002	SC.CKL= 000002
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	NULL = 000000	SC.CKP= 000004
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	QLDPAR= 000002	SC.CLO= 000000
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	OWEN = 024020	SC.HLD= 000010
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PAERR = 000010	SC.SCA= 000012
BKMOD = 000020	DT.ST0= 000010	KIPDR6= 172314	PARPRE= 002000	SENDLS= 177777
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PARSTA= 000100	SOFCNT= 000042
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PASCNT= 000034	SOFPAS= 000046
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PDPLSI= 020000	SPACE = 000040
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PDP60 = 004000	SPOINT= 000032
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PDP70 = 010000	SPVALU= 002200
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PRI0 = 000000	SR0 = 177572
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PRI1 = 000040	SR1 = 177574
CLKPRE= 000001	ECCMEM= 000100	MAPSTA= 000200	PRI4 = 000200	SR2 = 177576
CONFIG= 000056	ECCSTA= 000010	MED = 076600	PRI5 = 000240	SR3 = 172516
CQOVF = 000001	ENBEDP= 010000	MEMPAS= 040000	PRI6 = 000300	STAT = 000026
CR = 000015	ENBNUL= 000001	MODEXH= 004000	PRI7 = 000340	STATBI= 064757
CSRA = 000100	ENDLST= 000000	MODHOL= 002000	PRO = 000000	STAT1 = 000027
CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	PR4 = 000200	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	\$\$SERFL= 000000
SVR4 = 000072	UIPDR7= 177616	\$F\$GDO= 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\$LDO= 000200	\$NSK3 = 000110	\$\$LOCN= 000000
SYSERR= 000100	WBUFPA= 000134	\$F\$NAM= 000160	\$NSK4 = 000130	\$\$REG = 177777
TABL = 000000	WBUFRQ= 000140	\$F\$ND = 000403	\$NSK5 = 000120	\$\$RETU= 000000
TMPIO = 000002	WBUFSZ= 000142	\$F\$OR = 000320	SSAVLE= 177777	\$\$RTN1= 050000
TQOVF = 000002	WDFR = 000116	\$F\$RTI= 000350	SSSK0 = 050021	\$\$RTN2= 050001
UIPAR0= 177640	WDT0 = 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	
UIPDRO= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:15  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0757

3 COMMON EQUATE MODULE  
565 COMMON DEFINITIONS AND REFERENCES FOR NUMCHK  
568 000000' .PRINT ;SPMAC: VERSION 1.1  
593 NUMCHK ROUTINE

```
508 .TITLE NUMCHK OCTAL ASCII TO BINARY CONVERSION ROUTINE
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     NUMCHK
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS ROUTINE CONVERTS A 6 DIGIT OCTAL ASCII NUMBER
517 ;     INTO A 16 BIT BINARY NUMBER. IT WILL VERIFY THAT THE NUMBER
518 ;     IS NOT MORE THAN 16 BITS IN LENGTH(LEADING ZEROES ARE
519 ;     ARE NOT COUNTED AS PART OF THE 16 BITS) AND THAT IT IS AN OCTAL NUMBER.
520 ;     A RETURN WITH ERROR OCCURS IF THE NUMBER IS GREATER THAN 16
521 ;     BITS IN LENGTH OR IT IS NOT OCTAL ( I.E. SOMETHING
522 ;     OTHER THAN THE NUMBERS 0-7 WAS DETECTED IN THE NUMBER).
523 ;
524 ; INPUTS:
525 ;     1. ADDRESS OF DATA TABLE
526 ;     2. COMMAND DECODE BUFFER POINTER
527 ;
528 ; IMPLICIT INPUTS:
529 ;     NONE
530 ;
531 ; OUTPUTS:
532 ;     1. UPDATED COMMAND DECODE BUFFER POINTER
533 ;     2. THE BINARY NUMBER
534 ;
535 ; IMPLICIT OUTPUTS:
536 ;     DT.KBRSP - THE COMMAND RESPONSE LOCATION
537 ;
538 ; PATHOLOGICAL CONNECTIONS:
539 ;     CM.NUM          ;ERROR MESSAGE
540 ;     CM.NBAD        ;ERROR MESSAGE
541 ;
542 ; SUBORDINATE ROUTINES CALLED:
543 ;     SAVREG
544 ;     RESREG
545 ;
546 ; FUNCTIONAL SIDE EFFECTS:
547 ;     NONE
548 ;
549 ; CALLING SEQUENCE:
550 ;     CALL NUMCHK IN <DTADR,CMDPTR> OUT <NEWPTR,NUM>
551 ;     WHERE:
552 ;     DTADR = ADDRESS OF DATA TABLE
553 ;     CMDPTR = COMMAND DECODE BUFFER POINTER
554 ;     NEWPTR = UPDATED CMD DECODE BUFFER POINTER
555 ;     NUM = CONVERTED 16 BIT BINARY NUMBER
556 ;
557 ; VERSION:
558 ;     0.0
559 ;
560 ;     EDIT          DATE          BY          REASON
561 ;--
562
563
```

```
565          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR NUMCHK
566
567          .MCALL STRUCT
568 000000'    STRUCT
569 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
570          $LSTIN=1
571          $LSTTAG=1
572
573          ;*****
574          ;
575          ; REFERENCED BY OTHER MODULES
576          ;
577          .GLOBL NUMCHK          ;MODULE ENTRY POINT
578          ;
579          ;*****
580          ;
581          ; GLOBAL REFERENCES:
582          ;
583          .GLOBL CM.NUM          ;'NUMBER TOO LARGE' MSG ADDRESS
584          .GLOBL SAVREG         ;SAVE REGISTERS
585          .GLOBL RESREG         ;RESTORE REGISTERS
586          .GLOBL CM.NBAD        ;'NOT AN OCTAL NUMBER' MSG ADDRESS
587          ;
588          ;*****
589
590
591
```

```

593          .SBTTL NUMCHK ROUTINE
594
595 000000'          ROUTINE NUMCHK <DTADR,CMDBUF,NEWPTR,NUM>
(2) 000000'
596
597          ;+
598          ; SAVE REGISTERS, DATA TABLE ADDRESS AND CMD DECODE BUFFER POINTER
599          ; -
600
601 000000'          CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC,SAVREG
602
603 000004'          LET R0 := DTADR(R5)
(4) 000004' 016500 000000          MOV      DTADR(R5),R0
604 000010'          LET R1 := CMDBUF(R5)
(4) 000010' 016501 000002          MOV      CMDBUF(R5),R1
605
606          ;+
607          ; INITIALIZE THE OUTPUT REG
608          ; -
609
610 000014'          LET R2 := #0
(4) 000014' 005002          CLR      R2
611
612          ;+
613          ; WHILE THERE IS AN OCTAL ASCII CHAR IN CMD DECODE BUFFER.....
614          ; -
615
616 000016'          WHILEB (R1) GE #'0 ANDB (R1) LE #'7 DO
(4) 000016'
(6) 000016' 121127 000060          50002$:  CMPB   (R1),#'0
(9) 000022' 002425          BLT    50003$
(6) 000024' 121127 000067          CMPB   (R1),#'7
(9) 000030' 003022          BGT    50003$
617
618
619          ;+
620          ; INITIALIZE THE COUNTER.
621          ; -
622
623 000032'          LET R4 := #0
(4) 000032' 005004          CLR      R4
624
625          ;+
626          ; GET A CHARACTER, SUBTRACT ASCII ZERO, ROTATE 3 LEFT (TO MAKE ROOM),
627          ; AND CHECK TO SEE IF NUM IS NOW > 16 BITS. IF IT IS, STUFF
628          ; ERROR MSG AND RETURN. IF NOT, MOVE R3 TO R2.
629          ; -
630
631 000034'          LET R3 := (R1)+
(4) 000034' 112103          MOVB   (R1)+,R3
632 000036'          LET R3 := R3 - #'0
(6) 000036' 162703 000060          SUB    #'0,R3
633 000042'          WHILE R4 NE #3 DO
(4) 000042'
(6) 000042' 020427 000003          50004$:  CMP    R4,#3
    
```







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

SVR6 = 000076	UIPDR7= 177616	\$F\$DEC= 000220	\$LOCTA= 177777	\$\$CASE= 000000
SYS CNT= 000052	WASADR= 000104	\$F\$DO = 000340	\$LSTIN= 000001	\$\$DST = 000000
SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LSTTA= 000001	\$\$SELOC= 000403
TMPIO = 000002	WBUFEA= 000136	\$F\$GOD= 000400	\$NESTL= 177777	\$\$SERFL= 000000
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NSKO = 000300	\$\$FLAG= 000001
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	\$NSK1 = 000110	\$\$FROM= 000000
UIPAR1= 177642	WBUFSZ= 000142	\$F\$L00= 000200	\$NSK2 = 000120	\$\$LOC = 000116R
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK3 = 000110	\$\$LOCN= 000000
UIPAR3= 177646	WDTC = 000114	\$F\$NO = 000403	SSAVLE= 177777	\$\$REG = 177777
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	SSKO = 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	\$TSKO = 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	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:16

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0765

3 COMMON EQUATE MODULE  
555 000000'

.PRINT ;SPMAC: VERSION 1.1

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552

```
.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
;
;---
```

554  
555 000000'  
(1) 000000'  
556 000001  
557 000001  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571

```
.MCALL STRUCT
STRUCT
.PRINT ;SPMAC: VERSION 1.1
$LSTIN=1
$LSTTAG=1
;*****
;REFERENCED BY OTHER MODULES
;
.GLOBL PBREAK
;
;*****
; GLOBAL REFERENCES
;
.GLOBL ENQCQ
.GLOBL SAVREG
.GLOBL RESREG
;*****
```

```
573 000000'          ROUTINE PBREAK <DTA>
(2) 000000'
574
575
576 ;+
577 ; SAVE REGISTERS NEEDED
578 ; -
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)
(5) 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'
(3) 000044'
(2) 000044' 000207          500005:
602                                     50001$:
603 000001          .END          RTS      PC
```

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



TQOVF = 000002	WASADR= 000104	\$FSBLA= 000170	\$FSUNT= 000130	\$SELOC= 000000
UIPAR0= 177640	WBSTAT= 000040	\$FSCAS= 000150	\$FSWHI= 000120	\$\$SERFL= 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\$G00= 000400	\$LSTIN= 000001	\$\$L0CN= 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	SNSKO = 000300	\$\$RTN1= 050000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$\$AVLE= 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	\$\$TD = 000000
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$\$CASE= 000000	\$\$TAG = 050000
UIPDR7= 177616	\$F\$BAD= 000401	\$F\$TRU= 000404	\$\$DST = 000000	= 000045R

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:16  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0771

3 COMMON EQUATE MODULE  
551 COMMON DEFINITIONS AND REFERENCES  
557 000000' .PRINT ;SPMAC: VERSION 1.1  
581 PROCESS END TRAP CALL

```
508  
509 .TITLE PEND - PROCESS END TRAP CALL  
510 .IDENT /V0.0/  
511  
512  
513  
514 ;++  
515 ; MODULE NAME  
516 ; PEND  
517 ;  
518 ; FUNCTIONAL DESCRIPTION:  
519 ; THIS ROUTINE PROCESSES THE 'ENDMOD' TRAP CALL (END$).  
520 ;  
521 ; INPUTS:  
522 ; 1. DATA TABLE  
523 ;  
524 ; IMPLICIT INPUTS:  
525 ; 1. DT.PC  
526 ;  
527 ; OUTPUTS:  
528 ; NONE  
529 ;  
530 ; IMPLICIT OUTPUTS:  
531 ; NONE  
532 ;  
533 ; PATHOLOGICAL CONNECTIONS:  
534 ; NONE  
535 ;  
536 ; SUBORDINATE ROUTINES CALLED:  
537 ; 1. DRPMOD ;ROUTINE TO DROP MODULE FROM EXECUTION  
538 ;  
539 ; FUNCTIONAL SIDE EFFECTS:  
540 ; NONE  
541 ;  
542 ; CALLING SEQUENCE:  
543 ; CALL PEND IN <DTABLE>  
544 ;  
545 ; VERSION:  
546 ; 0.0  
547 ;  
548 ; EDIT BY DATE REASON  
549 ;--
```

```
551 .SBTTL COMMON DEFINITIONS AND REFERENCES
552
553
554
555
556 .MCALL STRUCT
557 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
558 000001 $LSTIN = 1
559 000001 $LSTTAG = 1
560
561
562
563
564
565 ;
566 ;*****
567 ;
568 ; REFERENCED BY OTHER MODULES
569 ;
570 .GLOBL PEND ;MODULE'S ENTRY POINT
571 ;
572 ;*****
573 ;
574 ; GLOBAL REFERENCES
575 ;
576 .GLOBL DRPMOD
577 ;
578 ;*****
579 ;
```

```
581 .SBTTL PROCESS END TRAP CALL
582
583
584 ROUTINE PEND <DTABLE>
585
586
587
588 ;+
589 ; INIT, GET DTABLE ADDR. AND HEADER
590 ; -
591
592 PUSH R0,R1
593
594 LET R0 := DTABLE(R5)
595
596 ;+
597 ; GO DROP THE OPTION MODULE FROM EXECUTION,
598 ; -
599
600 CALL DRPMD IN <R0,R1>
601
602
603 POP R1,R0
604
605 ENDRTN
606
607 .END
```

PEND:

MOV R0,-(SP)  
MOV R1,-(SP)  
MOV DTABLE(R5),R0  
MOV @DT.PC(R0),R1  
MOV R5,-(SP)  
MOV R1,-(R5)  
MOV R0,-(R5)  
JSR PC,DRPMD  
MOV (SP)+,R5  
MOV (SP)+,R1  
MOV (SP)+,R0  
50000\$:  
50001\$: RTS PC

ACSR = 000102	CTRLC = 000003	ENDLST= 000000	MODHOL= 002000	RBUFV= 000132
ACTBIT= 004000	CTRL0 = 000017	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124
ADDR22= 001000	CTRLU = 000025	ERRTYP= 000106	MSGCKD= 000010	RDSERV= 000101
ADR = 000006	DCEVNT= 000011	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
APTFER= 000004	DEFRTN= 000400	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
APTPRE= 000200	DIAGMC= 000000	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
ASB = 000106	DROPMO= 100000	EVNTPE= 000202	MSGECH= 177777	RELTIM= 010000
ASSEMB= 000010	DRPMOD= ***** G	EVNTRE= 000201	MSGEOP= 000013	RES1 = 000056
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
AUTO = 000010	DTABLE= 000000	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUBOUT= 000177
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT01 = 000002	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	R5VALU= 001740
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBADR = 000102
BIT04 = 000020	DT.ESI= 000044	INHEPR= 020000	MSGSKE= 000003	SBKMOD= 000000
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBKSEL= 010000
BIT06 = 000100	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SC.ADR= 000006
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.ALC= 000014
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.APC= 000016
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKL= 000002
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CKP= 000004
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CLD= 000000
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.HLD= 000010
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SC.SCA= 000012
BIT13 = 020000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SENDLS= 177777
BIT14 = 040000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SOFcnt= 000042
BIT15 = 100000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOFPAS= 000046
BIT2 = 000004	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPACE = 000040
BIT3 = 000010	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPOINT= 000032
BIT4 = 000020	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPVALU= 002200
BIT5 = 000040	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 023000	SR0 = 177572
BIT6 = 000100	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR1 = 177574
BIT7 = 000200	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR2 = 177576
BIT8 = 000400	DT.REL= 000040	KIPDR1= 172302	PEND 000000RG	SR3 = 172516
BIT9 = 001000	DT.SCT= 000066	KIPDR2= 172304	PRI0 = 000000	STAT = 000026
BKDEF = 000002	DT.SMX= 000106	KIPDR3= 172306	PRI1 = 000040	STATBI= 064757
BKMOD = 000020	DT.SP = 000006	KIPDR4= 172310	PRI4 = 000200	STAT1 = 000027
BKMODE= 040000	DT.SSI= 000046	KIPDR5= 172312	PRI5 = 000240	SUSPND= 000001
BKSLSH= 000134	DT.ST0= 000010	KIPDR6= 172314	PRI6 = 000300	SVR0 = 000062
CAPRES= 000004	DT.ST1= 000012	KIPDR7= 172316	PRI7 = 000340	SVR1 = 000064
CASSTAT= 000004	DT.SWR= 000056	KTERR0= 000040	PR0 = 000000	SVR2 = 000066
CDERCT= 000146	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR3 = 000070
CDWDCT= 000144	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR4 = 000072
CKTIM = 100000	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR5 = 000074
CLKPRE= 000001	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR6 = 000076
CONFIG= 000056	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SYSCNT= 000052
COQVF = 000001	ECCMEM= 000100	MAPSTA= 000200	PSW = 177776	YSERR= 000100
CR = 000015	ECCSTA= 000010	MED = 076600	RANNUM= 000054	TMPID = 000002
CSRA = 000100	ENBEOP= 010000	MEMPAS= 040000	RBUFEA= 000130	TQOVF = 000002
CSRC = 000102	ENBNUL= 000001	MODEXH= 004000	RBUFPA= 000126	UIPAR0= 177640

UIPAR1= 177642	WBUFEA= 000136	\$F\$DEC= 000220	SF\$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	WBUF SZ= 000142	\$F\$G00= 000400	\$LSTIN= 000001	\$LOCN= 000000
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$REG = 177777
UIPAR6= 177654	WDT0 = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$RETU= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$LOD= 000200	\$NSKO = 000300	\$RTN1= 050000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$RTN2= 050001
UIPDR1= 177602	XFLAG = 000005	\$F\$ND = 000403	\$TAGLE= 177777	\$SRC = 000000
UIPDR2= 177604	XOFF = 000023	\$F\$DR = 000320	\$TAGNU= 050002	\$TGSV= 000000
UIPDR3= 177606	XDN = 000021	\$F\$RTI= 000350	\$TEMP = 000300	\$TGS1= 000000
UIPDR4= 177610	\$BGNLE= 177777	\$F\$RTN= 000300	\$SARGC= 000002	\$TGS2= 000000
UIPDR5= 177612	\$ERFLG= 000400	\$F\$SEL= 000140	\$S\$BYTE= 000000	\$TO = 000002
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$SCASE= 000000	\$STAG= 050000
UIPDR7= 177616	\$F\$BAD= 000401	\$F\$TRU= 000404	\$SDST = 000000	= 000036R
WASADR= 000104	\$F\$BLA= 000170	\$F\$UNT= 000130	\$SELOC= 000000	
WBSTAT= 000040	\$F\$CAS= 000150	\$F\$WHI= 000120	\$SERFL= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:17  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0777

3 COMMON EQUATE MODULE  
562 COMMON DEFINITIONS AND REFERENCES  
566 000000' .PRINT ;SPMAC: VERSION 1.1  
592 PROCESS ENDIT ROUTINE





562  
563  
564  
565  
566  
(1)  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590

.SBTTL COMMON DEFINITIONS AND REFERENCES

.MCALL STRUCT  
STRUCT  
.PRINT ;SPMAC: VERSION 1.1  
\$LSTIN = 1  
\$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

```
592 .SBTTL PROCESS ENDIT ROUTINE
593
594
595 000000'          ROUTINE PENDIT <DTABLE>
(2) 000000'          PENDIT:
596
597 ;+
598 ; DO THE NECESSARY SET-UP STUFF, WHICH CONSISTS OF SAVING THE GPR'S
599 ; AND MARKING THE MODULE FOR AN IMMEDIATE RETURN AFTER
600 ; THE PROCESSING OF THIS TRAP CALL
601 ;-
602
603 000000'          CALL SAVREG
(3) 000000' 004767 000000G          JSR    PC, SAVREG
604 000004'          LET R4 := #0
(4) 000004' 005004          CLR    R4
605 000006'          LET R0 := DTABLE(R5)
(4) 000006' 016500 000000          MOV    DTABLE(R5), R0
606 000012'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #DEFRTN
(6) 000012' 042760 000400 000010          BIC    #DEFRTN, DT.STO(R)
607
608 ;+
609 ; GET THE MODULE'S HEADER, THEN GET ITS STATUS WORD, WITH ALL BUT
610 ; THE STATUS BITS MASKED OUT.
611 ;-
612
613 000020'          LET R1 := @DT.PC(R0)
(4) 000020' 017001 000002          MOV    @DT.PC(R0), R1
614 000024'          LET R2 := STAT(R1)
(4) 000024' 016102 000026          MOV    STAT(R1), R2
615 000030'          LET R2 := R2 CLR.BY #STATBITS
(6) 000030' 042702 064757          BIC    #STATBITS, R2
616
617 ;+
618 ; IF THE MODULE IS SBK OR NBK, INDICATE THE A DEFERRED RETURN MUST TAKE
619 ; PLACE.
620 ;-
621
622 000034'          IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
(6) 000034' 020227 000000          CMP    R2, #SBKMOD
(8) 000040' 001403          BEQ    50002$
(6) 000042' 020227 001000          CMP    R2, #NBKMOD
(9) 000046' 001003          BNE    50003$
(6) 000050'          50002$:
623 000050'          LET DT.STO(R0) := DT.STO(R0) SET.BY #DEFRTN
(6) 000050' 052760 000400 000010          BIS    #DEFRTN, DT.STO(R)
624 000056'          ENDIF
(4) 000056'          50003$:
625
626 ;+
627 ; UPDATE THE RETURN PC.
628 ;-
629
630 000056'          LET DT.PC(R0) := DT.PC(R0) + #2
(6) 000056' 062760 000002 000002          ADD    #2, DT.PC(R0)
631
```

```
632 ;+
633 ; INCREMENT THE MODULE'S ITERATION VARIABLE, AND DETERMINE IF AN
634 ; END OF PASS IS IN ORDER (I.E., IF ITERATIONS EQUAL MAX. OR
635 ; INITIAL PASS OF A BKMOD)
636 ;-
637
638 000064' LET ICOUNT(R1) := ICOUNT(R1) + #1
(6) 000064' 005261 000040
639 000070' IF ICOUNT(R1) HIS ICOUNT(R1) OR #TMPID SETIN DT.STO(R0) THEN INC ICOUNT(R1)
(6) 000070' 026161 000040 000036 CMP ICOUNT(R1),ICOUNT
(8) 000076' 103004 BHIS 50004$
(6) 000100' 032760 000002 000010 BIT #TMPID,DT.STO(R0)
(9) 000106' 001501 BEQ 50005$
(6) 000110' 50004$:
640
641 ;+
642 ; ITS END-OF-PASS TIME, SO, MARK THE MODULE AS NOT REQUIRING AN IMMEDIATE
643 ; RETURN FROM THIS CALL, RESET THE ERRORS PER PASS COUNTS,
644 ; BUMP THE END OF PASS COUNT AND RESET THE ITERATION COUNT.
645 ;-
646
647 000110' LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM BIS #CKTIM,DT.ST1(R0)
(6) 000110' 052760 100000 000012
648 000116' LET DT.STO(R0) := DT.STO(R0) SET.BY #DEFRTN BIS #DEFRTN,DT.STO(R0)
(6) 000116' 052760 000400 000010
649 000124' LET SOFPAS(R1) := #0 CLR SOFPAS(R1)
(4) 000124' 005061 000046
650 000130' LET HRDPAS(R1) := #0 CLR HRDPAS(R1)
(4) 000130' 005061 000050
651 000134' LET ICOUNT(R1) := #0 CLR ICOUNT(R1)
(4) 000134' 005061 000040
652 000140' LET PASCNT(R1) := PASCNT(R1) + #1 INC PASCNT(R1)
(6) 000140' 005261 000034
653
654 ;+
655 ; NOTE THAT AN END OF PASS MESSAGE IS IN ORDER
656 ;-
657
658
659 000144' LET R4 := R4 + #1 INC R4
(6) 000144' 005204
660
661 ;+
662 ; IF THE MODULE TYPE IS BK, BSK, OR NBK, DO THE FOLLOWING:
663 ; 1. CLEAR THE RETURN ADDRESS FOR THE MESSAGE ENQUEUE CALL.
664 ; 2. CLEAR THE ACTIVE BIT TO TAKE THE MODULE OFF THE AIR.
665 ; 3. IF NBK, SET THE "DROP" BIT.
666 ; 4. IF SBK OR NBK, UNLOCK THE MODULE QUEUE LIST.
667 ; OTHERWISE, LOAD THE MODULE'S RETURN ADDRESS FOR THE MESSAGE ENQUEUE CALL.
668 ;-
669
670 000146' IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD OR R2 EQ #BKMOD THEN
(6) 000146' 020227 000000 CMP R2,#SBKMOD
(8) 000152' 001406 BEQ 50006$
(6) 000154' 020227 001000 CMP R2,#NBKMOD
(8) 000160' 001403 BEQ 50006$
```

```
(6) 000162' 020227 000020
(9) 000166' 001025
(6) 000170'
671 000170' LET R3 := #0
(4) 000170' 005003
672 000172' LET STAT(R1) := STAT(R1) CLR.BY #ACTBIT
(6) 000172' 042761 004000 000026
673 000200' IF R2 EQ #NBKMOD THEN
(6) 000200' 020227 001000
(9) 000204' 001003
674 000206' LET STAT(R1) := STAT(R1) SET.BY #BIT13
(6) 000206' 052761 020000 000026
675 000214' ENDIF
(4) 000214'
676 000214' IF R2 EQ #SBKMOD OR R2 EQ #NBKMOD THEN
(6) 000214' 020227 000000
(8) 000220' 001403
(6) 000222' 020227 001000
(9) 000226' 001004
(6) 000230'
677 000230' LET DT.STO(R0) := DT.STO(R0) CLR.BY #MODHOLD
(6) 000230' 042760 002000 000010
678 000236' LET R3 := #0
(4) 000236' 005003
679 000240' ENDIF
(4) 000240'
680 000240' ELSE
(4) 000240' 000402
(3) 000242'
681 000242' LET R3 := RSTRT(R1)
(4) 000242' 016103 000112
682 000246' ENDIF
(4) 000246'
683
684
685 ;+
686 ; SET THE MODULE'S EOP BIT, UNLESS THIS IS A BKMOD IN DISGUISE AS A IOMOD
687 ; (TMPIO = 1), IN WHICH CASE WE MUST CHANGE
688 ; THE BKMOD'S STAT BITS BACK FROM IOMOD TO BKMOD AND
689 ; CLEAR TMPIO.
690 ;-
691 000246' IF #TMPIO NOTSETIN DT.STO(R0) THEN
(6) 000246' 032760 000002 000010
(9) 000254' 001004
692 000256' LET XFLAG(R1) := XFLAG(R1) SET.BY #EOPBIT
(6) 000256' 152761 000001 000005
693 000264' ELSE
(4) 000264' 000412
(3) 000266'
694 000266' LET R3 := #0
(4) 000266' 005003
695 000270' LET STAT(R1) := STAT(R1) CLR.BY #IOMOD
(6) 000270' 042761 100000 000026
696 000276' LET STAT(R1) := STAT(R1) SET.BY #BKMOD
(6) 000276' 052761 000020 000026
697 000304' LET DT.STO(R0) := DT.STO(R0) CLR.BY #TMPIO!MODHOLD
```



```
(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,#MSGEQP,#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      #MSGEQP,-(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$:
```

PENDIT - PROCESS ENDIT TRAP CALL  
PENDIT.MAC 14-SEP-78 10:57

MACY11 30A(1052) 20-SEP-78 18:17 PAGE 21-5  
PROCESS ENDIT ROUTINE

SEQ 0785

(2) 000524' 000207  
755  
756  
757 000001

RTS PC

.END ;THAT'S ALL FOLKS!!!



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

SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$SERFL= 000000
TMPID = 000002	WBUFEA= 000136	\$F\$G00= 000400	\$LSTIN= 000001	\$\$FLAG= 000001
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$LSTTA= 000001	\$\$FROM= 000000
UIPAR0= 177640	WBUFQR= 000140	\$F\$INC= 000210	\$NESTL= 177777	\$\$LOC = 000474R
UIPAR1= 177642	WBUFSZ= 000142	\$F\$L00= 000206	SNSK0 = 000300	\$\$LOCN= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	SNSK1 = 000110	\$\$REG = 177777
UIPAR3= 177646	WDTO = 000114	\$F\$ND = 000403	SNSK2 = 000110	\$\$REU= 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$OR = 000320	SNSK3 = 000110	\$\$RTN1= 050000
UIPAR5= 177652	WTWHMI= 000222	\$F\$RTI= 000350	SSAVLE= 177777	\$\$RTN2= 050001
UIPAR6= 177654	XFLAG = 000005	\$F\$RTN= 000300	STAGLE= 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	\$ISKO = 000001	\$\$CASE= 000000	
UIPDR7= 177616	\$F\$DEC= 000220	\$ISK1 = 000001	\$\$DST = 000000	
WASADR= 000104	\$F\$DO = 000340	\$ISK2 = 000001	\$\$ELDC= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 10:19

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0788

3 COMMON EQUATE MODULE  
556 PDATER (COMMON DEFINITIONS AND REFERENCES)  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
582 PDATER (CODE)

```
508 .TITLE PDATER (PROCESS DATERS$ TRAP)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PDATER
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE IS CALLED AS A RESULT OF A DATERS$ TRAP
517 ; EXECUTED BY AN OPTION MODULE. IT ENQUEUES AN ERROR MESSAGE.
518 ;
519 ; INPUTS:
520 ; 1. DTABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ; 1. DT.PC
524 ; 2. DT.CFO
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; NONE
534 ;
535 ; SUBORDINATE ROUTINES CALLED:
536 ; 1. ENQTQ ;ENQUE THE ERROR MESSAGE
537 ; 2. SAVREG
538 ; 3. RESREG
539 ; 4. APTSER
540 ; 5. DRPMOD
541 ;
542 ; FUNCTIONAL SIDE EFFECTS:
543 ; NONE
544 ;
545 ; CALLING SEQUENCE:
546 ; CALL PDATER IN <DT>
547 ; WHERE DT = DATA TABLE ADDRESS
548 ;
549 ; VERSION:
550 ; 0.0
551 ;
552 ; EDIT DATE BY REASON
553 ;
554 ;
```

PDATER (PROCESS DATERS TRAP)  
PDATER.MAC 28-JUL-78 09:24

MACY11 30A(1052) 20-SEP-78 18:19 PAGE 19-1  
PDATER (COMMON DEFINITIONS AND REFERENCES)

SEQ 0790

```
556 .SBTTL PDATER (COMMON DEFINITIONS AND REFERENCES)
557
558 .MCALL STRUCT
559 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
560 000001 $LSTIN=1
561 000001 $LSTTAG=1
562
563 ;*****
564 ;
565 ; REFERENCED BY OTHER MODULES
566 ;
567 .GLOBL PDATER ;MODULE ENTRY POINT
568 ;
569 ;*****
570 ;
571 ; GLOBAL REFERENCES
572 ;
573 .GLOBL ENQTQ ;ENQUES ERROR MSG
574 .GLOBL SAVREG ;
575 .GLOBL RESREG ;
576 .GLOBL APTSER ;APT ROUTINE
577 .GLOBL DRPMOD ;DROP MODULE
578 ;
579 ;*****
580
```

```

582          .SBTTL PDATER (CODE)
583
584 000000'  ROUTINE PDATER <AA>
(2) 000000'
585
586
587          ;+
588          ; THIS ROUTINE ENQUES THE DATA ERROR MESSAGE IN THE TYPE QUEUE.
589          ;
590          ;           R0 = DTABLE ADDRESS
591          ;           R1 = PC+2 OF CALL
592          ;           R2 = PC OF CALL AND IS PASSED AS ADDR OF MSG
593          ;           R3 = RETURN ADDRESS
594          ;           MSGDER = DATA ERROR TYPE MSG CODE
595          ;-
596 000000'  CALL SAVREG
(3) 000000' 004767 000000G
597
598 000004'  LET R0 := AA(R5)
(4) 000004' 016500 000000
599 000010'  LET R1 := DT.PC(R0)
(4) 000010' 016001 000002
600 000014'  LET R4 := @DT.PC(R0)
(4) 000014' 017004 000002
601
602          ;+
603          ; INCREMENT SOFCNT, AND SOFPAS AND IF UNDER APT, CALL APT SOFT
604          ; ERROR ROUTINE
605          ;-
606
607 000020'  LET SOFCNT(R4) := SOFCNT(R4) + #1
(6) 000020' 005264 000042
608 000024'  LET SOFPAS(R4) := SOFPAS(R4) + #1
(6) 000024' 005264 000046
609 000030'  IF #APTPRES SETIN DT.CF0(R0) THEN
(6) 000030' 032760 000200 000014
(9) 000036' 001406
610 000040'  CALL APTSER IN <R0>
(3) 000040' 010546
(4) 000042' 010045
(3) 000044' 004767 000000G
(3) 000050' 012605
611 000052'  ELSE
(4) 000052' 000431
(3) 000054'
612
613          ;+
614          ; THE ENVIRONMENT IS NOT APT.
615          ; IF THE SWITCH REGISTER SAYS DROP THE MODULE BECAUSE OF THE
616          ; ERROR, DO IT.
617          ;-
618
619 000054'  IF #BIT15 SETIN DT.SWR(R0) THEN
(6) 000054' 032760 100000 000056
(9) 000062' 001407
620 000064'  CALL DRPMOD IN <R0,R4>

```

PDATER:

JSR PC, SAVREG

MOV AA(R5), R0

MOV DT.PC(R0), R1

MOV @DT.PC(R0), R4

INC SOFCNT(R4)

INC SOFPAS(R4)

BIT #APTPRES, DT.CF0(50002\$)  
 BEQ

MOV R5, -(SP)  
 MOV R0, -(R5)  
 JSR PC, APTSER  
 MOV (SP)+, R5

BR 50003\$

50002\$:

BIT #BIT15, DT.SWR(R0)  
 BEQ 50004\$

```

(3) 000064' 010546
(5) 000066' 010445
(4) 000070' 010045
(3) 000072' 004767 000000G
(3) 000076' 012605
621 000100' ELSE
(4) 000100' 000416
(3) 000102' 50004$: BR 50005$
622
623
624
625 ;+
626 ; IF THE ERROR COUNT HAS REACHED THE LIMIT, AND IF THE SWITCH REGISTER
627 ; INDICATES THE MODULE SHOULD BE DROPPED, DO IT.
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
(4) 000136' 010102 MOV R1,R2
(6) 000140' 162702 000002 SUB #2,R2
641
642 ;+
643 ; PUT THE OPTION MODULE'S RETURN ADDRESS IN R3.
644 ; -
645
646 000144' LET R3 := R1 + #2
(4) 000144' 010103 MOV R1,R3
(6) 000146' 082703 000002 ADD #2,R3
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)
    
```

(3) 000170' 004767 000000G  
(3) 000174' 012605  
648  
649 000176'  
(3) 000176' 004767 000000G  
650  
651 000202'  
(3) 000202'  
(3) 000202'  
(2) 000202' 000207  
652  
653 000001

CALL RESREG

ENDRTN

.END

JSR PC,ENQTQ  
MOV (SP)+,R5

JSR PC,RESREG

50000\$:  
50001\$:

RTS PC



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

SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$ERFL= 000000
TMPID = 000002	WBUFEA= 000136	\$F\$G00= 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 = 000120R
UIPAR1= 177642	WBUFSZ= 000142	\$F\$L00= 000200	\$NSK0 = 000300	\$\$LOCN= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$REG = 177777
UIPAR3= 177646	WDTO = 000114	\$F\$ND = 000403	\$NSK2 = 000110	\$\$RETU= 000000
UIPAR4= 177650	WTINRE= 000352	\$F\$DR = 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	\$\$TD = 000000
UIPDR3= 177606	\$F\$AND= 000310	\$F\$WHI= 000120	\$TSK2 = 050006	\$\$\$TAG= 050000
UIPDR4= 177610	\$F\$BAD= 000401	\$F\$YES= 000402	\$\$ARGC= 000002	= 000204R
UIPDR5= 177612	\$F\$BLA= 000170	\$IFLEV= 177777	\$\$BYTE= 000403	
UIPDR6= 177614	\$F\$CAS= 000150	\$ISKO = 000001	\$\$CASE= 000000	
UIPDR7= 177616	\$F\$DEC= 000220	\$ISK1 = 000001	\$\$DST = 000000	
WASADR= 000104	\$F\$DQ = 000340	\$ISK2 = 000001	\$\$ELOC= 000402	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:20

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 0796

3 COMMON EQUATE MODULE  
565 PGETPA (COMMON DEFINITIONS & REFERENCES)  
568 000000' .PRINT ;SPMAC: VERSION 1.1  
587 PGETPA (CODE)

```
508 .TITLE PGETPA (GET PHYSICAL ADDRESS MODULE)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PGETPA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE SERVICES GETPA TRAP CALLS.
517 ; IT RETURNS A 16-BIT OR 18-BIT VIRTUAL
518 ; ADDRESS FOR A 16-BIT VIRTUAL ADDRESS GIVEN
519 ; (DEPENDING ON IF KT IS TURNED ON).
520 ; A POINTER TO A 3-WORD AREA IS RECEIVED. THE FIRST
521 ; WORD CONTAINS THE 16-BIT VIRTUAL ADDRESS, THE NEXT TWO
522 ; WORDS WILL CONTAIN, ON RETURN, THE LOW 18-BITS OF THE ADDRESS AND
523 ; THE EXTENDED ADDRESS BITS.
524 ;
525 ;
526 ; INPUTS:
527 ; DTABLE ADDRESS
528 ;
529 ; IMPLICIT INPUTS:
530 ; 1. DT.PC ;DT.PC+2 POINTS TO TABLE CONTAINING VA,PA,EA
531 ; ; WHERE VA = VIRTUAL ADDRESS
532 ; ; PA = PHYSICAL ADDRESS
533 ; ; EA = EXTENDED ADDRESS
534 ; 2. DT.STO
535 ; 3. DT.OFFSET
536 ;
537 ; OUTPUTS:
538 ; NONE
539 ;
540 ; IMPLICIT OUTPUTS:
541 ; 1. PHYSICAL ADDRESS WORD
542 ; 2. EXTENDED ADDRESS WORD
543 ;
544 ; PATHOLOGICAL CONNECTIONS:
545 ; NONE
546 ;
547 ; SUBORDINATE ROUTINES CALLED:
548 ; SAVREG ;SAVE REGISTERS
549 ; RESREG ;RESTORE REGISTERS
550 ;
551 ; FUNCTIONAL SIDE EFFECTS:
552 ; NONE
553 ;
554 ; CALLING SEQUENCE:
555 ; CALL PGETPA IN <DT>
556 ; WHERE DT = DTABLE ADDRESS
557 ;
558 ; VERSION:
559 ; 0.0
560 ;
561 ; EDIT DATE BY REASON
562 ;
563 ;
```

PGETPA (GET PHYSICAL ADDRESS MODULE)  
PGETPA.MAC 28-JUL-78 09:24

MACY11 30A(1052) 20-SEP-78 18:20 PAGE 19-1  
PGETPA (COMMON DEFINITIONS & REFERENCES)

SEQ 0798

565  
566  
567  
568  
(1)  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585

000000'  
000000'  
000001  
000001

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

.MCALL STRUCT
STRUCT
.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

;*****
```

```

587      .SBTTL PGETPA (CODE)
588
589      000000'      ROUTINE PGETPA <DT>
(2)      000000'
590
591      ;+
592      ; SAVE REGISTERS
593      ; -
594
595      000000'      CALL SAVREG
(3)      000000' 004767 000000G      JSR      PC, SAVREG
596
597
598      ;+
599      ; SET R0 TO THE START OF THE DATA TABLE
600      ; -
601
602      000004'      LET R0 := DT(R5)
(4)      000004' 016500 000000      MOV      DT(R5), R0
603
604
605      ;+
606      ; SET R1 TO THE START OF THE ADDRESS TABLE.
607      ; -
608
609      000010'      LET R2 := DT.PC(R0) + #2
(4)      000010' 016002 000002      MOV      DT.PC(R0), R2
(6)      000014' 062702 000002      ADD      #2, R2
610
611      000020'      LET R1 := (R2)
(4)      000020' 011201      MOV      (R2), R1
612
613
614      ;+
615      ; UPDATE DT.PC TO REFLECT THE OPTION MODULE'S RETURN ADDRESS.
616      ; -
617
618      000022'      LET DT.PC(R0) := DT.PC(R0) + #4
(6)      000022' 062760 000004 000002      ADD      #4, DT.PC(R0)
619
620
621      ;+
622      ; IF MEMORY MANAGEMENT IS NOT TURNED ON, SIMPLY MOVE THE
623      ; 16-BIT VIRTUAL ADDRESS TO THE OUTPUT POSITION OF THE TABLE, CLEAR THE
624      ; HIGH BIT TABLE POSITION, AND RETURN.
625      ; -
626
627      000030'      IF #KTSTAT NOTSETIN DT.ST0(R0) THEN
(6)      000030' 032760 000020 000010      BIT      #KTSTAT, DT.ST0(R
(9)      000036' 001004      BNE      50002$
628      000040'      LET (R1) := (R1)+
(4)      000040' 012111      MOV      (R1)+, (R1)
629      000042'      LET 2(R1) := #0
(4)      000042' 005061 000002      CLR      2(R1)
630
631      000046'      ELSE

```

```
(4) 000046' 000446
(3) 000050'
632
633
634
635
636
637
638
639
640
641
642
643
644
645 000050'
(4) 000050' 011102
646 000052'
(6) 000052' 000302
647 000054'
(7) 000054' 006202
(7) 000056' 006202
(7) 000060' 006202
(7) 000062' 006202
648 000064'
(6) 000064' 042702 177761
649 000070'
(4) 000070' 012703 172340
(6) 000074' 060203
650
651
652
653
654
655
656 000076'
(4) 000076' 011302
657
658
659
660
661
662
663
664 000100'
(6) 000100' 166002 000070
665
666
667
668
669
670
671
672
673
674 000104'
```

```

;+
; ELSE MEMORY MANAGEMENT IS ON.
;-

;+
; DETERMINE THE PAR REGISTER REFERENCED BY THE VIRTUAL ADDRESS
; BY PLACING THE VA IN R2, DETERMINING THE ADDRESS
; OF THE PROPER PAR REGISTER, AND PLACING THE ADDRESS OF THIS
; REGISTER IN R3.
;-

          LET R2 := (R1)
                                MOV      (R1),R2
          LET R2 := SWAP R2
                                SWAB    R2
          LET R2 := R2 SHIFT #-4
                                ASR     R2
                                ASR     R2
                                ASR     R2
                                ASR     R2
          LET R2 := R2 CLR.BY #177761
                                BIC     #177761,R2
          LET R3 := #KIPAR0 + R2
                                MOV     #KIPAR0,R3
                                ADD     R2,R3

;+
; NOW GET THE CONTENTS OF THIS PAR REGISTER.
;-

          LET R2 := (R3)
                                MOV     (R3),R2

; WE NOW HAVE THE 22-BIT PHYSICAL ADDRESS IN PAR FORMAT.
; MAKE THIS A VIRTUAL 19-BIT VALUE (STILL PAR FORMAT) BY SUBTRACTING
; THE CURRENT OFFSET VALUE.
;-

          LET R2 := R2 - DT.OFFSET(R0)
                                SUB     DT.OFFSET(R0),R2

;+
; NOW CONVERT THIS VALUE TO 18 BITS.
; FIRST FORM THE LOW-ORDER 16-BITS BY SHIFTING THE HIGH
; ORDER BITS OUT OF THE PAR CONTENTS AND ADDING THE LOW 12 BITS OF THE
; ORIGINAL VA. THEN IF THERE IS OVERFLOW, CARRY IT TO THE HIGH BITS.
;-

          LET R3 := R2 SHIFT #6
```

675	000122'		LET R4 := (R1) CLR.BY #160000	MOV	R2,R3
(4)	000122'	011104		ASL	R3
(6)	000124'	042704	160000	ASL	R3
676	000130'		LET 2(R1) := R3 + R4	ASL	R3
(4)	000130'	010361	000002	ASL	R3
(6)	000134'	060461	000002	ASL	R3
677	000140'		IFCOND CS THEN	ASL	R3
(6)	000140'	103002		ASL	R3
678	000142'		LET R2 := R2 + #200	ASL	R3
(6)	000142'	062702	000200		
679	000146'		ENDIF	BCC	50004\$
(4)	000146'			ADD	#200,R2
680					
681					
682					
683					
684					
685					
686					
687	000146'		LET R2 := SWAP R2		
(6)	000146'	000302		SWAB	R2
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		
(4)	000164'				50003\$:
694					
695					
696					
697					
698					
699					
700	000164'		CALL RESREG		
(3)	000164'	004767	000000G	JSR	PC,RESREG
701					
702	000170'		ENDRTN		
(3)	000170'				50000\$:
(3)	000170'				50001\$:
(2)	000170'	000207		RTS	PC
703					
704		000001	.END		



ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124
ACTBIT= 004000	CTRL0 = 000017	ERRTP= 000106	MSGCKD= 000010	RDSERV= 000101
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RELTIM= 010000
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGEQP= 000013	RESREG= ***** G
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES1 = 000056
ASTAT = 000104	DT = 000000	HRDCNT= 000044	MSGHNG= 000022	RES2 = 000060
AUTO = 000010	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RICHAR= 000007
AUTOST= 020000	DT.AP = 000100	ICDNT = 000035	MSGMAP= 000021	RPTDAT= 002000
AWAS = 000110	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RSTRT = 000112
BIT0 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUBOUT= 000177
BIT00 = 000001	DT.CF0= 000014	IE = 000100	MSGPRM= 177776	RUNMOD= 100000
BIT01 = 000002	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	R5VALU= 001740
BIT02 = 000004	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SAM = 075464
BIT03 = 000010	DT.ESI= 000044	INHPR= 020000	MSGSKE= 000003	SAVREG= ***** G
BIT04 = 000020	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBADR = 000102
BIT05 = 000040	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SBKMOD= 000000
BIT06 = 000100	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SBKSEL= 010000
BIT07 = 000200	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.ADR= 000006
BIT08 = 000400	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.ALC= 000014
BIT09 = 001000	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.APC= 000016
BIT1 = 000002	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CKL= 000002
BIT10 = 002000	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.CKP= 000004
BIT11 = 004000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SC.CLO= 000000
BIT12 = 010000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.HLD= 000010
BIT13 = 020000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SC.SCA= 000012
BIT14 = 040000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SENDLS= 177777
BIT15 = 100000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SOFCNT= 000042
BIT2 = 000004	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SOFPAS= 000046
BIT3 = 000010	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPACE = 000040
BIT4 = 000020	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SPOINT= 000032
BIT5 = 000040	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SPVALU= 002200
BIT6 = 000100	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR0 = 177572
BIT7 = 000200	DT.REL= 000040	KIPDR1= 172302	PGETPA 000000RG	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	KTERRO= 000040	PRO = 000000	SVR0 = 000062
CASTAT= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CDERCT= 000146	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CDWDCT= 000144	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR3 = 000070
CKTIM = 100000	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR4 = 000072
CLKPRE= 000001	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SVR5 = 000074
CONFIG= 000056	ECCMEM= 000100	MAPSTA= 000200	PSW = 177776	SVR6 = 000076
CQDVF = 000001	ECCSTA= 000010	MED = 076600	RANNUM= 000054	SYS CNT= 000052
CR = 000015	ENBEOP= 010000	MEMPAS= 040000	RBUFVA= 000130	SYSERR= 000100
CSRA = 000100	ENBNUL= 000001	MODEXH= 004000	RBUFPA= 000126	TMPIO = 000002
CSRC = 000102	ENDLST= 000000	MODHOL= 002000	RBUFSZ= 000132	TQOVF = 000002

UIPAR0= 177640	WBUFEA= 000136	\$F\$DD = 000340	\$ISK0 = 000001	\$\$DST = 000000
UIPAR1= 177642	WBUFPA= 000134	\$F\$FAL= 000405	\$ISK1 = 000001	\$\$ELOC= 000402
UIPAR2= 177644	WBUFRQ= 000140	\$F\$GOD= 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\$L0D= 000200	\$NESTL= 177777	\$\$LOC = 000140R
UIPAR6= 177654	WTINRE= 000352	\$F\$NAM= 000160	\$NSKO = 000300	\$\$LOCN= 000000
UIPAR7= 177656	WTWHMI= 000222	\$F\$ND = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:21  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0804

3 COMMON EQUATE MODULE  
550 COMMON DEFINITIONS AND REFERENCES  
552 000000' .PRINT ;SPMAC: VERSION 1.1  
570 PMAP22 ROUTINE

```
508 .TITLE PMAP22 - PROCESS MAP22
509 .IDENT /VO.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     PMAP22
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     PROCESSES MAP22 TRAP CALLS
517 ;
518 ; INPUTS:
519 ;     DATA TABLE
520 ;
521 ; IMPLICIT INPUTS:
522 ;     DT.PC
523 ;
524 ; OUTPUTS:
525 ;     NONE
526 ;
527 ; IMPLICIT OUTPUTS:
528 ;     DT.PC
529 ;
530 ; PATHOLOGICAL CONNECTIONS:
531 ;     NONE
532 ;
533 ; SUBORDINATE MODULES CALLED:
534 ;     UNIPA ;GET PHYS. ADDR. FROM UNIBUS MAP
535 ;
536 ; FUNCTIONAL SIDE EFFECTS:
537 ;     NONE
538 ;
539 ; CALLING SEQUENCE:
540 ;     CALL PMAP22 IN <A>
541 ;         A=ADDRESS OF DATA TABLE
542 ;
543 ; VERSION:
544 ;     0.0
545 ;
546 ;     EDIT          BY          DATE          REASON
547 ;
548 ;--
```

```
550 .SBTTL COMMON DEFINITIONS AND REFERENCES
551 .MCALL STRUCT
552 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
553 000001 $LSTIN=1
554 000001 $LSTTAG=1
555
556 ;
557 ;*****
558 ;
559 ; REFERENCED BY OTHER MODULES
560 ;
561 .GLOBL PMAP22 ;MODULE ENTRY POINT
562 ;
563 ;*****
564 ;
565 ; GLOBAL REFERENCES
566 ;
567 .GLOBL UNIPA ;GET PHYS. ADDR. FROM UNIBUS MAP
568 ;
```

```

570          .SBTTL PMAP22 ROUTINE
571
572 000000'    ROUTINE PMAP22 <TABL>
(2) 000000'
573
574
575          ;+
576          ; SAVE REGISTERS
577          ; -
578
579 000000'    PUSH R0,R1
(2) 000000' 010046          MOV     R0,-(SP)
(3) 000002' 010146          MOV     R1,-(SP)
580
581
582          ;+
583          ; SET R0 TO START OF DATA TABLE
584          ; -
585
586 000004'    LET R0 := TABL(R5)
(4) 000004' 016500 000000          MOV     TABL(R5),R0
587
588
589          ;+
590          ; GET THE ADDRESS OF THE TABLE REQUIRED BY UNIPA. THIS TABLE
591          ; CONTAINS THE 18-BIT ADDRESS TO BE CONVERTED ALONG WITH A STORAGE
592          ; AREA FOR THE 22-BIT RESULT.
593          ; -
594
595 000010'    LET R1 := DT.PC(R0) + #2
(4) 000010' 016001 000002          MOV     DT.PC(R0),R1
(6) 000014' 062701 000002          ADD     #2,R1
596
597
598          ;+
599          ; CALL UNIPA
600          ; -
601
602 000020'    CALL UNIPA IN <(R1)>
(3) 000020' 010546          MOV     R5,-(SP)
(4) 000022' 011145          MOV     (R1),-(R5)
(3) 000024' 004767 000000G          JSR     PC,UNIPA
(3) 000030' 012605          MOV     (SP)+,R5
603
604
605          ;+
606          ; UPDATE DT.PC TO REFLECT THE OPTION MODULE'S RE-ENTRY
607          ; ADDRESS. (THAT IS, JUMP OVER THE INPUT ARGUMENT OF
608          ; THE TRAP CALL.)
609          ; -
610
611 000032'    LET DT.PC(R0) := DT.PC(R0) + #4
(6) 000032' 062760 000004 000002          ADD     #4,DT.PC(R0)
612
613
614          ;+

```

```
615 ; RESTORE REGISTERS
616 ; -
617
618 000040' POP R1,R0
(2) 000040' 012601 MOV (SP)+,R1
(3) 000042' 012600 MOV (SP)+,R0
619
620
621 ; +
622 ; GOOD-BYE.
623 ; -
624
625 000044' ENDRTN
(3) 000044'
(3) 000044' 50000$:
(2) 000044' 000207 50001$:
626 RTS PC
627 000001 .END
```

ACSR = 000102	CTRLC = 000003	ERRTYP= 000106	MSGCKD= 000010	RDSERV= 000101
ACTBIT= 004000	CTRL0 = 000017	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
APTFER= 000004	DEFRTN= 000400	EVNTPF= 000202	MSGGECH= 177777	RELTIM= 010000
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGEOP= 000013	RES1 = 000056
ASB = 000106	DROPMO= 100000	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AUTO = 000010	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
AUTOST= 020000	DT,APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUBOUT= 000177
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT0 = 000001	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	RSVALU= 001740
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBADR = 000102
BIT02 = 000004	DT.ESI= 000044	INHPRP= 020000	MSGSKE= 000003	SBKMOD= 000000
BIT03 = 000010	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBKSEL= 010000
BIT04 = 000020	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SC.ADR= 000006
BIT05 = 000040	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.ALC= 000014
BIT06 = 000100	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.APC= 000016
BIT07 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKL= 000002
BIT08 = 000400	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CKP= 000004
BIT09 = 001000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CLO= 000000
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.HLD= 000010
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SC.SCA= 000012
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SENDLS= 177777
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	QWEN = 024020	SOFCNT= 000042
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOPPAS= 000046
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPACE = 000040
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPOINT= 000032
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPVALU= 002200
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SR0 = 177572
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR1 = 177574
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR2 = 177576
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PMP22 000000RG	SR3 = 172516
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PRI0 = 000000	STAT = 000026
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRI1 = 000040	STATBI= 064757
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI4 = 000200	STAT1 = 000027
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI5 = 000240	SUSPND= 000001
BKMOD = 000020	DT.ST0= 000010	KIPDR6= 172314	PRI6 = 000300	SVR0 = 000062
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRI7 = 000340	SVR1 = 000064
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PR0 = 000000	SVR2 = 000066
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR3 = 000070
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR4 = 000072
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR5 = 000074
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR6 = 000076
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SYSCNT= 000052
CLKPRE= 000001	ECCMEM= 000100	MAPSTA= 000200	PSW = 177776	SYSERR= 000100
CONFIG= 000056	ECCSTA= 000010	MED = 076600	RANNUM= 000054	TABL = 000000
CQOVF = 000001	ENBEOP= 010000	MEMPAS= 040000	RBUFEA= 000130	TMPID = 000002
CR = 000015	ENBNUL= 000001	MODEXH= 004000	RBUFPA= 000126	TQOVF = 000002
CSRA = 000100	ENDLST= 000000	MODHOL= 002000	RBUFSZ= 000132	UIPAR0= 177640
CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124	UIPAR1= 177642



UIPAR2= 177644	WBUFEA= 000136	\$F\$DEC= 000220	\$F\$YES= 000402	\$F\$FLAG= 000000
UIPAR3= 177646	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$F\$FROM= 000000
UIPAR4= 177650	WBUFRQ= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$F\$LOC = 000000
UIPAR5= 177652	WBUFSZ= 000142	\$F\$G00= 000400	\$LSTIN= 000001	\$F\$LOCN= 000000
UIPAR6= 177654	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$F\$REG = 177777
UIPAR7= 177656	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$F\$RETN= 000000
UIPDR0= 177600	WTINRE= 000352	\$F\$L00= 000200	\$NSKO = 000300	\$F\$RTN1= 050000
UIPDR1= 177602	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$F\$RTN2= 050001
UIPDR2= 177604	XFLAG = 000005	\$F\$ND = 000403	\$TAGLE= 177777	\$F\$SRC = 000000
UIPDR3= 177606	XOFF = 000023	\$F\$OR = 000320	\$TAGNU= 050002	\$F\$TGSV= 000000
UIPDR4= 177610	XON = 000021	\$F\$RTI= 000350	\$TEMP = 000300	\$F\$TGS1= 000000
UIPDR5= 177612	\$BGNLE= 177777	\$F\$RTN= 000300	\$ARGC= 000002	\$F\$TGS2= 000000
UIPDR6= 177614	\$ERFLG= 000400	\$F\$SEL= 000140	\$BYTE= 000000	\$F\$TD = 000001
UIPDR7= 177616	\$F\$AND= 000310	\$F\$THE= 000330	\$CASE= 000000	\$F\$TAG= 050000
UNIPA = ***** G	\$F\$BAD= 000401	\$F\$TRU= 000404	\$DST = 000000	. = 000046R
WASADR= 000104	\$F\$BLA= 000170	\$F\$UNT= 000130	\$EL0C= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:22  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0811

3 COMMON EQUATE MODULE  
526 KPON/KPOFF PROCESS THE KEYBOARD COMMANDS: 'PON'/'POFF'  
584 COMMON DEFINITIONS AND REFERENCES FOR PONOF  
587 000000' .PRINT ;SPMAC: VERSION 1.1  
624 KPON/KPOFF ROUTINE

```
508 .TITLE PONOF PROCESS THE KEYBOARD COMMANDS 'PON' AND 'POFF'  
509 .IDENT /V0.0/  
510  
511 ;++  
512 ; MODULE PACKAGE NAME:  
513 ; PONOF  
514 ;  
515 ; FUNCTIONAL DESCRIPTION:  
516 ; THIS MODULE PACKAGE CONTAINS TWO ROUTINES:  
517 ; 1. KPON - PROCESS THE KEYBOARD COMMAND 'PON'  
518 ; 2. KPOFF - PROCESS THE KEYBOARD COMMAND 'POFF'  
519 ;  
520 ; VERSION:  
521 ; 0.0  
522 ;  
523 ; EDIT BY DATE REASON  
524 ;--
```

```
526 .SBTTL KPON/KPOFF PROCESS THE KEYBOARD COMMANDS: 'PON'/'POFF'  
527 .IDENT /V0.0/  
528  
529 ;++  
530 ; MODULE NAME:  
531 ; KPON/KPOFF  
532 ;  
533 ; FUNCTIONAL DESCRIPTION:  
534 ; THIS ROUTINE PROCESSES THE 'PON'/'POFF' KEYBOARD COMMANDS.  
535 ; IT WILL FIRST VERIFY THAT NO JUNK ARGUMENTS EXIST.  
536 ; IT WILL THEN VERIFY THAT PARITY IS PRESENT ON THE SYSTEM.  
537 ; ALL PARITY CSR'S IN THE PARITY TABLE WILL HAVE THE APPROPRIATE BITS  
538 ; SET OR CLEARED DEPENDING ON COMMAND ISSUED. THE  
539 ; REGISTERS MAY BE NORMAL PARITY OR ECC(ERROR CORRECTION CIRCUITRY).  
540 ; IF CACHE THEN THE CACHE CONTROL  
541 ; REGISTER IS CLEARED. IF THE CPU IS AN 11/70, BITS 0,1  
542 ; IN THE MEMORY CONTROL REGISTER ARE CLEARED. THE PARITY ENABLED  
543 ; MESSAGE WILL BE STUFFED. IF INDIRECT ECC CSR'S THEN THIS COMMAND IS  
544 ; NOT VALID IN BUSY MODE.  
545 ;  
546 ; INPUTS:  
547 ; 1. ADDRESS OF DATA TABLE  
548 ; 2. COMMAND DECODE BUFFER POINTER  
549 ;  
550 ; IMPLICIT INPUTS:  
551 ; DT.PTA  
552 ; DT.CFO  
553 ; DT.STO  
554 ;  
555 ; OUTPUTS:  
556 ; NONE  
557 ;  
558 ; IMPLICIT OUTPUTS:  
559 ; DT.KBRSP  
560 ;  
561 ; PATHOLOGICAL CONNECTIONS:  
562 ; CM.RUN - COMMAND NOT ALLOWED IN RUN MODE MESSAGE  
563 ; CM.ARG - ILLEGAL ARGUMENT MESSAGE  
564 ;  
565 ; SUBORDINATE ROUTINES CALLED:  
566 ; ARGCHK - CHECK AN ARGUMENT  
567 ; ICSROO - TURN ON/OFF INDIRECT ECC CSRS  
568 ;  
569 ; FUNCTIONAL SIDE EFFECTS:  
570 ; NONE  
571 ;  
572 ; CALLING SEQUENCE:  
573 ; CALL KPON IN <DTADR,CMDBUF>  
574 ; WHERE: DTADR = ADDRESS OF DATA TABLE  
575 ; CMDBUF = COMMAND DECODE BUFFER POINTER  
576 ;  
577 ; VERSION:  
578 ; 0.0  
579 ;  
580 ; EDIT DATE BY REASON  
581 ;
```

```

583
584 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR PONOF
585
586 .MCALL STRUCT
587 000000' STRUCT
588 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
589
590 000001 $LSTTAG=1
591 000001 $LSTIN=1
592
593 ;*****
594 ; REFERENCED BY OTHER MODULES:
595 ;
596 .GLOBL KPON ;KPON ENTRY POINT
597 .GLOBL KPOFF ;KPOFF ENTRY POINT
598 ;
599 ;*****
600 ; GLOBAL REFERENCES:
601 ;
602 .GLOBL CCNTRL ;CACHE OR MEMORY CONTROL REG
603 .GLOBL ARGCHK ;CHECK AN ARGUMENT
604 .GLOBL ICSROO ;INDIRECT ECC CSR ROUTINE
605 .GLOBL CM.ARG ;'INVALID OR ILLEGAL ARGUMENT' MSG
606 .GLOBL CM.RUN ;'ILLEGAL COMMAND IN BUSY MODE' MSG
607 .GLOBL KONTRL ;CONTROL WORD
608 ;*****
609 ;
610 ; LOCAL STORAGE:
611 ;
612 000000' 000000 PO.FLAG: .WORD 0 ;PON/POFF FLAG
613 000002' 047516 050040 051101 PO.NUN: .ASCIZ /NO PARITY%/
614 000010' 052111 022531 000
615 000015' 120 051101 052111 PO.OFF: .ASCII /PARITY MEMORY OFF%/
616 000022' 020131 042515 047515
617 000030' 054522 047440 043106
618 000036' 045
619 000037' 000 PO.MB0: .BYTE 0 ;WILL BE STUFFED WITH A % IF
620 ;ECC MEM IS PRESENT
621 000040' 041505 020103 042515 PO.EOF: .ASCIZ /ECC MEMORY OFF%/
622 000046' 047515 054522 047440
623 000054' 043106 000045
624 000060' 040520 044522 054524 PO.ON: .ASCII /PARITY MEMORY ON%/
625 000066' 046440 046505 051117
626 000074' 020131 047117 045
627 000101' 000 PO.MB1: .BYTE 0 ;WILL BE STUFFED WITH A % IF
628 ;ECC MEM IS PRESENT
629 000102' 041505 020103 042515 PO.EON: .ASCIZ /ECC MEMORY ON%/
630 000110' 047515 054522 047440
631 000116' 022516 000
632 000122' .EVEN
  
```

```

624          .SBTTL KPON/KPOFF ROUTINE
625
626 000122'   ROUTINE KPON <DTADR,BUFPTR>
(2) 000122'
627
628          ;+
629          ; SET THE FLAG TO REFLECT PON COMMAND
630          ; -
631
632 000122'   LET PO.FLAG := #0
(4) 000122' 005067 177652
633 000126'   INLINE <BR PARITY>
(2) 000126' 000404
634 000130'   ENDRTN
(3) 000130'
(3) 000130'
(2) 000130' 000207
635
636
637 000132'   ROUTINE KPOFF <DTADR,BUFPTR>
(2) 000132'
638
639          ;+
640          ; SET THE FLAG TO REFLECT POFF COMMAND
641          ; -
642
643 000132'   LET PO.FLAG := #1
(4) 000132' 012767 000001 177640
644
645          ;+
646          ; SAVE REGISTERS, DTABLE ADDRESS AND BUFPTR
647          ; -
648 000140'   INLINE <PARITY:>
(2) 000140'
649 000140'   PUSH R0,R1
(2) 000140' 010046
(3) 000142' 010146
650 000144'   LET R0 := DTADR(R5)
(4) 000144' 016500 000000
651 000150'   LET R1 := BUFPTR(R5)
(4) 000150' 016501 000002
652
653          ;+
654          ; PLACE MESSAGE TERMINATORS IN MESSAGES
655          ; -
656
657 000154'   LET PO.MB0 :B= #0
(4) 000154' 105067 177657
658 000160'   LET PO.MB1 :B= #0
(4) 000160' 105067 177715
659
660          ;+
661          ; INSURE NO JUNK ARGUMENTS EXIST
662          ; -
663
664 000164'   CALL ARGCHK IN <R1> OUT <R1>
    
```

KPON:

CLR PO.FLAG  
BR PARITY

50000\$:  
50001\$:

RTS PC

KPOFF:

MOV #1,PO.FLAG

PARITY:

MOV R0,-(SP)  
MOV R1,-(SP)

MOV DTADR(R5),R0

MOV BUFPTR(R5),R1

CLRB PO.MB0

CLRB PO.MB1

```

(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
(6) 000204' 103404          BCS      50002$
667
668
669      ;+
669      ; IT'S NOT A CR SO STUFF ERROR MSG AND SCRAM...
670      ; -
671
672 000206'          LET DT.KBRSP(R0) := #CM.ARG
(4) 000206' 012760 000000G 000022      MOV      #CM.ARG,DT.KBRSP
673
674 000214'          ELSE
(4) 000214' 000564          BR      50003$
(3) 000216'          50002$:
675
676      ;+
677      ; IT'S A CR SO SEE IF PARITY OR ECC EXISTS AT ALL....
678      ; -
679
680 000216'          IF #PARPRES!ECCMEM NOTSETIN DT.CFO(R0) THEN
(6) 000216' 032760 002100 000014      BIT      #PARPRES!ECCMEM,
(9) 000224' 001004          BNE      50004$
681
682      ;+
683      ; NO PARITY..OR ECC... STUFF ERROR MSG AND LEAVE....
684      ; -
685
686 000226'          LET DT.KBRSP(R0) := #PO.NUN
(4) 000226' 012760 000002' 000022      MOV      #PO.NUN,DT.KBRSP
687
688 000234'          ELSE
(4) 000234' 000554          BR      50005$
(3) 000236'          50004$:
689
690      ;+
691      ; DETERMINE IF INDIRECT PARITY REGISTERS IF NOT THEN GO CHECK TABLE.
692      ; -
693
694 000236'          IF #INDPAR NOTSETIN DT.CFO(R0) THEN
(6) 000236' 032760 000040 000014      BIT      #INDPAR,DT.CFO(R
(9) 000244' 001016          BNE      50006$
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)
(4) 000246' 016001 000064          MOV      DT.PTA(R0),R1

```

```

703 000252'                               WHILE (R1) NE #0 DO
(4) 000252'
(6) 000252' 005711                          50007$: TST (R1)
(9) 000254' 001411                          BEQ 50010$
704
705 ;+
706 ; DETERMINE WHETHER PON OR POFF COMMAND AND SET OR CLEAR APPROPRIATE BITS
707 ;-
708
709 000256'                               IF PO.FLAG EQ #0 THEN
(6) 000256' 005767 177516                      TST PO.FLAG
(9) 000262' 001003                      BNE 50011$
710 000264'                               LET @(R1)+ := @(R1)+ SET.BY #3
(6) 000264' 052731 000003                      BIS #3,@(R1)+
711 000270'                               ELSE
(4) 000270' 000402
(3) 000272'                               50011$: BR 50012$
712 000272'                               LET @(R1)+ := @(R1)+ CLR.BY #3
(6) 000272' 042731 000003                      BIC #3,@(R1)+
713 000276'                               ENDIF
(4) 000276'
714 000276'                               ENDDO
(4) 000276' 000765
(3) 000300'                               50012$: BR 50007$
715                                     50010$:
716 000300'                               ELSE
(4) 000300' 000420
(3) 000302'                               50013$: BR 50013$
717                                     50006$:
718 ;+
719 ; THERE ARE INDIRECT ECC CSRS THEN IF IN RUN MODE DO NOT LOOK BUT RETURN
720 ;-
721
722 000302'                               IF #RUNMODE SETIN DT.STO(R0) THEN
(6) 000302' 032760 100000 000010                      BIT #RUNMODE,DT.STO(
(9) 000310' 001405                      BEQ 50014$
723 000312'                               LET DT.KBRSP(R0) := #CM.RUN
(4) 000312' 012760 000000G 000022                      MOV #CM.RUN,DT.KBRSP
724 000320'                               INLINE <BR 100$>
(2) 000320' 000522                      BR 100$
725 000322'                               ELSE
(4) 000322' 000407
(3) 000324'                               50014$: BR 50015$
726                                     50014$:
727
728 ;+
729 ; IT IS OKAY TO GO TO INDIRECT CSRS
730 ; SEND FLAG A '0' TO INDICATE PON COMMAND
731 ; OR SEND FLAG A '1' TO INDICATE POFF COMMAND
732 ;-
733 000324'                               CALL ICSROO IN <R0,PO.FLAG>
(3) 000324' 010546                      MOV R5,-(SP)
(5) 000326' 016745 177446                      MOV PO.FLAG,-(R5)
(4) 000332' 010045                      MOV R0,-(R5)
(3) 000334' 004767 000000G                      JSR PC,ICSROO
(3) 000340' 012605                      MOV (SP)+,R5

```



```

734 000342'                               ENDIF
(4) 000342'                               50015$:
735 000342'                               ENDIF
(4) 000342'                               50013$:
736
737
738                                     ;+
739                                     ; IF CACHE PRESENT .....
740                                     ; -
741 000342'                               IF #CAPRES SETIN DT.CFO(R0) THEN
(6) 000342' 032760 000004 000014          BIT      #CAPRES,DT.CFO(R
(9) 000350' 001415                          BEQ      50016$
742
743                                     ;+
744                                     ; IF PON COMMAND TURN ON CACHE PARITY OTHERWISE
745                                     ; MUST BE A PARITY POFF COMMAND
746                                     ; -
747
748 000352'                               IF PO.FLAG EQ #0 THEN
(6) 000352' 005767 177422                  TST      PO.FLAG
(9) 000356' 001004                          BNE      50017$
749 000360'                               LET KONTRL := KONTRL CLR.BY #203
(6) 000360' 042767 000203 000000G          BIC      #203,KONTRL
750 000366'                               ELSE
(4) 000366' 000403                          BR       50020$
(3) 000370'                               50017$:
751 000370'                               LET KONTRL := KONTRL SET.BY #203
(6) 000370' 052767 000203 000000G          BIS      #203,KONTRL
752 000376'                               ENDIF
(4) 000376'                               50020$:
753 000376'                               LET @CCNTRL := KONTRL
(4) 000376' 016777 000000G 000000G          MOV      KONTRL,@CCNTRL
754
755 000404'                               ENDIF
(4) 000404'                               50016$:
756                                     ;+
757                                     ; NOW STUFF THE APPROPRIATE BITS IN DT.STO AND LOAD THE ASSOCIATE MESSAGES
758                                     ; -
759
760
761 000404'                               IF #PARPRES SETIN DT.CFO(R0) THEN
(6) 000404' 032760 002000 000014          BIT      #PARPRES,DT.CFO(
(9) 000412' 001445                          BEQ      50021$
762 000414'                               IF #ECCMEM SETIN DT.CFO(R0) THEN
(6) 000414' 032760 000100 000014          BIT      #ECCMEM,DT.CFO(R
(9) 000422' 001420                          BEQ      50022$
763 000424'                               IF PO.FLAG EQ #0 THEN
(6) 000424' 005767 177350                  TST      PO.FLAG
(9) 000430' 001007                          BNE      50023$
764 000432'                               LET PO.MB1 :B= #'%
(4) 000432' 112767 000045 177441          MOVB     #'%,PO.MB1
765 000440'                               LET DT.STO(R0) := DT.STO(R0) SET.BY #ECCSTAT
(6) 000440' 052760 000010 000010          BIS      #ECCSTAT,DT.STO(
766 000446'                               ELSE
(4) 000446' 000406                          BR       50024$
(3) 000450'                               50023$:

```

```

767 000450'          LET PO.MBO :B= #'%
(4) 000450' 112767 000045 177361          MOVB  #'%,PO.MBO
768 000456'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #ECCSTAT
(6) 000456' 042760 000010 000010          BIC   #ECCSTAT,DT.STO(
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.STO(R0) := DT.STO(R0) SET.BY #PARSTAT
(6) 000500' 052760 000100 000010          BIS   #PARSTAT,DT.STO(
774 000506'          ELSE
(4) 000506' 000406          BR    50026$
(3) 000510'          50025$:
775 000510'          LET DT.KBRSP(R0) := #PO.OFF
(4) 000510' 012760 000015' 000022          MOV   #PO.OFF,DT.KBRSP
776 000516'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #PARSTAT
(6) 000516' 042760 000100 000010          BIC   #PARSTAT,DT.STO(
777 000524'          ENDIF
(4) 000524'          50026$:
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.STO(R0) := DT.STO(R0) SET.BY #ECCSTAT
(6) 000542' 052760 000010 000010          BIS   #ECCSTAT,DT.STO(
782 000550'          ELSE
(4) 000550' 000406          BR    50031$
(3) 000552'          50030$:
783 000552'          LET DT.KBRSP(R0) := #PO.EOF
(4) 000552' 012760 000040' 000022          MOV   #PO.EOF,DT.KBRSP
784 000560'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #ECCSTAT
(6) 000560' 042760 000010 000010          BIC   #ECCSTAT,DT.STO(
785 000566'          ENDIF
(4) 000566'          50031$:
786          ENDIF
787          ENDIF
(4) 000566'          50027$:
788          ENDIF
(4) 000566'          50005$:
789          ENDIF
790          ENDIF
(4) 000566'          50003$:
791
792
793          ;+
794          ; CLEAN UP MESS...
795          ;-
```

```
796
797 000566'          INLINE <100$:>
(2) 000566'
798 000566'          POP R1,R0
(2) 000566' 012601
(3) 000570' 012600
799
800 000572'          ENDRTN
(3) 000572'
(3) 000572'          50000$:
(2) 000572' 000207          50001$:
801          000001          RTS      PC
          .END
```

ACSR = 000102	CONFIG= 000056	ECCMEM= 000100	KTSTAT= 000020	PRI0 = 000000
ACTBIT= 004000	CQOVF = 000001	ECCSTA= 000010	KTXTND= 040000	PRI1 = 000040
ADDR22= 001000	CR = 000015	ENBEOB= 010000	LF = 000012	PRI4 = 000200
ADR = 000006	CSRA = 000100	ENBNUL= 000001	LPSTAT= 000001	PRI5 = 000240
APTFER= 000004	CSRC = 000102	ENDLST= 000000	MAPSTA= 000200	PRI6 = 000300
APTPRE= 000200	CTRLC = 000003	EOPBIT= 000001	MED = 076600	PRI7 = 000340
ARGCHK= ***** G	CTRLO = 000017	ERRTYP= 000106	MEMPAS= 040000	PRO = 000000
ASB = 000106	CTRLU = 000025	EVNTBE= 000200	MODEXH= 004000	PR4 = 000200
ASSEMB= 000010	DCEVNT= 000011	EVNTHD= 000200	MODHOL= 002000	PR5 = 000240
ASTAT = 000104	DEFRTN= 000400	EVNTKT= 000203	MODSEL= 001000	PR6 = 000300
AUTD = 000010	DIAGMC= 000000	EVNTPE= 000202	MSGCKD= 000010	PR7 = 000340
AUTDST= 020000	DROPMO= 100000	EVNTRE= 000201	MSGCKS= 000011	PS = 177776
AWAS = 000110	DSEVNT= 000014	FATERR= 100000	MSGDOR= 000005	PSW = 177776
BIT0 = 000001	DTADR = 000000	HRDCNT= 000044	MSGDRP= 000017	RANNUM= 000054
BIT00 = 000001	DT.ADD= 000042	HRDPAS= 000050	MSGECH= 177777	RBUFEA= 000130
BIT01 = 000002	DT.AP = 000100	ICONT = 000036	MSGEOP= 000013	RBUFPA= 000126
BIT02 = 000004	DT.APK= 000076	ICOUNT= 000040	MSGHDR= 000004	RBUFSZ= 000132
BIT03 = 000010	DT.BLS= 000034	ICSROO= ***** G	MSGHNG= 000022	RBUFVA= 000124
BIT04 = 000020	DT.CFO= 000014	IDNUM = 000122	MSGHRD= 000007	RDSERV= 000101
BIT05 = 000040	DT.CF1= 000016	IE = 000100	MSGMAP= 000021	RDWHMI= 000022
BIT06 = 000100	DT.ERR= 000020	INDPAR= 000040	MSGNUL= 177775	RELERR= 000020
BIT07 = 000200	DT.ESI= 000044	INHDRP= 040000	MSGPOP= 000002	RELMOD= 020000
BIT08 = 000400	DT.EVN= 000000	INHPR= 020000	MSGPRM= 177776	RELTIM= 010000
BIT09 = 001000	DT.EXS= 000060	INHREL= 001000	MSGRES= 000001	RES1 = 000056
BIT1 = 000002	DT.FCH= 000037	INHRE= 000400	MSGSFT= 000006	RES2 = 000060
BIT10 = 002000	DT.FCN= 000036	INIT = 000030	MSGSKE= 000003	RICHAR= 031060
BIT11 = 004000	DT.HMX= 000104	INTR = 000120	MSGSMB= 000015	RPTDAT= 002000
BIT12 = 010000	DT.KBE= 000024	IOMOD = 100000	MSGSMH= 000014	RSTRT = 000112
BIT13 = 020000	DT.KBP= 000026	IOMODP= 102000	MSGSMS= 000016	RUBOUT= 000177
BIT14 = 040000	DT.KBR= 000022	IOMODR= 112000	MSGSTD= 000000	RUNMOD= 100000
BIT15 = 100000	DT.KBU= 000030	IOMODX= 110000	MSGSYS= 000012	RSVALU= 001740
BIT2 = 000004	DT.MLS= 000032	JACK = 035060	MSGVEC= 000020	SAM = 075464
BIT3 = 000010	DT.MTI= 000110	KIPAR0= 172340	NBKMOD= 001000	SBADR = 000102
BIT4 = 000020	DT.OFF= 000070	KIPAR1= 172342	NPCUOP= 000020	SBKMOD= 000000
BIT5 = 000040	DT.PAS= 000074	KIPAR2= 172344	NOPTY= 000002	SBKSEL= 010000
BIT6 = 000100	DT.PC = 000002	KIPAR3= 172346	NULL = 000000	SC.ADR= 000006
BIT7 = 000200	DT.PFL= 000062	KIPAR4= 172350	OWEN = 024020	SC.ALC= 000014
BIT8 = 000400	DT.PSW= 000004	KIPAR5= 172352	PAERR = 000010	SC.APC= 000016
BIT9 = 001000	DT.PTA= 000064	KIPAR6= 172354	PARITY 000140R	SC.CKL= 000002
BKDEF = 000002	DT.RCS= 000102	KIPAR7= 172356	PARPRE= 002000	SC.CKP= 000004
BKMOD = 000020	DT.REL= 000040	KIPDR0= 172300	PARSTA= 000100	SC.CLO= 000000
BKMODE= 040000	DT.SCT= 000066	KIPDR1= 172302	PASCNT= 000034	SC.HLD= 000010
BKSLSH= 000134	DT.SMX= 000106	KIPDR2= 172304	PDPLSI= 020000	SC.SCA= 000012
BUFPTR= 000002	DT.SP = 000006	KIPDR3= 172306	PDP60 = 004000	SENDLS= 177777
CAPRES= 000004	DT.SSI= 000046	KIPDR4= 172310	PDP70 = 010000	SOFCNT= 000042
CASTAT= 000004	DT.ST0= 000010	KIPDR5= 172312	PO.EOF 000040R	SOFPAS= 000046
CCNTRL= ***** G	DT.ST1= 000012	KIPDR6= 172314	PO.EON 000102R	SPACE = 000040
CDERCT= 000146	DT.SWR= 000056	KIPDR7= 172316	PO.FLA 000000R	SPOINT= 000032
CDWDCT= 000144	DT.SYP= 000072	KONTRL= ***** G	PO.MB0 000037R	SPVALU= 002200
CKTIM = 100000	DT.WBU= 000050	KPOFF 000132RG	PO.MB1 000101R	SR0 = 177572
CLKPRE= 000001	DT.WHL= 000054	KPON 000122RG	PO.NUN 000002R	SR1 = 177574
CM.ARG= ***** G	DT.WLL= 000052	KTERRO= 000040	PO.OFF 000015R	SR2 = 177576
CM.RUN= ***** G	DVID1 = 000014	KTPRES= 000400	PO.ON 000060R	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\$GOD= 000400	\$NESTL= 177777	\$\$FLAG= 000001
SVR2 = 000066	UIPDR7= 177616	\$F\$IF = 000110	\$NSKO = 000300	\$\$FROM= 000000
SVR3 = 000070	WASADR= 000104	\$F\$INC= 000210	\$NSK1 = 000110	\$\$LOC = 000532R
SVR4 = 000072	WBSTAT= 000040	\$F\$LDD= 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	WBUFRO= 000140	\$F\$OR = 000320	\$NSK5 = 000110	\$\$RTN1= 050000
YSERR= 000100	WBUFSA= 000142	\$F\$RTI= 000350	\$\$AVLE= 177777	\$\$RTN2= 050001
TMPID = 000002	WDFR = 000116	\$F\$RTN= 000300	\$\$SKO = 050010	\$\$SRC = 000000
TQOVF = 000002	WDT0 = 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	\$TSKO = 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	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:24  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0823

3 COMMON EQUATE MODULE  
556 COMMON DEFINITIONS AND REFERENCES FOR PRBTOD  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
578 PRBTOD ROUTINE

```
508 .TITLE PRBTOD PROCESS THE 'BTOD' MACRO IN OPTION MODULES
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRBTOD
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'BTOD' TRAP ISSUED BY OPTION
517 ; MODULES. IT WILL RETRIEVE THE NUMBER TO BE CONVERTED TO
518 ; DECIMAL ASCII FROM DT.PC+2 AND THE TABLE ADDRESS AT WHICH
519 ; TO STORE THE DECIMAL ASCII FROM DT.PC +4. IT CALLS THE
520 ; BINARY TO DECIMAL ASCII CONVERSION ROUTINE. THE RETURN PC
521 ; IS UPDATED SINCE THIS IS DIRECT SERVICE.
522 ;
523 ; INPUTS:
524 ; DTABLE ADDRESS
525 ;
526 ; IMPLICIT INPUTS:
527 ; DT.PC
528 ;
529 ; OUTPUTS:
530 ; NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ; NONE
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ; NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ; BDACNV - BINARY TO DECIMAL ASCII CONVERSION ROUTINE
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ; NONE
543 ;
544 ; CALLING SEQUENCE:
545 ; CALL PRBTOD IN <DTADR>
546 ; WHERE DTADR = ADDRESS OF DATA TABLE
547 ;
548 ; VERSION:
549 ; 0.0
550 ;
551 ; EDIT DATE BY REASON
552 ;--
553
554
```

```
556          .SBTTL COMMON DEFINITIONS AND REFERENCES FOR PRBTOD
557
558          .MCALL STRUCT
559 000000'    STRUCT
(1) 000000'    .PRINT ;SPMAC: VERSION 1.1
560
561          000001    $LSTIN=1
562          000001    $LSTTAG=1
563
564          ;*****
565          ;
566          ; REFERENCED BY OTHER MODULES:
567          ;
568          .GLOBL PRBTOD          ;MODULE ENTRY POINT
569          ;
570          ;*****
571          ;
572          ; GLOBAL REFERENCES:
573          ;
574          .GLOBL BDACNV          ;BINARY TO DECIMAL ASCII CONVERSION ROUTINE
575          ;
576
```



```

578          .SBTTL PRBTOD ROUTINE
579
580 000000'   ROUTINE PRBTOD <DTADR>
(2) 000000'
581
582          ;+
583          ;INITIALIZE AND SAVE DATA TABLE ADDRESS
584          ;--
585
586
587 000000'   PUSH R0,R1
(2) 000000' 010046
(3) 000002' 010146
588 000004'   LET R0 := DTADR(R5)
(4) 000004' 016500 000000
589
590          ;+
591          ; GET DT.PC AND CALL BDACNV
592          ;--
593
594 000010'   LET R1 := DT.PC(R0)
(4) 000010' 016001 000002
595 000014'   CALL BDACNV IN <@2(R1),4(R1)>
(3) 000014' 010546
(5) 000016' 016145 000004
(4) 000022' 017145 000002
(3) 000026' 004767 000000G
(3) 000032' 012605
596
597          ;+
598          ; UPDATE RETURN PC
599          ;--
600
601 000034'   LET DT.PC(R0) := DT.PC(R0) + #6
(6) 000034' 062760 000006 000002
602
603          ;+
604          ; CLEAN UP AND GOODBYE
605          ;--
606
607 000042'   POP R1,R0
(2) 000042' 012601
(3) 000044' 012600
608
609 000046'   ENDRTN
(3) 000046'
(3) 000046'
(2) 000046' 000207
610
611          .END
    
```

PRBTOD:

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV DTADR(R5),R0
MOV DT.PC(R0),R1
MOV R5,-(SP)
MOV 4(R1),-(R5)
MOV @2(R1),-(R5)
JSR PC,BDACNV
MOV (SP)+,R5
ADD #6,DT.PC(R0)
MOV (SP)+,R1
MOV (SP)+,R0
50000$:
50001$:
RTS PC
    
```

ACSR = 000102	CSRC = 000102	ENDLST= 000000	MODHOL= 002000	RBUFSZ= 000132
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124
ADDR22= 001000	CTRLO = 000017	ERRTYP= 000106	MSGCKD= 000010	RDSERV= 000101
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RELTIM= 010000
ASSEMB= 000010	DROPMO= 100000	EVNTRE= 000201	MSGEQP= 000013	RES1 = 000056
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
AUTO = 000010	DTADR = 000000	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
BDACNV= ***** G	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	R5VALU= 001740
BIT01 = 000002	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT02 = 000004	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBADR = 000102
BIT03 = 000010	DT.ESI= 000044	INHPR= 020000	MSGSKE= 000003	SBKMOD= 000000
BIT04 = 000020	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBKSEL= 010000
BIT05 = 000040	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SC.ADR= 000006
BIT06 = 000100	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.ALC= 000014
BIT07 = 000200	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.APC= 000016
BIT08 = 000400	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKL= 000002
BIT09 = 001000	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CKP= 000004
BIT1 = 000002	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CLO= 000000
BIT10 = 002000	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.HLD= 000010
BIT11 = 004000	DT.KBU= 000030	JACK = 035060	NCAPTY= 000002	SC.SCA= 000012
BIT12 = 010000	DT.MLS= 000032	KIPARO= 172340	NULL = 000000	SENDLS= 177777
BIT13 = 020000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SOFCNT= 000042
BIT14 = 040000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOFPAS= 000046
BIT15 = 100000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPACE = 000040
BIT2 = 000004	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPOINT= 000032
BIT3 = 000010	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPVALU= 002200
BIT4 = 000020	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SR0 = 177572
BIT5 = 000040	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR1 = 177574
BIT6 = 000100	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR2 = 177576
BIT7 = 000200	DT.REL= 000040	KIPDR1= 172302	PRBTOD 000000RG	SR3 = 172516
BIT8 = 000400	DT.SCT= 000066	KIPDR2= 172304	PRI0 = 000000	STAT = 000026
BIT9 = 001000	DT.SMX= 000106	KIPDR3= 172306	PRI1 = 000040	STATBI= 064757
BKDEF = 000002	DT.SP = 000006	KIPDR4= 172310	PRI4 = 000200	STAT1 = 000027
BKMOD = 000020	DT.SSI= 000046	KIPDR5= 172312	PRI5 = 000240	SUSPND= 000001
BKMODE= 040000	DT.ST0= 000010	KIPDR6= 172314	PRI6 = 000300	SVR0 = 000062
BKSLSH= 000134	DT.ST1= 000012	KIPDR7= 172316	PRI7 = 000340	SVR1 = 000064
CAPRES= 000004	DT.SWR= 000056	KTERRO= 000040	PRO = 000300	SVR2 = 000066
CASTAT= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR3 = 000070
CDERCT= 000146	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR4 = 000072
CDWDCT= 000144	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR5 = 000074
CKTIM = 100000	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR6 = 000076
CLKPRE= 000001	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SYSCNT= 000052
CONFIG= 000056	ECCMEM= 000100	MAPSTA= 000200	PSW = 177776	YSERR= 000100
CQOVF = 000001	ECCSTA= 000010	MED = 076800	RANNUM= 000054	TMPIO = 000002
CR = 000015	ENBEOP= 010000	MEMPAS= 040000	RBUFEA= 000130	TQOVF = 000002
CSRA = 000100	ENBNUL= 000001	MODEXH= 004000	RBUFPA= 000126	UIPARO= 177640

UIPAR1= 177642	WBUFEA= 000136	\$F\$DEC= 000220	\$F\$YES= 000402	\$F\$FLAG= 000000
UIPAR2= 177644	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$F\$FROM= 000000
UIPAR3= 177646	WBUFRQ= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$F\$LOC = 000000
UIPAR4= 177650	WBUFSZ= 000142	\$F\$GOD= 000400	\$LSTIN= 000001	\$F\$LOCN= 000000
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTYA= 000001	\$F\$REG = 177777
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$F\$RETU= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$LDD= 000200	\$NSKO = 000300	\$F\$RTN1= 050000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$F\$RTN2= 050001
UIPDR1= 177602	XFLAG = 000005	\$F\$ND = 000403	\$TAGLE= 177777	\$F\$SRC = 000000
UIPDR2= 177604	XOFF = 000023	\$F\$OR = 000320	\$TAGNU= 050002	\$F\$TGSV= 000000
UIPDR3= 177606	XON = 000021	\$F\$RTI= 000350	\$TEMP = 000300	\$F\$TGS1= 000000
UIPDR4= 177610	\$BGNLE= 177777	\$F\$RTN= 000300	\$ARGC= 000002	\$F\$TGS2= 000000
UIPDR5= 177612	\$ERFLG= 000400	\$F\$SEL= 000140	\$BYTE= 000000	\$F\$TO = 000002
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$CASE= 000000	\$F\$TAG= 050000
UIPDR7= 177616	\$F\$BAD= 000401	\$F\$TRU= 000404	\$DST = 000000	= 000050R
WASADR= 000104	\$F\$BLA= 000170	\$F\$UNT= 000130	\$ELQC= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:25  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0829

3 COMMON EQUATE MODULE  
563 COMMON DEFINITIONS & REFERENCES  
566 000000' .PRINT ;SPMAC: VERSION 1.1  
638 PREVT (CODE)

```
508 .TITLE PREVT (PROCESS EVENT MODULE)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     PREVT
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE DISPATCHES CONTROL TO VARIOUS EVENT
517 ;     HANDLING ROUTINES.  THE ADDRESS OF A DATA TABLE
518 ;     IS PASSED TO THIS MODULE.  THE EVENT CODE IS
519 ;     EXTRACTED FROM THE TABLE AND AN OFFSET IS GENERATED.
520 ;     THIS OFFSET IS USED TO FORM ADDRESS IN THE DISPATCH
521 ;     TABLE TO SEND CONTROL TO THE PROPER EVENT HANDLING
522 ;     ROUTINE.
523
524 ;     NOTE:  IF EVENT CODE IS 'ZERO', THE STACK IS POPPED
525 ;           AND CONTROL GOES BACK TO CALLER.
526
527 ; INPUTS:
528 ;     ADDRESS OF DATA TABLE
529
530 ; IMPLICIT INPUTS:
531 ;     EVENT CODE
532
533 ; OUTPUTS:
534 ;     NONE
535
536 ; IMPLICIT OUTPUTS:
537 ;     NONE
538
539 ; PATHOLOGICAL CONNECTIONS:
540 ;     NONE
541
542 ; SUBORDINATE ROUTINES CALLED:
543 ;     IT CALLS THE ACTUAL EVENT HANDLING ROUTINE
544 ;     THROUGH A POINTER AND THE FORMAT IS:
545 ;         CALL PTR IN <DTADDR>
546 ;         WHERE DTADDR = DATA TABLE ADDRESS
547
548 ; FUNCTIONAL SIDE EFFECTS:
549 ;     NONE
550
551 ; CALLING SEQUENCE:
552 ;     CALL PREVT IN <DTADR>
553 ;         WHERE DTADR = DATA TABLE ADDRESS
554
555 ; VERSION:
556 ;     0.0
557
558 ;     EDIT      DATE      BY      REASON
559 ;     -----
560
561
```

```
563 .SBTTL COMMON DEFINITIONS & REFERENCES
564
565 .MCALL STRUCT
566 000000' STRUCT
567 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
568 000001 $LSTIN=1
569 000001 $LSTTAG=1
570 ;
571 ;*****
572 ;REFERENCED BY OTHER MODULES:
573 ;
574 .GLOBL PREVT ;MODULE ENTRY POINT
575
576 ;*****
577 ; GLOBAL REFERENCES
578 ;
579
580
581 .GLOBL PENDIT ;ENDIT HANDLER
582 .GLOBL PEND ;END HANDLER
583 .GLOBL PRHARD ;HRDER HANDLER
584 .GLOBL PDATER ;DATER HANDLER
585 .GLOBL PRMSG ;MSG HANDLER
586 .GLOBL PBREAK ;BREAK HANDLER
587 .GLOBL PRSOFT ;SOFER HANDLER
588 .GLOBL PRMSGN ;MSGN HANDLER
589 .GLOBL PRGWB ;GWBUFF HANDLER
590 .GLOBL PGETPA ;GETPA HANDLER
591 .GLOBL PDATCK ;DATACK HANDLER
592 .GLOBL PCDATA ;CDATA HANDLER
593 .GLOBL PMAP22 ;MAP22 HANDLER
594 .GLOBL PRMSGS ;MSGS HANDLER
595 .GLOBL PRRAND ;RAND TRAP HANDLER
596 .GLOBL PARERR ;MEMORY PARITY ERROR HANDLER
597 .GLOBL KTERR ;MEMORY MGMT VIOL. HANDLER
598 .GLOBL BUSERR ;TIMEOUT,ILLEGAL INSTR. ETC.
599 .GLOBL PROTOA ;OCTAL-TO-DECIMAL EVENT
600 .GLOBL PRBTOD ;BIN-TO-DECIMAL EVENT
601
602 ;*****
603 ;
604 ; LOCAL STORAGE:
605 ;
606 ; DISPATCH TABLE FOR SOFT-TRAP EVENTS
607 ;
608 000000' PR.TRPTBL: ;ADDRESS OF EVENT HANDLER
609 000000' 000000 .WORD 0 ; EXIT IS ZERO - DOES NOT NEED A ROUTINE
610 000002' 000000G PRMSG ;
611 000004' 000000G PRMSGS ;
612 000006' 000000G PRMSGN ;
613 000010' 000000G PDATER ;
614 000012' 000000G PRHARD ;
615 000014' 000000G PRSOFT ;
616 000016' 000000G PBREAK ;
617 000020' 000000G PEND ;
```

PREVT (PROCESS EVENT MODULE)  
PREVT.MAC 28-JUL-78 09:25

MACY11 30A(1052) 20-SEP-78 18:25 PAGE 19-2  
COMMON DEFINITIONS & REFERENCES

SEQ 0832

```
618 000022' 000000G          PDATCK          ;
619 000024' 000000G          PCDATA          ;
620 000026' 000000G          PENDIT          ;
621 000030' 000000G          PRGWB           ;
622 000032' 000000G          PGETPA          ;
623 000034' 000000G          PMAP22          ;
624 000036' 000000G          PRRAND          ;
625 000040' 000000G          PROTOA          ;
626 000042' 000000G          PRBTOD          ;
627
628
629          ;*****
630          ; DISPATCH TABLE FOR HARD TRAPS & MISC. EVENTS
631          ;
631 000044'          PR.NTRPTBL:          ;ADDRESS OF EVENT HANDLER
632 000044' 000000G          BUSERR          ;
633 000046' 000000G          BUSERR          ;
634 000050' 000000G          PARERR          ;
635 000052' 000000G          KTERR          ;
636
```

```
638 .SBTTL PREVT (CODE)
639
640
641 000054' ROUTINE PREVT <DTBL>
(2) 000054' PREVT:
642
643 ;+
644 ; SAVE REGISTERS,FETCH DTABLE ADDRESS & EVENT CODE.
645 ; IF EVENT CODE IS 0, RETURN.
646 ; ELSE CHECK IF EVENT CODE IS A HARDWARE-TRAP TYPE.
647 ;-
648
649 000054' PUSH R0,R1
(2) 000054' 010046 MOV R0,-(SP)
(3) 000056' 010146 MOV R1,-(SP)
650 000060' LET R0 := DTBL(R5)
(4) 000060' 016500 000000 MOV DTBL(R5),R0
651 000064' LET R1 := DT.EVNT(R0)
(4) 000064' 016001 000000 MOV DT.EVNT(R0),R1
652 000070' IF R1 NE #0 THEN
(6) 000070' 005701 TST R1
(9) 000072' 001421 BEQ 50002$
653 ;+
654 ; DEVELOP OFFSET INTO TABLE
655 ;-
656 000074' LET R1 := R1 CLR.BY #177700
(6) 000074' 042701 177700 BIC #177700,R1
657 000100' LET R1 := R1 SHIFT 1
(7) 000100' 006301 ASL R1
658
659 000102' IF DT.EVNT(R0) GE #EVNTHD THEN
(6) 000102' 026027 000000 000200 CMP DT.EVNT(R0),#EVN
(9) 000110' 002403 BLT 50003$
660
661 ;+
662 ; IT IS A HARDWARE-TRAP EVENT.
663 ; DEVELOP ADDRESS.
664 ;-
665
666 000112' LET R1 := R1 + #PR.NTRPTBL
(6) 000112' 062701 000044' ADD #PR.NTRPTBL,R1
667 000116' ELSE
(4) 000116' 000402 BR 50004$
(3) 000120' 50003$:
668
669 ;+
670 ; IT IS A SOFTWARE TRAP EVENT.
671 ; FORM AN ADDRESS
672 ;-
673
674 000120' LET R1 := R1 + #PR.TRPTBL
(6) 000120' 062701 000000' ADD #PR.TRPTBL,R1
675 000124' ENDIF
(4) 000124' 50004$:
676 000124' CALL @(R1) IN <R0>
(3) 000124' 010546 MOV R5,-(SP)
```



PREVT (PROCESS EVENT MODULE)  
PREVT.MAC 28-JUL-78 09:25

MACY11 30A(1052) 20-SEP-78 18:25 PAGE 19-4  
PREVT (CODE)

SEQ 0834

(4) 000126' 010045  
(3) 000130' 004771 000000  
(3) 000134' 012605  
677  
678 000136'  
(4) 000136'  
679  
680  
681  
682  
683 000136'  
(2) 000136' 012601  
(3) 000140' 012600  
684 000142'  
(3) 000142'  
(3) 000142'  
(2) 000142' 000207  
685 000001

ENDIF

;  
;+  
; RESTORE REGISTERS AND RETURN  
;-

POP R1,R0

ENDRTN

.END

MOV R0,-(R5)  
JSR PC,@(R1)  
MOV (SP)+,R5

50002\$:

MOV (SP)+,R1  
MOV (SP)+,R0

50000\$:  
50001\$:

RTS PC

ACSR = 000102	CSRC = 000102	ENDLST= 000000	MODEXH= 004000	PRI5 = 000240
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	MODHOL= 002000	PRI6 = 000300
ADDR22= 001000	CTRLQ = 000017	ERRTYP= 000106	MODSEL= 001000	PRI7 = 000340
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	MSGCKD= 000010	PRMSG = ***** G
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	MSGCKS= 000011	PRMSGN= ***** G
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	MSGDER= 000005	PRMSGS= ***** G
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	MSGDRP= 000017	PROTOA= ***** G
ASSEMB= 000010	DROPMO= 100000	EVNTRE= 000201	MSGECH= 177777	PRRAND= ***** G
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGEOP= 000013	PRSOFT= ***** G
AUTO = 000010	DTBL = 000000	HRDCNT= 000044	MSGHDR= 000004	PR.NTR 000044R
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHNG= 000022	PR.TRP 000000R
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGHRD= 000007	PRO = 000000
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGMAP= 000021	PR4 = 000200
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGNUL= 177775	PR5 = 000240
BIT01 = 000002	DT.CF0= 000014	IE = 000100	MSGPOP= 000002	PR6 = 000300
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MSGPRM= 177776	PR7 = 000340
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MSGRES= 000001	PS = 177776
BIT04 = 000020	DT.ESI= 000044	INHEPR= 020000	MSGSFT= 000006	PSW = 177776
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MSGSKF= 000003	RANNUM= 000054
BIT06 = 000100	DT.EXS= 000060	INHRE= 000400	MSGSMB= 000015	RBUFEA= 000130
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MSGSMH= 000014	RBUFPA= 000126
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGSMS= 000016	RBUFSZ= 000132
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGSTD= 000000	RBUFVA= 000124
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGSYS= 000012	RDSERV= 000101
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	MSGVEC= 000020	RDWHMI= 000022
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	NBKM0D= 001000	RELERR= 000020
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	NCPUOP= 000020	RELMOD= 020000
BIT13 = 020000	DT.MLS= 000032	KIPAR0= 172340	NCAPTY= 000002	RELTIM= 010000
BIT14 = 040000	DT.MTI= 000110	KIPAR1= 172342	NULL = 000000	RES1 = 000056
BIT15 = 100000	DT.OFF= 000070	KIPAR2= 172344	OWEN = 024020	RES2 = 000060
BIT2 = 000004	DT.PAS= 000074	KIPAR3= 172346	PAERR = 000010	RICHAR= 031060
BIT3 = 000010	DT.PC = 000002	KIPAR4= 172350	PARERR= ***** G	RPTDAT= 002000
BIT4 = 000020	DT.PFL= 000062	KIPAR5= 172352	PARPRE= 002000	RSTRT = 000112
BIT5 = 000040	DT.PSW= 000004	KIPAR6= 172354	PARSTA= 000100	RUBOUT= 000177
BIT6 = 000100	DT.PTA= 000064	KIPAR7= 172356	PASCNT= 000034	RUNMOD= 100000
BIT7 = 000200	DT.RCS= 000102	KIPDR0= 172300	PBREAK= ***** G	R5VALU= 001740
BIT8 = 000400	DT.REL= 000040	KIPDR1= 172302	PCDATA= ***** G	SAM = 075464
BIT9 = 001000	DT.SCT= 000066	KIPDR2= 172304	PDATCK= ***** G	SBADR = 000102
BKDEF = 000002	DT.SMX= 000106	KIPDR3= 172306	PDATER= ***** G	SBKMOD= 000000
BKMOD = 000020	DT.SP = 000006	KIPDR4= 172310	PDPLSI= 020000	SBKSEL= 010000
BKMODE= 040000	DT.SSI= 000046	KIPDR5= 172312	PDP60 = 004000	SC.ADR= 000006
BKSLSH= 000134	DT.STO= 000010	KIPDR6= 172314	PDP70 = 010000	SC.ALCL= 000014
BUSERR= ***** G	DT.ST1= 000012	KIPDR7= 172316	PEND = ***** G	SC.APC= 000016
CAPRES= 000004	DT.SWR= 000056	KTERR = ***** G	PENDIT= ***** G	SC.CKL= 000002
CASTAT= 000004	DT.SYP= 000072	KTERRO= 000040	PGETPA= ***** G	SC.CKP= 000004
CDCRECT= 000146	DT.WBU= 000050	KTPRES= 000400	PMAP22= ***** G	SC.CLO= 000000
CDWDCT= 000144	DT.WHL= 000054	KTSTAT= 000020	PRBTOD= ***** G	SC.HLD= 000010
CKTIM = 100000	DT.WLL= 000052	KTXTND= 040000	PREVT 000054RG	SC.SCA= 000012
CLKPRE= 000001	DVID1 = 000014	LF = 000012	PRGWB = ***** G	SENDLS= 177777
CONFIG= 000056	ECCMEM= 000100	LPSTAT= 000001	PRHARD= ***** G	SDFCNT= 000042
CQOVF = 000001	ECCSTA= 000010	MAPSTA= 000200	PRI0 = 000000	SDFPAS= 000046
CR = 000015	ENBEOP= 010000	MED = 076600	PRI1 = 000040	SPACE = 000040
CSRA = 000100	ENBNUL= 000001	MEMPAS= 040000	PRI4 = 000200	SPOINT= 000032

SPVALU= 002200	UIPAR2= 177644	WTINRE= 000352	\$F\$RTN= 000300	\$TSK1 = 050004
SRO = 177572	UIPAR3= 177646	WTWHMI= 000222	\$F\$SEL= 000140	\$\$ARGC= 000002
SR1 = 177574	UIPAR4= 177650	XFLAG = 000005	\$F\$THE= 000330	\$\$BYTE= 000403
SR2 = 177576	UIPAR5= 177652	XOFF = 000023	\$F\$TRU= 000404	\$\$CASE= 000000
SR3 = 172516	UIPAR6= 177654	XON = 000021	\$F\$UNT= 000130	\$\$DST = 000000
STAT = 000026	UIPAR7= 177656	\$BGNLE= 177777	\$F\$WHI= 000120	\$\$ELOC= 000402
STATBI= 064757	UIPDR0= 177600	\$ERFLG= 000400	\$F\$YES= 000402	\$\$ERFL= 000000
STAT1 = 000027	UIPDR1= 177602	\$F\$AND= 000310	\$IFLEV= 177777	\$\$FLAG= 000001
SUSPND= 000001	UIPDR2= 177604	\$F\$BAD= 000401	\$ISK0 = 000001	\$\$FROM= 000000
SVR0 = 000062	UIPDR3= 177606	\$F\$BLA= 000170	\$ISK1 = 000001	\$\$LOC = 000110R
SVR1 = 000064	UIPDR4= 177610	\$F\$CAS= 000150	\$LOCTA= 177777	\$\$LOCN= 000000
SVR2 = 000066	UIPDR5= 177612	\$F\$DEC= 000220	\$LSTIN= 000001	\$\$REG = 177777
SVR3 = 000070	UIPDR6= 177614	\$F\$DO = 000340	\$LSTTA= 000001	\$\$RETN= 000000
SVR4 = 000072	UIPDR7= 177616	\$F\$FAL= 000405	\$NESTL= 177777	\$\$RTN1= 050000
SVR5 = 000074	WASADR= 000104	\$F\$G00= 000400	\$NSK0 = 000300	\$\$RTN2= 050001
SVR6 = 000076	WBSTAT= 000040	\$F\$IF = 000110	\$NSK1 = 000110	\$\$SRC = 000000
SYSCNT= 000052	WBUFEA= 000136	\$F\$INC= 000210	\$NSK2 = 000110	\$\$TGSV= 000000
SYSERR= 000100	WBUFPA= 000134	\$F\$L00= 000200	\$SAVLE= 177777	\$\$TGS1= 000000
TMPID = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	\$TAGLE= 177777	\$\$TGS2= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$NO = 000403	\$TAGNU= 050005	\$\$TO = 000001
UIPAR0= 177640	WDFR = 000116	\$F\$OR = 000320	\$TEMP = 000300	\$\$\$TAG= 050000
UIPAR1= 177642	WDT0 = 000114	\$F\$RTI= 000350	\$TSK0 = 050002	. = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:25  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0837

3 COMMON EQUATE MODULE  
554 COMMON DEFINITIONS AND REFERENCES  
556 000000' .PRINT ;SPMAC: VERSION 1.1  
586 PRGWB ROUTINE

```
508 .TITLE PRGWB - PROCESS GWBUF
509 .IDENT /VO.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRGWB
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; PROCESSES GWBUF CALLS FROM OPTION MODULES. DETERMINES
517 ; AND PASSES TO A REQUESTING OPTION MODULE A BUFFER
518 ; SIZE AND STARTING ADDRESS.
519 ;
520 ; INPUTS:
521 ; DATA TABLE ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ; DT.ADDR,,DT.SSIZ,DT.ESIZ,DT.PC,DT.STO,DT.WBUF,
525 ; DT.WLLMT,DT.WHLMT,DT.OFFSET
526 ;
527 ; OUTPUTS:
528 ; NONE
529 ;
530 ; IMPLICIT OUTPUTS:
531 ; DT.WBUF
532 ;
533 ; PATHOLOGICAL CONNECTIONS:
534 ; NONE
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ; SAVREG ;SAVE REGISTERS
538 ; RESREG ;RESTORE REGISTERS
539 ;
540 ; FUNCTIONAL SIDE EFFECTS:
541 ; NONE
542 ;
543 ; CALLING SEQUENCE:
544 ; CALL PRGWB IN <A>
545 ; A=ADDRESS OF DATA TABLE
546 ;
547 ; VERSION:
548 ; 0.0
549 ;
550 ; EDIT BY DATE REASON
551 ;
552 ;--
```

```
554 .SBTTL COMMON DEFINITIONS AND REFERENCES
555 .MCALL STRUCT
556 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
557 000001 $LSTIN=1
558 000001 $LSTTAG=1
559
560
561 ;
562 ;*****
563 ;
564 ; REFERENCED BY OTHER MODULES:
565 ;
566 .GLOBL PRGWB ;MODULE ENTRY POINT
567 ;
568 ;*****
569 ;
570 ; GLOBAL REFERENCES
571 ;
572 .GLOBL SAVREG ;SAVE REGISTERS
573 .GLOBL RESREG ;RESTORE REGISTERS
574 ;
575 ;*****
576 ;
577 ; LOCAL STORAGE
578 ;
579 000000' 000000 PG.AVAIL: .WORD 0 ;AVAILABLE BUFFER SIZE - PAR FORMAT
580 000002' 000000 PG.SIZE: .WORD 0 ;ACTUAL WRITE BUFFER SIZE - PAR FORMAT
581 ;
582 ;*****
583 ;
584 ;
```

```

586          .SBTTL PRGWB ROUTINE
587 000004'  ROUTINE PRGWB <TABL>
(2) 000004'
588
589          ;+
590          ; SAVE REGISTERS
591          ; -
592
593 000004'  CALL SAVREG
(3) 000004' 004767 000000G          JSR      PC,SAVREG
594
595
596          ;+
597          ; SET R0 TO THE START OF THE DATA TABLE
598          ; -
599
600 000010'  LET R0 := TABL(R5)
(4) 000010' 016500 000000          MOV      TABL(R5),R0
601
602
603          ;+
604          ; GET THE OPTIONS MODULE'S HEADER ADDRESS
605          ; -
606
607 000014'  LET R1 := @DT.PC(R0)
(4) 000014' 017001 000002          MOV      @DT.PC(R0),R1
608
609
610          ;+
611          ; CALCULATE THE LARGEST AVAILABLE BUFFER SIZE.  IF THE BUFFER
612          ; ADDRESS POINTER IS POINTING BELOW THE MOVABLE PORTION OF
613          ; THE EXERCISER, THEN THE AVAILABLE BUFFER SIZE IS EVERYTHING FROM THE
614          ; CURRENT POINTER POSITION UP TO THE BASE ADDRESS OF THE MOVABLE PORTION
615          ; OF THE EXERCISER.  IF THE BUFFER ADDRESS POINTER IS ABOVE THE TOP OF
616          ; THE EXERCISER, THEN THE AVAILABLE BUFFER SIZE IS EVERYTHING FROM
617          ; THE CURRENT POINTER POSITION UP TO THE HIGH BUFFER LIMIT.
618          ; -
619
620 000020'  IF DT.WBUF(R0) LO DT.ADDR(R0) THEN
(6) 000020' 026060 000050 000042          CMP      DT.WBUF(R0),DT.A
(9) 000026' 103007          BHIS     50002$
621          LET PG.AVAIL := DT.ADDR(R0) - DT.WBUF(R0)
(4) 000030' 016067 000042 177742          MOV      DT.ADDR(R0),PG.A
(6) 000036' 166067 000050 177734          SUB      DT.WBUF(R0),PG.A
622 000044'  ELSE
(4) 000044' 000406          BR      50003$
(3) 000046'          50002$:
623          LET PG.AVAIL := DT.WHLMT(R0) - DT.WBUF(R0)
(4) 000046' 016067 000054 177724          MOV      DT.WHLMT(R0),PG.A
(6) 000054' 166067 000050 177716          SUB      DT.WBUF(R0),PG.A
624 000062'  ENDIF
(4) 000062'          50003$:
625
626
627          ;+
628          ; GET THE REQUESTED BUFFER SIZE FROM THE OPTION MODULE, CONVERT IT TO
    
```

```

629          ; PAR FORMAT, DOUBLE IT TO CHANGE IT FROM A WORD COUNT TO A MEMORY AREA SIZE, AND
630          ; COMPARE IT TO THE AVAILABLE BUFFER SIZE.  IF THE REQUESTED SIZE IS LARGER THAN THAT
631          ; AVAILABLE, LET THE ACTUAL BUFFER SIZE EQUAL THE AVAILABLE SIZE.  OTHERWISE, LET
632          ; THE ACTUAL SIZE BE THE FULL REQUESTED SIZE.  THEN PASS THE ACTUAL SIZE TO THE
633          ; OPTION MODULE, AFTER CONVERTING IT TO A PHYSICAL WORD COUNT FROM PAR FORMAT.
634          ; -
635
636          LET R2 := WBUF(R1) SHIFT #-6
637          (4) 000062' 016102 000140          MOV      WBUF(R1),R2
638          (7) 000066' 006202          ASR      R2
639          (7) 000070' 006202          ASR      R2
640          (7) 000072' 006202          ASR      R2
641          (7) 000074' 006202          ASR      R2
642          (7) 000076' 006202          ASR      R2
643          (7) 000100' 006202          ASR      R2
644          (6) 000102' 042702 176000          BIC      #176000,R2
645          (7) 000106' 006302          LET R2 := R2 SHIFT #+1
646          (7) 000110' 006302          IF R2 HI PG.AVAIL THEN
647          (6) 000110' 020267 177664          CMP      R2,PG.AVAIL
648          (9) 000114' 101404          BLOS    50004$
649          (4) 000116' 016767 177656 177656          LET PG.SIZE := PG.AVAIL
650          (6) 000124' 000402          ELSE
651          (3) 000126'          50004$:
652          (4) 000126' 010267 177650          LET PG.SIZE := R2
653          (4) 000132'          ENDIF
654          (4) 000132'          50005$:
655          (4) 000132' 016761 177644 000142          LET WBUF(SZ(R1)) := PG.SIZE SHIFT #+6
656          (7) 000140' 006361 000142          MOV      PG.SIZE,WBUF(SZ(R
657          (7) 000144' 006361 000142          ASL      WBUF(SZ(R1))
658          (7) 000150' 006361 000142          ASL      WBUF(SZ(R1))
659          (7) 000154' 006361 000142          ASL      WBUF(SZ(R1))
660          (7) 000160' 006361 000142          ASL      WBUF(SZ(R1))
661          (7) 000164' 006361 000142          ASL      WBUF(SZ(R1))
662          (7) 000170'          LET WBUF(SZ(R1)) := WBUF(SZ(R1)) SHIFT #-1
663          (7) 000170' 006261 000142          ASR      WBUF(SZ(R1))
664          (6) 000174' 042761 100000 000142          LET WBUF(SZ(R1)) := WBUF(SZ(R1)) CLR.BY #100000
665          (6) 000174'          BIC      #100000,WBUF(SZ(R
666
667          ; +
668          ; CONVERT THE BUFFER ADDRESS POINTER FROM PAR FORMAT TO AN 18-BIT
669          ; VIRTUAL ADDRESS, AND PASS IT TO THE OPTION MODULE.
670          ; -
671
672          LET R2 := DT.WBUF(R0) - DT.OFFSET(R0)
673          (4) 000202' 016002 000050          MOV      DT.WBUF(R0),R2
674          (6) 000206' 166002 000070          SUB      DT.OFFSET(R0),R2
675          (4) 000212'          LET R3 := R2
676          (4) 000212' 010203          MOV      R2,R3
677          (4) 000214'          LET R2 := R2 SHIFT #+6
    
```



657	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
	(6)	000230'	000303	LET R3 := SWAP R3	SWAB	R3
658	000232'			LET R3 := R3 SHIFT #+2		
	(7)	000232'	006303		ASL	R3
	(7)	000234'	006303		ASL	R3
659	000236'			LET R3 := R3 CLR.BY #177717		
	(6)	000236'	042703	177717	BIC	#177717,R3
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		
	(6)	000252'	032760	000040	000010	BIT #WBSTAT,DT.STO(R
	(9)	000260'	001470			BEQ 50006\$
670						
671						
672				;		
673				LET THE NEW BUFFER ADDRESS 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          ;+
697          ; NOW CHECK TO SEE IF THE POINTER HAS BEEN PLACED WITHIN THE CURRENT
698          ; POSITION OF THE EXERCISER. TO DO THIS, WE FIRST CALCULATE THE CURRENT POSITION
699          ; OF THE TOP OF THE EXERCISER (WITHIN 100 BYTES, BECAUSE WE ARE WORKING WITH
700          ; PAR-FORMATTED WORDS). THEN SEE IF THE CONTENTS OF THE POINTER IS LESS THAN
701          ; THIS VALUE BUT GREATER THAN THE BASE ADDRESS OF THE MOVABLE PORTION OF THE
702          ; EXERCISER MINUS 1/2 K. IF IT IS NOT, THEN WE HAVE A GOOD POINTER VALUE.
703          ; BUT IF IT IS, WE MUST THEN CHECK TO SEE IF THE TOP OF THE
704          ; EXERCISER IS WITHIN 1/2 K OF THE HIGH BUFFER LIMIT. IF YES,
705          ; AND IF THE MOVABLE PORTION OF THE EXERCISER IS NOT IN LOWEST MEMORY, MOVE THE
706          ; BUFFER ADDRESS POINTER DOWN TO THE LOW BUFFER LIMIT.
707          ; OTHERWISE, MOVE THE POINTER TO THE TOP OF THE EXERCISER.
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'
722 000436'          LET DT.WBUF(R0) := R2          50011$:
(4) 000436' 010260 000050          MQV          R2,DT.WBUF(R0)
723 000442'          ENDIF
(4) 000442'          ENDIF          50012$:
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
(6) 000442' 062760 000002 000002          ADD          #2,DT.PC(R0)
735
736
737          ;+
738          ; RESTORE REGISTERS AND RETURN
739          ; -
740
741 000450'          CALL RESREG
(3) 000450' 004767 000000G          JSR          PC,RESREG
742
743
744 000454'          ENDRTN
(3) 000454'
(3) 000454'          50000$:
(2) 000454' 000207          50001$:
745 000001          .END          RTS          PC
```

ACSR = 000102	CTRLC = 000003	ERRTP= 000106	MSGCKD= 000010	RBUFSZ= 000132
ACTBIT= 004000	CTRLU = 000017	EVNTBE= 000200	MSGCKS= 000011	RBUFVA= 000124
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RDSERV= 000101
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RDWHMI= 000022
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	RELERR= 000020
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGEOP= 000013	RELMOD= 020000
ASB = 000106	DROPMO= 100000	FATERR= 100000	MSGHDR= 000004	RELTIM= 010000
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RESREG= ***** G
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RES1 = 000056
AUTO = 000010	DT.AP = 000100	ICONT = 000035	MSGMAP= 000021	RES2 = 000060
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RICHAR= 031060
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPDP= 000002	RPTDAT= 002000
BIT0 = 000001	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	RSTRT = 000112
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	RJBOUT= 000177
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	RUNMOD= 100000
BIT02 = 000004	DT.ESI= 000044	INHPR= 020000	MSGSKE= 000003	R5VALU= 001740
BIT03 = 000010	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SAM = 075464
BIT04 = 000020	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SAVREG= ***** G
BIT05 = 000040	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SBAOR = 000102
BIT06 = 000100	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SBKMOD= 000000
BIT07 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SBKSEL= 010000
BIT08 = 000400	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.ADR= 000006
BIT09 = 001000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.ALC= 000014
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NCPUDP= 000020	SC.APC= 000016
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SC.CKL= 000004
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.CKP= 000002
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SC.CLO= 000000
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SC.HLD= 000010
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SC.SCA= 000012
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SENDLS= 177777
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SDFCNT= 000042
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SDFPAS= 000046
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SFACE = 000040
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SPOINT= 000032
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PG.AVA 000000R	SPVALU= 002200
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PG.SIZ 000002R	SRO = 177572
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRGWB 000004RG	SR1 = 177574
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI0 = 000000	SR2 = 177576
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI1 = 000040	SR3 = 172516
BKMOD = 000020	DT.STO= 000010	KIPDR6= 172314	PRI4 = 000200	STAT = 000026
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRI5 = 000240	STATBI= 064757
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PRI6 = 000300	STAT1 = 000027
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PRI7 = 000340	SUSPND= 000001
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PRO = 000000	SVRO = 000062
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR4 = 000200	SVR1 = 000064
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PR5 = 000240	SVR2 = 000066
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PR6 = 000300	SVR3 = 000070
CLKPRE= 000001	ECCMEM= 000100	MAPSTA= 000200	PR7 = 000340	SVR4 = 000072
CONFIG= 000056	ECCSTA= 000010	MED = 076600	PS = 177776	SVRE = 000074
CQDVF = 000001	ENBEOP= 010000	MEMPAS= 040000	PSW = 177776	SVR6 = 000076
CR = 000015	ENBNUL= 000001	MODEXH= 004000	RANNUM= 000054	SYSCNT= 000052
CSRA = 000100	ENDLST= 000000	MODHOL= 002000	RBUFEA= 000130	SYSERR= 000100
CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RBUFPA= 000126	TABL = 000000

TMPIO = 000002	WBSTAT= 000040	\$F\$DD = 000340	\$ISK1 = 000001	\$\$CASE= 000000
TQDVF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$ISK2 = 000001	\$\$DST = 000000
UIPAR0= 177640	WBUFPA= 000134	\$F\$GDD= 000400	\$LOCTA= 177777	\$\$ELDC= 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\$LDD= 000200	\$NESTL= 177777	\$\$FROM= 000000
UIPAR4= 177650	WDTO = 000114	\$F\$NAM= 000160	\$NSKO = 000300	\$\$LOC = 000424R
UIPAR5= 177652	WTINRE= 000352	\$F\$ND = 000403	\$NSK1 = 000110	\$\$LQCN= 000000
UIPAR6= 177654	WTWHMI= 000222	\$F\$OR = 000320	\$NSK2 = 000110	\$\$REG = 177777
UIPAR7= 177656	XFLAG = 000005	\$F\$RTI= 000350	\$NSK3 = 000110	\$\$RETI= 000000
UIPDR0= 177600	XOFF = 000023	\$F\$RTN= 000300	\$\$AVLE= 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	\$TSKO = 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	\$\$TD = 000000
UIPDR7= 177616	\$F\$CAS= 000150	\$IFLEV= 177777	\$\$ARGC= 000002	\$\$\$TAG= 050000
WASADR= 000104	\$F\$DEC= 000220	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:26  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0847

3 COMMON EQUATE MODULE  
573 PRHARD (COMMON DEFINITIONS & REFERENCES)  
576 000000' .PRINT ;SPMAC: VERSION 1.1  
596 PRHARD (CODE)

```
508 .TITLE PRHARD (PROCESS HARD ERRORS)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRHARD
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE IS CALLED AS A RESULT OF A 'HRDERR' TRAP EXECUTED
517 ; BY AN OPTION MODULE.
518 ;
519 ; FIRST THE FOLLOWING IS DONE
520 ;
521 ; 1. THE MODULE'S HARD ERROR COUNT IS INCREMENTED.
522 ; 2. ERROR MESSAGE IS QUEUED UP.
523 ;
524 ; IF THE ENVIRONMENT IS APT, THEN AN APT ERROR HANDLING ROUTINE
525 ; IS CALLED.
526 ;
527 ; IF NOT UNDER APT ENVIRONMENT, THEN
528 ;
529 ; 1. IF BIT15 OF THE SOFTWARE SWITCH REGISTER IS SET, THE
530 ; MODULE IS DROPPED WITH 'MODULE DROPPED' MESSAGE PRINTED,
531 ; EVEN ON THE FIRST ERROR.
532 ; 2. IF THE ERROR COUNT IS EQUAL TO OR GREATER THAN THE
533 ; LIMIT, THEN BIT14 OF THE SOFTWARE SWITCH REGISTER IS
534 ; CHECKED. IF BIT14 IS SET, THE MODULE IS NOT DROPPED.
535 ;
536 ;
537 ; INPUTS:
538 ; DATA TABLE ADDRESS
539 ;
540 ; IMPLICIT INPUTS:
541 ; DT.ST0,DT.CF0,DT.HMX
542 ;
543 ; OUTPUTS:
544 ; NONE
545 ;
546 ; IMPLICIT OUTPUTS:
547 ; NONE
548 ;
549 ; PATHOLOGICAL CONNECTIONS:
550 ; NONE
551 ;
552 ; SUBORDINATE ROUTINES CALLED:
553 ; 1. DRPMOD ;DROP A MODULE ROUTINE
554 ; 2. ENQTQ ;ENQUE AN ERROR MSG ROUTINE
555 ; 3. APTHER ;APT HARD ERROR ROUTINE
556 ; 4. SAVREG
557 ; 5. RESREG
558 ;
559 ; FUNCTIONAL SIDE EFFECTS:
560 ; UNDER APT ENVIRONMENT IT CAN SHUT DOWN THE EXERCISER.
561 ;
562 ; CALLING SEQUENCE:
563 ; CALL PRHARD IN <DT>
```

PRHARD (PROCESS HARD ERRORS)  
PRHARD.MAC 28-JUL-78 09:25

MACY11 30A(1052) 20-SEP-78 18:26 PAGE 19-1  
COMMON EQUATE MODULE

SEQ 0849

```
564 ; WHERE DT = DATA TABLE ADDRESS
565 ;
566 ; VERSION:
567 ; 0.0
568 ;
569 ; EDIT DATE BY REASON
570 ;--
571
```



PRHARD (PROCESS HARD ERRORS)  
PRHARD.MAC 28-JUL-78 09:25

MACY11 30A(1052) 20-SEP-78 18:26 PAGE 19-2  
PRHARD (COMMON DEFINITIONS & REFERENCES)

SEQ 0850

```
573 .SBTTL PRHARD (COMMON DEFINITIONS & REFERENCES)
574
575 .MCALL STRUCT
576 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
577 000001 $LSTIN=1
578 000001 $LSTTAG=1
579
580 ;*****
581 ; REFERENCED BY OTHER MODULES
582 ;
583 .GLOBL PRHARD ;MODULE ENTRY POINT
584
585 ;*****
586 ; GLOBAL REFERENCES
587 ;
588 .GLOBL APHER ;HARD ERROR HANDLER UNDER APT
589 .GLOBL ENQTQ ;ENQUE A MESSAGE INTO THE TYPE QUEUE
590 .GLOBL DRPMOD ;'DROP THE MODULE' ROUTINE
591 .GLOBL SAVREG
592 .GLOBL RESREG
593
594
```

```

596          .SBTTL PRHARD (CODE)
597
598 000000'    ROUTINE PRHARD <DTA>
(2) 000000'
599
600          ;+
601          ; SAVE REGS, GET DTABLE ADDRESS AND MODULE
602          ; HEADER ADDRESS.
603          ; -
604
605 000000'    CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC, SAVREG
606 000004'    LET R0 := DTA(R5)
(4) 000004' 016500 000000          MOV      DTA(R5), R0
607 000010'    LET R1 := DT.PC(R0)
(4) 000010' 016001 000002          MOV      DT.PC(R0), R1
608 000014'    LET R2 := (R1)
(4) 000014' 011102          MOV      (R1), R2
609
610          ;+
611          ; POINT TO TRAP PC
612          ; -
613 000016'    LET R3 := R1 - #2
(4) 000016' 010103          MOV      R1, R3
(6) 000020' 162703 000002          SUB      #2, R3
614
615          ;+
616          ; POINT R1 TO RETURN PC
617          ; -
618
619 000024'    LET R1 := R1 + #4
(6) 000024' 062701 000004          ADD      #4, R1
620
621          ;+
622          ; INCREMENT MODULE'S ERROR COUNT, AND ENQUE ERROR MESSAGE.
623          ; -
624
625 000030'    LET HRDCNT(R2) := HRDCNT(R2) + #1
(6) 000030' 005262 000044          INC      HRDCNT(R2)
626 000034'    LET HRDPAS(R2) := HRDPAS(R2) + #1
(6) 000034' 005262 000050          INC      HRDPAS(R2)
627 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
628
629          ;+
630          ; IF APT ENVIRONMENT, CALL APT ERROR HANDLING ROUTINE
631          ; -
632
633 000064'    IF #APTPRE SETIN DT.CFO(R0) THEN

```

```

(6) 000064' 032760 000200 000014
(9) 000072' 001406
634 000074'
      CALL APTHERR IN <R0>
(3) 000074' 010546
(4) 000076' 010045
(3) 000100' 004767 000000G
(3) 000104' 012605
635 000106'
      ELSE
(4) 000106' 000431
(3) 000110'
      50002$:
636
637
638
639
640
641
642
643 000110'
      IF #BIT15 SET IN DT.SWR(R0) THEN
(6) 000110' 032760 100000 000056
(9) 000116' 001407
644 000120'
      CALL DRPMOD IN <R0,R2>
(3) 000120' 010546
(5) 000122' 010245
(4) 000124' 010045
(3) 000126' 004767 000000G
(3) 000132' 012605
645 000134'
      ELSE
(4) 000134' 000416
(3) 000136'
      50004$:
646
647
648
649
650 000136'
      IF HRCNT(R2) HIS DT.HMX(R0) AND #BIT14 NOTSET IN DT.SWR(R0) THEN
(6) 000136' 026260 000044 000104
(9) 000144' 103412
(6) 000146' 032760 040000 000056
(9) 000154' 001006
651 000156'
      CALL DRPMOD IN <R0,R2>
(3) 000156' 010546
(5) 000160' 010245
(4) 000162' 010045
(3) 000164' 004767 000000G
(3) 000170' 012605
652 000172'
      ENDIF
(4) 000172'
      50006$:
653 000172'
      ENDIF
(4) 000172'
      50005$:
654 000172'
      ENDIF
(4) 000172'
      50003$:
655
656
657
658
659
660
      ;+
      ; RESTORE REGS AND RETURN
      ;-
```

```

BIT #APTPRE,DT.CF0(R)
BEQ 50002$
MOV R5,-(SP)
MOV R0,-(R5)
JSR PC,APTHERR
MOV (SP)+,R5
BR 50003$
```

```

BIT #BIT15,DT.SWR(R0)
BEQ 50004$
MOV R5,-(SP)
MOV R2,-(R5)
MOV R0,-(R5)
JSR PC,DRPMOD
MOV (SP)+,R5
BR 50005$
```

```

CMP HRCNT(R2),DT.HM
BLO 50006$
BIT #BIT14,DT.SWR(R0)
BNE 50006$
MOV R5,-(SP)
MOV R2,-(R5)
MOV R0,-(R5)
JSR PC,DRPMOD
MOV (SP)+,R5
```

```
661 000172'          CALL RESREG
(3) 000172' 004767 000000G          JSR    PC,RESREG
662
663 000176'          ENDRTN
(3) 000176'
(3) 000176'          50000S:
(2) 000176' 000207          50001S:
664                                     RTS    PC
665          000001          .END
```

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

SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$ERFL= 000000
TMPIO = 000002	WBUFEA= 000136	\$F\$GDO= 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\$LDO= 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	\$\$AVLE= 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	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:27

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0856

3 COMMON EQUATE MODULE  
553 COMMON REFERENCES AND DEFINITIONS  
556 000000' .PRINT ;SPMAC: VERSION 1.1  
579 PRMSG ROUTINE

```
508 .TITLE PRMSG - PROCESS MSG
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRMSG
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; PROCESSES MSG CALLS FROM OPTION MODULES BY LOADING
517 ; THE TYPE-QUEUE WITH THE SPECIFIED MESSAGE
518 ;
519 ; INPUTS:
520 ; DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ; DT.PC
524 ;
525 ; OUTPUTS:
526 ; NONE
527 ;
528 ; IMPLICIT OUTPUTS:
529 ; NONE
530 ;
531 ; PATHOLOGICAL CONNECTIONS:
532 ; NONE
533 ;
534 ; SUBORDINATE MODULES CALLED:
535 ; ENQTQ - TYPE-QUEUE ENQUEUER
536 ; SAVREG
537 ; RESREG
538 ;
539 ; FUNCTIONAL SIDE EFFECTS:
540 ; NONE
541 ;
542 ; CALLING SEQUENCE:
543 ; CALL PRMSG IN <A>
544 ; A=ADDRESS OF DATA TABLE
545 ;
546 ; VERSION:
547 ; 0.0
548 ;
549 ; EDIT BY DATE REASON
550 ;
551 ;--
```



```
553          .SBTTL COMMON REFERENCES AND DEFINITIONS
554
555          .MCALL STRUCT
556 000000'    STRUCT
557 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
558          $LSTIN=1
559          $LSTTAG=1
560
561          ;
562          ;*****
563          ;
564          ; REFERENCED BY OTHER MODULES
565          ;
566          .GLOBL PRMSG          ;MODULE ENTRY POINT
567          ;
568          ;*****
569          ;
570          ;GLOBAL REFERENCES
571          ;
572          .GLOBL ENQTQ          ;TYPE-QUEUE ENQUEUER
573          .GLOBL SAVREG
574          .GLOBL RESREG
575          ;
576          ;*****
577          ;
```

```

579          .SBTTL PRMSG ROUTINE
580
581
582 000000'          ROUTINE PRMSG <TABL>
(2) 000000'
583
584          ;+
585          ;SAVE REGISTERS
586          ;+
587
588 000000'          CALL SAVREG
(3) 000000' 004767 000000G
589
590
591          ;+
592          ;SET R0 TO START OF DTABLE
593          ;+
594
595 000004'          LET R0 := TABL(R5)
(4) 000004' 016500 000000
596
597
598          ;+
599          ;GET PC+2 OF THE MSG TRAP CALL FROM THE DATA TABLE
600          ;+
601
602 000010'          LET R1 := DT.PC(R0)
(4) 000010' 016001 000002
603
604
605          ;+
606          ;GET THE OPTION MODULE HEADER ADDRESS AND MESSAGE ADDRESS.
607          ;+
608
609 000014'          LET R2 := (R1)+
(4) 000014' 012102
610 000016'          LET R3 := (R1)+
(4) 000016' 012103
611
612          ;+
613          ;ENQUEUE THE MESSAGE, USING THE STANDARD MESSAGE CODE
614          ;+
615
616 000020'          CALL ENQTQ IN <R0,#MSGSTD,R3,R2,R1>
(3) 000020' 010546
(8) 000022' 010145
(7) 000024' 010245
(6) 000026' 010345
(5) 000030' 012745 000000
(4) 000034' 010045
(3) 000036' 004767 000000G
(3) 000042' 012605
617
618
619          ;+
620          ;RESTORE REGISTERS

```

PRMSG:

```

JSR    PC, SAVREG
MOV    TABL(R5), R0
MOV    DT.PC(R0), R1
MOV    (R1)+, R2
MOV    (R1)+, R3
MOV    R5, -(SP)
MOV    R1, -(R5)
MOV    R2, -(R5)
MOV    R3, -(R5)
MOV    #MSCSTD, -(R5)
MOV    R0, -(R5)
JSR    PC, ENQTQ
MOV    (SP)+, R5

```

```
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'
(3) 000050' 50000$:
(2) 000050' 000207 50001$:
631 000001 .END RTS PC
```

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  
CTRL0 = 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  
ENBEDP= 010000  
ENBNUL= 000001  
ENDLST= 000000  
ENQTQ = \*\*\*\*\* G

EOPBIT= 000001  
ERRTYP= 000106  
EVNTBE= 000200  
EVNTHD= 000200  
EVNTKT= 000203  
EVNTP= 000202  
EVNTRE= 000201  
FATERR= 100000  
HRDCNT= 000044  
HRDPAS= 000050  
ICONT = 000036  
ICOUNT= 000040  
IDNUM = 000122  
IE = 000100  
INDPAR= 000040  
INHDRP= 040000  
INHPR= 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  
KTXND= 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  
MSGGECH= 177777  
MSGGEOP= 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  
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  
SENDL= 177777  
SOFcnt= 000042  
SOFPAS= 000046  
SPACE = 000040  
SPCINT= 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  
TMPID = 000002

TQOVF = 000002	WASADR= 000104	\$\$BBLA= 000170	\$F\$UNT= 000130	\$\$ELOC= 000000
UIPAR0= 177640	WBSTAT= 000040	\$\$FCAS= 000150	\$F\$WHI= 000120	\$\$ERFL= 000000
UIPAR1= 177642	WBUFEA= 000136	\$\$FDEC= 000220	\$F\$YES= 000402	\$\$FLAG= 000000
UIPAR2= 177644	WBUFPA= 000134	\$\$FDO = 000340	\$IFLEV= 177777	\$\$FROM= 000000
UIPAR3= 177646	WBUFRQ= 000140	\$\$F\$FAL= 000405	\$LOCTA= 177777	\$\$LOC = 000000
UIPAR4= 177650	WBUFSZ= 000142	\$\$FGOO= 000400	\$LSTIN= 000001	\$\$LOCN= 000000
UIPAR5= 177652	WDFR = 000116	\$\$F\$IF = 000110	\$LSTA= 000001	\$\$REG = 177777
UIPAR6= 177654	WDT0 = 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	\$\$TD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:28  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0863

3 COMMON EQUATE MODULE  
554 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
580 PRMSGN ROUTINE

```
508 .TITLE PRMSGN - PROCESS MSGN
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRMSGN
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; PROCESSES MSGN CALLS FROM OPTION MODULES BY LOADING
517 ; THE TYPE-QUEUE WITH, FIRST, AN ENTRY FOR THE HEADER
518 ; MESSAGE AND, THEN, AN ENTRY FOR EACH ADDRESS IN THE
519 ; MESSAGE ADDRESS TABLE
520 ;
521 ; INPUTS:
522 ; DATA TABLE ADDRESS
523 ;
524 ; IMPLICIT INPUTS:
525 ; DT.PC
526 ;
527 ; OUTPUTS:
528 ; NONE
529 ;
530 ; IMPLICIT OUTPUTS:
531 ; NONE
532 ;
533 ; PATHOLOGICAL CONNECTIONS:
534 ; NONE
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ; PRMSGS - PROCESS MSGS ROUTINE
538 ; ENQTQ - TYPE-QUEUE ENQUEUER
539 ;
540 ; FUNCTIONAL SIDE EFFECTS:
541 ; NONE
542 ;
543 ; CALLING SEQUENCE:
544 ; CALL PRMSGN IN <A>
545 ; A=ADDRESS OF DATA TABLE
546 ;
547 ; VERSION:
548 ; 0.0
549 ;
550 ; EDIT BY DATE REASON
551 ;
552 ;--
```

```
554 .SBTTL COMMON DEFINITIONS AND REFERENCES
555
556
557 .MCALL STRUCT
558 STRUCT
559 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
560 000001 $LSTIN=1
561 000001 $LSTTAG=1
562
563 ;
564 ;*****
565 ;
566 ; REFERENCED BY OTHER MODULES
567 ;
568 .GLOBL PRMSGN ;MODULE ENTRY POINT
569 ;
570 ;*****
571 ;
572 ; GLOBAL REFERENCES
573 ;
574 .GLOBL ENQTQ ;TYPE-QUEUE ENQUEUER
575 .GLOBL PRMSGS ;PROCESS MSGS ROUTINE
576 ;
577 ;*****
578 ;
```



```

580          .SBTTL PRMSGN ROUTINE
581
582 000000'    ROUTINE PRMSGN <TABL>
(2) 000000'
583
584          ;+
585          ;SAVE REGISTERS
586          ;-
587
588 000000'    PUSH R0,R1
(2) 000000' 010046          MOV     R0,-(SP)
(3) 000002' 010146          MOV     R1,-(SP)
589
590
591          ;+
592          ;SET R0 TO START OF DATA TABLE
593          ;-
594
595 000004'    LET R0 := TABL(R5)
(4) 000004' 016500 000000          MOV     TABL(R5),R0
596
597
598          ;+
599          ;ENQUEUE THE SKELETAL HEADER MESSAGE, SETTING THE MESSAGE ADDRESS
600          ;PARAMETER TO THE TRAP ADDRESS IN THE OPTION MODULE AND
601          ;THE RETURN ADDRESS PARAMETER TO #0.
602          ;(THE TRAP ADDRESS IS NEEDED FOR THE HEADER MESSAGE.)
603          ;-
604
605 000010'    LET R1 := DT.PC(R0) - #2
(4) 000010' 016001 000002          MOV     DT.PC(R0),R1
(6) 000014' 162701 000002          SUB     #2,R1
606 000020'    CALL ENQTQ IN <R0,#MSGHDR,R1,2(R1),#0>
(3) 000020' 010546          MOV     R5,-(SP)
(8) 000022' 012745 000000          MOV     #0,-(R5)
(7) 000026' 016145 000002          MOV     2(R1),-(R5)
(6) 000032' 010145          MOV     R1,-(R5)
(5) 000034' 012745 000004          MOV     #MSGHDR,-(R5)
(4) 000040' 010045          MOV     R0,-(R5)
(3) 000042' 004767 000000G        JSR     PC,ENQTQ
(3) 000046' 012605          MOV     (SP)+,R5
607
608
609          ;+
610          ;ENQUEUE THE MESSAGES LISTED IN THE MESSAGE ADDRESS TABLE
611          ;BY CALLING "PROCESS MSGS".
612          ;-
613
614 000050'    CALL PRMSG IN <R0>
(3) 000050' 010546          MOV     R5,-(SP)
(4) 000052' 010045          MOV     R0,-(R5)
(3) 000054' 004767 000000G        JSR     PC,PRMSG
(3) 000060' 012605          MOV     (SP)+,R5
615
616
617          ;+
    
```

```
618 ;RESTORE REGISTERS
619 ;-
620
621 000062' POP R1,R0
(2) 000062' 012601
(3) 000064' 012600 MOV (SP)+,R1
MOV (SP)+,R0
622
623
624 ;+
625 ;RETURN
626 ;-
627
628 000066' ENDRTN
(3) 000066'
(3) 000066' 50000$:
(2) 000066' 000207 50001$:
629 RTS PC
630 000001 .END
```

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RBUFSZ= 000132
ACTBIT= 004000	CTRLD = 000017	ERRTYP= 000106	MSGCKD= 000010	RBUFVA= 000124
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RDSERV= 000101
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RDWHMI= 000022
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELERR= 000020
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RELMOD= 020000
ASB = 000106	DROPMQ= 100000	EVNTRE= 000201	MSGEOP= 000013	RELTIM= 010000
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES1 = 000056
ASTAT = 000104	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RES2 = 000060
AUTO = 000010	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RICHAR= 031060
AUTOST= 020000	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RPTDAT= 002000
AWAS = 000110	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RSTRT = 000112
BIT0 = 000001	DT.CF0= 000014	IDNUM = 000122	MSGPOP= 000002	RUBOUT= 000177
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RUNMOD= 100000
BIT01 = 000002	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	R5VALU= 001740
BIT02 = 000004	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SAM = 075464
BIT03 = 000010	DT.EVN= 000000	INHPR= 020000	MSGSKE= 000003	SBADR = 000102
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	SBKMOD= 000000
BIT05 = 000040	DT.FCH= 000037	INHRE= 000400	MSGSMH= 000014	SBKSEL= 010000
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SC.ADR= 000006
BIT07 = 000200	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.ALC= 000014
BIT08 = 000400	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.APC= 000016
BIT09 = 001000	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.CKL= 000002
BIT1 = 000002	DT.KBR= 000022	IOMODR= 112000	NBKMOD= 001000	SC.CKP= 000004
BIT10 = 002000	DT.KBU= 000030	IOMODX= 110000	NCPUOP= 000020	SC.CLO= 000000
BIT11 = 004000	DT.MLS= 000032	JACK = 035060	NCAPTY= 000002	SC.HLD= 000010
BIT12 = 010000	DT.MTI= 000110	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT13 = 020000	DT.OFF= 000070	KIPAR1= 172342	OWEN = 024020	SENDLS= 177777
BIT14 = 040000	DT.PAS= 000074	KIPAR2= 172344	PAERR = 000010	SQFCNT= 000042
BIT15 = 100000	DT.PC = 000002	KIPAR3= 172346	PARPRE= 002000	SQFPAS= 000046
BIT2 = 000004	DT.PFL= 000062	KIPAR4= 172350	PARSTA= 000100	SPACE = 000040
BIT3 = 000010	DT.PSW= 000004	KIPAR5= 172352	PASCNT= 000034	SPOINT= 000032
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PDP60 = 004000	SPVALU= 002200
BIT5 = 000040	DT.RCS= 000102	KIPAR7= 172356	PDP70 = 010000	SRO = 177572
BIT6 = 000100	DT.REL= 000040	KIPDR0= 172300	PRI0 = 000000	SR1 = 177574
BIT7 = 000200	DT.SCT= 000066	KIPDR1= 172302	PRI1 = 000040	SR2 = 177576
BIT8 = 000400	DT.SMX= 000106	KIPDR2= 172304	PRI2 = 000200	SR3 = 172516
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PRI3 = 000240	STAT = 000026
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI4 = 000300	STATBI= 064757
BKMOD = 000020	DT.ST0= 000010	KIPDR5= 172312	PRI5 = 000340	STAT1 = 000027
BKMODE= 040000	DT.ST1= 000012	KIPDR6= 172314	PRI6 = 000340	SUSPND= 000001
BKSLSH= 000134	DT.SWR= 000056	KIPDR7= 172316	PRMSGN 000000RG	SVR0 = 000062
CAPRES= 000004	DT.SYP= 000072	KTERRO= 000040	PRMSGS= ***** G	SVR1 = 000064
CASTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PRO = 000000	SVR2 = 000066
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR4 = 000200	SVR3 = 000070
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR5 = 000240	SVR4 = 000072
CKTIM = 100000	DVID1 = 000014	LF = 000012	PR6 = 000300	SVR5 = 000074
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PR7 = 000340	SVR6 = 000076
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	PS = 177776	SYSCNT= 000052
CQOVF = 000001	ENBEOP= 010000	MED = 076600	PSW = 177776	SYSERR= 000100
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RANNUM= 000054	TABL = 000000
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFEA= 000130	TMPIO = 000002
CSRC = 000102	ENQTQ = ***** G	MODHOL= 002000	RBUFPA= 000126	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	WBUFQR= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$LOC = 000000
UIPAR4= 177650	WBUFSS= 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\$LQO= 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	\$\$TD = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:28  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0870

3 COMMON EQUATE MODULE  
554 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
581 PRMSGs ROUTINE

```
508 .TITLE PRMSGSG - PROCESS MSGS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRMSGSG
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; PROCESSES MSGS CALLS FROM OPTION MODULES BY LOADING
517 ; THE TYPE-QUEUE WITH AN ENTRY FOR EACH ADDRESS
518 ; IN THE MESSAGE ADDRESS TABLE
519 ;
520 ; INPUTS:
521 ; ADDRESS OF DATA TABLE
522 ;
523 ; IMPLICIT INPUTS:
524 ; DT.PC
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; NONE
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ; ENQTQ - TYPE-QUEUE ENQUEUER
537 ; SAVREG - SAVE REGISTERS
538 ; RESREG - RESTORE REGISTERS
539 ;
540 ; FUNCTIONAL SIDE EFFECTS:
541 ; NONE
542 ;
543 ; CALLING SEQUENCE:
544 ; CALL PRMSGSG IN <A>
545 ; A=ADDRESS OF DATA TABLE
546 ;
547 ; VERSION:
548 ; 0.0
549 ;
550 ; EDIT BY DATE REASON
551 ;
552 ;
```

```
554 .SBTTL COMMON DEFINITIONS AND REFERENCES
555
556
557 .MCALL STRUCT
558 STRUCT
559 000000' .PRINT ;SPMAC: VERSION 1.1
560 (1) 000000' $LSTIN=1
561 000001 $LSTTAG=1
562 000001
563
564 ;
565 ;*****
566 ; REFERENCED BY OTHER MODULES
567 ;
568 .GLOBL PRMSGs ;MODULE ENTRY POINT
569 ;
570 ;*****
571 ;
572 ; GLOBAL REFERENCES
573 ;
574 .GLOBL ENQTQ ;TYPE-QUEUE ENQUEUEUER
575 .GLOBL SAVREG ;SAVE REGISTERS
576 .GLOBL RESREG ;RESTORE REGISTERS
577 ;
578 ;*****
579 ;
```

```

581          .SBTTL PRMSGSGS ROUTINE
582
583 000000'  ROUTINE PRMSGSGS <TABL>
(2) 000000'
584
585          ;+
586          ;SAVE REGISTERS
587          ;--
588
589 000000'  CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC,SAVREG
590
591
592          ;+
593          ;SET R0 TO START OF DATA TABLE
594          ;--
595
596 000004'  LET R0 := TABL(R5)
(4) 000004' 016500 000000          MOV      TABL(R5),R0
597
598
599          ;+
600          ;GET PC+2 OF THE MSGSGS TRAP CALL
601          ;--
602
603 000010'  LET R1 := DT.PC(R0)
(4) 000010' 016001 000002          MOV      DT.PC(R0),R1
604
605
606          ;+
607          ;GET THE OPTIONS MODULE'S HEADER ADDRESS, THE MESSAGE ADDRESS TABLE,
608          ;AND THE RETURN ADDRESS. (THE MESSAGE ADDRESS TABLE ADDRESS
609          ;IS PLACED IN R2).
610          ;--
611
612 000014'  LET R3 := (R1)+
(4) 000014' 012103          MOV      (R1)+,R3
613 000016'  LET R2 := (R1)+
(4) 000016' 012102          MOV      (R1)+,R2
614
615
616          ;+
617          ;BUILD AND ENQUEUE A TYPE-QUEUE ENTRY FOR EACH ADDRESS IN THE MESSAGE
618          ;ADDRESS TABLE. THE RETURN ADDRESS FOR EACH ENTRY EXCEPT THE LAST
619          ;ONE WILL BE 0. THE LAST ENTRY WILL CONTAIN THE ACTUAL RETURN ADDRESS
620          ;FOR THE OPTION MODULES.
621          ;--
622
623 000020'  WHILE (R2) NE #-1 DO
(4) 000020'
(6) 000020' 021227 177777          50002$:  CMP      (R2),#-1
(9) 000024' 001422          BEQ      50003$
624 000026'          IF 2(R2) EQ #-1 THEN
(6) 000026' 026227 000002 177777          CMP      2(R2),#-1
(9) 000034' 001002          BNE      50004$
625 000036'          LET R4 := R1
    
```



```

(4) 000036' 010104
626 000040'
(4) 000040' 000401
(3) 000042'
627 000042'
(4) 000042' 005004
628 000044'
(4) 000044'
S29 000044'
(3) 000044' 010546
(8) 000046' 010445
(7) 000050' 010345
(6) 000052' 012245
(5) 000054' 012745 000000
(4) 000060' 010045
(3) 000062' 004767 000000G
(3) 000066' 012605
630 000070'
(4) 000070' 000753
(3) 000072'
631
632
633
634
635
636
637
638 000072'
(3) 000072' 004767 000000G
639
640
641
642
643
644
645 000076'
(3) 000076'
(3) 000076'
(2) 000076' 000207
646
647 000001
  
```

```

ELSE
LET R4 := #0
ENDIF
CALL ENQTQ IN <R0,#MSGSTD,(R2)+,R3,R4>
ENDDO
;+
;RESTORE REGISTERS
;-
CALL RESREG
;+
;RETURN
;-
ENDRTN
.END
  
```

```

MOV R1,R4
BR 50005$
50004$:
CLR R4
50005$:
MOV R5,-(SP)
MOV R4,-(R5)
MOV R3,-(R5)
MOV (R2)+,-(R5)
MOV #MSGSTD,-(R5)
MOV R0,-(R5)
JSR PC,ENQTQ
MOV (SP)+,R5
BR 50002$
50003$:
JSR PC,RESREG
50000$:
50001$:
RTS PC
  
```

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124
ACTBIT= 004000	CTRL0 = 000017	ERRTP= 000106	MSGCKD= 000010	RDSERV= 000101
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RELTIM= 010000
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGEOP= 000013	RESREG= ***** G
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES1 = 000056
ASTAT = 000104	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RES2 = 000060
AUTO = 000010	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RICHAR= 031060
AUTDST= 020000	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RPTDAT= 002000
AWAS = 000110	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RSTRT = 000112
BIT0 = 000001	DT.CF0= 000014	IDNUM = 000122	MSGPOP= 000002	RUBOUT= 000177
BIT00 = 000001	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RUNMOD= 100000
BIT01 = 000002	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	RSVALU= 001740
BIT02 = 000004	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SAM = 075464
BIT03 = 000010	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	SAVREG= ***** G
BIT04 = 000020	DT.EXS= 000060	INHREL= 001000	MSGSMS= 000015	SBADR = 000102
BIT05 = 000040	DT.FCH= 000037	INHRR= 000400	MSGSMH= 000014	SBKMOD= 000000
BIT06 = 000100	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SBKSEL= 010000
BIT07 = 000200	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.ADR= 000006
BIT08 = 000400	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.ALC= 000014
BIT09 = 001000	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.APC= 000016
BIT1 = 000002	DT.KBR= 000022	IOMODR= 112000	NCPUOP= 000020	SC.CKL= 000002
BIT10 = 002000	DT.KBU= 000030	IOMODX= 110000	NOAPTY= 000002	SC.CKP= 000004
BIT11 = 004000	DT.MLS= 000032	JACK = 035060	NULL = 000000	SC.CLO= 000000
BIT12 = 010000	DT.MTI= 000110	KIPAR0= 172340	OWEN = 024020	SC.HLD= 000010
BIT13 = 020000	DT.OFF= 000070	KIPAR1= 172342	PAERR = 000010	SC.SCA= 000012
BIT14 = 040000	DT.PAS= 000074	KIPAR2= 172344	PARPRE= 002000	SENDLS= 177777
BIT15 = 100000	DT.PC = 000002	KIPAR3= 172346	PARSTA= 000100	SJFCNT= 000042
BIT2 = 000004	DT.PFL= 000062	KIPAR4= 172350	PASCNT= 000034	SOFPAS= 000046
BIT3 = 000010	DT.PSW= 000004	KIPAR5= 172352	PDPLSI= 020000	SPACE = 000040
BIT4 = 000020	DT.PTA= 000064	KIPAR6= 172354	PDP60 = 004000	SPOINT= 000032
BIT5 = 000040	DT.RCS= 000102	KIPAR7= 172356	PDP70 = 010000	SPVALU= 002200
BIT6 = 000100	DT.REL= 000040	KIPDR0= 172300	PRI0 = 000000	SR0 = 177572
BIT7 = 000200	DT.SCT= 000066	KIPDR1= 172302	PRI1 = 000040	SR1 = 000000
BIT8 = 000400	DT.SMX= 000106	KIPDR2= 172304	PRI4 = 000200	SR2 = 177576
BIT9 = 001000	DT.SP = 000006	KIPDR3= 172306	PRI5 = 000240	SR3 = 172516
BKDEF = 000002	DT.SSI= 000046	KIPDR4= 172310	PRI6 = 000300	STAT = 000026
BKMOD = 000020	DT.ST0= 000010	KIPDR5= 172312	PRI7 = 000340	STATBI= 064757
BKMODE= 040000	DT.ST1= 000012	KIPDR6= 172314	PRMSGS 000000RG	STAT1 = 000027
BKSLSH= 000134	DT.SWR= 000056	KIPDR7= 172316	PRO = 000000	SUSPND= 000001
CAPRES= 000004	DT.SYP= 000072	KTERRO= 000040	PR4 = 000200	SVR0 = 000062
CASSTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PR5 = 000240	SVR1 = 000064
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR6 = 000300	SVR2 = 000066
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR7 = 000340	SVR3 = 000070
CKTIM = 100000	DVID1 = 000014	LF = 000012	PS = 177776	SVR4 = 000072
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PSW = 177776	SVR5 = 000074
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	RANNUM= 000054	SVR6 = 000076
CQOVF = 000001	ENBEOP= 010000	MED = 076600	RBUFVA= 000130	SYSNT= 000052
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFPA= 000126	SYSERR= 000100
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132	TABL = 000000
CSRC = 000102	ENQTQ = ***** G	MODHOL= 002000		TMPID = 000002

TQOVF = 000002	WBUFEA= 000136	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$ERFL= 000000
UIPAR0= 177640	WBUFPA= 000134	\$F\$G00= 000400	\$LSTTA= 000001	\$\$FLAG= 000001
UIPAR1= 177642	WBUFRQ= 000140	\$F\$IF = 000110	\$NESTL= 177777	\$\$FROM= 000000
UIPAR2= 177644	WBUFSZ= 000142	\$F\$INC= 000210	\$NSKO = 000300	\$\$LOC = 000034R
UIPAR3= 177646	WDFR = 000116	\$F\$L00= 000200	\$NSK1 = 000120	\$\$LOCN= 000000
UIPAR4= 177650	WDT0 = 000114	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$REG = 177777
UIPAR5= 177652	WTINRE= 000352	\$F\$ND = 000403	\$\$SAVLE= 177777	\$\$RETU= 000000
UIPAR6= 177654	WTWHMI= 000222	\$F\$OR = 000320	\$\$SSKO = 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	\$TSKO = 050002	\$\$TGS1= 000000
UIPDR3= 177606	\$ERFLG= 000400	\$F\$TRU= 000404	\$TSK1 = 050003	\$\$TGS2= 000000
UIPDR4= 177610	\$F\$AND= 000310	\$F\$UNT= 000130	\$TSK2 = 050005	\$\$TD = 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	\$ISKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:29  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0877

3 COMMON EQUATE MODULE  
556 COMMON DEFINITIONS AND REFERENCES FOR PROTOA  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
577 PROTOA ROUTINE

```
508 .TITLE PROTOA PROCESS THE 'OTOA' MACRO IN OPTION MODULES
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PROTOA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS ROUTINE PROCESSES THE 'OTOA' TRAP ISSUED BY OPTION
517 ; MODULES. IT WILL RETRIEVE THE NUMBER TO BE CONVERTED TO
518 ; OCTAL ASCII FROM DT.PC+2 AND THE TABLE ADDRESS AT WHICH
519 ; TO STORE THE OCTAL ASCII FROM DT.PC +4. IT CALLS THE
520 ; BINARY TO OCTAL ASCII CONVERSION ROUTINE. THE RETURN PC
521 ; IS UPDATED SINCE THIS IS DIRECT SERVICE.
522 ;
523 ; INPUTS:
524 ; DTABLE ADDRESS
525 ;
526 ; IMPLICIT INPUTS:
527 ; DT.PC
528 ;
529 ; OUTPUTS:
530 ; NONE
531 ;
532 ; IMPLICIT OUTPUTS:
533 ; NONE
534 ;
535 ; PATHOLOGICAL CONNECTIONS:
536 ; NONE
537 ;
538 ; SUBORDINATE ROUTINES CALLED:
539 ; BOA16 - BINARY TO OCTAL ASCII CONVERSION ROUTINE
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ; NONE
543 ;
544 ; CALLING SEQUENCE:
545 ; CALL PROTOA IN <DTADR>
546 ; WHERE DTADR = ADDRESS OF DATA TABLE
547 ;
548 ; VERSION:
549 ; 0.0
550 ;
551 ; EDIT DATE BY REASON
552 ;--
553
554
```

```
556 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR PROTOA
557
558 .MCALL STRUCT
559 000000' STRUCT
560 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
561 000001 $LSTIN=1
562 000001 $LSTTAG=1
563
564 ;*****
565 ;
566 ; REFERENCED BY OTHER MODULES:
567 ;
568 .GLOBL PROTOA ;MODULE ENTRY POINT
569 ;
570 ;*****
571 ;
572 ; GLOBAL REFERENCES:
573 ;
574 .GLOBL BOA16 ;BINARY TO OCTAL ASCII CONVERSION ROUTINE
575
```

```

577          .SBTTL  PROTOA ROUTINE
578
579 000000'    ROUTINE PROTOA <DTADR>
(2) 000000'
580
581          ;+
582          ;INITIALIZE AND SAVE DATA TABLE ADDRESS
583          ;-
584
585
586 000000'    PUSH R0,R1
(2) 000000' 010046
(3) 000002' 010146
587 000004'    LET R0 := DTADR(R5)
(4) 000004' 016500 000000
588
589          ;+
590          ; GET DT.PC AND CALL BOA16
591          ;-
592
593 000010'    LET R1 := DT.PC(R0)
(4) 000010' 016001 000002
594 000014'    CALL BOA16 IN <@2(R1),4(R1)>
(3) 000014' 010546
(5) 000016' 016145 000004
(4) 000022' 017145 000002
(3) 000026' 004767 000000G
(3) 000032' 012605
595
596          ;+
597          ; UPDATE RETURN PC
598          ;-
599
600 000034'    LET DT.PC(R0) := DT.PC(R0) + #6
(6) 000034' 062760 000006 000002
601
602          ;+
603          ; CLEAN UP AND GOODBYE
604          ;-
605
606 000042'    POP R1,R0
(2) 000042' 012601
(3) 000044' 012600
607
608 000046'    ENDRTN
(3) 000046'
(3) 000046'
(2) 000046' 000207
609
610          .END
    
```

PROTOA:

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV DTADR(R5),R0
MOV DT.PC(R0),R1
MOV R5,-(SP)
MOV 4(R1),-(R5)
MOV @2(R1),-(R5)
JSR PC,BOA16
MOV (SP)+,R5
ADD #6,DT.PC(R0)
MOV (SP)+,R1
MOV (SP)+,R0
    
```

50000\$:  
50001\$:  
RTS PC

ACSR = 000102	CSRC = 000102	ENDLST= 000000	MODHOL= 002000	RBUFSZ= 000132
ACTBIT= 004000	CTRLC = 000003	EOPBIT= 000001	MODSEL= 001000	RBUFVA= 000124
ADDR22= 001000	CTRLO = 000017	ERRTYP= 000103	MSGCKD= 000010	RDSERV= 000101
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	MSGCKS= 000011	RDWHMI= 000022
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	MSGDER= 000005	RELERR= 000020
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	MSGDRP= 000017	RELMOD= 020000
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	MSGECH= 177777	RELTIM= 010000
ASSEMB= 000010	DROPMQ= 100000	EVNTRE= 000201	MSGEOP= 000013	RES1 = 000056
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
AUTO = 000010	DTADR = 000000	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUBOUT= 000177
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT01 = 000002	DT.CF0= 000014	IE = 000100	MSGPRM= 177776	RSVALU= 001740
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBADR = 000102
BIT04 = 000020	DT.ESI= 000044	INHEPR= 020000	MSGSKE= 000003	SBKMOD= 000000
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SBKSEL= 010000
BIT06 = 000100	DT.EXS= 000060	INHRE= 000400	MSGSMH= 000014	SC.ADR= 000006
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.ALC= 000014
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.APC= 000016
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKL= 000002
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CKP= 000004
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.CLO= 000000
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	NCPUQP= 000020	SC.HLD= 000010
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	NOPTY= 000002	SC.SCA= 000012
BIT13 = 020000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SENDLS= 177777
BIT14 = 040000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SOFCNT= 000042
BIT15 = 100000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOFPAS= 000046
BIT2 = 000004	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPACE = 000040
BIT3 = 000010	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPOINT= 000032
BIT4 = 000020	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPVALU= 002200
BIT5 = 000040	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SR0 = 177572
BIT6 = 000100	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR1 = 177574
BIT7 = 000200	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR2 = 177576
BIT8 = 000400	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	SR3 = 172516
BIT9 = 001000	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	STAT = 000026
BKDEF = 000002	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STATBI= 064757
BKMOD = 000020	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	STAT1 = 000027
BKMODE= 040000	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	SUSPND= 000001
BKSLSH= 000134	DT.ST0= 000010	KIPDR6= 172314	PRI7 = 000340	SVR0 = 000062
BOA16 = ***** G	DT.ST1= 000012	KIPDR7= 172316	PROTOA 000000RG	SVR1 = 000064
CAPRES= 000004	DT.SWR= 000056	KTERRO= 000040	PRO = 000000	SVR2 = 000066
CASTAT= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR3 = 000070
CDERCT= 000146	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR4 = 000072
CDWDCT= 000144	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR5 = 000074
CKTIM = 100000	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR6 = 000076
CLKPRE= 000001	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SYSCNT= 000052
CONFIG= 000056	ECCMEM= 000100	MAPSTA= 000200	PSW = 177776	YSERR= 000100
CQOVF = 000001	ECCSTA= 000010	MED = 076600	RANUM= 000054	TMPID = 000002
CR = 000015	ENBEOP= 010000	MEMPAS= 040000	RBUFEA= 000130	TQOVF = 000002
CSRA = 000100	ENBNUL= 000001	MODEXH= 004000	RBUFPA= 000126	UIPARO= 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\$G00= 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	\$\$RETN= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$L00= 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	\$\$ELDC= 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:29  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0883

3 COMMON EQUATE MODULE  
556 COMMON DEFINITIONS AND REFERENCES  
559 000000' .PRINT ;SPMAC: VERSION 1.1  
577 PRRAND (CODE)

PRRAND (PROCESS 'RAND' TRAP CALL)  
PRRAND.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:29 PAGE 19  
COMMON EQUATE MODULE

SEQ 0884

508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554

```
.TITLE PRRAND (PROCESS 'RAND' TRAP CALL)
.IDENT /VO.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
;-----
```

PRRAND (PROCESS 'RAND' TRAP CALL)  
PRRAND.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:29 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 0885

556  
557  
558  
559 000000'  
(1) 000000'  
560 000001  
561 000001  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575

.SBTTL COMMON DEFINITIONS AND REFERENCES  
.MCALL STRUCT  
STRUCT  
.PRINT ;SPMAC: VERSION 1.1  
\$LSTIN=1  
\$LSTTAG=1  
  
;\*\*\*\*\*  
;  
; REFERENCED BY OTHER MODULES  
;  
.GLOBL PRRAND ;MODULE'S ENTRY POINT  
;  
;\*\*\*\*\*  
; GLOBAL REFERENCES  
;  
.GLOBL RANDOM ;ROUTINE 'RANDOM'

```

577 .SBTTL PRRAND (CODE)
578
579 000000' ROUTINE PRRAND <AA>
(2) 000000' PRRAND:
580
581 ;+
582 ; SAVE REGISTERS
583 ; -
584
585 000000' PUSH R0,R1
(2) 000000' 010046 MOV R0,-(SP)
(3) 000002' 010146 MOV R1,-(SP)
586
587 ;+
588 ; GET DTABLE ADDRESS ,GET HEADER ADDRESS AND GET RANDOM NUMBER ADDRESS
589 ; IN HEADER
590 ; -
591
592 000004' LET R0 := AA(R5)
(4) 000004' 016500 000000 MOV AA(R5),R0
593 000010' LET R1 := DT.PC(R0)
(4) 000010' 016001 000002 MOV DT.PC(R0),R1
594 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)
595 000026' LET R1 := (R1)
(4) 000026' 011101 MOV (R1),R1
596
597 ;+
598 ; GET RANDOM NUMBER
599 ; -
600
601 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)
602
603 ;+
604 ; RESTORE REGISTERS AND RETURN
605 ; -
606
607 000044' POP R1,R0
(2) 000044' 012601 MOV (SP)+,R1
(3) 000046' 012600 MOV (SP)+,R0
608 000050' ENDRTN
(3) 000050'
(3) 000050'
(2) 000050' 000207 50000$:
609 RTS 50001$:
610 000001 .END PC

```

AA = 000000	CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RBUFSZ= 000132
ACSR = 000102	CTRLC = 000003	ERRTYP= 000106	MSGCKD= 000010	RBUFVA= 000124
ACTBIT= 004000	CTRLD = 000017	EVNTBE= 000200	MSGCKS= 000011	RDSERV= 000101
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RDWHMI= 000022
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RELERR= 000020
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	RELMOD= 020000
APTPRE= 000200	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	RSVALU= 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	INHRR= 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	NCPUQP= 000020	SC.CLO= 000000
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NQAPTY= 000002	SC.HLD= 000010
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SENDLS= 177777
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	SOFCNT= 000042
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SOFPAS= 000046
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPACE = 000040
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPOINT= 000032
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PDP1SI= 020000	SPVALU= 002200
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR0 = 177572
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR1 = 177574
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	SR2 = 177576
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	SR3 = 172516
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STAT = 000026
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	STATBI= 064757
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	STAT1 = 000027
BKMOD = 000020	DT.ST0= 000010	KIPDR6= 172314	PRI7 = 000340	SUSPND= 000001
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRRAND 000000RG	SVR0 = 000062
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PRO = 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	RANDOM= ***** G	SYSERR= 000100
CQOVF = 000001	ENBEOP= 010000	MEMPAS= 040000	RANNUM= 000054	TMPID = 000002
CR = 000015	ENBNUL= 000001	MODEXH= 004000	RBUFEA= 000130	TQOVF = 000002
CSRA = 000100	ENDLST= 000000	MODHOL= 002000	RBUFPA= 000126	UIPARO= 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\$GDO= 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\$LDO= 000200	\$NSKO = 000300	\$\$RTN1= 050000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$\$RTN2= 050001
UIPDR1= 177602	XFLAG = 000005	\$F\$ND = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:30  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0889

3 COMMON EQUATE MODULE  
576 PRSOFT (COMMON DEFINITIONS & REFERENCES)  
579 000000' .PRINT ;SPMAC: VERSION 1.1  
599 PRSOFT (CODE)



```
508 .TITLE PRSOFT (PROCESS SOFT ERRORS)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; PRSOFT
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE IS CALLED AS A RESULT OF A 'SOFERR' TRAP EXECUTED
517 ; BY AN OPTION MODULE.
518 ;
519 ; FIRST THE FOLLOWING IS DONE:
520 ;
521 ; 1. THE MODULE'S SOFT ERROR COUNT IS INCREMENTED.
522 ; 2. ERROR MESSAGE IS QUEUED UP.
523 ;
524 ; IF THE ENVIRONMENT IS APT, THEN AN APT ERROR HANDLING ROUTINE
525 ; IS CALLED AND ON RETURN, THE CONTROL GOES BACK TO THE CALLER.
526 ;
527 ; IF NOT UNDER APT ENVIRONMENT, THEN
528 ;
529 ; 1. IF BIT15 OF THE SOFTWARE SWITCH REGISTER IS SET, THE
530 ; MODULE IS DROPPED WITH 'MODULE DROPPED' MESSAGE PRINTED,
531 ; EVEN ON THE FIRST ERROR.
532 ; 2. IF THE ERROR COUNT IS EQUAL TO OR GREATER THAN THE
533 ; LIMIT, THEN BIT14 OF THE SOFTWARE SWITCH REGISTER IS
534 ; CHECKED. IF BIT14 IS SET, THE MODULE IS NOT DROPPED.
535 ; 3. CONTROL RETURNS TO THE CALLER.
536 ;
537 ;
538 ; INPUTS:
539 ; DATA TABLE ADDRESS
540 ;
541 ; IMPLICIT INPUTS:
542 ; 1. DATA TABLE OFFSETS
543 ; 2. MODULE HEADER OFFSETS
544 ;
545 ; OUTPUTS:
546 ; NONE
547 ;
548 ; IMPLICIT OUTPUTS:
549 ; 1. DATA TABLE
550 ; 2. MODULE HEADER
551 ;
552 ; PATHOLOGICAL CONNECTIONS:
553 ; NONE
554 ;
555 ; SUBORDINATE ROUTINES CALLED:
556 ; 1. DRPMOD ;DROP A MODULE ROUTINE
557 ; 2. ENQTQ ;ENQUE AN ERROR MSG ROUTINE
558 ; 3. APTSER ;APT SOFT ERROR ROUTINE
559 ; 4. SAVREG
560 ; 5. RESREG
561 ;
562 ; FUNCTIONAL SIDE EFFECTS:
563 ; UNDER APT ENVIRONMENT IT CAN SHUT DOWN THE EXERCISER.
```

PRSOFT (PROCESS SOFT ERRORS)  
PRSOFT.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:30 PAGE 19-1  
COMMON EQUATE MODULE

SEQ 0891

```
564 ;  
565 ; CALLING SEQUENCE:  
566 ; CALL PRSOFT IN <DT>  
567 ; WHERE DT = DATA TABLE ADDRESS  
568 ;  
569 ; VERSION:  
570 ; 0.0  
571 ;  
572 ; EDIT DATE BY REASON  
573 ;--  
574
```

PRSOFT (PROCESS SOFT ERRORS)  
PRSOFT.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:30 PAGE 19-2  
PRSOFT (COMMON DEFINITIONS & REFERENCES)

SEQ 0892

```
576          .SBTTL PRSOFT (COMMON DEFINITIONS & REFERENCES)
577
578          .MCALL STRUCT
579 000000'    STRUCT
(1) 000000'    .PRINT ;SPMAC: VERSION 1.1
580          000001    $LSTIN=1
581          000001    $LSTTAG=1
582
583          ;*****
584          ; REFERENCED BY OTHER MODULES
585          ;
586          .GLOBL PRSOFT          ;MODULE ENTRY POINT
587
588          ;*****
589          ; GLOBAL REFERENCES
590          ;
591          .GLOBL ENQTQ          ;ENQUE A MESSAGE INTO THE TYPE QUEUE
592          .GLOBL APTSER        ;APT SOFT ERROR HANDLER
593          .GLOBL DRPMOD        ;'DROP THE MODULE' ROUTINE
594          .GLOBL SAVREG
595          .GLOBL RESREG
596
597          ;*****
```

```

599          .SBTTL PRSOFT (CODE)
600
601 000000'  ROUTINE PRSOFT <DTA>
(2) 000000'
602
603          ;+
604          ; SAVE REGS, GET DTABLE ADDRESS AND MODULE
605          ; HEADER ADDRESS.
606          ; -
607
608 000000'  CALL SAVREG
(3) 000000' 004767 000000G
609 000004'  LET R0 := DTA(R5)
(4) 000004' 016500 000000
610 000010'  LET R1 := DT.PC(R0)
(4) 000010' 016001 000002
611 000014'  LET R2 := (R1)
(4) 000014' 011102
612
613          ;+
614          ; POINT R3 TO TRAP ADDRESS
615          ; -
616
617 000016'  LET R3 := R1 - #2
(4) 000016' 010103
(6) 000020' 162703 000002
618
619          ;+
620          ; POINT R1 TO RETURN ADDRESS
621          ; -
622
623 000024'  LET R1 := R1 + #4
(6) 000024' 062701 000004
624
625          ;+
626          ; INCREMENT MODULE'S SOFT ERROR COUNT AND ENQUE THE ERROR
627          ; MESSAGE.
628          ; -
629 000030'  LET SOFCNT(R2) := SOFCNT(R2) + #1
(6) 000030' 005262 000042
630 000034'  LET SOFPAS(R2) := SOFPAS(R2) + #1
(6) 000034' 005262 000046
631 000040'  CALL ENQTQ IN <R0,#MSGSFTR3,R2,R1>
(3) 000040' 010546
(8) 000042' 010145
(7) 000044' 010245
(6) 000046' 010345
(5) 000050' 012745 000006
(4) 000054' 010045
(3) 000056' 004767 000000G
(3) 000062' 012605
632
633          ;+
634          ; IF APT ENVIRONMENT, CALL APT ERROR HANDLING ROUTINE
635          ; -
636

```

PRSOFT:

```

JSR  PC,SAVREG
MOV  DTA(R5),R0
MOV  DT.PC(R0),R1
MOV  (R1),R2
MOV  R1,R3
SUB  #2,R3
ADD  #4,R1
INC  SOFCNT(R2)
INC  SOFPAS(R2)
MOV  R5,-(SP)
MOV  R1,-(R5)
MOV  R2,-(R5)
MOV  R3,-(R5)
MOV  #MSGSFTR5,-(R5)
MOV  R0,-(R5)
JSR  PC,ENQTQ
MOV  (SP)+,R5

```

```
637 000064' IF #APTPRE SETIN DT.CF0(R0) THEN
(6) 000064' 032760 000200 000014 BIT #APTPRE,DT.CF0(R
(9) 000072' 001406 BEQ 50002$
638 000074' 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
639 000106' ELSE
(4) 000106' 000431 BR 50003$
(3) 000110' 50002$:
640
641 ;+
642 ; THE ENVIRONMENT IS NOT APT.
643 ; CHECK SWITCH REGISTER IF THE MODULE IS TO BE DROPPED
644 ; DUE TO THIS ERROR.
645 ;-
646
647 000110' IF #BIT15 SETIN DT.SWR(R0) THEN
(6) 000110' 032760 100000 000056 BIT #BIT15,DT.SWR(R0
(9) 000116' 001407 BEQ 50004$
648 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
649 000134' ELSE
(4) 000134' 000416 BR 50005$
(3) 000136' 50004$:
650
651 ;+
652 ; IF THE COUNT IS EQUAL TO OR GREATER THAN THE LIMIT,
653 ; CHECK SWITCH REGISTER IF MODULE IS TO BE DROPPED.
654 ;-
655
656 000136' IF SOFCNT(R2) HI DT.SMX(R0) AND #BIT14 NOTSETIN DT.SWR(R0) THEN
(6) 000136' 026260 000042 000106 CMP SOFCNT(R2),DT.SM
(9) 000144' 101412 BLOS 50006$
(6) 000146' 032760 040000 000056 BIT #BIT14,DT.SWR(R0
(9) 000154' 001006 BNE 50006$
657 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
658 000172' ENDIF
(4) 000172' 50006$:
659 000172' ENDIF
(4) 000172' 50005$:
660
661 000172' ENDIF
(4) 000172' 50003$:
662
663 ;+
```

PRSOFT (PROCESS SOFT ERRORS)  
PRSOFT.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:30 PAGE 19-5  
PRSOFT (CODE)

SEQ 0895

```
664 ; RESTORE REGS AND RETURN
665 ; -
666
667 000172' CALL RESREG
(3) 000172' 004767 000000G JSR PC,RESREG
668
669 000176' ENDRTN
(3) 000176'
(3) 000176' 50000$:
(2) 000176' 000207 50001$:
670 RTS PC
671 000001 .END
```

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

SYSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LOCTA= 177777	\$SERFL= 000000
TMPID = 000002	WBUFEA= 000136	\$F\$GOD= 000400	\$LSTIN= 000001	\$FLAG= 000001
TQOVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$LSTTA= 000001	\$FROM= 000000
UIPAR0= 177640	WBUFQR= 000140	\$F\$INC= 000210	\$NESTL= 177777	\$LOC = 000154R
UIPAR1= 177642	WBUFSZ= 000142	\$F\$LDD= 000200	\$NSKO = 000300	\$LOCN= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK1 = 000110	\$REG = 177777
UIPAR3= 177646	WDT0 = 000114	\$F\$NO = 000403	\$NSK2 = 000110	\$RETN= 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	\$TSKO = 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	\$ISKO = 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:30  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0898

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
586 RANDOM (CODE)

```
508 .TITLE RANDOM (PSEUDO-RANDOM NUMBER GENERATOR)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     RANDOM
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE GENERATES AND RETURNS A 16-BIT RANDOM NUMBER
517 ;     TO THE CALLER.
518 ;
519 ;     THE ALGORITHM USED CONSISTS OF ROTATING TWO BASE WORDS
520 ;     ONE POSITION TO THE RIGHT AND THEN PERFORMING AN
521 ;     EXCLUSIVE-OR OPERATION ON CERTAIN BITS IN WORD-1. TO
522 ;     INSURE THOROUGH RANDOMNESS OF THE NUMBER, THIS OPERATION
523 ;     IS REPEATED 16 TIMES BEFORE RETURNING THE RANDOM NUMBER
524 ;     FROM THE SECOND WORD.
525 ;
526 ; INPUTS:
527 ;     NONE
528 ;
529 ; IMPLICIT INPUTS:
530 ;     NONE
531 ;
532 ; OUTPUTS:
533 ;     A RANDOM NUMBER
534 ;
535 ; IMPLICIT OUTPUTS:
536 ;     NONE
537 ;
538 ; PATHOLOGICAL CONNECTIONS:
539 ;     NONE
540 ;
541 ; SUBORDINATE ROUTINES CALLED:
542 ;     SAVREG
543 ;     RESREG
544 ;
545 ; FUNCTIONAL SIDE EFFECTS:
546 ;     NONE
547 ;
548 ; CALLING SEQUENCE:
549 ;     CALL RANDOM OUT <XXX>
550 ;
551 ;     WHERE XXX = RANDOM NUMBER RETURNED.
552 ;
553 ; VERSION:
554 ;     0.0
555 ;
556 ;     EDIT      DATE      BY      REASON
557 ;     -----
558 ;
559 ;
```

```
561 .SBTTL COMMON DEFINITIONS AND REFERENCES
562
563 .MCALL STRUCT
564 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
565 000001 $LSTIN=1
566 000001 $LSTTAG=1
567
568 ;*****
569 ; REFERENCED BY OTHER MODULES
570 ;
571 .GLOBL RANDOM ;MODULE'S ENTRY POINT
572
573 ;*****
574 ; GLOBAL REFERENCES
575 ;
576 .GLOBL SAVREG
577 .GLOBL RESREG
578 ;
579 ;*****
580 ; LOCAL STORAGE
581 ;
582 000000' 022333 WORK: 22333 ;WORD-1
583 000002' 123456 RAND: 123456 ;WORD-2
584 ;THE CHOICE OF VALUES IS ARBITRARY
```

```

586          .SBTTL RANDOM (CODE)
587
588          000004'          ROUTINE RANDOM <RSLT>
(2)         000004'
589
590          ;+
591          ; SAVE REGISTERS
592          ; GET THE TWO BASE WORDS INTO R0 AND R1.
593          ; -
594
595          000004'          CALL SAVREG
(3)         000004' 004767 000000G          JSR      PC, SAVREG
596          000010'          LET R0 := WORK          MOV      WORK, R0
(4)         000010' 016700 177764
597          000014'          LET R1 := RAND          MOV      RAND, R1
(4)         000014' 016701 177762
598
599          ;+
600          ; SET UP LOOP FOR 16 ITERATIONS, ROTATE THE TWO
601          ; WORDS ONE POSITION TO THE RIGHT, AND THEN DO
602          ; THE EXCLUSIVE-OR OPERATION ON BIT12 AND BIT10
603          ; OF WORD-1, BASED ON "CARRY" CONDITION.
604          ; -
605
606          000020'          DECR R2 FROM #15. TO #0 BY #1
(4)         000020' 012702 000017          MOV      #15., R2
(5)         000024' 000401          BR       50002$
(4)         000026'          50003$:
(7)         000026' 005302          DEC      R2
(5)         000030'          50002$:
(5)         000030' 005702          TST      R2
(7)         000032' 002423          BLT     50004$
607          000034'          LET R0 := R0 ROTATE -#1          ROR     R0
(7)         000034' 006000          LET R1 := R1 ROTATE -#1          ROR     R1
608          000036' 006001          IF.NO.CARRY THEN          BCS     50005$
(7)         000036'
609          000040'          LET R0 := R0 CLR.BY #BIT12          BIC     #BIT12, R0
(6)         000040' 103410          IF #BIT10 SETIN R0 THEN          BIT     #BIT10, R0
610          000042'          LET R0 := R0 SET.BY #BIT12          BEQ     50006$
(6)         000042' 042700 010000          LET R0 := R0 SET.BY #BIT12          BIS     #BIT12, R0
611          000046'          IF #BIT10 SETIN R0 THEN          BIT     #BIT10, R0
(6)         000046' 032700 002000          BEQ     50006$
(9)         000052' 001402          LET R0 := R0 CLR.BY #BIT12
612          000054'          ENDIF          50006$:
(6)         000054' 052700 010000          ELSE
613          000060'          LET R0 := R0 SET.BY #BIT12          BR       50007$
(4)         000060'          IF #BIT10 SETIN R0 THEN          50005$:
614          000060'          LET R0 := R0 CLR.BY #BIT12          BIS     #BIT12, R0
(4)         000060' 000407          BIT     #BIT10, R0
(3)         000062'          LET R0 := R0 SET.BY #BIT12          BEQ     50010$
615          000062' 052700 010000          IF #BIT10 SETIN R0 THEN
616          000066'          LET R0 := R0 CLR.BY #BIT12
(6)         000066' 032700 002000
(9)         000072' 001402
617          000074'          LET RC := R0 CLR.BY #BIT12
    
```

618	000100'		ENDIF				
619	000100'		ENDIF		50010\$:		
620	000100'		ENDDEC		50007\$:		
621							
622							
623							
624							
625							
626							
627							
628							
629	000102'		LET WORK := R0				
630	000106'	010067 177672	LET RAND := R1			MOV	R0,WORK
631	000112'		LET RSLT(R5) := R1			MOV	R1,RAND
632						MOV	R1,RSLT(R5)
633							
634							
635							
636							
637	000116'		CALL RESREG				
638							
639	000122'		ENDRTN				
640					50000\$:		
641	000001		.END		50001\$:	RTS	PC

BIC #BIT12,R0

50010\$:

50007\$:

50004\$:

BR 50003\$

MOV R0,WORK

MOV R1,RAND

MOV R1,RSLT(R5)

JSR PC,RESREG

50000\$:

50001\$:

RTS PC

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

TQOVF = 000002	WBUFPA= 000134	\$F\$GDD= 000400	\$LSTTA= 000001	\$\$ERFL= 000000
UIPAR0= 177640	WBUFRQ= 000140	\$F\$IF = 000110	\$NESTL= 177777	\$\$FLAG= 000001
UIPAR1= 177642	WBUFSZ= 000142	\$F\$INC= 000210	\$NSKO = 000300	\$\$FROM= 000000
UIPAR2= 177644	WDFR = 000116	\$F\$LDD= 000200	\$NSK1 = 000220	\$\$LQC = 000072R
UIPAR3= 177646	WDTO = 000114	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$LOCN= 000000
UIPAR4= 177650	WORK = 000000R	\$F\$NO = 000403	\$NSK3 = 000110	\$\$REG = 177777
UIPAR5= 177652	WTINRE= 000352	\$F\$OR = 000320	\$\$AVLE= 177777	\$\$RETU= 000000
UIPAR6= 177654	WTWHMI= 000222	\$F\$RTI= 000350	\$\$SKO = 050003	\$\$RTN1= 050000
UIPAR7= 177656	XFLAG = 000005	\$F\$RTN= 000300	\$TAGLE= 177777	\$\$RTN2= 050001
UIPDR0= 177600	XOFF = 000023	\$F\$SEL= 000140	\$TAGNU= 050011	\$\$SRC = 000000
UIPDR1= 177602	XQN = 000021	\$F\$THE= 000330	\$TEMP = 000300	\$\$TGSV= 000000
UIPDR2= 177604	\$BGNLE= 177777	\$F\$TRU= 000404	\$TSKO = 050004	\$\$TGS1= 000000
UIPDR3= 177606	\$ERFLG= 000400	\$F\$UNT= 000130	\$TSK1 = 050003	\$\$TGS2= 000000
UIPDR4= 177610	\$F\$AND= 000310	\$F\$WHI= 000120	\$TSK2 = 050007	\$\$TO = 000000
UIPDR5= 177612	\$F\$BAD= 000401	\$F\$YES= 000402	\$TSK3 = 050010	\$\$\$TAG= 050000
UIPDR6= 177614	\$F\$BLA= 000170	\$IFLEV= 177777	\$\$ARGC= 000002	. = 000124R
UIPDR7= 177616	\$F\$CAS= 000150	\$ISKO = 000001	\$\$BYTE= 000403	
WASADR= 000104	\$F\$DEC= 000220	\$ISK1 = 000001	\$\$CASE= 000000	
WBSTAT= 000040	\$F\$DDO = 000340	\$LOCTA= 177777	\$\$DST = 000000	
WBUFEA= 000136	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$ELOC= 000402	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:31

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0905

3 COMMON EQUATE MODULE  
559 COMMON DEFINITIONS AND REFERENCES  
562 000000' .PRINT ;SPMAC: VERSION 1.1  
609 RELCTL ROUTINE



```
508 .TITLE RELCTL - RELOCATION CONTROL
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; RELCTL
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; CONTROLS THE SEQUENCE OF EVENTS THAT MUST OCCUR BEFORE,
517 ; DURING, AND AFTER THE RELOCATION PROCESS.
518 ;
519 ; INPUTS:
520 ; DATA TABLE ADDRESS
521 ;
522 ; IMPLICIT INPUTS:
523 ; CURRENT BASE ADDRESS, CONFIGURATION WORD 0,
524 ; STATUS INDICATOR WORD 0, STATUS INDICATOR WORD 1
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; MESSAGE CODE IN MSGDEQ (MD.COD)
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ; CLKOFF ;TURN OFF SYSTEM CLOCK
537 ; CLKON ;TURN ON SYSTEM CLOCK
538 ; PRRLOC ;PROCESS RELOCATION
539 ; WBFLIM ;DETERMINE WRITE BUFFER LIMITS
540 ; UNIMAP ;LOAD UNIBUS MAP
541 ; WSTBUS ;WRITE WORST-CASE UNIBUS PATTERNS
542 ; MSGDEQ ;MESSAGE DEQUEUER
543 ; DX.INI ;INITIALIZE QUEUES AND KEYBOARD
544 ;
545 ; FUNCTIONAL SIDE EFFECTS:
546 ; NONE
547 ;
548 ; CALLING SEQUENCE:
549 ; CALL RELCTL IN <A>
550 ; A=DATA TABLE ADDRESS
551 ;
552 ; VERSION:
553 ; 0.0
554 ;
555 ; EDIT BY DATE REASON
556 ;
557 ;--
```

```
559          .SBTTL COMMON DEFINITIONS AND REFERENCES
560
561          .MCALL STRUCT
562 000000'    STRUCT
563 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
564          000001 $LSTIN=1
565          000001 $LSTTAG=1
566          ;
567          ;*****
568          ;
569          ; REFERENCED BY OTHER MODULES
570          ;
571          .GLOBL RELCTL          ;MODULE ENTRY POINT
572          ;
573          ;*****
574          ;
575          ; GLOBAL REFERENCES
576          ;
577          .GLOBL CLKOFF
578          .GLOBL CMDPRC          ;PROCESS A KEYBOARD COMMAND
579          .GLOBL DX.KFL          ;KEYBOARD FLAG
580          .GLOBL CLKCN
581          .GLOBL PRRLOC
582          .GLOBL WBFLIM
583          .GLOBL UNIMAP
584          .GLOBL WSTBUS
585          .GLOBL DX.INI
586          .GLOBL MSGDEQ          ;MESSAGE DEQUEUER
587          .GLOBL MD.COD          ;MESSAGE CODE IN MESDEQ
588          ;
589          ;*****
590          ;
591          ; GLOBAL EQUATES
592          ;
593          .GLOBL DT.ST0          ;STATUS INDICATOR WORD 0
594          .GLOBL DT.ST1          ;STATUS INDICATOR WORD 1
595          .GLOBL DT.CF0          ;CONFIGURATION WORD 0
596          .GLOBL DT.ADDR          ;CURRENT BASE ADDRESS
597          .GLOBL RELTIME          ;RELOCATION TIME INDICATOR
598          .GLOBL MSGNUL          ;NULL MESSAGE TYPE CODE
599          .GLOBL ADDR22          ;22-BIT ADDRESSING
600          .GLOBL RCSR          ;KB INPUT CSR
601          .GLOBL IE          ;INTERRUPT ENABLE
602          .GLOBL CLKPRES          ;SYSTEM CLOCK PRESENT
603          .GLOBL MEMPAS          ;LAST SYSTEM EOP WAS IN LOWEST MEMORY
604          .GLOBL AUTO          ;AUTOMATIC MODE
605          ;
606          ;*****
607          ;
```

```
609          .SBTTL RELCTL ROUTINE
610
611
612 000000'          ROUTINE RELCTL <TABL>
(2) 000000'
613
614
615          ;+
616          ; SAVE REGISTERS
617          ; -
618
619 000000'          PUSH R0
(2) 000000' 010046
620
621
622          ;+
623          ; SET R0 TO THE START OF THE DATA TABLE
624          ; -
625
626 000002'          LET R0 := TABL(R5)
(4) 000002' 016500 000000
627
628
629          ;+
630          ; SHUT OFF THE SYSTEM CLOCK IF IT EXISTS.
631          ; -
632
633 000006'          IF #CLKPRES SETIN DT.CFO(R0) THEN
(6) 000006' 032760 000001 000014
(9) 000014' 001405
634          CALL CLKOFF IN <R0>
(3) 000016' 010546
(4) 000020' 010045
(3) 000022' 004767 000000G
(3) 000026' 012605
635          ENDIF
(4) 000030'
636
637          ;+
638          ; SERVICE ANY FINAL KEYBOARD COMMANDS
639          ; -
640          IF DX.KFL EQ #1 THEN
(6) 000030' 026727 000000G 000001
(9) 000036' 001007
641          CALL CMDPRC IN <R0>
(3) 000040' 010546
(4) 000042' 010045
(3) 000044' 004767 000000G
(3) 000050' 012605
642          LET DX.KFL := #0
(4) 000052' 005067 000000G
643          ENDIF
(4) 000056'
644
645          ;+
646          ; WAIT FOR THE TYPE-QUEUE TO EMPTY
```

RELCTL:

MOV R0,-(SP)

MOV TABL(R5),R0

BIT #CLKPRES,DT.CFO(  
BEQ 50002\$

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,CLKOFF  
MOV (SP)+,R5

50002\$:

CMP DX.KFL,#1  
BNE 50003\$

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,CMDPRC  
MOV (SP)+,R5

CLR DX.KFL

50003\$:

```

647      ;+
648      ;+
649      000056' REPEAT
(3) 000056'
650      000056' CALL MSGDEQ IN <R0>
(3) 000056' 010546
(4) 000060' 010045
(3) 000062' 004767 000000G
(3) 000066' 012605
651      000070' UNTIL MD.COD EQ #MSGNUL
(3) 000070' 026727 000000G 177775
(6) 000076' 001367
652
653
654      ;+
655      ; RELOCATE
656      ;+
657
658      000100' CALL PRRLOC IN <R0,#0>
(3) 000100' 010546
(5) 000102' 012745 000000
(4) 000106' 010045
(3) 000110' 004767 000000G
(3) 000114' 012605
659
660
661      ;+
662      ; TURN THE SYSTEM CLOCK BACK ON IF IT EXISTS.
663      ;+
664
665      000116' IF #CLKPRES SETIN DT.CF0(R0) THEN
(6) 000116' 032760 000001 000014
(9) 000124' 001405
666      000126' CALL CLKON IN <R0>
(3) 000126' 010546
(4) 000130' 010045
(3) 000132' 004767 000000G
(3) 000136' 012605
667      000140' ENDIF
(4) 000140'
668
669
670
671      ;+
672      ; SET THE WRITE BUFFER LIMITS AND INITIALIZE THE WRITE BUFFER POINTER
673      ;+
674      000140' CALL WBFLIM IN <R0>
(3) 000140' 010546
(4) 000142' 010045
(3) 000144' 004767 000000G
(3) 000150' 012605
675
676
677      ;+
678      ; IF THE UNIBUS MAP EXISTS, LOAD THE MAP REGISTERS.
679      ;+

```

50004\$:

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,MSGDEQ  
MOV (SP)+,R5  
CMP MD.COD,#MSGNUL  
BNE 50004\$

MOV R5,-(SP)  
MOV #0,-(R5)  
MOV R0,-(R5)  
JSR PC,PRRLOC  
MOV (SP)+,R5

BIT #CLKPRES,DT.CF0(  
BEQ 50005\$

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,CLKON  
MOV (SP)+,R5

50005\$:

MOV R5,-(SP)  
MOV R0,-(R5)  
JSR PC,WBFLIM  
MOV (SP)+,R5

```

680
681 000152' IF #ADDR22 SETIN DT.CF0(R0) THEN
(6) 000152' 032760 001000 000014
(9) 000160' 001405
682 000162' CALL UNIMAP IN <R0>
(3) 000162' 010546
(4) 000164' 010045
(3) 000166' 004767 000000G
(3) 000172' 012605
683 000174' ENDF
(4) 000174'
684
685
686
687 ;+
688 ; IF THIS RELOCATION WAS TO THE LOWEST PART OF MEMORY, WRITE THE
689 ; WORST-CASE UNIBUS PATTERNS INTO ALL OF FREE MEMORY,
690 ; -
691 000174' IF DT.ADDR(R0) EQ #200 THEN
(6) 000174' 026027 000042 000200
(9) 000202' 001014
692 000204' CALL WSTBUS IN <R0>
(3) 000204' 010546
(4) 000206' 010045
(3) 000210' 004767 000000G
(3) 000214' 012605
693 000216' IF #AUTO SETIN DT.CF0(R0) THEN
(6) 000216' 032760 000010 000014
(9) 000224' 001403
694 000226' LET DT.STi(R0) := DT.ST1(R0) SET.BY #MEMPAS
(6) 000226' 052760 040000 000012
695 000234' ENDF
(4) 000234'
696 000234' ENDF
(4) 000234'
697
698
699
700 ;+
701 ; INITIALIZE THE QUEUES AND KEYBOARD.
702 ; -
703 000234' CALL DX.INI
(3) 000234' 004767 000000G
704
705
706 ;+
707 ; CLEAR THE RELOCATION TIME INDICATOR IN STATUS INDICATOR WORD 0
708 ; -
709
710
711 000240' LET DT.ST0(R0) := DT.ST0(R0) CLR.BY #RELTIME
(6) 000240' 042760 010000 000010
712
713
714 ;+
715 ; RESTORE REGISTERS
  
```

BIT #ADDR22,DT.CF0(R  
 BEQ 50006\$

MOV R5,-(SP)  
 MOV R0,-(R5)  
 JSR PC,UNIMAP  
 MOV (SP)+,R5

50006\$:

CMP DT.ADDR(R0),#200  
 BNE 50007\$

MOV R5,-(SP)  
 MOV R0,-(R5)  
 JSR PC,WSTBUS  
 MOV (SP)+,R5

BIT #AUTO,DT.CF0(R0)  
 BEQ 50010\$

BIS #MEMPAS,DT.ST1(R

50010\$:

50007\$:

JSR PC,DX.INI

BIC #RELTIME,DT.ST0(

RELCTL - RELOCATION CONTROL  
RELCTL.MAC 28-JUL-78 09:26

MACY11 30A(1052) 20-SEP-78 18:31 PAGE 19-5  
RELCTL ROUTINE

SEQ 0911

```
716          ;-
717
718 000246'    POP R0
(2) 000246' 012600          MOV      (SP)+,R0
719
720
721 000250'    ENDRTN
(3) 000250'
(3) 000250'
(2) 000250' 000207
722
723          000001          .END
```

50000\$:  
50001\$: RTS PC

ACSR = 000102	CR = 000015	ECCSTA= 000010	MD.COD= ***** G	PS = 177776
ACTBIT= 004000	CSRA = 000100	ENBEEP= 010000	MED = 076600	PSW = 177776
ADDR22= 001000 G	CSRC = 000102	ENBNUL= 000001	MEMPAS= 040000 G	RANNUM= 000054
ADR = 000006	CTRLC = 000003	ENDLST= 000000	MODEXH= 004000	RBUFEA= 000130
APTFER= 000004	CTRLO = 000017	EOPBIT= 000001	MODHOL= 002000	RBUFPA= 000126
APTPRE= 000200	CTRLU = 000025	ERRTYP= 000106	MODSEL= 001000	RBUFSZ= 000132
ASB = 000106	DCEVNT= 000011	EVNTBE= 000200	MSGCKD= 000010	RBUFVA= 000124
ASSEMB= 000010	DEFRTN= 000400	EVNTHD= 000200	MSGCKS= 000011	RCSR = ***** G
ASTAT = 000104	DIAGMC= 000000	EVNTKT= 000203	MSGDEQ= ***** G	RDSERV= 000101
AUTO = 000010 G	DROPMO= 100000	EVNTPE= 000202	MSGDER= 000005	RDWHMI= 000022
AUTOST= 020000	DSEVNT= 000014	EVNTRE= 000201	MSGDRP= 000017	RELCTL 000000RG
AWAS = 000110	DT.ADD= 000042 G	FATERR= 100000	MSGECH= 177777	RELERR= 000020
BIT0 = 000001	DT.AP = 000100	HRDCNT= 000044	MSGEOP= 000013	RELMOD= 020000
BIT00 = 000001	DT.APK= 000076	HRDPAS= 000050	MSGHDR= 000004	RELTIM= 010000 G
BIT01 = 000002	DT.BLS= 000034	ICONT = 000036	MSGHNG= 000022	RES1 = 000056
BIT02 = 000004	DT.CFO= 000014 G	ICOUNT= 000040	MSGHRD= 000007	RES2 = 000060
BIT03 = 000010	DT.CF1= 000016	IDNUM = 000122	MSGMAP= 000021	RICHAR= 000160
BIT04 = 000020	DT.ERR= 000020	IE = 000100 G	MSGNUL= 177775 G	RPTDAT= 002000
BIT05 = 000040	DT.ESI= 000044	INDPAR= 000040	MSGPOP= 000002	RSTRT = 000112
BIT06 = 000100	DT.EVN= 000000	INHDRP= 040000	MSGPRM= 177776	RUBOUT= 000177
BIT07 = 000200	DT.EXS= 000060	INHEPR= 020000	MSGRES= 000001	RUNMOD= 100000
BIT08 = 000400	DT.FCH= 000037	INHREL= 001000	MSGSFT= 000006	RSVALU= 001740
BIT09 = 001000	DT.FCN= 000036	INHRR= 000400	MSGSKE= 000003	SAM = 075464
BIT1 = 000002	DT.HMX= 000104	INIT = 000030	MSGSMB= 000015	SBADR = 000102
BIT10 = 002000	DT.KBE= 000024	INTR = 000120	MSGSMH= 000014	SBKMOD= 000000
BIT11 = 004000	DT.KBP= 000026	IOMOD = 100000	MSGSMS= 000016	SBKSEL= 010000
BIT12 = 010000	DT.KBR= 000022	IOMODP= 102000	MSGSTD= 000000	SC.ADR= 000006
BIT13 = 020000	DT.KBU= 000030	IOMODR= 112000	MSGSYS= 000012	SC.ALC= 000014
BIT14 = 040000	DT.MLS= 000032	IOMODX= 110000	MSGVEC= 000020	SC.APC= 000016
BIT15 = 100000	DT.MTI= 000110	JACK = 035060	NBKMOD= 001000	SC.CKL= 000002
BIT2 = 000004	DT.OFF= 000070	KIPAR0= 172340	NCPUOP= 000020	SC.CKP= 000004
BIT3 = 000010	DT.PAS= 000074	KIPAR1= 172342	NOAPTY= 000002	SC.CLO= 000000
BIT4 = 000020	DT.PC = 000002	KIPAR2= 172344	NULL = 000000	SC.HLD= 000010
BIT5 = 000040	DT.PFL= 000062	KIPAR3= 172346	OWEN = 024020	SC.SCA= 000012
BIT6 = 000100	DT.PSW= 000004	KIPAR4= 172350	PAERR = 000010	SENDLS= 177777
BIT7 = 000200	DT.PTA= 000064	KIPAR5= 172352	PARPRE= 002000	SOFCNT= 000042
BIT8 = 000400	DT.RCS= 000102	KIPAR6= 172354	PARSTA= 000100	SOFPAS= 000046
BIT9 = 001000	DT.REL= 000040	KIPAR7= 172356	PASCNT= 000034	SPACE = 000040
BKDEF = 000002	DT.SCT= 000066	KIPDR0= 172300	PDPLSI= 020000	SPOINT= 000032
BKMOD = 000020	DT.SMX= 000106	KIPDR1= 172302	PDP60 = 004000	SPVALU= 002200
BKMODE= 040000	DT.SP = 000006	KIPDR2= 172304	PDP70 = 010000	SRO = 177572
BKSLSH= 000134	DT.SSI= 000046	KIPDR3= 172306	PRI0 = 000000	SR1 = 177574
CAPRES= 000004	DT.STO= 000010 G	KIPDR4= 172310	PRI1 = 000040	SR2 = 177576
CASTAT= 000004	DT.ST1= 000012 G	KIPDR5= 172312	PRI4 = 000200	SR3 = 172516
CDERCT= 000146	DT.SWR= 000056	KIPDR6= 172314	PRI5 = 000240	STAT = 000026
CDWDCT= 000144	DT.SYP= 000072	KIPDR7= 172316	PRI6 = 000300	STATBI= 064757
CKTIM = 100000	DT.WBU= 000050	KTERRO= 000040	PRI7 = 000340	STAT1 = 000027
CLKOFF= ***** G	DT.WHL= 000054	KTPRES= 000400	PRRLOC= ***** G	SUSPND= 000001
CLKON = ***** G	DT.WLL= 000052	KTSTAT= 000020	PRO = 000000	SVR0 = 000062
CLKPRE= 000001 G	DVID1 = 000014	KTXTND= 040000	PR4 = 000200	SVR1 = 000064
CMDPRC= ***** G	DX.INI= ***** G	LF = 000012	PR5 = 000240	SVR2 = 000066
CONFIG= 000056	DX.KFL= ***** G	LPSTAT= 000001	PR6 = 000300	SVR3 = 000070
CQOVF = 000001	ECCMEM= 000100	MAPSTA= 000200	PR7 = 000340	SVR4 = 000072

SVR5 = 000074	UIPDR5= 177612	\$ERFLG= 000400	\$FSUNT= 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	\$ISKO = 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\$GDO= 000400	\$LSTTA= 000001	\$\$LOCN= 000000
UIPAR2= 177644	WBUFRQ= 000140	\$F\$IF = 000110	\$NESTL= 177777	\$\$REG = 177777
UIPAR3= 177646	WBUFSZ= 000142	\$F\$INC= 000210	\$NSKO = 000300	\$\$RETU= 000000
UIPAR4= 177650	WDFR = 000116	\$F\$LCO= 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\$DR = 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	\$TSKO = 050007	\$\$TD = 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:31  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0914

3 COMMON EQUATE MODULE  
529 COMMON DEFINITIONS AND REFERENCES FOR RONOF  
532 000000' .PRINT ;SPMAC: VERSION 1.1  
563 KROTON & KROTOF - PROCESS THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS  
617 KROTON - KROTOF ROUTINE

```
508 .TITLE RONOF PROCESS THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS
509 .IDENT /V0.0/
510
511
512 ;++
513 ; MODULE PACKAGE NAME:
514 ;     RONOF
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ;     THIS ROUTINE CONSISTS OF TWO MODULES:
518 ;     1. KROTON - PROCESS THE KEYBOARD COMMAND 'ROTON'
519 ;     2. KROTOF - PROCESS THE KEYBOARD COMMAND 'ROTOFF'
520 ;
521 ; VERSION:
522 ;     0.0
523 ;
524 ; EDIT          BY          DATE          REASON
525 ;--
526
527
```

```

529 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR RONOF
530
531 .MCALL STRUCT
532 000000' STRUCT
533 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
534 000001 $LSTIN=1
535 000001 $LSTTAG = 1
536
537 ;*****
538 ; REFERENCED BY OTHER MODULES:
539 ;
540 .GLOBL KROTON ;'ROTON' MODULE ENTRY POINT
541 .GLOBL KROTOF ;'ROTOFF' MODULE ENTRY POINT
542 ;
543 ;*****
544 ;
545 ; GLOBAL REFERENCES:
546 ;
547 .GLOBL CM.ARG ;ILLEGAL ARGUMENT MESSAGE
548 .GLOBL ARGCHK ;CHECK AN ARGUMENT
549 .GLOBL SAVREG
550 .GLOBL RESREG
551 ;
552 ;*****
553 ;
554 ; LOCAL STORAGE:
555 ;
556 000000' 051127 052111 020105 RO.MGN: .ASCIZ /WRITE BUFFER ROTATION ON%/
557 000006' 052502 043106 051105
558 000014' 051040 052117 052101
559 000022' 047511 020116 047117
560 000030' 000045
561 000032' 051127 052111 020105 RO.MGF: .ASCIZ /WRITE BUFFER ROTATION OFF%/
562 000040' 052502 043106 051105
563 000046' 051040 052117 052101
564 000054' 047511 020116 043117
565 000062' 022506 000
566 000065' 000 RO.FLG: .BYTE 0 ;1 IF ROTOF, 0 IF ROTON
567 ;
568 .EVEN

```

563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615

```
.SBTTL KROTON & KROTOF - PROCESS THE 'ROTON' AND 'ROTOFF' KEYBOARD COMMANDS
.IDENT /VO.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
; --
```

```

617          .SBTTL KROTON - KROTOF ROUTINE
618
619
620
621 000066'      INLINE <KROTOF:>
(2) 000066'
622
623          ;+
624          ; INDICATE IT'S 'ROTOFF'
625          ; -
626
627 000066'      LET RO.FLG :B= RO.FLG + #1
(6) 000066' 105267 177773
628 000072'      INLINE <BR      GUNCH>
(2) 000072' 000402
629
630
631 000074'      ROUTINE KROTON <DTADR,BUFPTR>
(2) 000074'
632
633
634          ;+
635          ; INDICATE 'ROTON' COMMAND
636          ; -
637
638 000074'      LET RO.FLG :B= #0
(4) 000074' 105067 177765
639
640 000100'      INLINE <GUNCH:>
(2) 000100'
641
642          ;+
643          ; SAVE REGISTERS AND SAVE DTABLE ADDRESS AND DECODE BUFFER POINTER
644          ; -
645 000100'      CALL SAVREG
(3) 000100' 004767 000000G
646 000104'      LET R0 := DTADR(R5)
(4) 000104' 016500 000000
647 000110'      LET R1 := BUFPTR(R5)
(4) 000110' 016501 000002
648
649
650          ;+
651          ; MAKE SURE NO ADDITIONAL ARGUMENTS ARE IN DECODE BUFFER
652          ; -
653
654 000114'      CALL ARGCHK IN <R1> OUT <R1>
(4) 000114' 162705 000002
(3) 000120' 010546
(4) 000122' 010145
(3) 000124' 004767 000000G
(3) 000130' 012605
(4) 000132' 012501
655
656 000134'      IF.NO.ERROR THEN
(6) 000134' 103404

```

KROTOF:

INCB RO.FLG

BR GUNCH

KROTON:

CLRB RO.FLG

GUNCH:

JSR PC, SAVREG

MOV DTADR(R5), R0

MOV BUFPTR(R5), R1

SUB #1\*2, R5

MOV R5, -(SP)

MOV R1, -(R5)

JSR PC, ARGCHK

MOV (SP)+, R5

MOV (R5)+, R1

BCS 50002\$

```

657
658
659      ;+
660      ; THERE IS NO <CR> YET SO IT'S A BAD COMMAND...STUFF ERROR
661      ; MESSAGE AND BEAT IT....
662      ; -
663      000136'          LET DT.KBRSP(R0) := #CM.ARG
664      (4) 000136' 012760 000000G 000022          MOV      #CM.ARG,DT.KBRSP
665      ELSE
666      000144'
667      (4) 000144' 000420          BR      50003$
668      (3) 000146'          50002$:
669      ;+
670      ; IT'S A <CR> SO GO PROCESS.... SET THE WBSTAT BIT
671      ; IN DT.STO IF 'ROTON' AND CLEAR IT IF 'ROTOF'
672      ; -
673      000146'          IFB RO.FLG EQ #0 THEN
674      (6) 000146' 105767 177713          TSTB   RO.FLG
675      (9) 000152' 001007          BNE    50004$
676      000154'          LET DT.STO(R0) := DT.STO(R0) SET.BY #WBSTAT
677      (6) 000154' 052760 000040 000010          BIS    #WBSTAT,DT.STO(R
678      000162'          LET DT.KBRSP(R0) := #RO.MGN
679      (4) 000162' 012760 000000' 000022          MOV    #RO.MGN,DT.KBRSP
680      ELSE
681      000170'
682      (4) 000170' 000406          BR      50005$
683      (3) 000172'          50004$:
684      000172'          LET DT.STO(R0) := DT.STO(R0) CLR.BY #WBSTAT
685      (6) 000172' 042760 000040 000010          BIC    #WBSTAT,DT.STO(R
686      000200'          LET DT.KBRSP(R0) := #RO.MGF
687      (4) 000200' 012760 000032' 000022          MOV    #RO.MGF,DT.KBRSP
688      000206'          ENDIF
689      (4) 000206'          50005$:
690      000206'          ENDIF
691      (4) 000206'          50003$:
692      ;+
693      ; CLEAN UP AND LEAVE
694      ; -
695      000206'          CALL RESREG
696      (3) 000206' 004767 000000G          JSR    PC,RESREG
697      ENDRTN
698      (3) 000212'          50000$:
699      (3) 000212'          50001$:
700      (2) 000212' 000207          RTS    PC
701      000212'          .END
  
```

ACSR = 000102	CR = 000015	ENBEEP= 010000	LPSTAT= 000001	PSW = 177776
ACTBIT= 004000	CSRA = 000100	ENBNUL= 000001	MAPSTA= 000200	RANNUM= 000054
ADDR22= 001000	CSRC = 000102	ENDLST= 000000	MED = 076600	RBUFEA= 000130
ADR = 000006	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	RBUFPA= 000126
APTFER= 000004	CTRLO = 000017	ERRTYP= 000106	MODEXH= 004000	RBUFSZ= 000132
APTPRE= 000200	CTRLU = 000025	EVNTBE= 000200	MODHOL= 002000	RBUFVA= 000124
ARGCHK= ***** G	DCEVNT= 000011	EVNTHD= 000200	MODSEL= 001000	RDSERV= 000101
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MSGCKD= 000010	RDWHMI= 000022
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MSGCKS= 000011	RELERR= 000020
ASTAT = 000104	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	RELMOD= 020000
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	RELTIM= 010000
AUTQST= 020000	DTADR = 000000	GUNCH 000100R	MSGECH= 177777	RESREG= ***** G
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGEOP= 000013	RES1 = 000056
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGHDR= 000004	RES2 = 000060
BIT00 = 000001	DT.APK= 000076	ICQNT = 000036	MSGHNG= 000022	RICHAR= 031060
BIT01 = 000002	DT.BLS= 000034	ICOUNT= 000040	MSGHRD= 000007	RO.FLG 000065R
BIT02 = 000004	DT.CFO= 000014	IDNUM = 000122	MSGMAP= 000021	RJ.MGF 000032R
BIT03 = 000010	DT.CF1= 000016	IE = 000100	MSGNUL= 177775	RJ.MGN 000000R
BIT04 = 000020	DT.ERR= 000020	INDPAR= 000040	MSGPOP= 000002	RPTDAT= 002000
BIT05 = 000040	DT.ESI= 000044	INHDRP= 040000	MSGPRM= 177776	RSTRT = 000112
BIT06 = 000100	DT.EVN= 000000	INHEPR= 020000	MSGRES= 000001	RUBOUT= 000177
BIT07 = 000200	DT.EXS= 000060	INHREL= 001000	MSGSFT= 000006	RUNMOD= 100000
BIT08 = 000400	DT.FCH= 000037	INHRR= 000400	MSGSKE= 000003	RSVALU= 001740
BIT09 = 001000	DT.FCN= 000036	INIT = 000030	MSGSMB= 000015	SAM = 075464
BIT1 = 000002	DT.HMX= 000104	INTR = 000120	MSGSMH= 000014	SAVREG= ***** G
BIT10 = 002000	DT.KBE= 000024	IOMOD = 100000	MSGSMS= 000016	SBADR = 000102
BIT11 = 004000	DT.KBP= 000026	IOMODP= 102000	MSGSTD= 000000	SBKMOD= 000000
BIT12 = 010000	DT.KBR= 000022	IOMODR= 112000	MSGSYS= 000012	SBKSEL= 010000
BIT13 = 020000	DT.KBU= 000030	IOMODX= 110000	MSGVEC= 000020	SC.ADR= 000006
BIT14 = 040000	DT.MLS= 000032	JACK = 035060	NBKMOD= 001000	SC.ALC= 000014
BIT15 = 100000	DT.MTI= 000110	KIPAR0= 172340	NCPUOP= 000020	SC.APC= 000016
BIT2 = 000004	DT.OFF= 000070	KIPAR1= 172342	NOAPTY= 000002	SC.CKL= 000002
BIT3 = 000010	DT.PAS= 000074	KIPAR2= 172344	NULL = 000000	SC.CKP= 000004
BIT4 = 000020	DT.PC = 000002	KIPAR3= 172346	OWEN = 024020	SC.CLD= 000000
BIT5 = 000040	DT.PFL= 000062	KIPAR4= 172350	PAERR = 000010	SC.HLD= 000010
BIT6 = 000100	DT.PSW= 000004	KIPAR5= 172352	PARPRE= 002000	SC.SCA= 000012
BIT7 = 000200	DT.PTA= 000064	KIPAR6= 172354	PARSTA= 000100	SENDLS= 177777
BIT8 = 000400	DT.RCS= 000102	KIPAR7= 172356	PASCNT= 000034	SOFCNT= 000042
BIT9 = 001000	DT.REL= 000040	KIPDR0= 172300	PDPLSI= 020000	SOFPAS= 000046
BKDEF = 000002	DT.SCT= 000066	KIPDR1= 172302	PDP60 = 004000	SPACE = 000040
BKMOD = 000020	DT.SMX= 000106	KIPDR2= 172304	PDP70 = 010000	SPOINT= 000032
BKMODE= 040000	DT.SP = 000006	KIPDR3= 172306	PRI0 = 000000	SPVALU= 002200
BKSLSH= 000134	DT.SSI= 000046	KIPDR4= 172310	PRI1 = 000040	SRO = 177572
BUFPTR= 000002	DT.STO= 000010	KIPDR5= 172312	PRI4 = 000200	SR1 = 177574
CAPRES= 000004	DT.ST1= 000012	KIPDR6= 172314	PRI5 = 000240	SR2 = 177576
CASTAT= 000004	DT.SWR= 000056	KIPDR7= 172316	PRI6 = 000300	SR3 = 172516
CDERCT= 000146	DT.SYP= 000072	KROTOF 000066RG	PRI7 = 000340	STAT = 000026
CDWDCT= 000144	DT.WBU= 000050	KROTON 000074RG	PRO = 000000	STATBI= 064757
CKTIM = 100000	DT.WHL= 000054	KTERRO= 000040	PR4 = 000200	STAT1 = 000027
CLKPRE= 000001	DT.WLL= 000052	KTPRES= 000400	PR5 = 000240	SUSPND= 000001
CM.ARG= ***** G	DVID1 = 000014	KTSTAT= 000020	PR6 = 000300	SVRO = 000062
CONFIG= 000056	ECCMEM= 000100	KTXTND= 040000	PR7 = 000340	SVR1 = 000064
CQOVF = 000001	ECCSTA= 000010	LF = 000012	PS = 177776	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\$DD = 000340	\$LOCTA= 177777	\$\$FROM= 000000
YSERR= 000100	WBSTAT= 000040	\$F\$FAL= 000405	\$LSTIN= 000001	\$\$LCC = 000152R
TMPID = 000002	WBUFEA= 000136	\$F\$GDO= 000400	\$LSTTA= 000001	\$\$LOCN= 000000
TQDVF = 000002	WBUFPA= 000134	\$F\$IF = 000110	\$NESTL= 177777	\$\$REG = 177777
UIPAR0= 177640	WBUFRQ= 000140	\$F\$INC= 000210	\$NSK0 = 000300	\$\$RETU= 000000
UIPAR1= 177642	WBUFSSZ= 000142	\$F\$LOD= 000200	\$NSK1 = 000110	\$\$RTN1= 050000
UIPAR2= 177644	WDFR = 000116	\$F\$NAM= 000160	\$NSK2 = 000110	\$\$RTN2= 050001
UIPAR3= 177646	WDTQ = 000114	\$F\$NO = 000403	\$\$AVLE= 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	\$\$TD = 000000
UIPDRO= 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:32  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0922

3 COMMON EQUATE MODULE  
526 SAVREG SAVE REGISTER R0 THRU R4  
568 COMMON DEFINITIONS AND REFERENCES  
571 000000' .PRINT ;SPMAC: VERSION 1.1  
581 SAVE REGISTER'S R0 THRU R4 ROUTINE  
607 RESREG RESTORE GENERAL REGISTERS R0 THRU R4  
648 COMMON DEFINITIONS AND REFERENCES  
657 RESTORE GENERAL REGISTERS R0 THRU R4 ROUTINE

SAVRES BINDER FOR SAVREG AND RESREG  
SAVRES.MAC 28-JUL-78 09:27

MACY11 30A(1052) 20-SEP-78 18:32 PAGE 19  
COMMON EQUATE MODULE

SEQ 0923

```
508 .TITLE SAVRES BINDER FOR SAVREG AND RESREG
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ;     SAVRES
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     BINDER FOR THE FOLLOWING MODULES:
517 ;         SAVREG - SAVE REGISTERS
518 ;         RESREG - RESTORE REGISTERS
519 ;
520 ; VERSION:
521 ;     0.0
522 ;
523 ;     EDIT           DATE           BY           REASON
524 ;--
```

```
526 .SBTTL SAVREG SAVE REGISTER R0 THRU R4
527 .IDENT /V0.0/
528
529 ;++
530 ; MODULE NAME:
531 ; SAVREG
532 ;
533 ; FUNCTIONAL DESCRIPTION:
534 ; THIS MODULE WILL SAVE GENERAL PURPOSE REGISTERS R0 THRU R4
535 ; ON THE R6 STACK AND RETURNS TO CALLER
536 ;
537 ; INPUTS:
538 ; NONE
539 ;
540 ; IMPLICIT INPUTS:
541 ; NONE
542 ;
543 ; OUTPUTS:
544 ; NONE
545 ;
546 ; IMPLICIT OUTPUTS:
547 ; NONE
548 ;
549 ; PATHOLOGICAL CONNECTIONS:
550 ; NONE
551 ;
552 ; SUBORDINATE ROUTINES CALLED:
553 ; NONE
554 ;
555 ; FUNCTIONAL SIDE EFFECTS:
556 ; NONE
557 ;
558 ; CALLING SEQUENCE:
559 ; CALL SAVREG
560 ;
561 ; VERSION:
562 ; 0.0
563 ;
564 ; EDIT DATE BY REASON
565 ;--
```

SAVRES BINDER FOR SAVREG AND RESREG  
SAVRES.MAC 28-JUL-78 09:27

MACY11 30A(1052) 20-SEP-78 18:32 PAGE 19-2  
SAVREG SAVE REGISTER R0 THRU R4

SEQ 0925

567  
568  
569  
570  
571 000000'  
(1) 000000'  
572 000001  
573 000001  
574  
575  
576  
577  
578  
579

.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

```
581 .SBTTL SAVE REGISTER'S R0 THRU R4 ROUTINE
582
583
584 000000' ROUTINE SAVREG
(2) 000000' SAVREG:
585
586 ;+
587 ; FIRST ALLOCATE 10 BYTES ON THE STACK POINTER BY SETTING STACK
588 ; POINTER TO SP + 12, THEN LOAD RETURN ADDRESS INTO NEW STACK POINTER
589 ; NOW SAVE REGISTERS
590 ;-
591
592 000000' LET SP := SP - #12
(6) 000000' 162706 000012 SUB #12,SP
593 000004' LET (SP) := +12(SP)
(4) 000004' 016616 000012 MOV +12(SP), (SP)
594 000010' LET 2(SP) := R4
(4) 000010' 010466 000002 MOV R4, 2(SP)
595 000014' LET 4(SP) := R3
(4) 000014' 010366 000004 MOV R3, 4(SP)
596 000020' LET 6(SP) := R2
(4) 000020' 010266 000006 MOV R2, 6(SP)
597 000024' LET 10(SP) := R1
(4) 000024' 010166 000010 MOV R1, 10(SP)
598 000030' LET 12(SP) := R0
(4) 000030' 010066 000012 MOV R0, 12(SP)
599
600 ;+
601 ; RETURN TO CALLER
602 ;-
603
604 000034' ENDRTN
(3) 000034' 50000$:
(3) 000034' 50001$:
(2) 000034' 000207 RTS PC
605
```

```
607 .SBTTL RESREG RESTORE GENERAL REGISTERS R0 THRU R4
608 .IDENT /V0.0/
609
610 ;++
611 ; MODULE NAME:
612 ; RESREG
613 ;
614 ; FUNCTIONAL DESCRIPTION:
615 ; THIS MODULE WILL RESTORE GENERAL REGISTERS R0 THRU R4 FROM THE
616 ; R6 STACK AND RETURN TO CALLER.
617 ;
618 ; INPUTS:
619 ; NONE
620 ;
621 ; IMPLICIT INPUTS:
622 ; NONE
623 ;
624 ; OUTPUTS:
625 ; NONE
626 ;
627 ; IMPLICIT OUTPUTS:
628 ; NONE
629 ;
630 ; PATHOLOGICAL CONNECTIONS:
631 ; NONE
632 ;
633 ; SUBORDINATE ROUTINES CALLED:
634 ; NONE
635 ;
636 ; FUNCTIONAL SIDE EFFECTS:
637 ; NONE
638 ;
639 ; CALLING SEQUENCE:
640 ; CALL RESREG
641 ;
642 ; VERSION:
643 ; 0.0
644 ;
645 ; EDIT DATE BY REASON
646 ;--
```

SAVRES BINDER FOR SAVREG AND RESREG  
SAVRES.MAC 28-JUL-78 09:27

MACY11 30A(1052) 20-SEP-78 18:32 PAGE 19-5  
COMMON DEFINITIONS AND REFERENCES

SEQ 0928

648  
649  
650  
651  
652  
653  
654  
655

.SBTTL COMMON DEFINITIONS AND REFERENCES  
  
;\*\*\*\*\*  
; REFERENCED BY OTHER MODULES  
;  
.GLOBL RESREG ;MODULE ENTRY POINT

```
657
658
659 000036'
(2) 000036'
660
661
662
663
664
665
666 000036'
(4) 000036' 016604 000002
667 000042'
(4) 000042' 016603 000004
668 000046'
(4) 000046' 016602 000006
669 000052'
(4) 000052' 016601 000010
670 000056'
(4) 000056' 016600 000012
671 000062'
(4) 000062' 011666 000012
672 000066'
(6) 000066' 062706 000012
673
674
675
676
677
678 000072'
(3) 000072'
(3) 000072'
(2) 000072' 000207
679
680 000001

.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	CTRLC = 000003	ERRTP= 000106	MSGCKD= 000010	RDWHMI= 000022
ACTBIT= 004000	CTRL0 = 000017	EVNTBE= 000200	MSGCKS= 000011	RELERR= 000020
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RELMOD= 020000
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RELTIM= 010000
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	RESREG 000036RG
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGGEP= 000013	RES1 = 000056
ASB = 000106	DROPMO= 100000	FATERR= 100000	MSGHDR= 000004	RES2 = 000060
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RICHAR= 031060
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RPTDAT= 002000
AUTO = 000010	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RSTRT = 000112
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUBOUT= 000177
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RUNMOD= 100000
BIT0 = 000001	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	RVALU= 001740
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SAM = 075464
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SAVREG 000000RG
BIT02 = 000004	DT.ESI= 000044	INHPR= 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	NBKM0D= 001000	SC.CKP= 000004
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NCPUOP= 000020	SC.CLO= 000000
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NDAPTY= 000002	SC.HLD= 000010
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SC.SCA= 000012
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SENDLS= 177777
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	PAERR = 000010	S0FCNT= 000042
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	S0FPAS= 000046
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPACE = 000040
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SPOINT= 000032
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SPVALU= 002200
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SRO = 177572
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR1 = 177574
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	SR2 = 177576
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	SR3 = 172516
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STAT = 000026
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	STATBI= 064757
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	STAT1 = 000027
BKMOD = 000020	DT.ST0= 000010	KIPDR6= 172314	PRI7 = 000340	SUSPND= 000001
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRO = 000000	SVR0 = 000062
BKSLSH= 000134	DT.SWR= 000056	KTERR0= 000040	PR4 = 000200	SVR1 = 000064
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PR5 = 000240	SVR2 = 000066
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR6 = 000300	SVR3 = 000070
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR7 = 000340	SVR4 = 000072
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PS = 177776	SVR5 = 000074
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PSW = 177776	SVR6 = 000076
CLKPRE= 000001	ECCMEM= 000100	MAPSTA= 000200	RANNUM= 000054	SYSCNT= 000052
CONFIG= 000056	ECCSTA= 000010	MED = 076600	RBUFEA= 000130	YSERR= 000100
CQOVF = 000001	ENBEOP= 010000	MEMPAS= 040000	RBUFPA= 000126	TMPIO = 000002
CR = 000015	ENBNUL= 000001	MODEXH= 004000	RBUFSZ= 000132	TQOVF = 000002
CSRA = 000100	ENDLST= 000000	MODHQL= 002000	RBUFVA= 000124	UIPAR0= 177640
CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RDSERV= 000101	UIPAR1= 177642

UIPAR2= 177644	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$F\$FROM= 000000
UIPAR3= 177646	WBUFRO= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$F\$LOC = 000000
UIPAR4= 177650	WBUF SZ= 000142	\$F\$G00= 000400	\$LSTIN= 000001	\$F\$L0CN= 000000
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$F\$REG = 177777
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$F\$RETU= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$L00= 000200	\$NSKO = 000300	\$F\$RTN1= 050000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$F\$RTN2= 050001
UIPDR1= 177602	XFLAG = 000005	\$F\$NO = 000403	\$TAGLE= 177777	\$F\$SRC = 000000
UIPDR2= 177604	XOFF = 000023	\$F\$OR = 000320	\$TAGNU= 050002	\$F\$TGSV= 000000
UIPDR3= 177606	XDN = 000021	\$F\$RTI= 000350	\$TEMP = 000300	\$F\$TGS1= 000000
UIPDR4= 177610	\$BGNLE= 177777	\$F\$RTN= 000300	\$ARGC= 000000	\$F\$TGS2= 000000
UIPDR5= 177612	\$ERFLG= 000400	\$F\$SEL= 000140	\$BYTE= 000000	\$F\$TO = 000000
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$CASE= 000000	\$F\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:33  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0932

3 COMMON EQUATE MODULE  
527 COMMON DEFINITIONS AND REFERENCES FOR SELDES  
530 000000' .PRINT ;SPMAC: VERSION 1.1  
563 KSEL PROCESS THE 'SEL' KEYBOARD COMMAND  
618 KSEL ROUTINE  
630 KDES PROCESS THE 'DES' KEYBOARD COMMAND  
684 KDES ROUTINE  
696 PRSEDE ROUTINE

```
508 .TITLE SELDES PROCESS THE KEYBOARD COMMANDS 'SEL' AND 'DES'  
509 .IDENT /V0.0/  
510  
511 ;++  
512 ; MODULE PACKAGE NAME:  
513 ; SELDES  
514 ;  
515 ; FUNCTIONAL DESCRIPTION:  
516 ; THIS PACKAGE CONTAINS TWO ROUTINES:  
517 ; 1. KSEL - PROCESS THE KEYBOARD COMMAND 'SEL'  
518 ; 2. KDES - PROCESS THE KEYBOARD COMMAND 'DES'  
519 ;  
520 ; VERSION:  
521 ; 0.0  
522 ;  
523 ; EDIT DATE BY REASON  
524 ;  
525 ;--
```

```
527 .SBTTL COMMON DEFINITIONS AND REFERENCES FOR SELDES
528
529 .MCALL STRUCT
530 000000' STRUCT
531 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
532
533 000001 $LSTIN=1
534 000001 $LSTTAG=1
535 ;*****
536 ;
537 ; REFERENCED BY OTHER MODULES:
538 ;
539 .GLOBL KSEL ;SELECT A MODULE
540 .GLOBL KDES ;DESELCT A MODULE
541 ;
542 ;*****
543 ; GLOBAL REFERENCES:
544 ;
545 .GLOBL NAMCHK ;MODULE NAME VERIFICATION ROUTINE
546 .GLOBL SAVREG ;SAVE REGISTERS
547 .GLOBL RESREG ;RESTORE REGISTERS
548 .GLOBL CM.BADNAME ;ADDRESS OF BAD MODULE NAME MESSAGE
549 .GLOBL CM.ARG ;ADDRESS OF INVALID ARGUMENT MESSAGE
550 .GLOBL ARGCHK ;ARGCHK ROUTINE
551 ;
552 ;*****
553 ;
554 ; LOCAL STORAGE:
555 ;
556 000000' 000000 SE.FLG: .WORD 0 ;WHEN A 1 - SELECT, WHEN A 0, DESELECT
557 ;
558 ;*****
559 ;
560 ;
561 ;
```

```
563 .SBTTL KSEL PROCESS THE 'SEL' KEYBOARD COMMAND
564 .IDENT /V0.0/
565
566 ;++
567 ; MODULE NAME:
568 ; KSEL
569 ;
570 ; FUNCTIONAL DESCRIPTION:
571 ; THIS ROUTINE PROCESSES THE 'SEL' KEYBOARD COMMAND.
572 ; IT SHALL DETERMINE IF A MODULE NAME HAS BEEN INCLUDED IN
573 ; THE COMMAND STRING. IF THERE IS NO MODULE NAME, ALL MODULES'
574 ; STATUS WORDS WILL HAVE THE SELECT BIT SET. IF THERE IS A
575 ; MODULE NAME IN THE COMMAND STRING, THE NAME MUST FIRST BE VERIFIED
576 ; AND IF VALID, THE STATUS WORD IN THE MODULE WILL BE UPDATED(SELECT
577 ; BIT SET).
578
579 ; INPUTS:
580 ; 1. DTABLE ADDRESS
581 ; 2. COMMAND DECODE BUFFER POINTER
582
583 ; IMPLICIT INPUTS:
584 ; DT.MLST
585 ; DT.ST0
586
587 ; OUTPUTS:
588 ; NONE
589
590 ; IMPLICIT OUTPUTS:
591 ; DT.KBRSP
592 ; DT.ST1
593 ; DT.ST0
594
595 ; PATHOLOGICAL CONNECTIONS:
596 ; CM.ARG
597 ; CM.BADNAME
598
599 ; SUBORDINATE ROUTINES CALLED:
600 ; SAVREG
601 ; RESREG
602
603 ; FUNCTIONAL SIDE EFFECTS:
604 ; NONE
605
606 ; CALLING SEQUENCE:
607 ; CALL KSEL IN <DTADR,CMDBUF>
608 ; WHERE DTADR = ADDRESS OF DTABLE
609 ; CMDBUF = COMMAND DECODE BUFFER POINTER
610
611 ; VERSION:
612 ; 0.0
613
614 ; EDIT DATE BY REASON
615 ;--
616
```

```
618 .SBTTL KSEL ROUTINE
619
620 000002' ROUTINE KSEL <DTADR,CMDBUF>
(2) 000002'
621 KSEL:
622 000002' LET SE.FLG := #1
(4) 000002' 012767 000001 177770 MOV #1,SE.FLG
623
624 000010' INLINE <BR PRSEDE>
(2) 000010' 000403 BR PRSEDE
625
626 000012' ENDRTN
(3) 000012' 50000$:
(3) 000012' 50001$:
(2) 000012' 000207 RTS PC
627
628
```

```
630 .SBTTL KDES PROCESS THE 'DES' KEYBOARD COMMAND
631 .IDENT /V0.0/
632
633 ;++
634 ; MODULE NAME:
635 ; KDES
636 ;
637 ; FUNCTIONAL DESCRIPTION:
638 ; THIS ROUTINE PROCESSES THE 'DES' KEYBOARD COMMAND.
639 ; IT SHALL DETERMINE IF A MODULE NAME HAS BEEN INCLUDED IN
640 ; THE COMMAND STRING. IF THERE IS NO MODULE NAME, ALL MODULES'
641 ; STATUS WORDS WILL HAVE THE SELECT BIT CLEARED. IF THERE IS A
642 ; MODULE NAME IN THE COMMAND STRING, THE NAME MUST FIRST BE VERIFIED
643 ; AND IF VALID, THE STATUS WORD IN THE MODULE WILL BE UPDATED(SELECT
644 ; BIT CLEARED).
645 ;
646 ; INPUTS:
647 ; 1. DTABLE ADDRESS
648 ; 2. COMMAND DECODE BUFFER POINTER
649 ;
650 ; IMPLICIT INPUTS:
651 ; DT.MLST
652 ; DT.ST0
653 ;
654 ; OUTPUTS:
655 ; NONE
656 ;
657 ; IMPLICIT OUTPUTS:
658 ; DT.KBRSP
659 ; DT.ST1
660 ; DT.ST0
661 ;
662 ; PATHOLOGICAL CONNECTIONS:
663 ; CM.ARG
664 ; CM.BADNAME
665 ;
666 ; SUBORDINATE ROUTINES CALLED:
667 ; SAVREG
668 ; RESREG
669 ;
670 ; FUNCTIONAL SIDE EFFECTS:
671 ; NONE
672 ;
673 ; CALLING SEQUENCE:
674 ; CALL KDES IN <DTADR,CMDBUF>
675 ; WHERE DTADR = ADDRESS OF DTABLE
676 ; CMDBUF = COMMAND DECODE BUFFER POINTER
677 ;
678 ; VERSION:
679 ; 0.0
680 ;
681 ; EDIT DATE BY REASON
682 ;--
```



```
684 .SBTTL KDES ROUTINE
685
686 000014' ROUTINE KDES <DTADR,CMDBUF>
(2) 000014' KDES:
687
688 ;+
689 ; SET THE FLAG TO DESELECT AND CALL THE COMMON ROUTINE
690 ; -
691
692 000014' LET SE.FLG := #0
(4) 000014' 005067 177760 CLR SE.FLG
693
694
```

```

696 .SBTTL PRSEDE ROUTINE
697
698 000020'          INLINE <PRSEDE:>
(2) 000020'
699
700          ;+
701          ; SAVE REGISTERS AND SAVE DTABLE ADDRESS
702          ; -
703
704 000020'          CALL SAVREG
(3) 000020' 004767 000000G          JSR      PC, SAVREG
705 000024'          LET R0 := DTADR(R5)
(4) 000024' 016500 000000          MOV      DTADR(R5), R0
706
707          ;+
708          ; GET ADDRESS OF MODULE LIST AND SAVE DECODE BUFFER POINTER
709          ; -
710
711 000030'          LET R1 := DT.MLST(R0)
(4) 000030' 016001 000032          MOV      DT.MLST(R0), R1
712 000034'          LET R2 := CMDBUF(R5)
(4) 000034' 016502 000002          MOV      CMDBUF(R5), R2
713
714
715          ;+
716          ; ADVANCE DECODE BUFFER PTR TO FIRST NON SPACE CHARACTER.
717          ; IF IT'S A <CR>, SELECT ALL MODULES - IF NOT A <CR>
718          ; GO VERIFY THE MODULE NAME
719          ; -
720
721
722
723
724 000040'          CALL ARGCHK IN <R2> OUT <R2>
(4) 000040' 162705 000002          SUB      #1*2, R5
(3) 000044' 010546          MOV      R5, -(SP)
(4) 000046' 010245          MOV      R2, -(R5)
(3) 000050' 004767 000000G          JSR      PC, ARGCHK
(3) 000054' 012605          MOV      (SP)+, R5
(4) 000056' 012502          MOV      (R5)+, R2
725
726 000060'          IF.ERROR THEN
(6) 000060' 103047          BCC      50002$
727
728          ;+
729          ; ITS A CR SO SELECT - DESELECT THEM ALL
730          ; -
731
732
733
734          ;+
735          ; IF WE ARE SELECTING, SET BIT 14.
736          ; IF WE ARE DESELECTING, CLEAR BIT 14.
737          ; -
738
739 000062'          IF SE.FLG GT #0 THEN
  
```

```

(6) 000062' 005767 177712
(9) 000066' 003431
740 000070'
(4) 000070'
(6) 000070' 005711
(9) 000072' 001417
741
742
743
744
745
746
747 000074'
(4) 000074' 012102
748
749 000076'
(6) 000076' 052762 040000 000026
750 000104'
(4) 000104' 016204 000026
(6) 000110' 042704 064757
751 000114'
(6) 000114' 020427 000000
(9) 000120' 001003
752 000122'
(6) 000122' 052760 010000 000012
753 000130'
(4) 000130'
754
755 000130'
(4) 000130' 000757
(3) 000132'
756 000132'
(6) 000132' 032760 100000 000010
(9) 000140' 001403
757 000142'
(6) 000142' 052760 001000 000010
758 000150'
(4) 000150'
759
760 000150'
(4) 000150' 000412
(3) 000152'
761
762 000152'
(6) 000152' 052760 100000 000012
763 000160'
(4) 000160'
(6) 000160' 005711
(9) 000162' 001405
764
765
766
767
768
769
770 000164'
    
```

WHILE (R1) NE #0 DO

```

50004$:
TST (R1)
BLE 50005$
BEQ 50005$
;+
; GET A MODULE HEADER ADDRESS FROM MODULE LIST, UPDATE
; STATUS WORD, ADVANCE TO GET NEXT MODULE HEADER ADDRESS.
;-
LET R2 := (R1)+
MOV (R1)+,R2
LET STAT(R2) := STAT(R2) SET.BY #BIT14
BIS #BIT14,STAT(R2)
LET R4 := STAT(R2) CLR.BY #STATBITS
MOV STAT(R2),R4
BIC #STATBITS,R4
IF R4 EQ #SBKMOD THEN
CMP R4,#SBKMOD
BNE 50006$
LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
BIS #SBKSEL,DT.ST1(R)
ENDIF
50006$:
ENDDO
BR 50004$
50005$:
IF #RUNMODE SETIN DT.ST0(R0) THEN
BIT #RUNMODE,DT.ST0(
BEQ 50007$
LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODSEL
BIS #MODSEL,DT.ST0(R)
ENDIF
50007$:
ELSE
BR 50010$
50003$:
LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
BIS #CKTIM,DT.ST1(R0)
WHILE (R1) NE #0 DO
50011$:
TST (R1)
BEQ 50012$
;+
; GET A MODULE HEADER ADDRESS FROM MODULE LIST, UPDATE
; STATUS WORD, ADVANCE TO GET NEXT MODULE HEADER ADDRESS.
;-
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
(4) 000174' 000771                                BR      50011$
(3) 000176'                                50012$:
774
775 000176'                                ENDIF
(4) 000176'                                50010$:
776
777
778 000176'                                ELSE
(4) 000176' 000472                                BR      50013$
(3) 000200'                                50002$:
779
780 ;+
781 ; THERE IS A MODULE NAME, SO GO CHECK IT OUT
782 ; -
783
784 000200'                                CALL NAMCHK IN <R0,R2> OUT <R3,R2>
(4) 000200' 162705 000004                        SUB      #2*2,R5
(3) 000204' 010546                        MOV      R5,-(SP)
(5) 000206' 010245                        MOV      R2,-(R5)
(4) 000210' 010045                        MOV      R0,-(R5)
(3) 000212' 004767 000000G                JSR      PC,NAMCHK
(3) 000216' 012605                        MOV      (SP)+,R5
(4) 000220' 012503                        MOV      (R5)+,R3
(4) 000222' 012502                        MOV      (R5)+,R2
785
786 ;+
787 ; SEE IF NAME O.K. IF IT WAS, THE MODULE HEADER ADDRESS
788 ; WILL BE IN R3.
789 ; -
790 000224'                                IF.NO.ERROR THEN
(6) 000224' 103454                                BCS     50014$
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
(6) 000246' 103037                                BCC     50015$
800
801 ;+
802 ; IT'S A CR, SO ALL IS WELL, SO IF SELECTING - SET BIT 14
803 ; AND IF DESELECTING - CLEAR BIT 14
    
```

```

804      ; -
805
806
807
808      000250'                IF SE.FLG GT #0 THEN
(6) 000250' 005767 177524                TST      SE.FLG
(9) 000254' 003425                BLE     50016$
809 000256'                LET STAT(R3) := STAT(R3) SET.BY #BIT14
(6) 000256' 052763 040000 000026        BIS     #BIT14,STAT(R3)
810 000264'                IF #RUNMODE SETIN DT.ST0(R0) THEN
(6) 000264' 032760 100000 000010        BIT     #RUNMODE,DT.ST0(
(9) 000272' 001403                BEQ     50017$
811 000274'                LET DT.ST0(R0) := DT.ST0(R0) SET.BY #MODSEL
(6) 000274' 052760 001000 000010        BIS     #MODSEL,DT.ST0(R
812 000302'                ENDIF
(4) 000302'                50017$:
813 000302'                LET R4 := STAT(R3) CLR.BY #STATBITS
(4) 000302' 016304 000026                MOV     STAT(R3),R4
(6) 000306' 042704 064757                BIC     #STATBITS,R4
814 000312'                IF R4 EQ #SBKMOD THEN
(6) 000312' 020427 000000                CMP     R4,#SBKMOD
(9) 000316' 001003                BNE     50020$
815 000320'                LET DT.ST1(R0) := DT.ST1(R0) SET.BY #SBKSEL
(6) 000320' 052760 010000 000012        BIS     #SBKSEL,DT.ST1(R
816 000326'                ENDIF
(4) 000326'                50020$:
817 000326'                ELSE
(4) 000326' 000406                BR      50021$
(3) 000330'                50016$:
818
819
820      ; +
821      ; IT'S A DESELECT.... DO IT
822      ; -
823
824 000330'                LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
(6) 000330' 052760 100000 000012        BIS     #CKTIM,DT.ST1(R0
825 000336'                LET STAT(R3) := STAT(R3) CLR.BY #BIT14!ACTBIT
(6) 000336' 042763 044000 000026        BIC     #BIT14!ACTBIT,ST
826 000344'                ENDIF
(4) 000344'                50021$:
827 000344'                ELSE
(4) 000344' 000403                BR      50022$
(3) 000346'                50015$:
828
829
830      ; +
831      ; BAD ! THERE ARE ADDITIONAL ARGUMENTS BEYOND THE MODULE NAME
832      ; SO STUFF THE BAD ARG MESSAGE AND GET OUT
833      ; -
834 000346'                LET DT.KBRSP(R0) := #CM.ARG
(4) 000346' 012760 000000G 000022        MOV     #CM.ARG,DT.KBRSP
834 000354'                ENDIF
(4) 000354'                50022$:
835 000354'                ELSE
(4) 000354' 000403                BR      50023$
(3) 000356'                50014$:
    
```

```
836
837          ;+
838          ; THERE WAS AN ERROR IN SYNTAX OF MODULE NAME
839          ; SO STUFF THE INVALID MODULE NAME ERROR MSG
840          ; -
841
842 000356'          LET DT.KBRSP(R0) := #CM.BADNAME
(4) 000356' 012760 000000G 000022          MOV          #CM.BADNAME,DT.K
843 000364'          ENDIF
(4) 000364'          50023$:
844
845 000364'          ENDIF
(4) 000364'          50013$:
846
847          ;+
848          ; CLEAN UP
849          ; -
850
851 000364'          CALL RESREG
(3) 000364' 004767 000000G          JSR          PC,RESREG
852
853 000370'          ENDRTN
(3) 000370'          50000$:
(3) 000370'          50001$:
(2) 000370' 000207          RTS          PC
854
855          000001          .END
```

ACSR = 000102	CQOVF = 000001	ECCSTA= 000010	LPSTAT= 000001	PR7 = 000340
ACTBIT= 004000	CR = 000015	ENBEOP= 010000	MAPSTA= 000200	PS = 177776
ADDR22= 001000	CSRA = 000100	ENBNUL= 000001	MED = 076600	PSW = 177776
ADR = 000006	CSRC = 000102	ENLST= 000000	MEMPAS= 040000	RANNUM= 000054
APTFER= 000004	CTRLC = 000003	EOPBIT= 000001	MODEXH= 004000	RBUFEA= 000130
APTPRE= 000200	CTRLD = 000017	ERRTYP= 000106	MODHOL= 002000	RBUFPA= 000126
ARGCHK= ***** G	CTRLU = 000025	EVNTBE= 000200	MODSEL= 001000	RBUFSZ= 000132
ASB = 000106	DCEVNT= 000011	EVNTHD= 000200	MSGCKD= 000010	RBUFVA= 000124
ASSEMB= 000010	DEFRTN= 000400	EVNTKT= 000203	MSGCKS= 000011	RDSERV= 000101
ASTAT = 000104	DIAGMC= 000000	EVNTPE= 000202	MSGDER= 000005	RDWHMI= 000022
AUTO = 000010	DRDPMO= 100000	EVNTRE= 000201	MSGDRP= 000017	RELERR= 000020
AUTOST= 020000	DSEVNT= 000014	FATERR= 100000	MSGECH= 177777	RELMOD= 020000
AWAS = 000110	DTADR = 000000	HRDCNT= 000044	MSGGEO= 000013	RELTIM= 010000
BIT0 = 000001	DT.ADD= 000042	HRDPAS= 000050	MSGHDR= 000004	RESREG= ***** G
BIT00 = 000001	DT.AP = 000100	ICONT = 000036	MSGHNG= 000022	RES1 = 000056
BIT01 = 000002	DT.APK= 000076	ICOUNT= 000040	MSGHRD= 000007	RES2 = 000060
BIT02 = 000004	DT.BLS= 000034	IDNUM = 000122	MSGMAP= 000021	RICHAR= 031060
BIT03 = 000010	DT.CFO= 000014	IE = 000100	MSGNUL= 177775	RPTDAT= 002000
BIT04 = 000020	DT.CF1= 000016	INDPAR= 000040	MSGPOP= 000002	RSTRT = 000112
BIT05 = 000040	DT.ERR= 000020	INHDRP= 040000	MSGPRM= 177776	RUBOUT= 000177
BIT06 = 000100	DT.ESI= 000044	INHPR= 020000	MSGRES= 000001	RUNMOD= 100000
BIT07 = 000200	DT.EVN= 000000	INHREL= 001000	MSGSFT= 000006	RVALU= 001740
BIT08 = 000400	DT.EXS= 000060	INHRE= 000400	MSGSKE= 000003	SAM = 075464
BIT09 = 001000	DT.FCH= 000037	INIT = 000030	MSGSMB= 000015	SAVREG= ***** G
BIT1 = 000002	DT.FCN= 000036	INTR = 000120	MSGSMH= 000014	SBADR = 000102
BIT10 = 002000	DT.HMX= 000104	IOMOD = 100000	MSGSMS= 000016	SBKMOD= 000000
BIT11 = 004000	DT.KBE= 000024	IOMODP= 102000	MSGSTD= 000000	SBKSEL= 010000
BIT12 = 010000	DT.KBP= 000026	IOMODR= 112000	MSGSYS= 000012	SC.ADR= 000006
BIT13 = 020000	DT.KBR= 000022	IOMODX= 110000	MSGVEC= 000020	SC.ALC= 000014
BIT14 = 040000	DT.KBU= 000030	JACK = 035060	NAMCHK= ***** G	SC.APC= 000016
BIT15 = 100000	DT.MLS= 000032	KDES 000014RG	NBKMOD= 001000	SC.CKL= 000002
BIT2 = 000004	DT.MTI= 000110	KIPAR0= 172340	NCPUOP= 000020	SC.CKP= 000004
BIT3 = 000010	DT.OFF= 000070	KIPAR1= 172342	NOAPTY= 000002	SC.CLO= 000000
BIT4 = 000020	DT.PAS= 000074	KIPAR2= 172344	NULL = 000000	SC.HLD= 000010
BIT5 = 000040	DT.PC = 000002	KIPAR3= 172346	OWEN = 024020	SC.SCA= 000012
BIT6 = 000100	DT.PFL= 000062	KIPAR4= 172350	PAERR = 000010	SENDLS= 177777
BIT7 = 000200	DT.PSW= 000004	KIPAR5= 172352	PARPRE= 002000	SE.FLG 000000R
BIT8 = 000400	DT.PTA= 000064	KIPAR6= 172354	PARSTA= 000100	SOFcnt= 000042
BIT9 = 001000	DT.RCS= 000102	KIPAR7= 172356	PASCNT= 000034	SOFPAS= 000046
BKDEF = 000002	DT.REL= 000040	KIPDR0= 172300	PDPLSI= 020000	SPACE = 000040
BKMOD = 000020	DT.SCT= 000066	KIPDR1= 172302	PDP60 = 004000	SPOINT= 000032
BKMODE= 040000	DT.SMX= 000106	KIPDR2= 172304	PDP70 = 010000	SPVALU= 002200
BKSLSH= 000134	DT.SP = 000006	KIPDR3= 172306	PRI0 = 000000	SR0 = 177572
CAPRES= 000004	DT.SSI= 000046	KIPDR4= 172310	PRI1 = 000040	SR1 = 177574
CASTAT= 000004	DT.ST0= 000010	KIPDR5= 172312	PRI4 = 000200	SR2 = 177576
CDERCT= 000146	DT.ST1= 000012	KIPDR6= 172314	PRI5 = 000240	SR3 = 172516
CDWDCT= 000144	DT.SWR= 000056	KIPDR7= 172316	PRI6 = 000300	STAT = 000026
CKTIM = 100000	DT.SYP= 000072	KSEL 000002RG	PRI7 = 000340	STATBI= 064757
CLKPRE= 000001	DT.WBU= 000050	KTERRO= 000040	PRSEDE 000020R	STAT1 = 000027
CMDBUF= 000002	DT.WHL= 000054	KTPRES= 000400	PRO = 000000	SUSPND= 000001
CM.ARG= ***** G	DT.WLL= 000052	KTSTAT= 000020	PR4 = 000200	SVR0 = 000062
CM.BAD= ***** G	DVID1 = 000014	KTXTND= 040000	PR5 = 000240	SVR1 = 000064
CONFIG= 000056	ECCMEM= 000100	LF = 000012	PR6 = 000300	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\$GOD= 000400	\$LSTIN= 000001	\$\$DST = 000000
SVR6 = 000076	WBSTAT= 000040	\$F\$IF = 000110	\$LSTTA= 000001	\$\$ELDC= 000402
SYSCNT= 000052	WBUFEA= 000136	\$F\$INC= 000210	\$NESTL= 177777	\$\$ERFL= 000000
SYSERR= 000100	WBUFPA= 000134	\$F\$L0D= 000200	\$NSK0 = 000300	\$\$FLAG= 000001
TMPID = 000002	WBUFRQ= 000140	\$F\$NAM= 000160	\$NSK1 = 000110	\$\$FROM= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$NO = 000403	\$NSK2 = 000110	\$\$L0C = 000316R
UIPAR0= 177640	WDFR = 000116	\$F\$DR = 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	XTWHMI= 000222	\$F\$SEL= 000140	\$\$AVLE= 177777	\$\$RTN1= 050000
UIPAR4= 177650	XFLAG = 000005	\$F\$THE= 000330	\$\$SK0 = 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	\$ISKO = 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	\$SARGC= 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:35  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 0946

3 COMMON EQUATE MODULE  
555 COMMON DEFINITIONS AND REFERENCES  
558 000000' .PRINT ;SPMAC: VERSION 1.1  
577 STRAP ROUTINE

STRAP (FIELD SOFTWARE TRAP)  
STRAP.MAC 28-JUL-78 09:27

MACY11 30A(1052) 20-SEP-78 18:35 PAGE 19  
COMMON EQUATE MODULE

SEQ 0947

```
508 .TITLE STRAP (FIELD SOFTWARE TRAP)
509 .IDENT /VO.0/
510
511 ;++
512 ; MODULE NAME:
513 ; STRAP
514
515 ; FUNCTIONAL DESCRIPTION:
516 ;
517 ; THIS ROUTINE IS ENTERED BY ANY ONE OF SEVERAL SOFTWARE TRAPS
518 ; (I.E. MSGN, GWBUFF, ETC.) FROM DECX/11 OPTION MODULES.
519 ; AN EVENT CODE IS GENERATED AND
520 ; PUSHED ONTO THE STACK.
521 ; THERE IS ONE ENTRY POINT FOR A SOFTWARE TRAP
522 ; THRU LOC. 34.
523
524 ; INPUTS:
525 ; NONE.
526
527 ; IMPLICIT INPUTS:
528 ; SOFTWARE TRAP INSTRUCTION (THRU LOCATION 34)
529
530 ; OUTPUTS:
531 ; NONE
532
533 ; IMPLICIT OUTPUTS:
534 ; EVENT CODE
535
536 ; PATHOLOGICAL CONNECTIONS:
537 ; NONE.
538
539 ; SUBORDINATE MODULES CALLED:
540 ; NONE.
541
542 ; FUNCTIONAL SIDE EFFECTS:
543 ; NONE.
544
545 ; CALLING SEQUENCE:
546 ; CALLED BY A 1044XX TRAP INSTRUCTION
547 ; XX = THE CODE FOR THE PARTICULAR TRAP INSTRUCTION.
548
549 ; VERSION:
550 ; 0.0
551
552 ; EDIT BY DATE REASON
553 ;---
```

STRAP (FIELD SOFTWARE TRAP)  
STRAP.MAC 28-JUL-78 09:27

MACY11 30A(1052) 20-SEP-78 18:35 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 0948

```
555          .SBTTL COMMON DEFINITIONS AND REFERENCES
556          ;
557          .MCALL STRUCT
558 000000'    STRUCT
(1) 559 000000' .PRINT ;SPMAC: VERSION 1.1
560          $LSTIN = 1
561          $LSTTAG = 1
562          ;*****
563          ;
564          ; REFERENCED BY OTHER MODULES
565          ;
566          .GLOBL STINT          ;ENTRY ADDRESS OF FIELD SOFTWARE TRAP ROUTINE
567          .GLOBL ST.EXT        ;ADDRESS TO EXIT THROUGH
568          ;
569          ;
570          ;*****
571          ;
572          ; LOCAL STORAGE:
573          ;
574 000000' 000000 ST.EXT: .WORD 0      ;ADDRESS TO EXIT THIS MODULE THRU
575          ;
```

```
577          .SBTTL STRAP ROUTINE
578
579
580          ;+
581          ;ENTER HERE FOR SOFTWARE TRAP
582          ;-
583
584 000002'    INLINE <STINT:>
(2) 000002'                                     STINT:
585
586          ;+
587          ; SAVE REGISTER AND RETRIEVE PC+2 FROM STACK AND DETERMINE ACTUAL TRAP ADDRESS
588          ;-
589
590 000002'    PUSH      R0
(2) 000002' 010046                                MOV      R0,-(SP)
591 000004'    LET      R0 := 2(SP)
(4) 000004' 016600 000002                        MOV      2(SP),R0
592 000010'    LET      R0 := -(R0)
(4) 000010' 014000                                MOV      -(R0),R0
593
594          ;+
595          ; IDENTIFY TRAP AND SAVE EVENT CODE ON STACK, RESTORE REGISTER
596          ; AND CLEAN UP STACK AND EXIT ROUTINE
597          ;-
598
599 000012'    LET      R0 := R0 CLR.BY #177400
(6) 000012' 042700 177400                        BIC      #177400,R0
600 000016'    PUSH      R0
(2) 000016' 010046                                MOV      R0,-(SP)
601 000020'    LET      R0 := 2(SP)
(4) 000020' 016600 000002                        MOV      2(SP),R0
602 000024'    LET      (SP) := (SP)+
(4) 000024' 012616                                MOV      (SP)+,(SP)
603 000026' 000177 177746                        JMP      @ST.EXT
604
605          000001          .END
```

ACSR = 000102	CTRLC = 000003	ERRTP= 000106	MSGCKD= 000010	ROWHMI= 000022
ACTBIT= 004000	CTRL0 = 000017	EVNTBE= 000200	MSGCKS= 000011	RELERR= 000020
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGDER= 000005	RELMOD= 020000
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDRP= 000017	RELTIM= 010000
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGECH= 177777	RES1 = 000056
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGEOP= 000013	RES2 = 000060
ASB = 000106	DROPM0= 100000	FATERR= 100000	MSGHDR= 000004	RICHAR= 031060
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHNG= 000022	RPTDAT= 002000
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHRD= 000007	RSTRT = 000112
AUTO = 000010	DT.AP = 000100	ICONT = 000036	MSGMAP= 000021	RUBOUT= 000177
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGNUL= 177775	RUNMOD= 100000
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGPOP= 000002	RSVALU= 001740
BIT0 = 000001	DT.CFO= 000014	IE = 000100	MSGPRM= 177776	SAM = 075464
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGRES= 000001	SBADR = 000102
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGSFT= 000006	SBKMOD= 000000
BIT02 = 000004	DT.ESI= 000044	INHEPR= 020000	MSGSKE= 000003	SBKSEL= 010000
BIT03 = 000010	DT.EVN= 000000	INHREL= 001000	MSGSMB= 000015	SC.ADR= 000006
BIT04 = 000020	DT.EXS= 000060	INHRR= 000400	MSGSMH= 000014	SC.ALC= 000014
BIT05 = 000040	DT.FCH= 000037	INIT = 000030	MSGSMS= 000016	SC.APC= 000016
BIT06 = 000100	DT.FCN= 000036	INTR = 000120	MSGSTD= 000000	SC.CKL= 000002
BIT07 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSYS= 000012	SC.CKP= 000004
BIT08 = 000400	DT.KBE= 000024	IOMODP= 102000	MSGVEC= 000020	SC.CLO= 000000
BIT09 = 001000	DT.KBP= 000026	IOMODR= 112000	NBKMOD= 001000	SC.HLD= 000010
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NCPUDP= 000020	SC.SCA= 000012
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NOAPTY= 000002	SENDLS= 177777
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NULL = 000000	SDFCNT= 000042
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	OWEN = 024020	SOPPAS= 000046
BIT13 = 020000	DT.DFF= 000070	KIPAR2= 172344	PAERR = 000010	SPACE = 000040
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PARPRE= 002000	SPOINT= 000032
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARSTA= 000100	SPVALU= 002200
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PASCNT= 000034	SRO = 177572
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PDPLSI= 020000	SR1 = 177574
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDP60 = 004000	SR2 = 177576
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP70 = 010000	SR3 = 172516
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PRI0 = 000000	STAT = 000026
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PRI1 = 000040	STATBI= 064757
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRI4 = 000200	STAT1 = 000027
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI5 = 000240	STINT 000002RG
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI6 = 000300	ST.EXT 000000RG
BKMOD = 000020	DT.STO= 000010	KIPDR6= 172314	PRI7 = 000340	SUSPND= 000001
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRO = 000000	SVRO = 000062
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PR4 = 000200	SVR1 = 000064
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PR5 = 000240	SVR2 = 000066
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR6 = 000300	SVR3 = 000070
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR7 = 000340	SVR4 = 000072
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PS = 177776	SVR5 = 000074
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PSW = 177776	SVR6 = 000076
CLKPRE= 000001	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	TMPIO = 000002
CR = 000015	ENBNUL= 000001	MODEXH= 004000	RBUFSZ= 000132	TQOVF = 000002
CSRA = 000100	ENDLST= 000000	MODHQL= 002000	RBUFVA= 000124	UIPAR0= 177640
CSRC = 000102	EOPBIT= 000001	MODSEL= 001000	RDSERV= 000101	UIPAR1= 177642

UIPAR2= 177644	WBUFPA= 000134	\$F\$DO = 000340	\$IFLEV= 177777	\$SLOC = 000000
UIPAR3= 177646	WBUFRO= 000140	\$F\$FAL= 000405	\$LOCTA= 177777	\$SLOCN= 000000
UIPAR4= 177650	WBUFSZ= 000142	\$F\$GDD= 000400	\$LSTIN= 000001	\$SREG = 177777
UIPAR5= 177652	WDFR = 000116	\$F\$IF = 000110	\$LSTTA= 000001	\$SRETU= 000000
UIPAR6= 177654	WDTO = 000114	\$F\$INC= 000210	\$NESTL= 177777	\$SRTN1= 000000
UIPAR7= 177656	WTINRE= 000352	\$F\$LDD= 000200	\$SAVLE= 177777	\$SRTN2= 000000
UIPDR0= 177600	WTWHMI= 000222	\$F\$NAM= 000160	\$TAGLE= 177777	\$SSRC = 000000
UIPDR1= 177602	XFLAG = 000005	\$F\$NO = 000403	\$TAGNU= 050000	\$STGSV= 000000
UIPDR2= 177604	XOFF = 000023	\$F\$OR = 000320	\$TEMP = 000402	\$STGS1= 000000
UIPDR3= 177606	XDN = 000021	\$F\$RTI= 000350	\$SARGC= 000000	\$STGS2= 000000
UIPDR4= 177610	\$BGNLE= 177777	\$F\$RTN= 000300	\$SBYTE= 000000	\$STO = 000000
UIPDR5= 177612	\$ERFLG= 000400	\$F\$SEL= 000140	\$SCASE= 000000	\$STAG= 050000
UIPDR6= 177614	\$F\$AND= 000310	\$F\$THE= 000330	\$SDST = 000000	. = 000032R
UIPDR7= 177616	\$F\$BAD= 000401	\$F\$TRU= 000404	\$SELOC= 000000	
WASADR= 000104	\$F\$BLA= 000170	\$F\$UNT= 000130	\$SERFL= 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)

3	COMMON EQUATE MODULE
534	CLKCHK - SYSTEM CLOCK CHECK
600	COMMON DEFINITIONS AND REFERENCES
603	000000' .PRINT ;SPMAC: VERSION 1.1
627	CHECK FOR SYSTEM CLOCK ROUTINE
779	CLKOFF - SYSTEM CLOCK-OFF
826	COMMON DEFINITIONS AND REFERENCES
833	CLKOFF ROUTINE
903	CLKON SYSTEM CLOCK-ON
951	COMMON DEFINITIONS AND REFERENCES
961	CLKON ROUTINE
990	PRHMS PROCESS HOURS, MINUTES, AND SECONDS CONVERSION
1045	COMMON DEFINITIONS AND REFERENCES
1060	PROCESS HOURS, MINUTES AND SECONDS CONVERSION ROUTINE
1129	LDTIME - LOAD CONVERTED TIME
1155	PRPSCNT PROCESS PASS COUNT TIME
1218	COMMON DEFINITIONS AND REFERENCES
1233	PROCESS PASS COUNT TIME ROUTINE
1328	HMS HOURS, MINUTES, AND SECONDS CONVERSION
1385	COMMON DEFINITIONS AND REFERENCES
1417	HOURS, MINUTES AND SECONDS CONVERSION ROUTINE
1547	UPDTIM UPDATE MODULE PASS TIME
1602	COMMON DEFINITIONS AND REFERENCES
1616	UPDATE TIME ROUTINE
1683	CLRTIM CLEAR TIME FROM SYSTEM CLOCK TIME TABLES
1731	COMMON DEFINITIONS AND REFERENCES
1740	CLEAR TIME ROUTINE
1781	CKHUNG CHECK FOR HUNG OPTION MODULES
1834	COMMON DEFINITIONS AND REFERENCES
1856	CHECK MODULE HUNG ROUTINE

```
508 .TITLE SYSCLK SYSTEM CLOCK BINDER MODULE
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ;     SYSCLK
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     BINDER FOR THE FOLLOWING MODULES:
517 ;         CLKCHK - SYSTEM CLOCK CHECK
518 ;         CLKOFF - TURN SYSTEM CLOCK OFF
519 ;         CLKON  - TURN SYSTEM CLOCK ON
520 ;         PRHMS  - PROCESS HOURS, MINUTES AND SECONDS CONVERSION
521 ;         PRPSCNT - PROCESS PASS COUNT TIME
522 ;         HMS    - HOURS, MINUTES AND SECONDS CONVERSION
523 ;         UPDTIM - UPDATE PASS TIME
524 ;         CLRTIM - CLEAR MODULE TIME TABLE
525 ;         CKHUNG - CHECK FOR HUNG OPTION MODULE
526 ;
527 ; VERSION:
528 ;     0.0
529 ;
530 ;     EDIT          DATE          BY          REASON
531 ;--
```



```
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588
```

.SBTTL CLKCHK - SYSTEM CLOCK CHECK  
.IDENT /V0.0/

;++  
; MODULE NAME:  
;     CLKCHK  
;  
; FUNCTIONAL DESCRIPTION:  
;     THIS MODULE DETERMINES IF A CLOCK IS AVAILABLE ON THE SYSTEM. THIS IS  
;     ACCOMPLISHED BY FIRST CHECKING TO SEE IF A KW11-L AND/OR A KW11-P CLOCK  
;     OPTION MODULES ARE PART OF THE RUNTIME EXERCISER AND SECONDLY,  
;     WHETHER THE CLOCK HARDWARE IS PART OF THE SYSTEM CONFIGURATION.  
;  
;     ONCE A CLOCK HAS BEEN DETERMINED TO BE AVAILABLE THEN THE APPROPRIATE  
;     CLOCK CODE IS SET IN THE SYSTEM CLOCK INDICATOR WORD, THIS INDICATES  
;     THE TYPE OF SYSTEM CLOCK, AND ALSO SYSTEM CLOCK PRESENT BIT IN CONFIGURA-  
;     TION WORD 0 AND FINALLY THE SYSTEM CLOCK MODULE BEGIN ADDRESS  
;     IS PLACED IN THE SYSTEM CLOCK WORD. IF THERE IS NOT A SYSTEM CLOCK  
;     AVAILABLE THEN THE ABOVE WORDS WILL REFLECT THE NO CLOCK CONDITION.  
;  
;     NOTE:    IF THE KW11-L IS AVAILABLE IT WILL BE USED AS THE SYSTEM  
;              CLOCK EVEN IF A KW11-P IS PRESENT.  
;  
;     THIS MODULE WILL BE A ONE TIME ONLY ROUTINE CALLED AT INITIALIZATION  
;     OF DEC/X11.  
;  
; INPUTS:  
;     ADDRESS OF DTABLE  
;  
; IMPLICIT INPUTS:  
;     DT.CFO           - CONFIGURATION WORD 0 LOCATED IN DTABLE  
;     DT.SCT           - SCTAB ADDRESS  
;     SC.CLO           - SYSTEM CLOCK WORD  
;     SC.CKL           - ADDRESS OF POINTER TO KW11-L HEADER ADDRESS  
;     SC.CLP           - ADDRESS OF POINTER TO KW11-P HEADER ADDRESS  
;     SC.SCA           - SYSTEM CLOCK ADDRESS  
;  
; OUTPUTS:  
;     NONE  
;  
; IMPLICIT OUTPUTS:  
;     DT.CFO           - CONFIGURATION WORD 0 LOCATED IN DTABLE  
;     SC.CLO           - SYSTEM CLOCK WORD  
;     SC.SCA           - SYSTEM CLOCK ADDRESS  
;  
; PATHOLOGICAL CONNECTIONS:  
;     NONE  
;  
; SUBORDINATE ROUTINES CALLED:  
;     SAVREG           - SAVE REGISTERS  
;     ENQTQ            - TYPE-QUEUE ENQUEUER  
;     HRDADRCHK        - HARDWARE ADDRESS CHECK  
;     RESREG           - RESTORE REGISTERS  
;  
; FUNCTIONAL SIDE EFFECTS:

SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 19-2  
CLKCHK - SYSTEM CLOCK CHECK

SEQ 0955

589  
590  
591  
592  
593  
594  
595  
596  
597  
598

```
;      NONE  
;  
; CALLING SEQUENCE:  
;      CALL CLKCHK IN <A>  
;      A= ADDRESS OF DTABLE  
;  
;      EDIT          DATE          BY          REASON  
;--
```

```
600 .SBTTL COMMON DEFINITIONS AND REFERENCES
601
602 .MCALL STRUCT
603 000000' STRUCT
604 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
605 000001 $LSTIN =1
606 000001 $LSTTAG =1
607
608 ;*****
609 ; REFERENCED BY OTHER MODULES
610 ;
611 .GLOBL CLKCHK ;MODULE ENTRY POINT
612
613 ;*****
614 ; GLOBAL REFERENCES
615 ;
616 .GLOBL SAVREG ;SAVE REGISTERS
617 .GLOBL HRDADRCHK ;HARDWARE ADDRESS CHECK
618 .GLOBL ENQTQ ;ENQUEUE INTO TYPE QUEUE
619 .GLOBL RESREG ;RESTORE REGISTERS
620
621 ;*****
622 ; LOCAL STORAGE
623 ;
624 000000' 054523 052123 046505 CC.CM1: .ASCIZ /SYSTEM CLOCK NOT AVAILABLE%/
625 000006' 041440 047514 045503
626 000014' 047040 052117 040440
627 000022' 040526 046111 041101
628 000030' 042514 000045
629
630 .EVEN
```

```
627 .SBTTL CHECK FOR SYSTEM CLOCK ROUTINE
628
629 000034' ROUTINE CLKCHK <DTA>
(2) 000034' CLKCHK:
630
631 ;+
632 ; SAVE REGISTERS ON STACK AND RETRIEVE
633 ; ADDRESS OF DTABLE FROM R5 STACK AND ADDRESS OF SCTAB
634 ; FROM DTABLE
635 ; -
636
637 000034' CALL SAVREG
(3) 000034' 004767 000000G JSR PC,SAVREG
638 000040' LET R2 := DTA(R5) MOV DTA(R5),R2
(4) 000040' 016502 000000 MOV DT.SCT(R2),R3
639 000044' LET R3 := DT.SCT(R2)
(4) 000044' 016203 000066
640
641 ;+
642 ; INITIALIZE SYSTEM CLOCK WORD AND SYSTEM CLOCK ADDRESS WORD
643 ; -
644
645 000050' LET SC.CLO(R3) := #0 CLR SC.CLO(R3)
(4) 000050' 005063 000000
646 000054' LET SC.SCA(R3) := #0 CLR SC.SCA(R3)
(4) 000054' 005063 000012
647
648 ;+
649 ; DETERMINE IF KW11-L OPTION MODULE
650 ; IS PART OF THE RUNTIME EXERCISER
651 ; -
652
653 000060' IF SC.CKL(R3) NE #0 THEN TST SC.CKL(R3)
(6) 000060' 005763 000002 BEQ 50002$
(9) 000064' 001423
654
655 ;+
656 ; GET KW11-L OPTION MODULE HEADER ADDRESS
657 ; -
658
659 000066' LET R0 := @SC.CKL(R3) MOV @SC.CKL(R3),R0
(4) 000066' 017300 000002
660
661 ;+
662 ; GET KW11-L HARDWARE CSR ADDRESS FROM KW11-L
663 ; OPTION MODULE'S HEADER
664 ; -
665
666 000072' LET R1 := ADR(R0) MOV ADR(R0),R1
(4) 000072' 016001 000006
667
668 ;+
669 ; CHECK TO SEE IF KW11-L HARDWARE IS AVAILABLE
670 ; -
671
672 000076' CALL HRDADRCHK IN <R1>
```

```

(3) 000076' 010546
(4) 000100' 010145
(3) 000102' 004767 000000G
(3) 000106' 012605
673
674
675 ;+
676 ; IF KW11L IS AVAILABLE THEN INDICATE KW11-L IS SYSTEM CLOCK
677 ; IN SYSTEM CLOCK WORD. BIT00 SET MEANS KW11-L IS
678 ; SYSTEM CLOCK.
679 ;-
680 000110' IF.NO.ERROR THEN
(6) 000110' 103411
681 000112' LET SC.CLO(R3) := #BIT00
(4) 000112' 012763 000001 000000
682
683 ;+
684 ; SET SYSTEM CLOCK PRESENT BIT IN CONFIGURATION WORD 0 IN DTABLE
685 ; AND SAVE SYSTEM CLOCK ADDRESS
686 ;-
687
688 000120' LET DT.CF0(R2) := DT.CF0(R2) SET.BY #CLKPRES
(6) 000120' 052762 000001 000014
689 000126' LET SC.SCA(R3) := R0
(4) 000126' 010063 000012
690
691 ;+
692 ; RETURN
693 ;-
694
695 000132' INLINE <BR 100$>
(2) 000132' 000461
696
697
698 000134' ENDF
(4) 000134' 50003$:
699 000134' ENDIF
(4) 000134' 50002$:
700
701
702 ;+
703 ; DETERMINE IF KW11-P OPTION MODULE IS
704 ; PART OF THE RUN TIME EXERCISER
705 ;-
706
707 000134' IF SC.CKP(R3) NE #0 THEN
(6) 000134' 005763 000004
(9) 000140' 001441
708
709 ;+
710 ; GET KW11-P OPTION MODULE HEADER ADDRESS
711 ;-
712
713 000142' LET R0 := @SC.CKP(R3)
(4) 000142' 017300 000004
714

```

```

MOV R5,-(SP)
MOV R1,-(R5)
JSR PC,HRDADRCHK
MOV (SP)+,R5

BCS 50003$
MOV #BIT00,SC.CLO(R3)

BIS #CLKPRES,DT.CF0(
MOV R0,SC.SCA(R3)

BR 100$

50003$:
50002$:

TST SC.CKP(R3)
BEQ 50004$

MOV @SC.CKP(R3),R0

```

```

715 ;+
716 ; GET KW11-P HARDWARE CSR ADDRESS
717 ; FROM KW11-P OPTION MODULE'S HEADER
718 ; -
719
720 000146'          LET R1 := ADR(R0)
(4) 000146' 016001 000006          MOV      ADR(R0),R1
721
722 ;+
723 ; CHECK TO SEE IF KW11-P HARDWARE
724 ; IS AVAILABLE
725 ; -
726
727 000152'          CALL HRDADRCHK IN <R1>
(3) 000152' 010546          MOV      R5,-(SP)
(4) 000154' 010145          MOV      R1,-(R5)
(3) 000156' 004767 000000G      JSR      PC,HRDADRCHK
(3) 000162' 012605          MOV      (SP)+,R5
728
729 ;+
730 ; IF ADDRESS WAS AVAILABLE THEN INDICATE KW11-P IS SYSTEM CLOCK
731 ; IN SYSTEM CLOCK WORD
732 ; -
733
734 000164'          IF.NO.ERROR THEN
(6) 000164' 103411          BCS      50005$
735
736 000166'          LET SC.CLD(R3) := #BIT01
(4) 000166' 012763 000002 000000  MOV      #BIT01,SC.CLD(R3)
737
738 ;+
739 ; SET SYSTEM CLOCK PRESENT BIT IN CONFIGURATION WORD 0 IN DTABLE
740 ; AND SAVE SYSTEM CLOCK ADDRESS
741 ; -
742
743 000174'          LET DT.CF0(R2) := DT.CF0(R2) SET.BY #CLKPRES
(6) 000174' 052762 000001 000014  BIS      #CLKPRES,DT.CF0(
744 000202'          LET SC.SCA(R3) := R0
(4) 000202' 010063 000012          MOV      R0,SC.SCA(R3)
745
746 000206'          ELSE
(4) 000206' 000415          BR      50006$
(3) 000210'          50005$:
747
748
749 ;+
750 ; OUTPUT MESSAGE INDICATING NO SYSTEM CLOCK
751 ; -
752
753 000210'          CALL ENQTQ IN <R2,#MSGSTD,#CC.CM1,#0,#C>
(3) 000210' 010546          MOV      R5,-(SP)
(8) 000212' 012745 000000          MOV      #0,-(R5)
(7) 000216' 012745 000000          MOV      #0,-(R5)
(6) 000222' 012745 000000'      MOV      #CC.CM1,-(R5)
(5) 000226' 012745 000000          MOV      #MSGSTD,-(R5)
(4) 000232' 010245          MOV      R2,-(R5)

```

```

(3) 000234' 004767 000000G
(3) 000240' 012605
754 000242'
(4) 000242'
755 000242'
(4) 000242' 000415
(3) 000244'
756
757
758
759
760
761 000244'
(3) 000244' 010546
(8) 000246' 012745 000000
(7) 000252' 012745 000000
(6) 000256' 012745 000000'
(5) 000262' 012745 000000
(4) 000266' 010245
(3) 000270' 004767 000000G
(3) 000274' 012605
762 000276'
(4) 000276'
763
764 000276'
(2) 000276'
765
766
767
768
769 000276'
(3) 000276' 004767 000000G
770
771
772
773
774
775 000302'
(3) 000302'
(3) 000302'
(2) 000302' 000207
776
777

```

```

                ENDIF
ELSE
                ;+
                ; OUTPUT MESSAGE INDICATING SYSTEM CLOCK IS NOT AVAILABLE
                ;-
                CALL ENQTQ IN <R2,#0,#CC.CM1,#0,#0>
                MOV     R5,-(SP)
                MOV     #0,-(R5)
                MOV     #0,-(R5)
                MOV     #CC.CM1,-(R5)
                MOV     #0,-(R5)
                MOV     R2,-(R5)
                JSR     PC,ENQTQ
                MOV     (SP)+,R5
ENDIF
                50006$:
                BR     50007$
                50004$:
                50007$:
                100$:
                CALL RESREG
                JSR     PC,RESREG
                ;+
                ; RETURN TO CALLER
                ;-
                ENDRTN
                50000$:
                50001$:
                RTS     PC

```

```
779 .SBTTL CLKOFF - SYSTEM CLOCK-OFF
780 .IDENT /V0.0/
781
782 ;++
783 ; MODULE NAME:
784 ;     CLKOFF
785 ;
786 ; FUNCTIONAL DESCRIPTION:
787 ;     THIS MODULE WILL SHUT OFF THE SYSTEM CLOCK (IF AVAILABLE). THE ROUTINE
788 ;     WILL SAVE THE CLOCK'S CONTROL STATUS REGISTER (CSR) ADDRESS AND IT'S
789 ;     CONTENTS. PLUS TURN THE CLOCK OFF BY CLEARING THE INTERRUPT ENABLE BIT.
790 ;
791 ; INPUTS:
792 ;     DTABLE - ADDRESS OF DATA TABLE
793 ;
794 ; IMPLICIT INPUTS:
795 ;     DT.SCT - ADDRESS OF SYSTEM CLOCK TABLE
796 ;     DT.CLO - SYSTEM CLOCK WORD
797 ;     SC.CKL - POINTER TO KW11L OPTION MODULE
798 ;     SC.CKP - POINTER TO KW11P OPTION MODULE
799 ;
800 ; OUTPUTS:
801 ;     NONE
802 ;
803 ; IMPLICIT OUTPUTS:
804 ;     SC.ADR - SAVE LOCATION FOR CLOCK CSR ADDRESS
805 ;     SC.HLD - SAVE LOCATION FOR CONTENTS OF CLOCK'S
806 ;             CSR
807 ;
808 ; PATHOLOGICAL CONNECTIONS:
809 ;     NONE
810 ;
811 ; SUBORDINATE ROUTINES CALLED:
812 ;     NONE
813 ;
814 ; FUNCTIONAL SIDE EFFECTS:
815 ;     NONE
816 ;
817 ; CALLING SEQUENCE:
818 ;     CALL CLKOFF IN <A>
819 ;             A= ADDRESS OF DTABLE
820 ;
821 ;     EDIT                DATE                BY                REASON
822 ;
823 ;
824 ;
```



SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 19-9  
COMMON DEFINITIONS AND REFERENCES

SEQ 0962

826  
827  
828  
829  
830  
831

.SBTTL COMMON DEFINITIONS AND REFERENCES

\*\*\*\*\*

; REFERENCED BY OTHER MODULES

;

.GLOBL CLKOFF ;MODULE ENTRY POINT

```
833 .SBTTL CLKOFF ROUTINE
834
835
836 000304' ROUTINE CLKOFF <DTA>
(2) 000304' CLKOFF:
837
838
839 ;+
840 ; SAVE REGISTERS ON STACK AND RETRIEVE ADDRESS OF DTABLE AND
841 ; GET ADDRESS OF SYSTEM CLOCK TABLE (SCTAB)
842 ;-
843
844 000304' PUSH R0,R1
(2) 000304' 010046 MOV R0,-(SP)
(3) 000306' 010146 MOV R1,-(SP)
845 000310' LET R1 := DTA(R5)
(4) 000310' 016501 000000 MOV DTA(R5),R1
846 000314' LET R1 := DT.SCT(R1)
(4) 000314' 016101 000066 MOV DT.SCT(R1),R1
847
848
849 ;+
850 ; IF KW11-L IS SYSTEM CLOCK THEN GET THE KW11-L
851 ; OPTION MODULE HEADER ADDRESS
852 ;-
853
854 000320' IF #BIT00 SETIN SC.CLO(R1) THEN
(6) 000320' 032761 000001 000000 BIT #BIT00,SC.CLO(R1)
(9) 000326' 001403 BEQ 50002$
855 000330' LET R0 := @SC.CKL(R1)
(4) 000330' 017100 000002 MOV @SC.CKL(R1),R0
856
857 000334' ELSE
(4) 000334' 000402 BR 50003$
(3) 000336' 50002$:
858
859 ;+
860 ; KW11-P IS SYSTEM CLOCK , GET THE KW11-P
861 ; OPTION MODULE HEADER ADDRESS
862 ;-
863
864 000336' LET R0 := @SC.CKP(R1)
(4) 000336' 017100 000004 MOV @SC.CKP(R1),R0
865 000342' ENDIF
(4) 000342' 50003$:
866
867 ;+
868 ; GET SYSTEM CLOCK CSR ADDRESS AND SAVE IN
869 ; SC.ADR. R0 CONTAINS THE OPTION
870 ; MODULE HEADER ADDRESS.
871 ;-
872
873 000342' LET SC.ADR(R1) := ADR(R0)
(4) 000342' 016061 000006 000006 MOV ADR(R0),SC.ADR(R)
874
875 ;+
```

```
876 ; GET CONTENTS OF SYSTEM CLOCK CSR AND SAVE
877 ; IN SC.HLD
878 ;-
879
880 000350' LET SC.HLD(R1) := @SC.ADR(R1)
(4) 000350' 017161 000006 000010 MOV @SC.ADR(R1),SC.H
881 ;+
882 ; TURN SYSTEM CLOCK OFF BY CLEARING
883 ; IT'S CSR
884 ;-
885
886 000356' LET @SC.ADR(R1) := #0
(4) 000356' 005071 000006 CLR @SC.ADR(R1)
887 ;+
888 ; RESTORE REGISTERS
889 ;-
890
891 000362' POP R1,R0
(2) 000362' 012601 MOV (SP)+,R1
(3) 000364' 012600 MOV (SP)+,R0
892
893 ;+
894 ; RETURN TO CALLING MODULE
895 ;-
896
897 000366' ENDRTN
(3) 000366'
(3) 000366'
(2) 000366' 000207 50000$:
50001$: RTS PC
898
899
900
901
```



SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 19-13  
COMMON DEFINITIONS AND REFERENCES

SEQ 0966

951  
952  
953  
954  
955  
956  
957  
958  
959

.SBTTL COMMON DEFINITIONS AND REFERENCES

\*\*\*\*\*

; REFERENCED BY OTHER MODULES

;

.GLOBL CLKON ;MODULE ENTRY POINT

```
961 .SBTTL CLKON ROUTINE
962
963
964 000370' ROUTINE CLKON <DTA>
(2) 000370' CLKON:
965
966
967
968 ;+
969 ; SAVE R0 AND RETRIEVE DTABLE FROM R5 AND GET SCTAB INTO R0
970 ; GET THE SYSTEM CLOCK CSR ADDRESS AND
971 ; RESTORE ITS CONTENTS, THIS WILL TURN THE
972 ; CLOCK ON. R0 CONTAINS THE ADDRESS OF SCTAB.
973 ;-
974
975 000370' PUSH R0
(2) 000370' 010046 MOV R0,-(SP)
976 000372' LET R0 := DTA(R5) MOV DTA(R5),R0
(4) 000372' 016500 000000 MOV DTA(R5),R0
977 000376' LET R0 := DT.SCT(R0) MOV DT.SCT(R0),R0
(4) 000376' 016000 000066 MOV DT.SCT(R0),R0
978 000402' LET @SC.ADR(R0) := SC.HLD(R0) MOV SC.HLD(R0),@SC.A
(4) 000402' 016070 000010 000006
979
980
981 ;+
982 ; RESTORE REGISTER AND RETURN TO CALLER
983 ;-
984
985 000410' POP R0
(2) 000410' 012600 MOV (SP)+,R0
986
987 000412' ENDRTN
(3) 000412' 50000$:
(3) 000412' 50001$:
(2) 000412' 000207 RTS PC
988
```

```
990 .SBTTL PRHMS PROCESS HOURS, MINUTES, AND SECONDS CONVERSION
991 .IDENT /V0.0/
992
993 ;++
994 ; MODULE NAME:
995 ; PRHMS
996 ;
997 ; FUNCTIONAL DESCRIPTION:
998 ; THIS MODULE WHEN CALLED WILL GET THE TOTAL RUNTIME IN SECONDS
999 ; FROM THE SYSTEM CLOCK OPTION MODULE (KW11-L OR KW11-P) HAVE IT
1000 ; CONVERTED TO HOURS, MINUTES AND SECONDS, AND RETURNS
1001 ; THE CONVERTED TIME TO THE CALLER.
1002 ;
1003 ; INPUTS:
1004 ; ADDRESS OF DTABLE
1005 ; ADDRESS OF CONVERTED TIME STORAGE
1006 ;
1007 ; IMPLICIT INPUTS:
1008 ; DT.SCT
1009 ; SC.CLO
1010 ; SC.CKL
1011 ; SC.CKP
1012 ;
1013 ; OUTPUTS:
1014 ; NONE
1015 ;
1016 ; IMPLICIT OUTPUTS:
1017 ; HOURS \
1018 ; MINUTES \ ---IN STORAGE SPECIFIED
1019 ; SECONDS /
1020 ;
1021 ; PATHOLOGICAL CONNECTIONS:
1022 ; NONE
1023 ;
1024 ; SUBORDINATE ROUTINES CALLED:
1025 ; SAVREG - SAVE REGISTERS
1026 ; HMS - CONVERSION TO HRS, MINS AND SECS
1027 ; RESREG - RESTORE REGISTERS
1028 ; LDTIME - LOAD RESULTANT TIME
1029 ;
1030 ; FUNCTIONAL SIDE EFFECTS:
1031 ; NONE
1032 ;
1033 ; CALLING SEQUENCE:
1034 ; CALL PRHMS IN <A,B>
1035 ; A = ADDRESS OF DTABLE
1036 ; B = ADDRESS OF CONVERTED TIME
1037 ;
1038 ; VERSION:
1039 ; 0.0
1040 ;
1041 ; EDIT DATE BY REASON
1042 ;--
```

```
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
```



```

1059
1060
1061
1062
1063 000414'
(2) 000414'
1064
1065
1066
1067
1068
1069
1070
1071
1072 000414'
(2) 000414' 010046
(3) 000416' 010146
1073 000420'
(4) 000420' 016500 000000
1074 000424'
(4) 000424' 016000 000066
1075
1076
1077
1078
1079
1080
1081 000430'
(6) 000430' 032760 000001 000000
(9) 000436' 001405
1082
1083
1084
1085
1086
1087
1088
1089 000440'
(4) 000440' 016001 000002
(6) 000444' 062701 000002
1090
1091
1092 000450'
(4) 000450' 000404
(3) 000452'
1093
1094
1095
1096
1097
1098
1099
1100 000452'
(4) 000452' 016001 000004
(6) 000456' 062701 000002
1101

```

```

.SBTTL PROCESS HOURS, MINUTES AND SECONDS CONVERSION ROUTINE

ROUTINE PRHMS <DTA,RESULT>
                                PRHMS:

;+
; SAVE REGISTERS ON STACK, RETRIEVE ADDRESS OF DTABLE FROM
; R5 STACK AND DETERMINE ADDRESS OF SCTAB.
;-

PUSH R0,R1
                                MOV     R0,-(SP)
                                MOV     R1,-(SP)

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

LET R0 := DT.SCT(R0)
                                MOV     DT.SCT(R0),R0

;+
; 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
                                BIT     #BIT00,SC.CLO(R0)
                                BEQ     50002$

;+
; 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
                                MOV     SC.CKL(R0),R1
                                ADD     #2,R1

ELSE
                                BR      50003$
                                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
                                MOV     SC.CKP(R0),R1
                                ADD     #2,R1

```

```

1102
1103 000462'
(4) 000462'
1104
1105
1106 ;+
1107 ; CALL HMS TO CONVERT TIME TO DECIMAL ASCII HOURS.
1108 ; MINUTES AND SECONDS.
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

```

```

1129 .SBTTL LDTIME - LOAD CONVERTED TIME
1130
1131 000534' ROUTINE LDTIME <ADR,RESULT>
(2) 000534'
1132
1133 ;+
1134 ; GET CALLING MODULE CONVERTED TIME STORAGE ADDRESS, THEN LOAD UP THE
1135 ; STORAGE WITH THE CONVERTED TIME AND INCLUDE COLONS(:) IN APPROPRIATE
1136 ; LOCATIONS. ADR(R5) WILL CONTAIN THE STORAGE ADDRESS WHILE RESULT(R5) CONTAINS
1137 ; THE RESULT ADDRESS
1138 ; -
1139 000534' PUSH R0,R1
(2) 000534' 010046 MOV R0,-(SP)
(3) 000536' 010146 MOV R1,-(SP)
1140 000540' LET R0 := RESULT(R5) MOV RESULT(R5),R0
(4) 000540' 016500 000002
1141 000544' LET R1 := ADR(R5) MOV ADR(R5),R1
(4) 000544' 016501 000000
1142 000550' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000550' 112120
1143 000552' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000552' 112120
1144 000554' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000554' 112120
1145 000556' LET (R0)+ :B= #' : MOV #' : ,(R0)+
(4) 000556' 112720 000072
1146 000562' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000562' 112120
1147 000564' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000564' 112120
1148 000566' LET (R0)+ :B= #' : MOV #' : ,(R0)+
(4) 000566' 112720 000072
1149 000572' LET (R0)+ :B= (R1)+ MOV (R1)+,(R0)+
(4) 000572' 112120
1150 000574' LET (R0) :B= (R1) MOV (R1),(R0)
(4) 000574' 111110
1151 000576' POP R1,R0 MOV (SP)+,R1
(2) 000576' 012601 MOV (SP)+,R0
(3) 000600' 012600
1152 000602' ENDRTN
(3) 000602'
(3) 000602'
(2) 000602' 000207

```

```

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 SUBTRACTING 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
;
```

SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 19-21  
PRPSCNT PROCESS PASS COUNT TIME

SEQ 0974

1210  
1211  
1212  
1213  
1214

```
; VERSION:  
; 0.0  
;  
; EDIT DATE BY REASON  
;--
```

SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 19-22  
PRPSCNT PROCESS PASS COUNT TIME

SEQ 0975

```
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230
```

```
.SBTTL COMMON DEFINITIONS AND REFERENCES  
  
;*****  
; REFERENCED BY OTHER MODULES  
;  
.GLOBL PRPSCNT ;MODULE ENTRY POINT  
  
;*****  
; GLOBAL REFERENCES  
;  
.GLOBL HMS ;HOURS MINUTES AND SECONDS CONVERSION  
.GLOBL SAVREG ;SAVE REGISTERS  
.GLOBL RESREG ;RESTORE REGISTERS
```

```
1232
1233
1234 .SBTTL PROCESS PASS COUNT TIME ROUTINE
1235
1236 000604' ROUTINE PRPSCNT <DTA,BGNADR,RESULT>
(2) 000604' PRPSCNT:
1237
1238
1239 ;+
1240 ; SAVE REGISTERS, RETRIEVE ADDRESS OF DTABLE FROM R5 STACK, THEN
1241 ; DETERMINE PTR TO MODULE LIST AND ADDRESS OF SCTAB
1242 ; -
1243
1244 000604' CALL SAVREG
(3) 000604' 004767 000000G JSR PC,SAVREG
1245
1246 000610' LET R1 := DTA(R5)
(4) 000610' 016501 000000 MOV DTA(R5),R1
1247 000614' LET R0 := DT.MLST(R1)
(4) 000614' 016100 000032 MOV DT.MLST(R1),R0
1248 000620' LET R1 := DT.SCT(R1)
(4) 000620' 016101 000066 MOV DT.SCT(R1),R1
1249
1250 ;+
1251 ; DETERMINE IF KW11-L IS SYSTEM CLOCK BY CHECKING IF BIT 0 IS SET IN SC.CLO
1252 ; WORD OF SCTAB
1253 ; -
1254
1255 000624' IF #BIT00 SETIN SC.CLO(R1) THEN
(6) 000624' 032761 000001 000000 BIT #BIT00,SC.CLO(R1)
(9) 000632' 001403 BEQ 50002$
1256
1257 ;+
1258 ; GET ADDRESS OF KW11-L ELAPSED TIME TABLE
1259 ; -
1260
1261 000634' LET R1 := SC.CKL(R1)
(4) 000634' 016101 000002 MOV SC.CKL(R1),R1
1262 000640' ELSE
(4) 000640' 000402 BR 50003$
(3) 000642' 50002$:
1263
1264 ;+
1265 ; MUST BE A KW11-P CLOCK GET ADDRESS OF KW11-P ELAPSED TIME
1266 ; TABLE
1267 ; -
1268
1269
1270 000642' LET R1 := SC.CKP(R1)
(4) 000642' 016101 000004 MOV SC.CKP(R1),R1
1271 000646' ENDF
(4) 000646' 50003$:
1272
1273 ;+
1274 ; GET R2 TO POINT TO MODULE TIME TABLE AND R1 TO POINT TO
1275 ; R.T.E ELAPSED TIME
```

```

1276      ; -
1277
1278      000646'      LET R2 := R1 + #6
      (4) 000646' 010102      MOV      R1,R2
      (6) 000650' 062702 000006      ADD      #6,R2
1279      000654'      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'
      (6) 000664' 021003      50004$:      CMP      (R0),R3
      (9) 000666' 001405      BEQ      50005$
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
; --
```

```
1385 .SBTTL COMMON DEFINITIONS AND REFERENCES
1386
1387 ;*****
1388 ; REFERENCED BY OTHER MODULES
1389 ;
1390 .GLOBL HMS ;MODULE ENTRY POINT
1391
1392 ;*****
1393 ; GLOBAL REFERENCES
1394 ;
1395 .GLOBL SAVREG ;SAVE REGISTERS
1396 .GLOBL BDACNV ;BINARY TO DECIMAL ASCII CONVERSION
1397 .GLOBL RESREG ;RESTORE REGISTERS
1398
1399 ;*****
1400 ; LOCAL STORAGE
1401 ;
1402 HM.CTT : ;CONVERTED TIME TABLE
1403 HM.HRS :.BLKB 3 ; HOURS
1404 HM.MIN :.BLKB 2 ; MINUTES
1405 HM.SEC :.BLKB 2 ; SECONDS
1406 .EVEN
1407 HM.CTE : ;END OF CONVERTED TIME TABLE
1408 HM.THR :.WORD 0 ;TOTAL TIME IN HOURS STORAGE
1409 HM.TMI :.WORD 0 ;TOTAL TIME IN MINUTES STORAGE
1410 HM.TSE :.WORD 0 ;TOTAL TIME IN SECONDS STORAGE
1411
1412 HM.CNV :.BLKB 5 ;STORAGE FOR BDACNV CALL
1413 .EVEN
1414
```

```

1416
1417
1418
1419
1420 001004'
1421 (2) 001004'
1422
1423
1424
1425
1426
1427 001004'
1428 (3) 001004' 004767 000000G
1429 001010'
1430 (4) 001010' 016500 000000
1431 001014'
1432 (4) 001014' 016501 000002
1433 001020'
1434 (4) 001020' 005067 177744
1435 001024'
1436 (4) 001024' 005067 177742
1437 001030'
1438 (4) 001030' 005067 177740
1439
1440
1441
1442
1443
1444
1445
1446 001040'
1447 (6) 001040' 020027 007020
1448 (9) 001044' 103007
1449 001046'
1450 (6) 001046' 062767 000022 177714
1451 001054'
1452 (6) 001054' 062700 001340
1453 001060'
1454 (6) 001060' 005301
1455 001062'
1456 (4) 001062' 000404
1457 (3) 001064'
1458 001064'
1459 (6) 001064' 005267 177700
1460 001070'
1461 (6) 001070' 162700 007020

```

.SBTTL HOURS, MINUTES AND SECONDS CONVERSION ROUTINE

ROUTINE HMS <TIME,XTIME,ADRCT>

HMS:

```

;+
; SAVE REGISTERS ON STACK, RETRIEVE TIME (R0) AND EXTENDED TIME (R1) FROM STACK,
; INITIALIZE LOCAL STORAGE
;-
CALL SAVREG
JSR PC,SAVREG
LET R0 := TIME(R5)
MOV TIME(R5),R0
LET R1 := XTIME(R5)
MOV XTIME(R5),R1
LET HM.THR := #0
CLR HM.THR
LET HM.TMI := #0
CLR HM.TMI
LET HM.TSE := #0
CLR HM.TSE
;+
; PROCESS (EXTENDED TIME BITS) INTO HOURS
;-
WHILE R1 NE #0 DO
50002$:
TST R1
BEQ 50003$
;+
; 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)
;-
IF R0 LO #^D3600 THEN
CMP R0,#^D3600
BHS 50004$
LET HM.THR := HM.THR + #^D18
ADD #^D18,HM.THR
LET R0 := R0 + #^D736
ADD #^D736,R0
LET R1 := R1 - #1
DEC R1
ELSE
BR 50005$
50004$:
LET HM.THR := HM.THR + #1
INC HM.THR
LET R0 := R0 - #^D3600
SUB #^D3600,R0

```

1453	001074'		ENDIF		
(4)	001074'				50005\$:
1454	001074'		ENDDO		
(4)	001074'	000757			BR 50002\$
(3)	001076'				50003\$:
1455					
1456			;		
1457			; PROCESS REMAINING HOURS CONTAINED IN R0 (TIME).		
1458			;		
1459					
1460	001076'		WHILE R0 HIS #^D3600 DO		50006\$:
(4)	001076'				
(6)	001076'	020027 007020			CMP R0,#^D3600
(9)	001102'	103405			BLO 50007\$
1461	001104'		LET HM.THR := HM.THR + #1		INC HM.THR
(6)	001104'	005267 177660			
1462	001110'		LET R0 := R0 - #^D3600		SUB #^D3600,R0
(6)	001110'	162700 007020			
1463	001114'		ENDDO		BR 50006\$
(4)	001114'	000770			50007\$:
(3)	001116'				
1464					
1465			;		
1466			; PROCESS MINUTES FROM REMAINING R0 (TIME)		
1467			;		
1468					
1469	001116'		WHILE R0 HIS #^D60 DO		50010\$:
(4)	001116'				
(6)	001116'	020027 000074			CMP R0,#^D60
(9)	001122'	103405			BLO 50011\$
1470	001124'		LET HM.TMI := HM.TMI + #1		INC HM.TMI
(6)	001124'	005267 177642			
1471	001130'		LET R0 := R0 - #^D60		SUB #^D60,R0
(6)	001130'	162700 000074			
1472	001134'		ENDDO		BR 50010\$
(4)	001134'	000770			50011\$:
(3)	001136'				
1473					
1474			;		
1475			; TIME LEFT STICK IN HM.TSE		
1476			;		
1477					
1478	001136'		LET HM.TSE := R0		MOV R0, HM.TSE
(4)	001136'	010067 177632			
1479					
1480			;		
1481			; CONVERT HOURS TO DECIMAL ASCII		
1482			;		
1483					
1484	001142'		CALL BDACNV IN <HM.THR,#HM.CNV>		MOV R5,-(SP)
(3)	001142'	010546			MOV #HM.CNV,-(R5)
(5)	001144'	012745 000776'			MOV HM.THR,-(R5)
(4)	001150'	016745 177614			JSR PC,BDACNV
(3)	001154'	004767 000000G			MOV (SP)+,R5
(3)	001160'	012605			
1485					

```

1486
1487 ;+
1488 ; INITIALIZE R2 TO POINT TO HM.CNV TABLE + 2 AND R3 TO POINT TO HM.HRS.
1489 ; THEN MOVE LOWEST THREE BYTES IN HM.CNV TABLE INTO HM.HRS STORAGE IN HM.CTT.
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

```

```
1521 ; TO HM.SEC. MOVE 2 ENTRIES IN HM.CNV TABLE INTO HM.SEC
1522 ; STORAGE IN HM.CTT
1523 ;-
1524
1525 001264' LET R2 := #HM.CNV + #3
(4) 001264' 012702 000776' MOV #HM.CNV,R2
(6) 001270' 062702 000003 ADD #3,R2
1526 001274' LET R3 := #HM.SEC
(4) 001274' 012703 000765' MOV #HM.SEC,R3
1527 001300' LET (R3)+ :B= (R2)+
(4) 001300' 112223 MOVB (R2)+,(R3)+
1528 001302' LET (R3)+ :B= (R2)+
(4) 001302' 112223 MOVB (R2)+,(R3)+
1529
1530 ;+
1531 ; LOAD HM.CTT ADDRESS (CONVERTED TIME TABLE) ON R5 STACK FOR
1532 ; CALLER
1533 ;-
1534
1535 001304' LET ADRCT(R5) := #HM.CTT
(4) 001304' 012765 000760' 000004 MOV #HM.CTT,ADRCT(R5)
1536
1537 ;+
1538 ; RESTORE REGISTERS AND RETURN TO CALLER
1539 ;-
1540
1541 001312' CALL RESREG
(3) 001312' 004767 000000G JSR PC,RESREG
1542
1543 001316' ENDRTN
(3) 001316' 50000$:
(3) 001316' 50001$:
(2) 001316' 000207 RTS PC
1544
1545
```

1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598

```
.SBTTL UPDTIM UPDATE MODULE PASS TIME
.IDENT /V0.0/

; ++
; MODULE NAME:
;   UPDTIM
;
; FUNCTIONAL DESCRIPTION:
;   THIS MODULE WILL ONLY BE CALLED IF THERE IS A SYSTEM CLOCK AVAILABLE
;   (KW11-L OR KW11-P OPTION MODULE). UPON BEING CALLED IT WILL SEARCH THE MODULE
;   LIST UNTIL THE MODULE IS FOUND THAT REQUIRES THE MODULE TIME TO BE UPDATED.
;   THE UPDATE IS DONE BY MOVING THE TOTAL ELAPSED TIME AND EXTENDED TIME
;   INTO THE APPROPRIATE STORAGE LOCATIONS (CONTAINED IN SYSTEM CLOCK MODULE;
;   KW11-L OR KW11-P) FOR THE MODULE BEING UPDATED.
;
; INPUTS:
;   DTABLE ADDRESS
;   MODULE BEGIN ADDRESS TO BE UPDATED
;
; IMPLICIT INPUTS:
;   DT.MLST  OFFSET TO MODULE LIST ADDR. IN DTABLE
;   DT.SCT   OFFSET TO SCTAB ADDR. IN DTABLE
;   SC.CLO
;   SC.CKL
;   SC.CKP
;
; OUTPUTS:
;   NONE
;
; IMPLICIT OUTPUTS:
;   NONE
;
; PATHOLOGICAL CONNECTIONS:
;   NONE
;
; SUBORDINATE ROUTINES CALLED:
;   SAVREG - SAVE REGISTERS
;   RESREG - RESTORE REGISTERS
;
; FUNCTIONAL SIDE EFFECTS:
;   NONE
;
; CALLING SEQUENCE:
;   CALL UPDTIM IN <A,B>
;           A - DTABLE ADDRESS
;           B - MODULE BEGIN ADDR. TO BE UPDATED
;
; VERSION:
;   0.0
;
;   EDIT          DATE          BY          REASON
; --
```



SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 20-7  
UPDTIM UPDATE MODULE PASS TIME

SEQ 0986

1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613

```
.SBTTL COMMON DEFINITIONS AND REFERENCES  
;*****  
; REFERENCED BY OTHER MODULES  
;  
.GLOBL UPDTIM ;MODULE ENTRY POINT  
;*****  
; GLOBAL REFERENCES  
;  
.GLOBL SAVREG ;SAVE REGISTERS  
.GLOBL RESREG ;RESTORE REGISTERS
```

```

1615
1616      .SBTTL UPDATE TIME ROUTINE
1617
1618      001320'      ROUTINE UPDTIME <DTA,UPDMOD>
1619      (2) 001320'
1620
1621      ;+
1622      ; SAVE REGISTERS, GET POINTERS TO MODULE LIST, R.T.E TIME AND
1623      ; MODULE TIME TABLE
1624      ; -
1625
1626      001320'      CALL SAVREG
1627      (3) 001320' 004767 000000G      JSR      PC, SAVREG
1628      001324'      LET R0 := DTA(R5)
1629      (4) 001324' 016500 000000      MOV      DTA(R5), R0
1630      001330'      LET R1 := DT.MLST(R0)
1631      (4) 001330' 016001 000032      MOV      DT.MLST(R0), R1
1632      001334'      LET R2 := DT.SCT(R0)
1633      (4) 001334' 016002 000066      MOV      DT.SCT(R0), R2
1634
1635      ;+
1636      ; DETERMINE IF SYSTEM CLOCK IS KW11L OR KW11P AND
1637      ; GET ADDRESS WITHIN MODULE (CLOCK)
1638      ; -
1639      001340'      IF #BIT00 SETIN SC.CLO(R2) THEN
1640      (6) 001340' 032762 000001 000000      BIT      #BIT00, SC.CLO(R2)
1641      (9) 001346' 001403      BEQ      50002$
1642      001350'      LET R2 := SC.CKL(R2)
1643      (4) 001350' 016202 000002      MOV      SC.CKL(R2), R2
1644      001354'      ELSE
1645      (4) 001354' 000402      BR      50003$
1646      (3) 001356'      50002$:
1647      001356'      LET R2 := SC.CKP(R2)
1648      (4) 001356' 016202 000004      MOV      SC.CKP(R2), R2
1649      001362'      ENDIF
1650      (4) 001362'      50003$:
1651
1652      ;+
1653      ; GET R3 TO POINT TO MODULE TIME TABLE AND R2
1654      ; TO POINT TO R.T.E. ELAPSED TIME
1655      ; -
1656      001362'      LET R3 := R2 + #6
1657      (4) 001362' 010203      MOV      R2, R3
1658      (6) 001364' 062703 000006      ADD      #6, R3
1659      001370'      LET R2 := R2 + #2
1660      (6) 001370' 062702 000002      ADD      #2, R2
1661
1662      ;+
1663      ; GET ADDRESS OF MODULE TO BE UPDATED FROM R5 STACK
1664      ; -
1665      001374'      LET R0 := UPDMOD(R5)
1666      (4) 001374' 016500 000002      MOV      UPDMOD(R5), R0
1667
1668
1669
1670

```

```
1655 ;+
1656 ; SEARCH MODULE LIST UNTIL MODULE LIST POINTER POINTS TO ADDRESS
1657 ; IN R0
1658 ;-
1659
1660 001400' WHILE (R1) NE R0 DO
(4) 001400' 50004$:
(6) 001400' 021100 CMP (R1),R0
(9) 001402' 001405 BEQ 50005$
1661 001404' LET R1 := R1 + #2 ADD #2,R1
(6) 001404' 062701 000002 LET R3 := R3 + #4 ADD #4,R3
1662 001410' ENDDO BR 50004$
(6) 001410' 062703 000004
1663 001414'
(4) 001414' 000771 50005$:
(3) 001416'
1664
1665 ;+
1666 ; HAVE FOUND MODULE UPDATE MODULE TIME TO REFLECT
1667 ; R.T.E ELAPSED TIME
1668 ;-
1669
1670 001416' LET (R3) := (R2) MOV (R2),(R3)
(4) 001416' 011213 LET 2(R3) := 2(R2) MOV 2(R2),2(R3)
1671 001420'
(4) 001420' 016263 000002 000002
1672
1673 ;+
1674 ; RESTORE STACK AND RETURN
1675 ;-
1676
1677 001426' CALL RESREG JSR PC,RESREG
(3) 001426' 004767 000000G
1678
1679 001432' ENDRTN
(3) 001432' 50000$:
(3) 001432' 50001$:
(2) 001432' 000207 RTS PC
1680
1681
```

1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727

```
.SBTTL CLRTIM CLEAR TIME FROM SYSTEM CLOCK TIME TABLES
.IDENT /V0.0/

; ++
; MODULE NAME:
;     CLRTIM
;
; FUNCTIONAL DESCRIPTION:
;     THIS MODULE WILL CALL THE APPROPRIATE ROUTINE CONTAINED IN THE SYSTEM
;     CLOCK MODULE (KW11L OR KW11P OPTION MODULE) WHICH WILL CLEAR OUT THE R.T.E
;     AND EXTENDED ELAPSED TIME WORDS AND ALSO THE MODULE TIME TABLE.
;
; INPUTS:
;     DTABLE ADDRESS
;
; IMPLICIT INPUTS:
;     DT.SCT - SYSTEM CLOCK TABLE ADDRESS
;     SC.CLO
;     SC.ALC
;     SC.APC
;
; OUTPUTS:
;     NONE
;
; IMPLICIT OUTPUTS:
;     NONE
;
; PATHOLOGICAL CONNECTIONS:
;     NONE
;
; SUBORDINATE ROUTINES CALLED:
;     NONE
;
; FUNCTIONAL SIDE EFFECTS:
;     NONE
;
; CALLING SEQUENCE:
;     CALL CLRTIM IN <A>
;                                     A - DTABLE ADDRESS
;
; VERSION:
;     0.0
;
;     EDIT           DATE           BY           REASON
; ---
```

SYSCLK SYSTEM CLOCK BINDER MODULE  
SYSCLK.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:36 PAGE 20-11  
CLRTIM CLEAR TIME FROM SYSTEM CLOCK TIME TABLES

SEQ 0990

1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737

.SBTTL COMMON DEFINITIONS AND REFERENCES

\*\*\*\*\*

; REFERENCED BY OTHER MODULES

;

.GLOBL CLRTIM

;MODULE ENTRY POINT

```

1739
1740      .SBTTL CLEAR TIME ROUTINE
1741
1742      001434'      ROUTINE CLRRTIM <DTA>
1743      (2) 001434'
1744
1745      ;+
1746      ; SAVE REGISTER, GET ADDRESS OF SCTAB
1747      ; -
1748      001434'      PUSH R0
1749      (2) 001434' 010046
1750      001436'      LET R0 := DTA(R5)
1751      (4) 001436' 016500 000000
1752      001442'      LET R0 := DT.SCT(R0)
1753      (4) 001442' 016000 000066
1754
1755      ;+
1756      ; DETERMINE IF KW11L IS SYSTEM CLOCK, IF IT IS
1757      ; GO CLEAR KW11L CLOCK TIME TABLES.
1758      ; -
1759      001446'      IF #BIT00 SETIN SC.CLO(R0) THEN
1760      (6) 001446' 032760 000001 000000
1761      (9) 001454' 001403
1762
1763      CALL @SC.ALC(R0)
1764      (3) 001456' 004770 000014
1765
1766      ELSE
1767      (4) 001462' 000402
1768      (3) 001464'
1769
1770      ;+
1771      ; MUST BE KW11P. GO CLEAR KW11P TIME TABLES
1772      ; -
1773      001464'      CALL @SC.APC(R0)
1774      (3) 001464' 004770 000016
1775
1776      ENDIF
1777
1778      ;+
1779      ; RESTORE REGISTER AND RETURN
1780      ; -
1781      001470'      POP R0
1782      (2) 001470' 012600
1783
1784      ENDRTN
1785      (3) 001472'
1786      (3) 001472'
1787      (2) 001472' 000207
1788
1789
1790

```

CLRRTIM:

```

MOV      R0,-(SP)
MOV      DTA(R5),R0
MOV      DT.SCT(R0),R0
BIT      #BIT00,SC.CLO(R0)
BEQ      50002$
JSR      PC,@SC.ALC(R0)
BR       50003$
50002$:
JSR      PC,@SC.APC(R0)
50003$:
MOV      (SP)+,R0
50000$:
50001$:
RTS      PC

```

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  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853

001474' 000000

.SBTTL COMMON DEFINITIONS AND REFERENCES

\*\*\*\*\*

; REFERENCED BY OTHER MODULES

.GLOBL CKHUNG ;MODULE ENTRY POINT

\*\*\*\*\*

; GLOBAL REFERENCES

.GLOBL SAVREG ;SAVE REGISTERS

.GLOBL ENQTQ ;ENQUEUE TYPE QUEUE ROUTINE

.GLOBL RESREG ;RESTORE REGISTERS

.GLOBL DUNCHK ;CHECK TO SEE IF ALL MODULES ARE DROPPED

.GLOBL DTABLE ;ADDRESS OF DATA TABLE

\*\*\*\*\*

; LOCAL STORAGE

CK.TEMP: .WORD 0 ;TEMP STORAGE FOR TIME COMPARSIONS



```
1855
1856 .SBTTL CHECK MODULE HUNG ROUTINE
1857
1858
1859 001476' ROUTINE CKHUNG
(2) 001476' CKHUNG:
1860
1861
1862 ;+
1863 ; SAVE REGISTERS, GET ADDRESSO OF DTABLE AND SCTAB
1864 ; -
1865
1866 001476' CALL SAVREG
(3) 001476' 004767 000000G JSR PC, SAVREG
1867 001502' LET R0 := #DTABLE MOV #DTABLE, R0
(4) 001502' 012700 000000G
1868 001506' LET R1 := DT.SCT(R0) MOV DT.SCT(R0), R1
(4) 001506' 016001 000066
1869
1870
1871
1872 ;+
1873 ; DETERMINE IF KW11L IS SYSTEM CLOCK
1874 ; -
1875
1876 001512' IF #BIT00 SETIN SC.CLO(R1) THEN
(6) 001512' 032761 000001 000000 BIT #BIT00, SC.CLO(R1)
(9) 001520' 001403 BEQ 50002$
1877
1878 ;+
1879 ; SET R1 EQUAL TO KW11L LABEL IN OPTION MODULE
1880 ; -
1881
1882 001522' LET R1 := SC.CKL(R1)
(4) 001522' 016101 000002 MOV SC.CKL(R1), R1
1883
1884 ;+
1885 ; SET R1 EQUAL TO KW11P LABEL IN OPTION MODULE
1886 ; -
1887
1888 001526' ELSE
(4) 001526' 000402 BR 50003$
(3) 001530' 50002$:
1889 001530' LET R1 := SC.CKP(R1) MOV SC.CKP(R1), R1
(4) 001530' 016101 000004
1890
1891 001534' ENDIF
(4) 001534' 50003$:
1892
1893
1894
1895 ;+
1896 ; POINT R3 TO FIRST ENTRY IN MODULE LIST,
1897 ; R2 POINTS TO FIRST ENTRY IN MOD TIME, AND R1 TO R.T.E TIME WORD.
1898 ; -
1899
```

1900	001534'			LET R3 := DT.MLST(R0)		
(4)	001534'	016003	000032		MOV	DT.MLST(R0),R3
1901	001540'			LET R2 := R1 + #6		
(4)	001540'	010102			MOV	R1,R2
(6)	001542'	062702	000006		ADD	#6,R2
1902	001546'			LET R1 := R1 + #2		
(6)	001546'	062701	000002		ADD	#2,R1
1903						
1904				;		
1905				; WHILE R3 IS NOT AT END OF MODULE LIST		
1906				;-		
1907						
1908	001552'			WHILE (R3) NE #ENDLST DO		
(4)	001552'				50004\$:	
(6)	001552'	021327	000000		CMP	(R3),#ENDLST
(9)	001556'	001500			BEQ	50005\$
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)		
(4)	001560'	011304			MOV	(R3),R4
1916						
1917				;		
1918				; MASK OUT UNWANTED BITS		
1919				;-		
1920						
1921	001562'			LET R4 := STAT(R4) CLR.BY #STATBIT		
(4)	001562'	016404	000026		MOV	STAT(R4),R4
(6)	001566'	042704	064757		BIC	#STATBIT,R4
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)		
(4)	001614'	011304			MOV	(R3),R4
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				; DETERMINE WHETHER THE DIFFERENCE OF MODULE TIME AND RTE. TIME		
1934						

```

1935 ; IS GREATER THAN THE MAXIMUM ALLOWABLE TIME BETWEEN PASSES
1936 ; -
1937 001636' LET CK.TEMP := (R1) - (R2)
(4) 001636' 011167 177632 MOV (R1),CK.TEMP
(6) 001642' 161267 177626 SUB (R2),CK.TEMP
1938 001646' IF CK.TEMP HI DT.MTIME(R0) THEN
(6) 001646' 026760 177622 000110 CMP CK.TEMP,DT.MTIME
(9) 001654' 101434 BLOS 50010$
1939
1940
1941 ; +
1942 ; THEN LOAD MESSAGE WITH MODULE HUNG AND OUTPUT MESSAGE AND DROP
1943 ; MODULE
1944 ; -
1945 001656' CALL ENQTQ IN <#DTABLE,#MSGHNG,#0,R4,#0>
(3) 001656' 010546 MOV R5,-(SP)
(8) 001660' 012745 000000 MOV #0,-(R5)
(7) 001664' 010445 MOV R4,-(R5)
(6) 001666' 012745 000000 MOV #0,-(R5)
(5) 001672' 012745 000022 MOV #MSGHNG,-(R5)
(4) 001676' 012745 000000G MOV #DTABLE,-(R5)
(3) 001702' 004767 000000G JSR PC,ENQTQ
(3) 001706' 012605 MOV (SP)+,R5
1946
1947
1948 ; +
1949 ; CLEAR THE ACTIVE BIT AND SET THE DROPPED BIT IN THE MODULES STATUS WORD
1950 ; -
1951
1952 001710' LET STAT(R4) := STAT(R4) SET.BY #BIT13
(6) 001710' 052764 020000 000026 BIS #BIT13,STAT(R4)
1953 001716' LET STAT(R4) := STAT(R4) CLR.BY #ACTBIT
(6) 001716' 042764 004000 000026 BIC #ACTBIT,STAT(R4)
1954
1955 ; +
1956 ; NOW SET BIT TO TELL MONITOR THAT A MODULE HAS BEEN DROPPED
1957 ; -
1958
1959 001724' LET DT.ST1(R0) := DT.ST1(R0) SET.BY #CKTIM
(6) 001724' 052760 100000 000012 BIS #CKTIM,DT.ST1(R0)
1960
1961 ; +
1962 ; NOW GO SEE IF ANY MODULES ARE STILL RUNNING IF NOT DUNCHK
1963 ; WILL DROP THE EXERCISOR.
1964 ; -
1965
1966 001732' CALL DUNCHK IN <R0,R4>
(3) 001732' 010546 MOV R5,-(SP)
(5) 001734' 010445 MOV R4,-(R5)
(4) 001736' 010045 MOV R0,-(R5)
(3) 001740' 004767 000000G JSR PC,DUNCHK
(3) 001744' 012605 MOV (SP)+,R5
1967 001746' ENDIF
(4) 001746' 50010$:
1968 001746' ENDIF
(4) 001746' 50007$:

```

```
1969 001746'          ENDIF
(4) 001746'          50006$:
1970
1971                ;+
1972                ; UPDATE MODULE LIST POINTER AND MODULE TIME POINTER
1973                ; -
1974
1975 001746'          LET R3 := R3 + #2
(6) 001746' 062703 000002          ADD #2, R3
1976 001752'          LET R2 := R2 + #4
(6) 001752' 062702 000004          ADD #4, R2
1977 001756'          ENDDO
(4) 001756' 000675          BR 50004$
(3) 001760'          50005$:
1978
1979                ;+
1980                ; RESTORE REGISTERS AND RETURN
1981                ; -
1982
1983 001760'          CALL RESREG
(3) 001760' 004767 000000G          JSR PC, RESREG
1984
1985 001764'          ENDRTN
(3) 001764'          50000$:
(3) 001764'          50001$:
(2) 001764' 000207          RTS PC
1986
1987 000001          .END
```

ACSR = 000102	CLKCHK 000034RG	DT.SWR= 000056	KIPAR3= 172346	PAERR = 000010
ACTBIT= 004000	CLKOFF 000304RG	DT.SYP= 000072	KIPAR4= 172350	PARPRE= 002000
ADDR22= 001000	CLKON 000370RG	DT.WBU= 000050	KIPAR5= 172352	PARSTA= 000100
ADR = 000000	CLKPRE= 000001	DT.WHL= 000051	KIPAR6= 172354	PASCNT= 000034
ADRCT = 000004	CLRTIM 001434RG	DT.WLL= 000052	KIPAR7= 172356	PDFLSI= 020000
APTFER= 000004	CONFIG= 000056	DUNCHK= ***** G	KIPDR0= 172300	PDP60 = 004000
APTPRE= 000200	CQOVF = 000001	DVID1 = 000014	KIPDR1= 172302	PDP70 = 010000
ASB = 000106	CR = 000015	ECCMEM= 000100	KIPDR2= 172304	PRHMS 000414RG
ASSEMB= 000010	CSRA = 000100	ECCSTA= 000010	KIPDR3= 172306	PRI0 = 000000
ASTAT = 000104	CSRC = 000102	ENBEO= 010000	KIPDR4= 172310	PRI1 = 000040
AUTO = 000010	CTRLC = 000003	ENBNUL= 000001	KIPDR5= 172312	PRI4 = 000200
AUTOST= 020000	CTRL0 = 000017	ENDLST= 000000	KIPDR6= 172314	PRI5 = 000240
AWAS = 000110	CTRLU = 000025	ENQTQ = ***** G	KIPDR7= 172316	PRI6 = 000300
BDACNV= ***** G	DCEVNT= 000011	EOPBIT= 000001	KTERRO= 000040	PRI7 = 000340
BGNADR= 000002	DEFRTN= 000400	ERRTYP= 000106	KTPRES= 000400	PRPSCN 000604RG
BIT0 = 000001	DIAGMC= 000000	EVNTBE= 000200	KTSTAT= 000020	PRO = 000000
BIT00 = 000001	DROPMO= 100000	EVNTHD= 000200	KXTXND= 040000	PR4 = 000200
BIT01 = 000002	DSEVNT= 000014	EVNTKT= 000203	LDTIME 000534R	PR5 = 000240
BIT02 = 000004	DTA = 000000	EVNTP= 000202	LF = 000012	PR6 = 000300
BIT03 = 000010	DTABLE= ***** G	EVNTRE= 000201	LPSTAT= 000001	PR7 = 000340
BIT04 = 000020	DT.ADD= 000042	FATERR= 100000	MAPSTA= 000200	PS = 177776
BIT05 = 000040	DT.AP = 000100	HMS 001004RG	MED = 076600	PSW = 177776
BIT06 = 000100	DT.APK= 000076	HM.CNV 000776R	MEMPAS= 040000	RANNUM= 000054
BIT07 = 000200	DT.BLS= 000034	HM.CTE 000770R	MODEXH= 004000	RBUFEA= 000130
BIT08 = 000400	DT.CFO= 000014	HM.CTT 000760R	MODHOL= 002000	RBUFPA= 000126
BIT09 = 001000	DT.CF1= 000016	HM.HRS 000760R	MODSEL= 001000	RBUFSZ= 000132
BIT1 = 000002	DT.ERR= 000020	HM.MIN 000763R	MSGCKD= 000010	RBUFVA= 000124
BIT10 = 002000	DT.ESI= 000044	HM.SEC 000765R	MSGCKS= 000011	RDSERV= 000101
BIT11 = 004000	DT.EVN= 000000	HM.THR 000770R	MSGDER= 000005	RDWHMI= 000022
BIT12 = 010000	DT.EXS= 000060	HM.TMI 000772R	MSGDRP= 000017	RELERR= 000020
BIT13 = 020000	DT.FCH= 000037	HM.TSE 000774R	MSGECH= 177777	RELMOD= 020000
BIT14 = 040000	DT.FCN= 000036	HRDADR= ***** G	MSGEOP= 000013	RELTIM= 010000
BIT15 = 100000	DT.HMX= 000104	HRDCNT= 000044	MSGHDR= 000004	RESREG= ***** G
BIT2 = 000004	DT.KBE= 000024	HRDPAS= 000050	MSGHNG= 000022	RESULT= 000004
BIT3 = 000010	DT.KBP= 000026	ICONT = 000036	MSGHRD= 000007	RES1 = 000056
BIT4 = 000020	DT.KBR= 000022	ICOUNT= 000040	MSGMAP= 000021	RES2 = 000060
BIT5 = 000040	DT.KBU= 000030	IDNUM = 000122	MSGNUL= 177775	RICHAR= 031060
BIT6 = 000100	DT.MLS= 000032	IE = 000100	MSGPOP= 000002	RPTDAT= 002000
BIT7 = 000200	DT.MTI= 000110	INDPAR= 000040	MSGPRM= 177776	RSTRT = 000112
BIT8 = 000400	DT.DFF= 000070	INHDRP= 040000	MSGRES= 000001	RUBOUT= 000177
BIT9 = 001000	DT.PAS= 000074	INHEPR= 020000	MSGSFT= 000006	RUNMOD= 100000
BKDEF = 000002	DT.PC = 000002	INHREL= 001000	MSGSKE= 000003	R5VALU= 001740
BKMOD = 000020	DT.PFL= 000062	INHRR= 000400	MSGSMB= 000015	SAM = 075464
BKMODE= 040000	DT.PSW= 000004	INIT = 000030	MSGSMH= 000014	SAVREG= ***** G
BKSLSH= 000134	DT.PTA= 000064	INTR = 000120	MSGSMS= 000016	SBADR = 000102
CAPRES= 000004	DT.RCS= 000102	IOMOD = 100000	MSGSTD= 000000	SBKMOD= 000000
CASSTAT= 000004	DT.REL= 000040	IOMODP= 102000	MSGSYS= 000012	SBKSEL= 010000
CC.CM1 000000R	DT.SCT= 000066	IOMODR= 112000	MSGVEC= 000020	SC.ADR= 000006
CDERCT= 000146	DT.SMX= 000106	IOMODX= 110000	NBKMOD= 001000	SC.ALC= 000014
CDWDCT= 000144	DT.SP = 000006	JACK = 035060	NCPUOP= 000020	SC.APC= 000016
CKHUNG 001476RG	DT.SSI= 000046	KIPAR0= 172340	NOAPTY= 000002	SC.CKL= 000002
CKTIM = 100000	DT.STO= 000010	KIPAR1= 172342	NULL = 000000	SC.CKP= 000004
CK.TEM 001474R	DT.ST1= 000012	KIPAR2= 172344	OWEN = 024020	SC.CLO= 000000

SC.HLD= 000010	TMPID = 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	\$SARGC= 000000
SOPPAS= 000046	UIPAR2= 177644	XFLAG = 000005	\$F\$WHI= 000120	\$S\$BYTE= 000403
SPACE = 000040	UIPAR3= 177646	XOFF = 000023	\$F\$YES= 000402	\$S\$CASE= 000000
SPOINT= 000032	UIPAR4= 177650	XON = 000021	\$IFLEV= 177777	\$S\$DST = 000000
SPVALU= 002200	UIPAR5= 177652	XTIME = 000002	\$ISKO = 000001	\$S\$ELOC= 000402
SR0 = 177572	UIPAR6= 177654	\$BGNLE= 177777	\$ISK1 = 000001	\$S\$ERFL= 000000
SR1 = 177574	UIPAR7= 177656	\$SERFLG= 000400	\$ISK2 = 000001	\$S\$FLAG= 000001
SR2 = 177576	UIPDR0= 177600	\$F\$AND= 000310	\$LOCTA= 177777	\$S\$FROM= 000000
SR3 = 172516	UIPDR1= 177602	\$F\$BAD= 000401	\$LSTIN= 000001	\$S\$LOC = 001654R
STAT = 000026	UIPDR2= 177604	\$F\$BLA= 000170	\$LSTTA= 000001	\$S\$LOCN= 000000
STATBI= 064757	UIPDR3= 177606	\$F\$CAS= 000150	\$NESTL= 177777	\$S\$REG = 177777
STAT1 = 000027	UIPDR4= 177610	\$F\$DEC= 000220	\$NSKO = 000300	\$S\$RETU= 000000
SUSPND= 000001	UIPDR5= 177612	\$F\$DO = 000340	\$NSK1 = 000120	\$S\$RTN1= 050000
SVR0 = 000062	UIPDR6= 177614	\$F\$FAL= 000405	\$NSK2 = 000110	\$S\$RTN2= 050001
SVR1 = 000064	UIPDR7= 177616	\$F\$GOD= 000400	\$NSK3 = 000110	\$S\$SRC = 000000
SVR2 = 000066	UPDMOD= 000002	\$F\$IF = 000110	\$NSK4 = 000110	\$S\$TGSV= 000000
SVR3 = 000070	UPDTIM 001320RG	\$F\$INC= 000210	\$SAVLE= 177777	\$S\$TGS1= 000000
SVR4 = 000072	WASADR= 000104	\$F\$LDD= 000200	\$SSKO = 050005	\$S\$TGS2= 000000
SVR5 = 000074	WBSTAT= 000040	\$F\$NAM= 000160	\$TAGLE= 177777	\$S\$TO = 000000
SVR6 = 000076	WBUFEA= 000136	\$F\$NO = 000403	\$TAGNU= 050011	\$S\$TAG= 050000
SYSCNT= 000052	WBUFPA= 000134	\$F\$OR = 000320	\$TEMP = 000300	. = 001766R
SYSERR= 000100	WBUFRQ= 000140	\$F\$RTI= 000350	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:39

EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1000

3 COMMON EQUATE MODULE  
536 COMMON DEFINITIONS & REFERENCES  
539 000000' .PRINT ;SPMAC: VERSION 1.1  
575 TQINI (FUNCTIONAL DESCRIPTION)  
617 TQINI (CODE)  
634 ENQTQ (FUNCTIONAL DESCRIPTION)  
686 ENQTQ (CODE)  
757 DEQTQ (MODULE FUNCTIONAL DESCRIPTION)  
808 DEQTQ (CODE)

```
508 .TITLE TYPQUE (TYPE QUEUE MODULES)
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE PACKAGE NAME:
513 ; TYPQUE
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE PACKAGE CONTAINS THREE ROUTINES. THE FIRST ONE INITIALIZES
517 ; THE TYPE QUEUE AND THE OTHER TWO ACTUALLY DO THE
518 ; ENQUEING AND DEQUEING OF ITEMS FROM THE TYPE QUEUE.
519 ; THE DESCRIPTION IS GIVEN UNDER THE RESPECTIVE MODULES.
520 ; THESE MODULES ARE:
521 ;
522 ; 1. TQINI - INITIALIZES THE TYPE QUEUE
523 ; 2. ENQTQ - ENQUES ITEMS ON THE TYPE QUEUE
524 ; 3. DEQTQ - DEQUES ITEMS FROM THE TYPE QUEUE.
525 ;
526 ; THE QUEUE POINTERS ARE CONTAINED
527 ; IN THIS MODULE FOR USE BY THE ABOVE NAMED MODULES LOCALLY.
528 ;
529 ; VERSION:
530 ; 0.0
531 ;
532 ; EDIT DATE BY REASON
533 ;---
534
```



```
536          .SBTTL COMMON DEFINITIONS & REFERENCES
537
538          .MCALL STRUCT
539 000000'    STRUCT
(1) 000000'    .PRINT ;SPMAC: VERSION 1.1
540          000001    $LSTIN=1
541          000001    $LSTTAG=1
542
543          ;*****
544          ;
545          ; REFERENCED BY OTHER MODULES:
546          ;
547          .GLOBL TQ.CT          ;TQ COUNT
548          .GLOBL TQ.INP        ;TQ IN PTR
549          .GLOBL TQ.OTP        ;TQ OUT PTR
550          .GLOBL TQ.INI        ;ROUTINE TO INIT TYPE QUEUE
551          .GLOBL ENQTQ         ;ROUTINE TO PUT ITEMS ON TQ
552          .GLOBL DEQTQ         ;ROUTINE TO DEQUE ITEMS.
553          ;*****
554          ; GLOBAL REFERENCES
555          ;
556          .GLOBL GETPSW        ;GET CALLER'S PSW ROUTINE
557          .GLOBL OV.TQ         ;TYPE QUEUE BUFFER
558          .GLOBL OV.HITQ
559          .GLOBL OV.TQSIZ
560
561          ;*****
562          ; LOCAL STORAGE
563          ;
564 000000' 000000    TQ.VEC: 0          ;SAVE LOC.4
565 000002' 000000    TQ.PSW: 0        ;LOC. TO SAVE CALLER'S PS WORD
566 000004' 000000G   TQ.INP: #OV.TQ       ;ENQUE POINTER
567 000006' 000000G   TQ.OTP: #OV.TQ       ;DEQUE POINTER
568 000010' 000000    TQ.CT: 0          ;CURRENT ENTRY COUNT
569 000012' 000000G   TQ.HI: #OV.HITQ      ;QUEUE'S HIGH ADDRESS
570 000014' 000000G   TQ.MAX: #OV.TQSIZ    ;MAX NO. OF ENTRIES ALLOWED
571 000016' 000000G   TQ.ITQ: #OV.TQ
572
573
```

```
575      .SBTTL TQINI (FUNCTIONAL DESCRIPTION)
576
577      ;++
578      ; MODULE NAME:
579      ;     TQINI
580      ;
581      ; FUNCTIONAL DESCRIPTION:
582      ;     THIS MODULE INITIALIZES THE TYPE QUEUE. IT RESETS THE ENQUE
583      ;     AND DEQUE POINTERS TO THE START OF THE QUEUE, AND SETS THE
584      ;     COUNT TO ZERO
585      ;
586      ; INPUTS:
587      ;     NONE
588      ;
589      ; IMPLICIT INPUTS:
590      ;     NONE
591      ;
592      ; OUTPUTS:
593      ;     NONE
594      ;
595      ; IMPLICIT OUTPUTS:
596      ;     NONE
597      ;
598      ; PATHOLOGICAL CONNECTIONS:
599      ;     NONE
600      ;
601      ; SUBORDINATE ROUTINES CALLED:
602      ;     NONE
603      ;
604      ; FUNCTIONAL SIDE EFFECTS:
605      ;     NONE
606      ;
607      ; CALLING SEQUENCE:
608      ;     CALL TQINI
609      ;
610      ; VERSION:
611      ;     0.0
612      ;
613      ;     EDIT      DATE      BY      REASON
614      ;--
615
```

TYPQUE (TYPE QUEUE MODULES)  
TYPQUE.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:39 PAGE 19-3  
TQINI (CODE)

SEQ 1004

```
617 .SBTTL TQINI (CODE)
618
619
620 000020' ROUTINE TQINI
(2) 000020'
621
622
623 ;+
624 ; INITIALIZE IN AND OUT POINTERS AND COUNT
625 ;-
626
627
628 000020' LET TQ.INP := TQ.ITQ
(4) 000020' 016767 177772 177756 MOV TQ.ITQ,TQ.INP
629 000026' LET TQ.OTP := TQ.ITQ
(4) 000026' 016767 177764 177752 MOV TQ.ITQ,TQ.OTP
630 000034' LET TQ.CT := #0
(4) 000034' 005067 177750 CLR TQ.CT
631 000040' ENDRTN
(3) 000040'
(3) 000040'
(2) 000040' 000207
632
```

TQINI:

50000\$:

50001\$:

RTS PC

```
634 .SBTTL ENQTQ (FUNCTIONAL DESCRIPTION)
635
636 ;++
637 ; MODULE NAME:
638 ; ENQTQ
639 ;
640 ; FUNCTIONAL DESCRIPTION:
641 ; THIS MODULE HANDLES QUEUEING OF ITEMS ONTO THE TYPE
642 ; QUEUE.
643 ;
644 ; INPUTS:
645 ; 1. DATA TABLE ADDRESS
646 ; 2. MESSAGE TYPE CODE
647 ; 3. MESSAGE ADDRESS
648 ; 4. MODULE'S HEADER ADDRESS
649 ; 5. RETURN ADDRESS
650 ;
651 ; IMPLICIT INPUTS:
652 ; DT.ERR
653 ;
654 ; OUTPUTS:
655 ; NONE
656 ;
657 ; IMPLICIT OUTPUTS:
658 ; ERROR WORD (DT.ERR)
659 ;
660 ; PATHOLOGICAL CONNECTIONS:
661 ; NONE
662 ;
663 ; SUBORDINATE ROUTINES CALLED:
664 ; 1. GETPSW ;GET CALLER'S PS WORD
665 ;
666 ; FUNCTIONAL SIDE EFFECTS:
667 ; ENQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
668 ; SERVES AS A LOCKING MECHANISM.
669 ;
670 ; CALLING SEQUENCE:
671 ; CALL ENQTQ IN <DTADR,TYPCOD,MSGADR,HDRADR,RETADR>
672 ;
673 ; WHERE DTADR = DATA TABLE ADDRESS
674 ; TYPCOD = TYPE CODE
675 ; MSGADR = MESSAGE ADDRESS
676 ; HDRADR = HEADER ADDRESS
677 ; RETADR = RETURN ADDRESS
678 ;
679 ; VERSION:
680 ; 0.0
681 ;
682 ; EDIT DATE BY REASON
683 ;--
684
```

```

686          .SBTTL ENQTQ (CODE)
687
688 000042'   ROUTINE ENQTQ <DT, CODE, MSG, HDR, RET>
(2) 000042'
689
690          ;+
691          ; SAVE REGISTER
692          ; -
693
694 000042'   PUSH R0
(2) 000042' 010046
695
696          ;+
697          ; IF THE COUNT IS EQUAL TO MAX. COUNT,
698          ; THE QUEUE IS FULL AND WE CANNOT MAKE
699          ; FURTHER ENTRIES INTO THE QUEUE. SAVE DTABLE ADDR AND SET
700          ; ERROR FLAG IN THE STATUS INDICATOR
701          ; AND RETURN WITH ERROR.
702          ; -
703
704 000044'   IF TQ.CT GE TQ.MAX THEN
(6) 000044' 026767 177740 177742
(9) 000052' 002410
705 000054'   LET R0 := DT(R5)
(4) 000054' 016500 000000
706 000060'   LET DT.ERR(R0) := DT.ERR(R0) SET.BY #TQOVF!FATERR
(6) 000060' 052760 100002 000020
707 000066'   POP R0
(2) 000066' 012600
708 000070'   RETURN ERROR
(2) 000070' 000261
(4) 000072' 000447
709 000074'   ENDIF
(4) 000074'
710
711          ;+
712          ; SAVE CALLER'S PRIORITY AND
713          ; CHANGE PRIORITY LEVEL TO 7
714          ; -
715
716 000074'   CALL GETPSW OUT <TQ.PSW>
(4) 000074' 162705 000002
(3) 000100' 004767 000000G
(4) 000104' 012567 177672
717 000110'   PUSH #PR7
(2) 000110' 012746 000340
718 000114'   PUSH #1$
(2) 000114' 012746 000122'
719 000120'   INLINE <RTI>
(2) 000120' 000002
720
721          ;+
722          ; IF QUEUE'S ENQUE POINTER IS EQUAL
723          ; TO QUEUE'S HIGH ADDRESS, RE-
724          ; INITIALIZE THE POINTER TO THE
725          ; START ADDRESS OF THE QUEUE.

```

ENQTQ:

MOV R0, -(SP)

CMP TQ.CT, TQ.MAX  
 BLT 50002\$

MOV DT(R5), R0

BIS #TQOVF!FATERR, DT

MOV (SP)+, R0

SEC  
 BR 50001\$

50002\$:

SUB #1\*2, R5  
 JSR PC, GETPSW  
 MOV (R5)+, TQ.PSW

MOV #PR7, -(SP)

MOV #1\$, -(SP)

RTI

```

726      ; -
727
728      000122'      INLINE <1$:>
(2)      000122'
729      000122'      IF TQ.INP EQ TQ.HI THEN
(6)      000122' 026767 177656 177662      CMP      TQ.INP,TQ.HI
(9)      000130' 001003      BNE      50003$
730      000132'      LET TQ.INP := #DV.TQ
(4)      000132' 012767 000000G 177644      MOV      #DV.TQ,TQ.INP
731      000140'      ENDIF
(4)      000140'      50003$:
732
733
734      ; +
735      ; NOW ENQUE THE ENTRIES.
736      ; -
737
738      000140'      LET R0 := TQ.INP
(4)      000140' 016700 177640      MOV      TQ.INP,R0
739      000144'      LET (R0)+ := CODE(R5)
(4)      000144' 016520 000002      MOV      CODE(R5),(R0)+
740      000150'      LET (R0)+ := MSG(R5)
(4)      000150' 016520 000004      MOV      MSG(R5),(R0)+
741      000154'      LET (R0)+ := HDR(R5)
(4)      000154' 016520 000006      MOV      HDR(R5),(R0)+
742      000160'      LET (R0)+ := RET(R5)
(4)      000160' 016520 000010      MOV      RET(R5),(R0)+
743      000164'      LET TQ.INP := R0
(4)      000164' 010067 177614      MOV      R0,TQ.INP
744      000170'      LET TQ.CT := TQ.CT + #1
(6)      000170' 005267 177614      INC      TQ.CT
745
746      ; +
747      ; RESTORE CALLER'S PS WORD AND RETURN
748      ; -
749
750      000174'      PUSH TQ.PSW
(2)      000174' 016746 177602      MOV      TQ.PSW,-(SP)
751      000200'      PUSH #2$
(2)      000200' 012746 000206'      MOV      #2$,-(SP)
752      000204'      INLINE <RTI>
(2)      000204' 000002      RTI
753      000206'      INLINE <2$:>
(2)      000206'      2$:
754      000206'      POP R0
(2)      000206' 012600      MOV      (SP)+,R0
755      000210'      ENDRTN
(3)      000210'      50000$:
(2)      000210' 000241      CLC
(3)      000212'      50001$:
(2)      000212' 000207      RTS      PC

```

TYPQUE (TYPE QUEUE MODULES)  
TYPQUE.MAC 28-JUL-78 09:28

MACY11 30A(1052) 20-SEP-78 18:39 PAGE 19-7  
DEQTQ (MODULE FUNCTIONAL DESCRIPTION)

SEQ 1008

```
757 .SBTTL DEQTQ (MODULE FUNCTIONAL DESCRIPTION)
758
759
760 ;++
761 ; MODULE NAME:
762 ; DEQTQ
763 ;
764 ; FUNCTIONAL DESCRIPTION:
765 ; THIS MODULE HANDLES DEQUEUEING OF ITEMS FROM THE TYPE
766 ; QUEUE.
767 ;
768 ; INPUTS:
769 ; NONE
770 ;
771 ; IMPLICIT INPUTS:
772 ; NONE
773 ;
774 ; OUTPUTS:
775 ; 1. MESSAGE TYPE CODE
776 ; 2. MESSAGE ADDRESS
777 ; 3. MODULE'S HEADER ADDRESS
778 ; 4. RETURN ADDRESS
779 ;
780 ; IMPLICIT OUTPUTS:
781 ; NONE
782 ;
783 ; PATHOLOGICAL CONNECTIONS:
784 ; NONE
785 ;
786 ; SUBORDINATE ROUTINES CALLED:
787 ; GETPSW ;GET CALLER'S PS WORD
788 ;
789 ; FUNCTIONAL SIDE EFFECTS:
790 ; DEQUEING IS DONE AT PRIORITY LEVEL 7 WHICH
791 ; SERVES AS A LOCKING MECHANISM.
792 ;
793 ; CALLING SEQUENCE:
794 ; CALL DEQTQ OUT <CODE,MSGADR,HDRADR,RETADR>
795 ;
796 ; WHERE CODE = TYPE CODE
797 ; MSGADR = MESSAGE ADDRESS
798 ; HDRADR = HEADER ADDRESS
799 ; RETADR = RETURN ADDRESS
800 ;
801 000214' ; VERSION:
802 ; 0.0
803 ;
804 ; EDIT DATE BY REASON
805 ;--
806
```

```
808 .SBTTL DEQTTQ (CODE)
809
810 000214' ROUTINE DEQTTQ <CODE,MSG,HDR,RET>
(2) 000214' DEQTTQ:
811
812 ;+
813 ; IF THE COUNT IS EQUAL TO ZERO, QUEUE
814 ; IS EMPTY. RETURN WITH ERROR.
815 ;-
816
817 000214' IF TQ.CT LE #0 THEN
(6) 000214' 005767 177570 TST TQ.CT
(9) 000220' 003002 BGT 50002$
818 000222' RETURN ERROR
(2) 000222' 000261 SEC
(4) 000224' 000450 BR 50001$
819 000226' ENDF
(4) 000226' 50002$:
820
821 ;+
822 ; SAVE CALLER'S PRIORITY AND
823 ; CHANGE PRIORITY TO 7
824 ;-
825
826 000226' CALL GETPSW OUT <TQ.PSW>
(4) 000226' 162705 000002 SUB #1*2,R5
(3) 000232' 004767 000000G JSR PC,GETPSW
(4) 000236' 012567 177540 MOV (R5)+,TQ.PSW
827 000242' PUSH #PR7
(2) 000242' 012746 000340 MOV #PR7,-(SP)
828 000246' PUSH #3$
(2) 000246' 012746 000254' MOV #3$,-(SP)
829 000252' INLINE <RTI>
(2) 000252' 000002 RTI
830
831 ;+
832 ; IF THE DEQUE POINTER IS AT HIGH
833 ; ADDRESS, RE-INITIALIZE IT TO THE
834 ; START ADDRESS OF THE QUEUE.
835 ;-
836
837 000254' INLINE <3$:>
(2) 000254' 3$:
838 000254' IF TQ.OTP EQ TQ.HI THEN
(6) 000254' 026767 177526 177530 CMP TQ.OTP,TQ.HI
(9) 000262' 001003 BNE 50003$
839 000264' LET TQ.OTP := #DV.TQ
(4) 000264' 012767 000000G 177514 MOV #DV.TQ,TQ.OTP
840 000272' ENDF
(4) 000272' 50003$:
841
842 ;+
843 ; NOW DEQUE THE ENTRIES.
844 ;-
845
846 ;+
```



```

847          ; SAVE REGISTERS
848          ; -
849
850 000272'   PUSH R0
(2) 000272' 010046          MOV    R0,-(SP)
851 000274'   LET R0 := TQ.OTP
(4) 000274' 016700 177506  MOV    TQ.OTP,R0
852 000300'   LET CODE(R5) := (R0)+
(4) 000300' 012065 000000  MOV    (R0)+,CODE(R5)
853 000304'   LET MSG(R5) := (R0)+
(4) 000304' 012065 000002  MOV    (R0)+,MSG(R5)
854 000310'   LET HDR(R5) := (R0)+
(4) 000310' 012065 000004  MOV    (R0)+,HDR(R5)
855 000314'   LET RET(R5) := (R0)+
(4) 000314' 012065 000006  MOV    (R0)+,RET(R5)
856 000320'   LET TQ.OTP := R0
(4) 000320' 010067 177462  MOV    R0,TQ.OTP
857
858          ; +
859          ; DECREMENT COUNT
860          ; -
861
862 000324'   LET TQ.CT := TQ.CT - #1
(6) 000324' 005367 177460  DEC    TQ.CT
863
864          ; +
865          ; RESTORE CALLER'S PS WORD AND RETURN
866          ; -
867
868 000330'   PUSH TQ.PSW
(2) 000330' 016746 177446  MOV    TQ.PSW,-(SP)
869 000334'   PUSH #4$
(2) 000334' 012746 000342'  MOV    #4$,-(SP)
870 000340'   INLINE <RTI>
(2) 000340' 000002          RTI
871 000342'   INLINE <4$:>
(2) 000342'          4$:
872 000342'   POP R0
(2) 000342' 012600          MOV    (SP)+,R0
873 000344'   ENDRTN
(3) 000344'          50000$:
(2) 000344' 000241          CLC
(3) 000346'          50001$:
(2) 000346' 000207          RTS    PC
874 000001   .END
```

ACSR = 000102	CSRC = 000102	ENBNUL= 000001	MAPSTA= 000200	PR6 = 000300
ACTBIT= 004000	CTRLC = 000003	ENDLST= 000000	MED = 076600	PR7 = 000340
ADDR22= 001000	CTRL0 = 000017	ENQIQ 000042RG	MEMPAS= 040000	PS = 177776
ADR = 000006	CTRLU = 000025	EOPBIT= 000001	MODEXH= 004000	PSW = 177776
APTFER= 000004	DCEVNT= 000011	ERRTYP= 000106	MODHOL= 002000	RANNUM= 000054
APTPRE= 000200	DEFRTN= 000400	EVNTBE= 000200	MODSEL= 001000	RBUFEA= 000130
ASB = 000106	DEQIQ 000214RG	EVNTHD= 000200	MSG = 000002	RBUFPA= 000126
ASSEMB= 000010	DIAGMC= 000000	EVNTKT= 000203	MSGCKD= 000010	RBUFSZ= 000132
ASTAT = 000104	DROPMO= 100000	EVNTPE= 000202	MSGCKS= 000011	RBUFVA= 000124
AUTO = 000010	DSEVNT= 000014	EVNTRE= 000201	MSGDER= 000005	RDSERV= 000101
AUTOST= 020000	DT = 000000	FATERR= 100000	MSGDRP= 000017	RDWHMI= 000022
AWAS = 000110	DT.ADD= 000042	GETPSW= ***** G	MSGECH= 177777	RELERR= 000020
BIT0 = 000001	DT.AP = 000100	HDR = 000004	MSGGEP= 000013	RELMOD= 020000
BIT00 = 000001	DT.APK= 000076	HRDCNT= 000044	MSGHDR= 000004	RELTIM= 010000
BIT01 = 000002	DT.BLS= 000034	HRDPAS= 000050	MSGHNG= 000022	RES1 = 000056
BIT02 = 000004	DT.CFO= 000014	ICONT = 000036	MSGHRD= 000007	RES2 = 000060
BIT03 = 000010	DT.CF1= 000016	ICOUNT= 000040	MSGMAP= 000021	RET = 000006
BIT04 = 000020	DT.ERR= 000020	IDNUM = 000122	MSGNUL= 177775	RICHAR= 031060
BIT05 = 000040	DT.ESI= 000044	IE = 000100	MSGPOP= 000002	RPTDAT= 002000
BIT06 = 000100	DT.EVN= 000000	INDPAR= 000040	MSGPRM= 177776	RSTRT = 000112
BIT07 = 000200	DT.EXS= 000060	INHDRP= 040000	MSGRES= 000001	RUBOUT= 000177
BIT08 = 000400	DT.FCH= 000037	INHPR= 020000	MSGSFT= 000006	RUNMOD= 100000
BIT09 = 001000	DT.FCN= 000036	INHREL= 001000	MSGSKE= 000003	REVALU= 001740
BIT1 = 000002	DT.HMX= 000104	INHRE= 000400	MSGSMB= 000015	SAM = 075464
BIT10 = 002000	DT.KBE= 000024	INIT = 000030	MSGSMH= 000014	SBADR = 000102
BIT11 = 004000	DT.KBP= 000026	INTR = 000120	MSGSMS= 000016	SBKMOD= 000000
BIT12 = 010000	DT.KBR= 000022	IOMOD = 100000	MSGSTD= 000000	SBKSEL= 010000
BIT13 = 020000	DT.KBU= 000030	IOMODP= 102000	MSGSYS= 000012	SC.ADR= 000006
BIT14 = 040000	DT.MLS= 000032	IOMODR= 112000	MSGVEC= 000020	SC.ALC= 000014
BIT15 = 100000	DT.MTI= 000110	IOMODX= 110000	NEKMOD= 001000	SC.APC= 000016
BIT2 = 000004	DT.OFF= 000070	JACK = 035060	NCPUOP= 000020	SC.CKL= 000002
BIT3 = 000010	DT.PAS= 000074	KIPAR0= 172340	NOPTY= 000002	SC.CKP= 000004
BIT4 = 000020	DT.PC = 000002	KIPAR1= 172342	NULL = 000000	SC.CLD= 000000
BIT5 = 000040	DT.PFL= 000062	KIPAR2= 172344	OV.HIT= ***** G	SC.HLD= 000010
BIT6 = 000100	DT.PSW= 000004	KIPAR3= 172346	OV.TQ = ***** G	SC.SCA= 000012
BIT7 = 000200	DT.PTA= 000064	KIPAR4= 172350	OV.TQS= ***** G	SENDLS= 177777
BIT8 = 000400	DT.RCS= 000102	KIPAR5= 172352	OWEN = 024020	SOFCNT= 000042
BIT9 = 001000	DT.REL= 000040	KIPAR6= 172354	PAERR = 000010	SOFPAS= 000046
BKDEF = 000002	DT.SCT= 000066	KIPAR7= 172356	PARPRE= 002000	SPACE = 000040
BKMOD = 000020	DT.SMX= 000106	KIPDR0= 172300	PASTA= 000100	SPOINT= 000032
BKMODE= 040000	DT.SP = 000006	KIPDR1= 172302	PARCNT= 000034	SPVALU= 002200
BKSLSH= 000134	DT.SSI= 000046	KIPDR2= 172304	PDPLSI= 020000	SR0 = 177572
CAPRES= 000004	DT.STO= 000010	KIPDR3= 172306	PDP60 = 004000	SR1 = 177574
CASTAT= 000004	DT.ST1= 000012	KIPDR4= 172310	PDP70 = 010000	SR2 = 177576
CDERCT= 000146	DT.SWR= 000056	KIPDR5= 172312	PRI0 = 000000	SR3 = 172516
CDWDCT= 000144	DT.SYP= 000072	KIPDR6= 172314	PRI1 = 000040	STAT = 000026
CKTIM = 100000	DT.WBU= 000050	KIPDR7= 172316	PRI4 = 000200	STATBI= 064757
CLKPRE= 000001	DT.WHL= 000054	KTERRO= 000040	PRI5 = 000240	STAT1 = 000027
CODE = 000000	DT.WLL= 000052	KTPRES= 000400	PRI6 = 000300	SUSPND= 000001
CONFIG= 000056	DVID1 = 000014	KTSTAT= 000020	PRI7 = 000340	SVR0 = 000062
CQOVF = 000001	ECCMEM= 000100	KTXTND= 040000	PRO = 000000	SVR1 = 000064
CR = 000015	ECCSTA= 000010	LF = 000012	PR4 = 000200	SVR2 = 000066
CSRA = 000100	ENBEOP= 010000	LPSTAT= 000001	PR5 = 000240	SVR3 = 000070

SVR4 = 000072	UIPAR5= 177652	WTWHMI= 000222	\$F\$RTN= 000300	\$S\$BYTE= 000403
SVR5 = 000074	UIPAR6= 177654	XFLAG = 000005	\$F\$SEL= 000140	\$S\$CASE= 000000
SVR6 = 000076	UIPAR7= 177656	XOFF = 000023	\$F\$THE= 000330	\$S\$DST = 000000
SYSCNT= 000052	UIPDR0= 177600	XON = 000021	\$F\$TRU= 000404	\$S\$ELOC= 000402
SYSERR= 000100	UIPDR1= 177602	\$BGNLE= 177777	\$F\$UNT= 000130	\$S\$ERFL= 000000
TMPIO = 000002	UIPDR2= 177604	\$ERFLG= 000400	\$F\$WHI= 000120	\$S\$FLAG= 000001
TQINI 000020RG	UIPDR3= 177606	\$F\$AND= 000310	\$F\$YES= 000402	\$S\$FROM= 000001
TQOVF = 000002	UIPDR4= 177610	\$F\$BAD= 000401	\$IFLEV= 177777	\$S\$LQC = 000262R
TQ.CT 000010RG	UIPDR5= 177612	\$F\$BLA= 000170	\$ISKO = 000001	\$S\$LOCN= 000000
TQ.HI 000012R	UIPDR6= 177614	\$F\$CAS= 000150	\$LOCTA= 177777	\$S\$REG = 177777
TQ.INP 000004RG	UIPDR7= 177616	\$F\$DEC= 000220	\$LSTIN= 000001	\$S\$RETU= 000001
TQ.ITQ 000016R	VERSIO 000214R	\$F\$DO = 000340	\$LSTTA= 000001	\$S\$RTN1= 050000
TQ.MAX 000014R	WASADR= 000104	\$F\$FAL= 000405	\$NESTL= 177777	\$S\$RTN2= 050001
TQ.OTP 000006RG	WBSTAT= 000040	\$F\$GDD= 000400	\$NSKO = 000300	\$S\$SRC = 000000
TQ.PSW 000002R	WBUFEA= 000136	\$F\$IF = 000110	\$NSK1 = 000110	\$S\$TGSV= 000000
TQ.VEC 000000R	WBUFPA= 000134	\$F\$INC= 000210	\$SAVLE= 177777	\$S\$TGS1= 000000
UIPAR0= 177640	WBUFRQ= 000140	\$F\$LDD= 000200	\$TAGLE= 177777	\$S\$TGS2= 000000
UIPAR1= 177642	WBUFSZ= 000142	\$F\$NAM= 000160	\$TAGNU= 050004	\$S\$TO = 000000
UIPAR2= 177644	WDFR = 000116	\$F\$ND = 000403	\$TEMP = 000300	\$S\$TAG= 050000
UIPAR3= 177646	WDTO = 000114	\$F\$DR = 000320	\$TSKO = 050003	= 000350R
UIPAR4= 177650	WTINRE= 000352	\$F\$RTI= 000350	\$S\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:40  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1013

3 COMMON EQUATE MODULE  
554 COMMON DEFINITIONS AND REFERENCES  
556 000000' .PRINT ;SPMAC: VERSION 1.1  
583 UNIMAP ROUTINE

UNIMAP - LOAD UNIBUS MAP  
UNIMAP.MAC 15-AUG-78 15:01

MACY11 30A(1052) 20-SEP-78 18:40 PAGE 19  
COMMON EQUATE MODULE

SEQ 1014

```
508 .TITLE UNIMAP - LOAD UNIBUS MAP
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; UNIMAP
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; LOADS THE UNIBUS MAP SO THAT THE WRITE BUFFER CAN BE
517 ; ADDRESSED BY UNIBUS DEVICES WHEN 22-BIT ADDRESSING
518 ; SPACE IS IN USE.
519 ;
520 ; INPUTS:
521 ; DATA TABLE
522 ;
523 ; IMPLICIT INPUTS:
524 ; DT.WLLMT
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; BL.BTMBNK ;CURRENT 124K BANK OF BOTTOM OF EXERCISER
534 ; BL.TOPBNK ;CURRENT 124K BANK OF TOP OF EXERCISER
535 ;
536 ; SUBORDINATE MODULES CALLED:
537 ; SAVREG ;SAVE REGISTERS
538 ; RESREG ;RESTORE REGISTERS
539 ;
540 ; FUNCTIONAL SIDE EFFECTS:
541 ; NONE
542 ;
543 ; CALLING SEQUENCE:
544 ; CALL UNIMAP IN <A>
545 ; A=ADDRESS OF DATA TABLE
546 ;
547 ; VERSION:
548 ; 0.0
549 ;
550 ; EDIT BY DATE REASON
551 ;
552 ;--
```

UNIMAP - LOAD UNIBUS MAP  
UNIMAP.MAC 15-AUG-78 15:01

MACY11 30A(1052) 20-SEP-78 18:40 PAGE 19-1  
COMMON DEFINITIONS AND REFERENCES

SEQ 1015

```
554 .SBTTL COMMON DEFINITIONS AND REFERENCES
555 .MCALL STRUCT
556 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
557 000001 $LSTIN=1
558 000001 $LSTTAG=1
559
560 ;
561 ;*****
562 ;
563 ; REFERENCED BY OTHER MODULES
564 ;
565 .GLOBL UNIMAP ;MODULE ENTRY POINT
566 ;
567 ;*****
568 ;
569 ; GLOBAL REFERENCES
570 ;
571 .GLOBL SAVREG ;SAVE REGISTERS
572 .GLOBL RESREG ;RESTORE REGISTERS
573 .GLOBL BL.BTMBNK ;CURRENT 124K BANK OF BOTTOM OF RTE
574 .GLOBL BL.TOPBNK ;CURRENT 124K BANK OF TOP OF RTE
575 ;
576 ;*****
577 ;
578 ; LOCAL EQUATES
579 ;
580 170200 MAPREG=170200 ;LOWEST ADDRESS OF UNIBUS MAP
581
```

```
583          .SBTTL UNIMAP ROUTINE
584
585 000000'    ROUTINE UNIMAP <TABL>
(2) 000000'
586
587          ;+
588          ; SAVE REGISTERS
589          ; -
590
591 000000'    CALL SAVREG
(3) 000000' 004767 000000G          JSR      PC, SAVREG
592
593
594          ;+
595          ; SET R0 TO THE START OF THE DATA TABLE
596          ; -
597
598 000004'    LET R0 := TABL(R5)
(4) 000004' 016500 000000          MOV      TABL(R5), R0
599
600          ;+
601          ; IF THE WRITE BUFFER LOW LIMIT IS 4K, IMPLYING THE CURRENT BANK IS THE
602          ; LOWEST 124K BANK OF MEMORY, THEN START
603          ; MAPPING FROM ADDRESS 0. HOWEVER, IF THIS IS A 128K SYSTEM
604          ; AND IF THE EXERCISER IS CROSSING THE 124K BOUNDARY, BEGIN MAPPING
605          ; FROM THE LOW BUFFER LIMIT.
606
607          ; -
608
609 000010'    IF DT.WLLMT(R0) EQ #200 THEN
(6) 000010' 026027 000052 000200          CMP      DT.WLLMT(R0), #200
(9) 000016' 001014          BNE      50002$
610          IF DT.SSIZ(R0) EQ #10000 AND BL.TOPBNK NE BL.BTM BNK THEN
(6) 000020' 026027 000046 010000          CMP      DT.SSIZ(R0), #10000
(9) 000026' 001005          BNE      50003$
(6) 000030' 026767 000000G 000000G          CMP      BL.TOPBNK, BL.BTM BNK
(9) 000036' 001401          BEQ      50003$
611          INLINE <BR 1$>
(2) 000040' 000403          BR      1$
612          ENDIF
(4) 000042'          50003$:
613          LET R1 := #0
(4) 000042' 005001          CLR      R1
614          LET R2 := #0
(4) 000044' 005002          CLR      R2
615
616          ELSE
(4) 000046' 000417          BR      50004$
(3) 000050'          50002$:
617
618
619          ;+
620          ; ELSE CONVERT THE WRITE BUFFER LOW LIMIT TO 22 BITS. (THE LOW 16 BITS
621          ; WILL BE IN R1, AND THE HIGH 6 BITS WILL BE IN R2.)
622          ; GET THE UPPER SIX BITS INTO THE UPPER SIX BITS OF R1
623          ; THEN SWAP THEM INTO THE LOWER BYTE OF R2
```

```
624 ; THEN LINE THEM UP IN BITS 0 THRU 5 AND MASK OUT UPPER 10 BITS
625 ;--
626
627 000050' INLINE <1$:>
(2) 000050' 1$:
628 000050' LET R1 := DT.WLLMT(R0) SHIFT #+6
(4) 000050' 016001 000052 MOV DT.WLLMT(R0),R1
(7) 000054' 006301 ASL R1
(7) 000056' 006301 ASL R1
(7) 000060' 006301 ASL R1
(7) 000062' 006301 ASL R1
(7) 000064' 006301 ASL R1
(7) 000066' 006301 ASL R1
629 000070' LET R2 := SWAP DT.WLLMT(R0)
(6) 000070' 016002 000052 MOV DT.WLLMT(R0),R2
(6) 000074' 000302 SWAB R2
630 000076' LET R2 := R2 SHIFT #-2
(7) 000076' 006202 ASR R2
(7) 000100' 006202 ASR R2
631 000102' LET R2 := R2 CLR.BY #177400
(6) 000102' 042702 177400 BIC #177400,R2
632 000106' ENDIF
(4) 000106' 50004$:
633
634
635 ;+
636 ; NOW MAP, BY SETTING R3 TO LOWEST MAP REGISTER AND R4 TO MAXIMUM
637 ; NUMBER OF REGISTER.
638 ;--
639
640 000106' LET R3 := #MAPREG
(4) 000106' 012703 170200 MOV #MAPREG,R3
641 000112' LET R4 := #^D31
(4) 000112' 012704 000037 MOV #^D31,R4
642
643 ;+
644 ; WHILE REGISTERS REMAIN CONTINUE TO MAP
645 ;--
646
647 000116' WHILE R4 NE #0 DO
(4) 000116' 50005$:
(6) 000116' 005704 TST R4
(9) 000120' 001407 BEQ 50006$
648
649 ;+
650 ; LOAD REGISTERS AND THEN UPDATE POINTERS
651 ;--
652
653 000122' LET (R3)+ := R1
(4) 000122' 010123 MOV R1,(R3)+
654 000124' LET (R3)+ := R2
(4) 000124' 010223 MOV R2,(R3)+
655 000126' LET R1 := R1 + #20000
(6) 000126' 062701 020000 ADD #20000,R1
656 000132' LET R2 := R2 + CARRY
(6) 000132' 005502 ADC R2
```



```
657 000134'          LET R4 := R4 - #1
(6) 000134' 005304
658 000136'          ENDDO
(4) 000136' 000767
(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	CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132
ACTBIT= 004000	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ADDR22= 001000	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RDSERV= 000101
ADR = 000006	CTRLO = 000017	EVNTBE= 000200	MSGCKD= 000010	RDWHMI= 000022
APTFER= 000004	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RELERR= 000020
APTPRE= 000200	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RELMOD= 020000
ASB = 000106	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RELTIM= 010000
ASSEMB= 000010	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RESREG= ***** G
ASTAT = 000104	DROPMO= 100000	FATERR= 100000	MSGEOP= 000013	RES1 = 000056
AUTO = 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHDR= 000004	RES2 = 000060
AUTOST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGHNG= 000022	RICHAR= 031060
AWAS = 000110	DT.AP = 000100	ICONT = 000036	MSGHRD= 000007	RPTDAT= 002000
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGMAP= 000021	RSTRT = 000112
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGNUL= 177775	RUBOUT= 000177
BIT01 = 000002	DT.CFO= 000014	IE = 000100	MSGPOP= 000002	RUNMOD= 100000
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MSGPRM= 177776	RSVALU= 001740
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MSGRES= 000001	SAM = 075464
BIT04 = 000020	DT.ESI= 000044	INHPR= 020000	MSGSFT= 000006	SAVREG= ***** G
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MSGSKE= 000003	SBADR = 000102
BIT06 = 000100	DT.EXS= 000060	INHRE= 000400	MSGSMB= 000015	SBKMOD= 000000
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MSGSMH= 000014	SBKSEL= 010000
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGSMS= 000016	SC.ADR= 000006
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGSTD= 000000	SC.ALC= 000014
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGSYS= 000012	SC.APC= 000016
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	MSGVEC= 000020	SC.CKL= 000002
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	NBKMOD= 001000	SC.CKP= 000004
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	NCPUOP= 000020	SC.CLO= 000003
BIT13 = 020000	DT.MLS= 000032	KIPAR0= 172340	NDAPTY= 000002	SC.HLD= 000010
BIT14 = 040000	DT.MTI= 000110	KIPAR1= 172342	NULL = 000000	SC.SCA= 000012
BIT15 = 100000	DT.OFF= 000070	KIPAR2= 172344	OWEN = 024020	SENDLS= 177777
BIT2 = 000004	DT.PAS= 000074	KIPAR3= 172346	PAERR = 000010	SOFCNT= 000042
BIT3 = 000010	DT.PC = 000002	KIPAR4= 172350	PARPRE= 002000	SOFPAS= 000046
BIT4 = 000020	DT.PFL= 000062	KIPAR5= 172352	PARSTA= 000100	SPACE = 000040
BIT5 = 000040	DT.PSW= 000004	KIPAR6= 172354	PASCNT= 000034	SPOINT= 000032
BIT6 = 000100	DT.PTA= 000064	KIPAR7= 172356	PDPLSI= 020000	SPVALU= 002200
BIT7 = 000200	DT.RCS= 000102	KIPDR0= 172300	PDP60 = 004000	SR0 = 177572
BIT8 = 000400	DT.REL= 000040	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
BIT9 = 001000	DT.SCT= 000066	KIPDR2= 172304	PRI0 = 000000	SR2 = 177576
BKDEF = 000002	DT.SMX= 000106	KIPDR3= 172306	PRI1 = 000040	SR3 = 172516
BKMOD = 000020	DT.SP = 000006	KIPDR4= 172310	PRI4 = 000200	STAT = 000026
BKMODE= 040000	DT.SSI= 000046	KIPDR5= 172312	PRI5 = 000240	STATBI= 064757
BKSLSH= 000134	DT.STO= 000010	KIPDR6= 172314	PRI6 = 000300	STAT1 = 000027
BL.BTM= ***** G	DT.ST1= 000012	KIPDR7= 172316	PRI7 = 000340	SUSPND= 000001
BL.TOP= ***** G	DT.SWR= 000056	KTERR0= 000040	PR0 = 000000	SVR0 = 000062
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR3 = 000070
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR4 = 000072
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SVR5 = 000074
CLKPRE= 000001	ECCMEM= 000100	MAPREG= 170200	PSW = 177776	SVR6 = 000076
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	RANNUM= 000054	SYSCNT= 000052
CQOVF = 000001	ENBEOP= 010000	MED = 076600	RBUFEA= 000130	SYSERR= 000100
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFPA= 000126	TABL = 000000

TMPIO = 000002	WASADR= 000104	\$F\$DEC= 000220	\$ISKO = 000001	\$\$DST = 000000
TQOVF = 000002	WBSTAT= 000040	\$F\$DD = 000340	\$ISK1 = 000001	\$\$ELDC= 000402
UIPAR0= 177640	WBUFEA= 000136	\$F\$FAL= 000405	\$LOCTA= 177777	\$\$ERFL= 000000
UIPAR1= 177642	WBUFPA= 000134	\$F\$GDD= 000400	\$LSTIN= 000001	\$\$FLAG= 000340
UIPAR2= 177644	WBUFRR= 000140	\$F\$IF = 000110	\$LSTTA= 000001	\$\$FROM= 000000
UIPAR3= 177646	WBUFSZ= 000142	\$F\$INC= 000210	\$NESTL= 177777	\$\$LOC = 000120R
UIPAR4= 177650	WDFR = 000116	\$F\$LDD= 000200	\$NSKO = 000300	\$\$LOCN= 000000
UIPAR5= 177652	WDTD = 000114	\$F\$NAM= 000160	\$NSK1 = 000120	\$\$REG = 177777
UIPAR6= 177654	WTINRE= 000352	\$F\$NO = 000403	\$NSK2 = 000110	\$\$RETN= 000000
UIPAR7= 177656	WTWHMI= 000222	\$F\$OR = 000320	\$\$SAVLE= 177777	\$\$RTN1= 050000
UIPDR0= 177600	XFLAG = 000005	\$F\$RTI= 000350	\$\$SSKO = 050006	\$\$RTN2= 050001
UIPDR1= 177602	XOFF = 000023	\$F\$RTN= 000300	\$TAGLE= 177777	\$\$SRC = 000000
UIPDR2= 177604	XDN = 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	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:40  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1021

3 COMMON EQUATE MODULE  
553 COMMON DEFINITIONS AND REFERENCES  
555 000000' .PRINT ;SPMAC: VERSION 1.1  
583 UNIPA ROUTINE

```
508 .TITLE UNIPA - GET PHYSICAL ADDRESS FORM UNIBUS MAP
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; UNIPA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; DETERMINES 22-BIT PHYSICAL ADDRESS FROM 18-BIT VIRTUAL
517 ; ADDRESS AND PROPER UNIBUS MAP REGISTER.
518 ;
519 ; INPUTS:
520 ; ADDRESS OF I/O TABLE - CONTAINS 16 BIT PA AND 2 EA BITS
521 ;
522 ; IMPLICIT INPUTS:
523 ; PA
524 ; EA
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; 16 BITS AND EA BITS
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; NONE
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ; SAVREG ;SAVE REGISTERS
537 ; RESREG ;RESTORE REGISTERS
538 ;
539 ; FUNCTIONAL SIDE EFFECTS:
540 ; NONE
541 ;
542 ; CALLING SEQUENCE:
543 ; CALL UNIPA IN <A>
544 ; A=ADDRESS OF I/O TABLE
545 ;
546 ; VERSION:
547 ; 0.0
548 ;
549 ; EDIT BY DATE REASON
550 ;
551 ;--
```

```
553 .SBTTL COMMON DEFINITIONS AND REFERENCES
554 .MCALL STRUCT
555 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
556 000001 $LSTIN=1
557 000001 $LSTTAG=1
558
559 ;
560 ;*****
561 ;
562 ; REFERENCED BY OTHER MODULES
563 ;
564 .GLOBL UNIPA ;MODULE ENTRY POINT
565 ;
566 ;*****
567 ;
568 ; GLOBAL REFERENCES
569 ;
570 .GLOBL SAVREG ;SAVE REGISTERS
571 .GLOBL RESREG ;RESTORE REGISTERS
572 ;
573 ;*****
574 ;
575 ; LOCAL EQUATES
576 ;
577 170200 MAPREG=170200
578 ;
579 ;*****
580 ;
581 ;
```

```

583          .SBTTL UNIPA ROUTINE
584
585
586 000000'          ROUTINE UNIPA <ADDR>
587 (2) 000000'
588
589          ;+
590          ; SAVE REGISTERS
591          ; -
592
593 000000'          CALL SAVREG
594 (3) 000000' 004767 000000G          JSR          PC, SAVREG
595
596          ;+
597          ; GET THE ADDRESS OF THE I/O TABLE
598          ; -
599
600 000004'          LET R0 := ADDR(R5)
601 (4) 000004' 016500 000000          MOV          ADDR(R5), R0
602
603          ;+
604          ; GET THE 18-BIT ADDRESS TO BE CONVERTED.  THE LOW 16 BITS WILL
605          ; BE IN R1 AND THE HIGH 2 BITS WILL BE IN R2 (POSITIONS 4 AND 5).
606          ; -
607
608 000010'          LET R1 := (R0)+
609 (4) 000010' 012001          MOV          (R0)+, R1
610 000012'          LET R2 := (R0)+
611 (4) 000012' 012002          MOV          (R0)+, R2
612
613          ;+
614          ; GET THE HIGH 5 BITS OF THIS ADDRESS TO DETERMINE THE PROPER
615          ; UNIBUS MAP REGISTER.
616          ; -
617
617 000014'          LET R3 := SWAP R1
618 (6) 000014' 010103          MOV          R1, R3
619 (6) 000016' 000303          SWAB          R3
620 000020'          LET R3 := R3 SHIFT #-3
621 (7) 000020' 006203          ASR          R3
622 (7) 000022' 006203          ASR          R3
623 (7) 000024' 006203          ASR          R3
624 000026'          LET R3 := R3 CLR.BY #177743
625 (6) 000026' 042703 177743          BIC          #177743, R3
626 000032'          LET R4 := R2 SHIFT #+1
627 (4) 000032' 010204          MOV          R2, R4
628 (7) 000034' 006304          ASL          R4
629 000036'          LET R4 := R4 + R3
630 (6) 000036' 060304          ADD          R3, R4
631
632
633
634          ;+
    
```

```

625 ; GET THE CONTENTS OF THE PROPER MAP REGISTER. THE LOW
626 ; 16 BITS WILL BE IN R2 AND THE HIGH 6 BITS WILL BE IN R3.
627 ;-
628
629 000040' LET R4 := R4 + #MAPREG ADD #MAPREG,R4
(6) 000040' 062704 170200
630 000044' LET R2 := (R4)+ MOV (R4)+,R2
(4) 000044' 012402
631 000046' LET R3 := (R4) MOV (R4),R3
(4) 000046' 011403
632
633
634 ;+
635 ; ADD THE LOW 12 BITS OF THE 18-BIT VIRTUAL ADDRESS TO THE
636 ; VALUE JUST OBTAINED FROM THE MAP REGISTER. THE RESULT
637 ; IS THE 22-BIT PHYSICAL ADDRESS.
638 ;-
639
640 000050' LET R1 := R1 CLR.BY #160000 BIC #160000,R1
(6) 000050' 042701 160000
641 000054' LET R2 := R2 + R1 ADD R1,R2
(6) 000054' 060102
642 000056' LET R3 := R3 + CARRY ADC R3
(6) 000056' 005503
643
644
645 ;+
646 ; STORE THE 22-BIT PHYSICAL ADDRESS IN THE I/O TABLE. THE LOW
647 ; 16 BITS GO IN FIRST, THEN THE HIGH 6.
648 ;-
649
650 000060' LET (R0)+ := R2 MOV R2,(R0)+
(4) 000060' 010220
651 000062' LET (R0) := R3 MOV R3,(R0)
(4) 000062' 010310
652
653
654 ;+
655 ; RESTORE REGISTERS
656 ;-
657
658 000064' CALL RESREG JSR PC,RESREG
(3) 000064' 004767 000000G
659
660
661 ;+
662 ; COME AGAIN, SOON.
663 ;-
664
665 000070' ENDRTN
(3) 000070' 50000$:
(3) 000070' 50001$:
(2) 000070' 000207 RTS PC
666
667 000001 .END
  
```



ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFVA= 000124
ACTBIT= 004000	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RDSERV= 000101
ADDR = 000000	CTRLD = 000017	EVNTBE= 000200	MSGCKD= 000010	RDWHMI= 000022
ADDR22= 001000	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RELERR= 000020
ADR = 000006	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RELMOD= 020000
APTFER= 000004	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RELTIM= 010000
APTPRE= 000200	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RESREG= ***** G
ASB = 000106	DROPMQ= 100000	FATERR= 100000	MSGEOP= 000013	RES1 = 000056
ASSEMB= 000010	DSEVNT= 000014	HRDCNT= 000044	MSGHDR= 000004	RES2 = 000060
ASTAT = 000104	DT.ADD= 000042	HRDPAS= 000050	MSGHNG= 300022	RICHAR= 031060
AUTO = 000010	DT.AP = 000100	ICONT = 000036	MSGHRD= 000007	RPTDAT= 002000
AUTOST= 020000	DT.APK= 000076	ICOUNT= 000040	MSGMAP= 000021	RSTRT = 000112
AWAS = 000110	DT.BLS= 000034	IDNUM = 000122	MSGNUL= 177775	RUBOUT= 000177
BIT0 = 000001	DT.CFO= 000014	IE = 000100	MSGPOP= 000002	RUNMOD= 100000
BIT00 = 000001	DT.CF1= 000016	INDPAR= 000040	MSGPRM= 177776	RVALU= 001740
BIT01 = 000002	DT.ERR= 000020	INHDRP= 040000	MSGRES= 000001	SAM = 075464
BIT02 = 000004	DT.ESI= 000044	INHPR= 020000	MSGSFT= 000006	SAVREG= ***** G
BIT03 = 000010	DT.EVN= 000000	INHREL= 001000	MSGSKE= 000003	SBADR = 000102
BIT04 = 000020	DT.EXS= 000060	INHRE= 000400	MSGSMB= 000015	SBKMOD= 000000
BIT05 = 000040	DT.FCH= 000037	INIT = 000030	MSGSMH= 000014	SBKSEL= 010000
BIT06 = 000100	DT.FCN= 000036	INTR = 000120	MSGSMS= 000016	SC.ADR= 000006
BIT07 = 000200	DT.HMX= 000104	IOMOD = 100000	MSGSTD= 000000	SC.ALC= 000014
BIT08 = 000400	DT.KBE= 000024	IOMODP= 102000	MSGSYS= 000012	SC.APC= 000016
BIT09 = 001000	DT.KBP= 000026	IOMODR= 112000	MSGVEC= 000020	SC.CKL= 000002
BIT1 = 000002	DT.KBR= 000022	IOMODX= 110000	NBKMOD= 001000	SC.CKP= 000004
BIT10 = 002000	DT.KBU= 000030	JACK = 035060	NCPUOP= 000020	SC.CLO= 000000
BIT11 = 004000	DT.MLS= 000032	KIPAR0= 172340	NOAPTY= 000002	SC.HLD= 000010
BIT12 = 010000	DT.MTI= 000110	KIPAR1= 172342	NULL = 000000	SC.SCA= 000012
BIT13 = 020000	DT.OFF= 000070	KIPAR2= 172344	OWEN = 024020	SENDLS= 177777
BIT14 = 040000	DT.PAS= 000074	KIPAR3= 172346	PAERR = 000010	SOFCNT= 000042
BIT15 = 100000	DT.PC = 000002	KIPAR4= 172350	PARPRE= 002000	SOFPAS= 000046
BIT2 = 000004	DT.PFL= 000062	KIPAR5= 172352	PARSTA= 000100	SPACE = 000040
BIT3 = 000010	DT.PSW= 000004	KIPAR6= 172354	PASCNT= 000034	SPOINT= 000032
BIT4 = 000020	DT.PTA= 000064	KIPAR7= 172356	PDPLSI= 020000	SPVALU= 002200
BIT5 = 000040	DT.RCS= 000102	KIPDR0= 172300	PDP60 = 004000	SR0 = 177572
BIT6 = 000100	DT.REL= 000040	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
BIT7 = 000200	DT.SCT= 000066	KIPDR2= 172304	PRI0 = 000000	SR2 = 177576
BIT8 = 000400	DT.SMX= 000106	KIPDR3= 172306	PRI1 = 000040	SR3 = 172516
BIT9 = 001000	DT.SP = 000006	KIPDR4= 172310	PRI4 = 000200	STAT = 000026
BKDEF = 000002	DT.SSI= 000046	KIPDR5= 172312	PRI5 = 000240	STATBI= 064757
BKMOD = 000020	DT.ST0= 000010	KIPDR6= 172314	PRI6 = 000300	STAT1 = 000027
BKMODE= 040000	DT.ST1= 000012	KIPDR7= 172316	PRI7 = 000340	SUSPND= 000001
BKSLSH= 000134	DT.SWR= 000056	KTERRO= 000040	PR0 = 000000	SVR0 = 000062
CAPRES= 000004	DT.SYP= 000072	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CASTAT= 000004	DT.WBU= 000050	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CDERCT= 000146	DT.WHL= 000054	KTXTND= 040000	PR6 = 000300	SVR3 = 000070
CDWDCT= 000144	DT.WLL= 000052	LF = 000012	PR7 = 000340	SVR4 = 000072
CKTIM = 100000	DVID1 = 000014	LPSTAT= 000001	PS = 177776	SVR5 = 000074
CLKPRE= 000001	ECCMEM= 000100	MAPREG= 170200	PSW = 177776	SVR6 = 000076
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	RANNUM= 000054	SYSCNT= 000052
CQCQVF = 000001	ENBEOP= 010000	MED = 076600	RBUFEA= 000130	YSERR= 000100
CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFPA= 000126	TMPID = 000002
CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFSZ= 000132	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\$G00= 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\$LDO= 000200	\$NSKO = 000300	\$RTN1= 050000
UIPDR1= 177602	WTWHMI= 000222	\$F\$NAM= 000160	\$SAVLE= 177777	\$RTN2= 050001
UIPDR2= 177604	XFLAG = 000005	\$F\$ND = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:41  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1028

3 COMMON EQUATE MODULE  
558 COMMON DEFINITIONS AND REFERENCES  
560 000000' .PRINT ;SPMAC: VERSION 1.1  
596 WBFLIM ROUTINE

```
508 .TITLE WBFLIM - DETERMINE WRITE BUFFER LIMITS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     WBFLIM
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     CALCULATES LOW AND HIGH WRITE BUFFER LIMITS AND
517 ;     INITIALIZES THE WRITE BUFFER ADDRESS POINTER
518 ;
519 ; INPUTS:
520 ;     DATA TABLE
521 ;
522 ; IMPLICIT INPUTS:
523 ;     DT.ADDR
524 ;     DT.ESIZ
525 ;     DT.SSIZ
526 ;     DT.STO
527 ;
528 ; OUTPUTS:
529 ;     NONE
530 ;
531 ; IMPLICIT OUTPUTS:
532 ;     DT.WLLMT
533 ;     DT.WHLMT
534 ;     DT.WBUF
535 ;     DT.OFFSET
536 ;
537 ; PATHOLOGICAL CONNECTIONS
538 ;     NONE
539 ;
540 ; SUBORDINATE MODULES CALLED:
541 ;     SAVREG ;SAVE REGISTERS
542 ;     RESREG ;RESTORE REGISTERS
543 ;
544 ; FUNCTIONAL SIDE EFFECTS:
545 ;     NONE
546 ;
547 ; CALLING SEQUENCE:
548 ;     CALL WBFLIM IN <A>
549 ;         A=ADDRESS OF DATA TABLE
550 ;
551 ; VERSION:
552 ;     0.0
553 ;
554 ;     EDIT          BY          DATE          REASON
555 ;
556 ;---
```

```
558 .SBTTL COMMON DEFINITIONS AND REFERENCES
559 .MCALL STRUCT
560 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
561 000001 $LSTIN=1
562 000001 $LSTTAG=1
563
564 ;
565 ;*****
566 ; REFERENCED BY OTHER MODULES
567 ;
568 .GLOBL WBFLIM ;MODULE ENTRY POINT
569 .GLOBL BL.BTMBNK
570 .GLOBL BL.TOPBNK
571 ;
572 ;*****
573 .GLOBL SAVREG ;SAVE REGISTERS
574 .GLOBL RESREG ;RESTORE REGISTERS
575 ;
576 ;*****
577 ;
578 ; LOCAL EQUATES
579 ;
580 020000 FOURK=20000 ;4K OF MEMORY
581 000200 PAR4K=200 ;4K OF MEMORY IN PAR FORMAT
582 007600 BANK=7600 ;PAR FORMAT FOR 124K BANK
583 ;
584 ;*****
585 ;
586 ; LOCAL STORAGE
587 ;
588 000000' 000000 BL.TOPBNK: .WORD 0 ;CURRENT 124K BANK OF TOP OF
589 ; MOVABLE PORTION OF EXERCISER
590 000002' 000000 BL.BTMBNK: .WORD 0 ;CURRENT 124K BANK OF BOTTOM OF
591 ; MOVABLE PORTION OF EXERCISER
592 ;
593 ;*****
594 ;
```

```
596 .SBTTL WBFLIM ROUTINE
597
598 000004' ROUTINE WBFLIM <TABL>
(2) 000004'
599
600
601 ;+
602 ; SAVE REGISTERS
603 ; -
604
605 000004' CALL SAVREG
(3) 000004' 004767 000000G JSR PC, SAVREG
606
607
608 ;+
609 ; SET R0 TO THE START OF THE DATA TABLE
610 ; -
611
612 000010' LET R0 := TABL(R5)
(4) 000010' 016500 000000 MOV TABL(R5), R0
613
614
615 ;+
616 ; FIND THE TOP ADDRESS OF THE MOVABLE PORTION OF THE EXERCISER.
617 ; TO DO THIS, FIND THE SIZE OF THE MOVABLE PORTION BY SUBTRACTING 4K FROM
618 ; EXERCISER SIZE, CONVERT IT TO PAR FORMAT.
619 ; -
620
621 000014' LET R1 := DT.ESIZ(R0) - #FOURK
(4) 000014' 016001 000044 MOV DT.ESIZ(R0), R1
(6) 000020' 162701 020000 SUB #FOURK, R1
622 000024'
(7) 000024' 006201 ASR R1
(7) 000026' 006201 ASR R1
(7) 000030' 006201 ASR R1
(7) 000032' 006201 ASR R1
(7) 000034' 006201 ASR R1
(7) 000036' 006201 ASR R1
623 000040' LET R1 := R1 CLR.BY #176000
(6) 000040' 042701 176000 BIC #176000, R1
624 000044' LET R1 := R1 + #1
(6) 000044' 005201 INC R1
625
626 ;+
627 ; NOW ADD IT TO THE BASE ADDRESS OF THE MOVABLE PORTION
628 ; -
629 000046' LET R1 := R1 + DT.ADDR(R0)
(6) 000046' 066001 000042 ADD DT.ADDR(R0), R1
630
631
632 ;+
633 ; INITIALIZE THE OFFSET VALUE TO ZERO.
634 ; -
635
636 000052' LET DT.OFFSET(R0) := #0
(4) 000052' 005060 000070 CLR DT.OFFSET(R0)
```

```

638
639
640 ;+
641 ; SEE IF MEMORY MANAGEMENT IS TURNED ON.
642 ; -
643 000056' IF #KTSTAT SETIN DT.STO(R0) THEN
(6) 000056' 032760 000020 000010 BIT #KTSTAT,DT.STO(R
(9) 000064' 001566 BEQ 50002$
644
645
646 ;+
647 ; MEMORY MANAGEMENT IS ON.
648 ; -
649
650
651 ;+
652 ; SEE IF THE MOVABLE PORTION OF THE EXERCISER LIES ACROSS A 124K
653 ; BOUNDARY. TO DO THIS, DETERMINE THE RESPECTIVE BANKS OF THE
654 ; TOP AND BOTTOM OF THE MOVABLE PORTION OF THE EXERCISER.
655 ; -
656
657 000066' LET R2 := #0
(4) 000066' 005002 CLR R2
658 000070' LET BL.TOPBNK := #-1
(4) 000070' 012767 177777 177702 MOV #-1,BL.TOPBNK
659 000076' WHILE R2 LO R1 DO
(4) 000076' 50003$:
(6) 000076' 020201 CMP R2,R1
(9) 000100' 103005 BHIS 50004$
660 000102' LET R2 := R2 + #BANK
(6) 000102' 062702 007600 ADD #BANK,R2
661 000106' LET BL.TOPBNK := BL.TOPBNK + #1
(6) 000106' 005267 177666 INC BL.TOPBNK
662 000112' ENDDO
(4) 000112' 000771 BR 50003$
(3) 000114' 50004$:
663
664 000114' LET R2 := #0
(4) 000114' 005002 CLR R2
665 000116' LET BL.BTMENK := #-1
(4) 000116' 012767 177777 177656 MOV #-1,BL.BTMENK
666 000124' WHILE R2 LO DT.ADDR(R0) DO
(4) 000124' 50005$:
(6) 000124' 020260 000042 CMP R2,DT.ADDR(R0)
(9) 000130' 103005 BHIS 50006$
667 000132' LET R2 := R2 + #BANK
(6) 000132' 062702 007600 ADD #BANK,R2
668 000136' LET BL.BTMENK := BL.BTMENK + #1
(6) 000136' 005267 177640 INC BL.BTMENK
669 000142' ENDDO
(4) 000142' 000770 BR 50005$
(3) 000144' 50006$:

```

```

671
672
673
674
675 000144'
(6) 000144' 026767 177630 177630
(9) 000152' 001441
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690 000154'
(4) 000154' 016060 000042 000052
691 000162'
(4) 000162' 016060 000042 000054
(6) 000170' 062760 007600 000054
692 000176'
(6) 000176' 026060 000054 000046
(9) 000204' 101420
693 000206'
(4) 000206' 016060 000046 000054
694 000214'
(4) 000214' 016060 000046 000052
(6) 000222' 162760 007600 000052
695 000230'
(6) 000230' 026027 000052 000200
(9) 000236' 103003
696 000240'
(4) 000240' 012760 000200 000052
697 000246'
(4) 000246'
698 000246'
(4) 000246'
699
700
701
702
703
704
705 000246'
(4) 000246' 016060 000052 000070
706
707
708 000254'
(4) 000254' 000471
(3) 000256'

```

```

;+
; IF THE BANKS ARE NOT THE SAME, THEN THE EXERCISER IS CROSSING A BOUNDARY.
;-
      IF BL.TOPBNK NE BL.BTMBNK THEN
                                CMP      BL.TOPBNK,BL.BTM
                                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.
;-
      LET DT.WLLMT(R0) := DT.ADDR(R0)
                                MOV      DT.ADDR(R0),DT.W
      LET DT.WHLMT(R0) := DT.ADDR(R0) + #BANK
                                MOV      DT.ADDR(R0),DT.W
                                ADD      #BANK,DT.WHLMT(R)
      IF DT.WHLMT(R0) HI DT.SSIZ(R0) THEN
                                CMP      DT.WHLMT(R0),DT.
                                BLOS     50010$
                                LET DT.WHLMT(R0) := DT.SSIZ(R0)
                                MOV      DT.SSIZ(R0),DT.W
                                LET DT.WLLMT(R0) := DT.SSIZ(R0) - #7600
                                MOV      DT.SSIZ(R0),DT.W
                                SUB      #7600,DT.WLLMT(R)
      IF DT.WLLMT(R0) LO #200 THEN
                                CMP      DT.WLLMT(R0),#20
                                BHIS     50011$
                                LET DT.WLLMT(R0) := #200
                                MOV      #200,DT.WLLMT(R0)
      ENDIF
                                50011$:
      ENDIF
                                50010$:
;+
; ALSO, THE OFFSET FROM THE ACTUAL 124K BANK BASE ADDRESS MUST
; BE STORED. IT EQUALS, IN THIS CASE, THE LOW BUFFER LIMIT.
;-
      LET DT.OFFSET(R0) := DT.WLLMT(R0)
                                MOV      DT.WLLMT(R0),DT.
      ELSE
                                BR      50012$
                                50007$:

```



```
710 ;+
711 ; OTHERWISE, THE MOVABLE PORTION OF THE EXERCISER IS WITHIN ONE BANK.
712 ; -
713
714 ;+
715 ; IF THE CURRENT BANK IS THE LOWEST ONE, THEN THE LOW BUFFER LIMIT
716 ; IS THE TOP OF THE BOTTOM 4K OF THE EXERCISER. OTHERWISE, THE LOW
717 ; BUFFER LIMIT IS THE BOTTOM OF THE CURRENT BANK.
718 ; -
719
720 IF BL.BTMBNK EQ #0 THEN
(6) 000256' 005767 177520 TST BL.BTMBNK
(9) 000262' 001004 BNE 50013$
721 000264' LET DT.WLLMT(R0) := #PAR4K
(4) 000264' 012760 000200 000052 MOV #PAR4K,DT.WLLMT(
722 000272' ELSE
(4) 000272' 000413 BR 50014$
(3) 000274' 50013$:
723 000274' LET R2 := #0
(4) 000274' 005002 CLR R2
724 000276' LET DT.WLLMT(R0) := #0 CLR DT.WLLMT(R0)
(4) 000276' 005060 000052 WHILE R2 LO BL.BTMBNK DO
725 000302' 50015$:
(4) 000302' CMP R2,BL.BTMBNK
(6) 000302' 020267 177474 BHIS 50016$
(9) 000306' 103005 LET DT.WLLMT(R0) := DT.WLLMT(R0) + #BANK
726 000310' ADD #BANK,DT.WLLMT(R
(6) 000310' 062760 007600 000052 LET R2 := R2 + #1
727 000316' INC R2
(6) 000316' 005202 ENDDO
728 000320' BR 50015$
(4) 000320' 000770 50016$:
(3) 000322' 50014$:
729
730 000322' ENDIF
(4) 000322'
731
732
733 ;+
734 ; IF THE CURRENT BANK IS THE HIGHEST ONE, THEN THE HIGH BUFFER LIMIT
735 ; IS THE TOP OF MEMORY. OTHERWISE, THE HIGH BUFFER LIMIT IS THE TOP OF
736 ; THE CURRENT BANK.
737 ; -
738
739 LET R2 := #0 CLR R2
(4) 000322' 005002 LET R3 := #-1 MOV #-1,R3
740 000324' WHILE R2 LO DT.SSIZ(R0) DO
(4) 000324' 012703 177777 50017$:
741 000330' CMP R2,DT.SSIZ(R0)
(4) 000330' 020260 000046 BHIS 50020$
(6) 000330' 020260 000046 LET R2 := R2 + #BANK
(9) 000334' 103004 ADD #BANK,R2
742 000336' LET R3 := R3 + #1 INC R3
(6) 000336' 062702 007600
743 000342'
(6) 000342' 005203
```

```
744 000344'          ENDDO
(4) 000344' 000771
(3) 000346'          50020$: BR 50017$
745
746 000346'          IF BL.TOPBNK EQ R3 THEN
(6) 000346' 026703 177426
(9) 000352' 001004
747 000354'          LET DT.WHLMT(R0) := DT.SSIZ(R0)
(4) 000354' 016060 000046 000054
748 000362'          ELSE
(4) 000362' 000415
(3) 000364'          50021$: BR 50022$
749 000364'          IF BL.BTMBNK EQ #0 THEN
(6) 000364' 005767 177412
(9) 000370' 001004
750 000372'          LET DT.WHLMT(R0) := #BANK
(4) 000372' 012760 007600 000054
751 000400'          ELSE
(4) 000400' 000406
(3) 000402'          50023$: BR 50024$
752 000402'          LET DT.WHLMT(R0) := DT.WLLMT(R0) + #BANK
(4) 000402' 016060 000052 000054
(6) 000410' 062760 007600 000054
753 000416'          ENDIF
(4) 000416'          50024$:
754 000416'          ENDIF
(4) 000416'          50022$:
```

```
756          ;+
757          ; DETERMINE THE OFFSET FORM THE BASE OF THE CURRENT 124K BANK.
758          ; -
759
760          WHILE BL.BTMBMK GT #0 DO
(4) 000416'
(6) 000416' 005767 177360
(9) 000422' 003406
761 000424'          LET DT.OFFSET(R0) := DT.OFFSET(R0) + #7600
(6) 000424' 062760 007600 000070          ADD      #7600,DT.OFFSET(
762 000432'          LET BL.BTMBNK := BL.BTMBNK - #1          DEC      BL.BTMBNK
(6) 000432' 005367 177344          ENDDO          BR      50025$
763 000436'
(4) 000436' 000767          50026$:
(3) 000440'
764
765          ENDIF          50012$:
766 000440'
(4) 000440'
767
768          ELSE
769 000440'          BR      50027$
(4) 000440' 000415          50002$:
(3) 000442'
770
771          ;+
772          ; MEMORY MANAGEMENT IS TURNED OFF
773          ; -
774
775          ;+
776          ; IN THIS CASE, THE LOW BUFFER LIMIT IS THE TOP OF THE EXERCISER.
777          ; -
778
779 000442'          LET DT.WLLMT(R0) := R1          MOV      R1,DT.WLLMT(R0)
(4) 000442' 010160 000052
780
781
782          ;+
783          ; IF TOTAL MEMORY IS GREATER THAN 28K, THEN THE HIGH BUFFER LIMIT
784          ; IS 28K. OTHERWISE, THE HIGH BUFFER LIMIT IS THE TOP OF MEMORY.
785          ; -
786
787 000446'          IF DT.SSIZ(R0) HIS #1600 THEN          CMP      DT.SSIZ(R0),#160
(6) 000446' 026027 000046 001600          BLO      50030$
(9) 000454' 103404
788 000456'          LET DT.WHLMT(R0) := #1600          MOV      #1600,DT.WHLMT(R
(4) 000456' 012760 001600 000054          ELSE          BR      50031$
789 000464'
(4) 000464' 000403          50030$:
(3) 000466'          LET DT.WHLMT(R0) := DT.SSIZ(R0)          MOV      DT.SSIZ(R0),DT.W
790 000466'
(4) 000466' 016060 000046 000054          ENDIF          50031$:
791 000474'
(4) 000474'
792
793 000474'          ENDIF
```

```
(4) 000474' 50027$:  
794  
795  
796  
797 ;+  
798 ; NOW INITIALIZE THE WRITE BUFFER ADDRESS POINTER. IF THE TOP OF  
799 ; THE EXERCISER IS AT AN ADDRESS THAT IS GREATER THAN  
800 ; OR EQUAL TO THE HIGH BUFFER LIMIT MINUS 1/2K, AND THE EXERCISER IS NOT  
801 ; AT THE BOTTOM OF MEMORY, THEN SET THE POINTER IS THE LOW BUFFER  
802 ; LIMIT. OTHERWISE, SET THE POINTER TO THE TOP OF  
803 ; THE EXERCISER.  
804 ;-  
805 000474' LET R2 := DT.SSIZ(R0) - #20  
(4) 000474' 016002 000046 MOV DT.SSIZ(R0),R2  
(6) 000500' 162702 000020 SUB #20,R2  
806 000504' IF R1 HIS R2 AND DT.ADDR(R0) HI #PAR4K THEN  
(6) 000504' 020102 CMP R1,R2  
(9) 000506' 103410 BLO 50032$  
(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	CR = 000015	ENBNUL= 000001	MEMPAS= 040000	RBUFEA= 000130
ACTBIT= 004000	CSRA = 000100	ENDLST= 000000	MODEXH= 004000	RBUFPA= 000126
ADDR22= 001000	CSRC = 000102	EOPBIT= 000001	MODHOL= 002000	RBUFSZ= 000132
ADR = 000006	CTRLC = 000003	ERRTYP= 000106	MODSEL= 001000	RBUFVA= 000124
APTFER= 000004	CTRLO = 000017	EVNTBE= 000200	MSGCKD= 000010	RDSERV= 000101
APTPRE= 000200	CTRLU = 000025	EVNTHD= 000200	MSGCKS= 000011	RDWHMI= 000022
ASB = 000106	DCEVNT= 000011	EVNTKT= 000203	MSGDER= 000005	RELERR= 000020
ASSEMB= 000010	DEFRTN= 000400	EVNTPE= 000202	MSGDRP= 000017	RELMOD= 020000
ASTAT = 000104	DIAGMC= 000000	EVNTRE= 000201	MSGECH= 177777	RELTIM= 010000
AUTO = 000010	DROPMO= 100000	FATERR= 100000	MSGEOP= 000013	RESREG= ***** G
AUTOST= 020000	DSEVNT= 000014	FOURK = 020000	MSGHDR= 000004	RES1 = 000056
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RES2 = 000060
BANK = 007600	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RICHAR= 031060
BIT0 = 000001	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RPTDAT= 002000
BIT00 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RSTRT = 000112
BIT01 = 000002	DT.CFO= 000014	IDNUM = 000122	MSGPOP= 000002	RUBOUT= 000177
BIT02 = 000004	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RUNMOD= 100000
BIT03 = 000010	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	RSVALU= 001740
BIT04 = 000020	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	SAM = 075464
BIT05 = 000040	DT.EVN= 000000	INHPRP= 020000	MSGSKS= 000003	SAVREG= ***** G
BIT06 = 000100	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	SBADR = 000102
BIT07 = 000200	DT.FCH= 000037	INHRRP= 000400	MSGSMH= 000014	SBKMOD= 000000
BIT08 = 000400	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	SBKSEL= 010000
BIT09 = 001000	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	SC.ADR= 000006
BIT1 = 000002	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	SC.ALC= 000014
BIT10 = 002000	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	SC.APC= 000016
BIT11 = 004000	DT.KBR= 000022	IOMODR= 112000	NBKMOD= 001000	SC.CKL= 000002
BIT12 = 010000	DT.KBU= 000030	IOMODX= 110000	NCPUOP= 000020	SC.CKP= 000004
BIT13 = 020000	DT.MLS= 000032	JACK = 035060	NDAPTY= 000002	SC.CLO= 000000
BIT14 = 040000	DT.MTI= 000110	KIPAR0= 172340	NULL = 000000	SC.HLD= 000010
BIT15 = 100000	DT.OFF= 000070	KIPAR1= 172342	OWEN = 024020	SC.SCA= 000012
BIT2 = 000004	DT.PAS= 000074	KIPAR2= 172344	PAERR = 000010	SENDLS= 177777
BIT3 = 000010	DT.PC = 000002	KIPAR3= 172346	PARPRE= 002000	SDFCNT= 000042
BIT4 = 000020	DT.PFL= 000062	KIPAR4= 172350	PARSTA= 000100	SDFPAS= 000048
BIT5 = 000040	DT.PSW= 000004	KIPAR5= 172352	PAR4K = 000200	SPACE = 000040
BIT6 = 000100	DT.PTA= 000064	KIPAR6= 172354	PASCNT= 000034	SPDINT= 000032
BIT7 = 000200	DT.RCS= 000102	KIPAR7= 172356	PDPLSI= 020000	SPVALU= 002200
BIT8 = 000400	DT.REL= 000040	KIPDR0= 172300	PDP60 = 004000	SR0 = 177572
BIT9 = 001000	DT.SCT= 000066	KIPDR1= 172302	PDP70 = 010000	SR1 = 177574
BKDEF = 000002	DT.SMX= 000106	KIPDR2= 172304	PRI0 = 000000	SR2 = 177576
BKMOD = 000020	DT.SP = 000006	KIPDR3= 172306	PRI1 = 000040	SR3 = 172516
BKMODE= 040000	DT.SSI= 000046	KIPDR4= 172310	PRI4 = 000200	STAT = 000026
BKSLSH= 000134	DT.STO= 000010	KIPDR5= 172312	PRI5 = 000240	STATBI= 064757
BL.BTM 000002RG	DT.ST1= 000012	KIPDR6= 172314	PRI6 = 000300	STAT1 = 000027
BL.TOP 000000RG	DT.SWR= 000056	KIPDR7= 172316	PRI7 = 000340	SUSPND= 000001
CAPRES= 000004	DT.SYP= 000072	KTERR0= 000040	PRO = 000000	SVR0 = 000062
CASTAT= 000004	DT.WBU= 000050	KTPRES= 000400	PR4 = 000200	SVR1 = 000064
CDERCT= 000146	DT.WHL= 000054	KTSTAT= 000020	PR5 = 000240	SVR2 = 000066
CDWDCT= 000144	DT.WLL= 000052	KTXTND= 040000	PR6 = 000300	SVR3 = 000070
CKTIM = 100000	DVID1 = 000014	LF = 000012	PR7 = 000340	SVR4 = 000072
CLKPRE= 000001	ECCMEM= 000100	LPSTAT= 000001	PS = 177776	SVR5 = 000074
CONFIG= 000056	ECCSTA= 000010	MAPSTA= 000200	PSW = 177776	SVR6 = 000076
CQOVF = 000001	ENBEOP= 010000	MED = 076600	RANNUM= 000054	SYSCNT= 000052

SYSERR= 000100	WBFLIM 000004RG	\$\$\$FAL= 000405	\$LOCTA= 177777	\$\$CASE= 000000
TABL = 000000	WBSTAT= 000040	\$\$\$GDD= 000400	\$LSTIN= 000001	\$\$DST = 000000
TMPID = 000002	WBUFEA= 000136	\$\$\$IF = 000110	\$LSTTA= 000001	\$\$ELOC= 000402
TQOVF = 000002	WBUFPA= 000134	\$\$\$INC= 000210	\$NESTL= 177777	\$\$ERFL= 000000
UIPAR0= 177640	WBUFRQ= 000140	\$\$\$LOO= 000200	\$NSKO = 000300	\$\$FLAG= 000001
UIPAR1= 177642	WBUFSZ= 000142	\$\$\$NAM= 000160	\$NSK1 = 000110	\$\$FROM= 000000
UIPAR2= 177644	WDFR = 000116	\$\$\$NO = 000403	\$NSK2 = 000110	\$\$LOC = 000516R
UIPAR3= 177646	WDTO = 000114	\$\$\$OR = 000320	\$NSK3 = 000120	\$\$LOCN= 000000
UIPAR4= 177650	WTINRE= 000352	\$\$\$RTI= 000350	\$NSK4 = 000110	\$\$REG = 177777
UIPAR5= 177652	WTWHMI= 000222	\$\$\$RTN= 000300	\$\$AVLE= 177777	\$\$RETU= 000000
UIPAR6= 177654	XFLAG = 000005	\$\$\$SEL= 000140	\$\$SKO = 050026	\$\$RTN1= 050000
UIPAR7= 177656	XOFF = 000023	\$\$\$THE= 000330	\$TAGLE= 177777	\$\$RTN2= 050001
UIPDR0= 177600	XON = 000021	\$\$\$TRU= 000404	\$TAGNU= 050034	\$\$SRC = 000000
UIPDR1= 177602	\$BGNLE= 177777	\$\$\$UNT= 000130	\$TEMP = 000300	\$\$TGSV= 000000
UIPDR2= 177604	\$ERFLG= 000400	\$\$\$WHI= 000120	\$TSKO = 050033	\$\$TGS1= 000000
UIPDR3= 177606	\$\$\$AND= 000310	\$\$\$YES= 000402	\$TSK1 = 050031	\$\$TGS2= 000000
UIPDR4= 177610	\$\$\$BAD= 000401	\$IFLEV= 177777	\$TSK2 = 050025	\$\$TO = 000000
UIPDR5= 177612	\$\$\$BLA= 000170	\$ISK0 = 000001	\$TSK3 = 050026	\$\$\$TAG= 050000
UIPDR6= 177614	\$\$\$CAS= 000150	\$ISK1 = 000001	\$TSK4 = 050016	= 000542R
UIPDR7= 177616	\$\$\$DEC= 000220	\$ISK2 = 000001	\$\$ARGC= 000002	
WASADR= 000104	\$\$\$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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:42  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1040

3 COMMON EQUATE MODULE  
557 COMMON DEFINITIONS AND REFERENCES  
560 000000' .PRINT ;SPMAC: VERSION 1.1  
631 WSTBUS ROUTINE  
782 ROUTINE TO CONTROL THE GENERATION OF TWO 16-BIT PATTERN WORDS.  
837 ROUTINE TO PRODUCE A ROTATING PATTERN ALTERNATING WITH A CONSTANT.  
868 ROUTINE TO PRODUCE A BLOCK OF TWO ALTERNATING CONSTANT VALUES.  
909 ROUTINE TO REPORT MEMORY WRITE ERRORS.

```
508 .TITLE WSTBUS - LOAD WORST-CASE UNIBUS PATTERN
509 .IDENT /V0.0/
510
511 ;++
512 ;
513 ; MODULE NAME:
514 ; WSTBUS
515 ;
516 ; FUNCTIONAL DESCRIPTION:
517 ; ASSUMES EXERCISER IS IN LOWEST MEMORY AND LOADS ALL
518 ; OF FREE MEMORY WITH WORST-CASE UNIBUS PATTERNS.
519 ;
520 ; INPUTS:
521 ; DATA TABLE ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ; DT.ST0,DT.SSIZ,DT.EXS
525 ;
526 ; OUTPUTS:
527 ; NONE
528 ;
529 ; IMPLICIT OUTPUTS:
530 ; NONE
531 ;
532 ; PATHOLOGICAL CONNECTIONS:
533 ; NONE
534 ;
535 ; SUBORDINATE MODULES CALLED:
536 ; SAVREG ;SAVE REGISTERS
537 ; RESREG ;RESTORE REGISTERS
538 ; BADMEM ;OUTPUT BAD MEMORY MESSAGE
539 ; GPA ;GET PHYSICAL ADDRESSES
540 ;
541 ; FUNCTIONAL SIDE EFFECTS:
542 ; NONE
543 ;
544 ; CALLING SEQUENCE:
545 ; CALL WSTBUS IN <A>
546 ; A=ADDRESS OF DATA TABLE
547 ;
548 ; VERSION:
549 ; 0.0
550 ;
551 ; EDIT BY DATE REASON
552 ;
553 ;
554 ;
555 ;
```



```
557 .SBTTL COMMON DEFINITIONS AND REFERENCES
558
559 .MCALL STRUCT
560 000000' STRUCT
561 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
562 000001 $LSTIN=1
563 000001 $LSTTAG=1
564 ;
565 ;*****
566 ;
567 ; REFERENCED BY OTHER MODULES
568 ;
569 .GLOBL WSTBUS ;MODULE ENTRY POINT
570 ;
571 ;*****
572 ;
573 ; GLOBAL REFERENCES
574 ;
575 .GLOBL SAVREG ;SAVE REGISTERS
576 .GLOBL RESREG ;RESTORE REGISTERS
577 .GLOBL BADMEM ;OUTPUT BAD MEMORY MESSAGE
578 .GLOBL GPA ;GET PHYSICAL ADDRESS
579 ;
580 ;*****
581 ;
582 ; LOCAL STORAGE
583 ;
584 000000' 000000 WB.SAV: .WORD 0 ;LOCAL STORAGE WORD
585 000002' 000000 WB.DONE: .WORD 0 ;CURRENT PATTERN COMPLETE FLAG
586 000004' 000000 WB.CNT: .WORD 0 ;CURRENT PATTERN WORD COUNT
587 000006' 000400 WB.MAX: .WORD ^D256 ;PATTERN BLOCK SIZE
588 000010' 000000 WB.ROTCNT: .WORD 0 ;WORD ROTATION COUNTER FOR ROUTINE PAT2
589 000012' 000000 WB.TMP: .WORD 0 ;TEMPORARY STORAGE ON ERROR
590 000014' 000003 WB.TBL: .WORD 3 ;TABLE FOR GPA ROUTINE
591 000016' 177777 WB.SD1: .WORD 177777 ;PATTERN SEEDS SELECTED BY THE
592 000020' 177776 WB.SD2: .WORD 177776 ;
593 ; INITIALIZATION ROUTINES. THESE
594 ; SEEDS CAN BE CHANGED IF IT IS DESIRED
595 ; TO CHANGE THE PATTERNS THAT ARE
596 ; PRODUCED.
597 ;*****
598 ;
599 ;
600 ; THE MODULE IS SET UP TO PRODUCE THE FOLLOWING PATTERNS:
601 ;
602 ; 1. BUBBLE ALTERNATING WITH ALL 1'S.
603 ; ... ,177777,177776,177777,177775,177777,...
604 ;
605 ; 2. BUBBLE ALTERNATING WITH ALL 0'S.
606 ; ... ,000000,177776,000000,177775,000000,...
607 ;
608 ; 3. TWO ALTERNATING CONSTANTS. ONE CONSTANT IS ALL 1'S, AND ONE
609 ; CONSTANT HAS A BUBBLE WHICH SHIFTS AFTER EACH BLOCK.
610 ;
611 ;
```

```
612 ;  
613 ; 177777,177776,177777,177776,...  
614 ; 177777,177775,177777,177775,...  
615 ;  
616 ;  
617 ;  
618 ;  
619 ; 4. TWO ALTERNATING CONSTANTS. ONE CONSTANT IS ALL 0'S, AND  
620 ; ONE CONSTANT HAS A BUBBLE WHICH SHIFTS AFTER EACH BLOCK.  
621 ;  
622 ;  
623 ;  
624 ; 000000,177776,000000,177776,...  
625 ; 000000,177775,000000,177775,...  
626 ;  
627 ;  
628 ;  
629 ;
```

```

631          .SBTTL WSTBUS ROUTINE
632
633 000022'  ROUTINE WSTBUS <TABL>
(2) 000022'
634
635          ;+
636          ; SAVE REGISTERS
637          ; -
638
639 000022'  CALL SAVREG
(3) 000022' 004767 000000G          JSR      PC, SAVREG
640
641
642          ;+
643          ; SET R0 TO THE START OF THE DATA TABLE
644          ; -
645
646 000026'  LET R0 := TABL(R5)
(4) 000026' 016500 000000          MOV      TABL(R5), R0
647
648
649          ;+
650          ; INITIALIZE THE DONE FLAG, THE PATTERN WORD COUNT, AND
651          ; THE ROTATION COUNTER.
652          ; -
653
654 000032'  LET WB.DONE := #1
(4) 000032' 012767 000001 177742          MOV      #1, WB.DONE
655 000040'  LET WB.CNT := #0
(4) 000040' 005067 177740          CLR      WB.CNT
656 000044'  LET WB.ROTCNT := #0
(4) 000044' 005067 177740          CLR      WB.ROTCNT
657
658
659          ;+
660          ; SEE IF MEMORY MANAGEMENT IS IN OPERATION.
661          ; -
662
663 000050'  IF #KTSTAT SETIN DT.STO(R0) THEN
(6) 000050' 032760 000020 000010          BIT      #KTSTAT, DT.STO(R
(9) 000056' 001452          BEQ      50002$
664
665
666          ;+
667          ; SAVE THE CONTENTS OF KIPAR6 AND THEN SET IT UP FOR USE AS THE
668          ; MEMORY LOCATION POINTER BY POINTING IT TO THE 4K BANK THAT
669          ; CONTAINS THE TOP OF THE EXERCISER.
670          ; -
671
672 000060'  PUSH @#KIPAR6
(2) 000060' 013746 172354          MOV      @#KIPAR6, -(SP)
673 000064'  LET R1 := DT.ESIZ(R0) SHIFT #-6
(4) 000064' 016001 000044          MOV      DT.ESIZ(R0), R1
(7) 000070' 006201          ASR      R1
(7) 000072' 006201          ASR      R1
(7) 000074' 006201          ASR      R1

```

```

(7) 000076' 006201
(7) 000100' 006201
(7) 000102' 006201
674 000104'
(4) 000104' 010137 172354
(6) 000110' 042737 176177 172354
675
676
677
678
679
680
681
682 000116'
(4) 000116' 016001 000044
(6) 000122' 042701 160000
683 000126'
(6) 000126' 052701 140000
684 000132'
(6) 000132' 062701 000002
685
686
687
688
689
690
691 000136'
(4) 000136'
(6) 000136' 023760 172354 000046
(9) 000144' 103014
692
693
694
695
696
697
698 000146'
(4) 000146'
(6) 000146' 020127 157776
(9) 000152' 103003
699 000154'
(3) 000154' 004767 000142
700 000160'
(4) 000160' 000772
(3) 000162'
701 000162'
(4) 000162' 012701 140000
702 000166'
(6) 000166' 062737 000200 172354
703
704 000174'
(4) 000174' 000760
(3) 000176'
705
706
707

```

```

;+
; SET R1 TO SELECT KIPAR6 AND POINT TO THE FIRST WORD ABOVE
; THE TOP OF THE EXERCISER.
;-

LET @#KIPAR6 := R1 CLR.BY #176177

MOV R1,@#KIPAR6
BIC #176177,@#KIPAR6

LET R1 := DT.ESIZ(R0) CLR.BY #160000

MOV DT.ESIZ(R0),R1
BIC #160000,R1

LET R1 := R1 SET.BY #140000

BIS #140000,R1

LET R1 := R1 + #2

ADD #2,R1

;+
; WHILE NOT AT THE TOP OF MEMORY...
;-

WHILE @#KIPAR6 LO DT.SSIZ(R0) DO

50003$:
CMP @#KIPAR6,DT.SSIZ
BHS 50004$

;+
; GENERATE PATTERN WORDS, TWO AT A TIME, AND WRITE THEM INTO MEMORY,
; ADJUSTING THE PAR- AND REGISTER- CONTENTS AS NECESSARY.
;-

WHILE R1 LO #157776 DO

50005$:
CMP R1,#157776
BHS 50006$

CALL GENPAT

ENDDO

JSR PC,GENPAT

BR 50005$

50006$:
LET R1 := #140000

MOV #140000,R1

LET @#KIPAR6 := @#KIPAR6 + #200

ADD #200,@#KIPAR6

ENDDO

BR 50003$

50004$:

```

```
708 ; RESTORE KIPAR6 TO ITS ORIGINAL VALUE.
709 ; -
710
711 000176' POP @#KIPAR6
(2) 000176' 012637 172354 MOV (SP)+,@#KIPAR6
712
713
714 ; +
715 ; ELSE MEMORY MANAGEMENT IS OFF.
716 ; -
717
718 000202' ELSE
(4) 000202' 000444 BR 50007$
(3) 000204' 50002$:
719
720 ; +
721 ; USE WB.SAV AS THE HIGH MEMORY LIMIT. BECAUSE OF
722 ; ADDRESSING LIMITATIONS WITHOUT KT, IF TOTAL MEMORY
723 ; IS GREATER THAN OR EQUAL TO 28K THEN THE HIGH LIMIT IS
724 ; 28K. OTHERWISE, THE HIGH LIMIT IS THE TOP OF MEMORY.
725 ; -
726
727 000204' IF DT.SSIZ(R0) HIS #1600 THEN
(6) 000204' 026027 000046 001600 CMP DT.SSIZ(R0),#160
(9) 000212' 103404 BLO 50010$
728 000214' LET WB.SAV := #160000
(4) 000214' 012767 160000 177556 MOV #160000,WB.SAV
729 000222' ELSE
(4) 000222' 000417 BR 50011$
(3) 000224' 50010$:
730 000224' LET WB.SAV := DT.SSIZ(R0) SHIFT #+6
(4) 000224' 016067 000046 177546 MOV DT.SSIZ(R0),WB.S
(7) 000232' 006367 177542 ASL WB.SAV
(7) 000236' 006367 177536 ASL WB.SAV
(7) 000242' 006367 177532 ASL WB.SAV
(7) 000246' 006367 177526 ASL WB.SAV
(7) 000252' 006367 177522 ASL WB.SAV
(7) 000256' 006367 177516 ASL WB.SAV
731 000262' ENDIF
(4) 000262' 50011$:
732
733
734 ; +
735 ; SUBTRACT 2 FROM THE HIGH LIMIT, SO WE CAN WRITE TWO WORDS AT
736 ; A TIME INTO MEMORY WITHOUT TRAPPING OUT AT THE TOP OF CORE.
737 ; -
738
739 000262' LET WB.SAV := WB.SAV - #2
(6) 000262' 162767 000002 177510 SUB #2,WB.SAV
740
741
742 ; +
743 ; SET UP R1 AS THE MEMORY ADDRESS POINTER
744 ; -
745 000270' LET R1 := DT.ESIZ(R0) + #2
(4) 000270' 016001 000044 MOV DT.ESIZ(R0),R1
```



```

777          ;+
778          ; *** SUBROUTINES ***
779          ; -
780
781          .SBTTL ROUTINE TO CONTROL THE GENERATION OF TWO 16-BIT PATTERN WORDS.
782
783          ROUTINE GENPAT
784          000322'
(2)          000322'
785
786
787          ;+
788          ; IF WE HAVE COMPLETED ONE PATTERN, CLEAR THE DONE FLAG AND SET UP
789          ; TO GET THE NEXT PATTERN ROUTINE. THEN RE-INITIALIZE THE PATTERN WORDS
790          ; (R3 AND R4) BY COMPLEMENTING ONE SEED AND LEAVING THE OTHER CONSTANT.
791          ; -
792
793          000322' IF WB.DONE EQ #1 THEN
(6)          000322' 026727 177454 000001          CMP          WB.DONE,#1
(9)          000330' 001020          BNE          50002$
794          000332'          LET WB.DONE := #0          CLR          WB.DONE
(4)          000332' 005067 177444          IF R2 EQ #PAT1 THEN          CMP          R2,#PAT1
795          000336'          IF R2 EQ #PAT1 THEN          BNE          50003$
(6)          000336' 020227 000432'          LET R2 := #PAT2          MOV          #PAT2,R2
(9)          000342' 001003          ELSE          BR          50004$
796          000344'          LET R2 := #PAT2          BR          50004$
(4)          000344' 012702 000502'          ELSE          BR          50004$
797          000350'          ELSE          BR          50004$
(4)          000350' 000402          LET R2 := #PAT1          BR          50004$
(3)          000352'          LET R2 := #PAT1          BR          50004$
798          000352'          LET R2 := #PAT1          BR          50004$
(4)          000352' 012702 000432'          ENDIF          BR          50004$
799          000356'          ENDIF          BR          50004$
(4)          000356'          ENDIF          BR          50004$
800
801          000356'          LET WB.SD1 := COMP WB.SD1          COM          WB.SD1
(6)          000356' 005167 177434          LET R3 := WB.SD1          MOV          WB.SD1,R3
802          000362'          LET R3 := WB.SD1          MOV          WB.SD1,R3
(4)          000362' 016703 177430          LET R4 := WB.SD2          MOV          WB.SD2,R4
803          000366'          LET R4 := WB.SD2          MOV          WB.SD2,R4
(4)          000366' 016704 177426          ENDIF
804
805          000372'          ENDIF          BR          50002$
(4)          000372'          ENDIF          BR          50002$
806
807
808          ;+
809          ; CALL THE PROPER PATTERN ROUTINE.
810          ; R2 = PAT1 OR PAT2
811          ; -
812
813          000372'          CALL (R2)          JSR          PC,(R2)
(3)          000372' 004712
814
815
816          ;+

```

```
B17 ; MOVE THE TWO PATTERN WORDS INTO MEMORY. AFTER EACH MOVE, CHECK FOR A
B18 ; WRITE ERROR. IF AN ERROR IS FOUND, GO REPORT IT.
B19 ;-
B20
B21 000374' LET (R1) := R3
(4) 000374' 010311 MOV R3,(R1)
B22 000376' IF (R1)+ NE R3 THEN
(6) 000376' 022103 CMP (R1)+,R3
(9) 000400' 001404 BEQ 50005$
B23 000402' LET WB.TMP := R3
(4) 000402' 010367 177404 MOV R3,WB.TMP
B24 000406' CALL ERROR
(3) 000406' 004767 000160 JSR PC,ERROR
B25 000412' ENDIF
(4) 000412' 50005$:
B26 000412' LET (R1) := R4
(4) 000412' 010411 MOV R4,(R1)
B27 000414' IF (R1)+ NE R4 THEN
(6) 000414' 022104 CMP (R1)+,R4
(9) 000416' 001404 BEQ 50006$
B28 000420' LET WB.TMP := R4
(4) 000420' 010467 177366 MOV R4,WB.TMP
B29 000424' CALL ERROR
(3) 000424' 004767 000142 JSR PC,ERROR
B30 000430' ENDIF
(4) 000430' 50006$:
B31
B32
B33 000430' ENDRTN
(3) 000430' 50000$:
(3) 000430' 50001$:
(2) 000430' 000207 RTS PC
B34
```



```

836
837          .SBTTL ROUTINE TO PRODUCE A ROTATING PATTERN ALTERNATING WITH A CONSTANT.
838
839 000432'  ROUTINE PAT1
(2) 000432'          PAT1:
840
841
842          ;+
843          ; CALCULATE A NEW SECOND PATTERN WORD. (THE FIRST PATTERN WORD IS
844          ; CONSTANT.)
845          ;-
846
847 000432'  IF #BIT15 SETIN R4 THEN
(6) 000432' 032704 100000          BIT    #BIT15,R4
(9) 000436' 001402          BEQ    50002$
848 000440'          INLINE <SEC>          SEC
(2) 000440' 000261          ELSE
849 000442'          ELSE
(4) 000442' 000401          BR    50003$
(3) 000444'          50002$:
850 000444'          INLINE <CLC>          CLC
(2) 000444' 000241          ENDIF
851 000446'          ENDIF
(4) 000446'          50003$:
852 000446'  LET R4 := R4 ROTATE #+1          ROL    R4
(7) 000446' 006104
853
854          ;+
855          ; ADD TWO TO THE PATTERN COUNT AND SEE IF WE HAVE REACHED THE MAXIMUM.
856          ; IF YES, THEN CLEAR THE COUNT AND SET THE DONE FLAG.
857          ;-
858
859 000450'  LET WB.CNT := WB.CNT + #2          ADD    #2,WB.CNT
(6) 000450' 062767 000002 177326          IF WB.CNT GE WB.MAX THEN
860 000456'          CMP    WB.CNT,WB.MAX
(6) 000456' 026767 177322 177322          BLT    50004$
(9) 000464' 002405          LET WB.CNT := #0          CLR    WB.CNT
861 000466'          LET WB.DONE := #1          MOV    #1,WB.DONE
(4) 000466' 005067 177312          ENDIF
862 000472'          LET WB.DONE := #1          MOV    #1,WB.DONE
(4) 000472' 012767 000001 177302          ENDIF
863 000500'          50004$:
(4) 000500'          50000$:
864          50001$:
865 000500'          ENDRTN          RTS    PC
(3) 000500'
(3) 000500'
(2) 000500' 000207
866

```

```

868          .SBTTL ROUTINE TO PRODUCE A BLOCK OF TWO ALTERNATING CONSTANT VALUES.
869
870
871          ;+
872          ; AFTER A BLOCK IS COMPLETE, ONE OF THE WORDS IS CHANGED (THE OTHER
873          ; ALWAYS REMAINS CONSTANT) AND ANOTHER BLOCK IS PRODUCED. A
874          ; ROTATING PATTERN IS USED IN THE WORD THAT IS CHANGED. THE PATTERN IS
875          ; COMPLETE WHEN THE CHANGING WORD HAS ROTATED ALL THE WAY AROUND
876          ; TO THE STARTING POINT.
877          ;-
878
879          ROUTINE PAT2                                PAT2:
(2) 000502'
880
881
882          ;+
883          ; ADD TWO TO THE PATTERN WORD COUNT.
884          ; IF THE PATTERN WORD COUNT HAS REACHED THE MAXIMUM, THEN ROTATE THE
885          ; CHANGING WORD. IF THE WORD HAS ROTATED COMPLETELY, THEN
886          ; SET THE DONE FLAG.
887          ;-
888
889          000502' LET WB.CNT := WB.CNT + #2
(6) 000502' 062767 000002 177274          ADD      #2,WB.CNT
890          000510' IF WB.CNT GE WB.MAX THEN
(6) 000510' 026767 177270 177270          CMP      WB.CNT,WB.MAX
(9) 000516' 002424          BLT      50002$
891          000520' LET WB.CNT := #0
(4) 000520' 005067 177260          CLR      WB.CNT
892          000524' IF #BIT15 SET IN R4 THEN
(6) 000524' 032704 100000          BIT      #BIT15,R4
(9) 000530' 001402          BEQ      50003$
893          000532'          INLINE <SEC>
(2) 000532' 000261          SEC
894          000534' ELSE
(4) 000534' 000401          BR      50004$
(3) 000536'          50003$:
895          000536'          INLINE <CLC>
(2) 000536' 000241          CLC
896          000540'          ENDIF
(4) 000540'          50004$:
897          000540' LET R4 := R4 ROTATE #+1
(7) 000540' 006104          ROL      R4
898          000542' LET WB.ROTCNT := WB.ROTCNT + #1
(6) 000542' 005267 177242          INC      WB.ROTCNT
899          000546' IF WB.ROTCNT EQ #^D16 THEN
(6) 000546' 026727 177236 000020          CMP      WB.ROTCNT,#^D16
(9) 000554' 001005          BNE      50005$
900          000556' LET WB.ROTCNT := #0
(4) 000556' 005067 177226          CLR      WB.ROTCNT
901          000562' LET WB.DGNE := #1
(4) 000562' 012767 000001 177212          MOV      #1,WB.DONE
902          000570'          ENDIF
(4) 000570'          50005$:
903          000570'          ENDIF
(4) 000570'          50002$:

```

904 000570'  
(3) 000570'  
(3) 000570'  
(2) 000570' 000207  
905  
906

ENDRTN

50000\$:  
50001\$:

RTS PC

```

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
(2) 000572'
921
922 000572'          INLINE <TST      -(R1)>
(2) 000572' 005741
923 000574'          LET WB.TBL := R1
(4) 000574' 010167 177214
924 000600'          CALL GPA IN <R0,#WB.TBL>
(3) 000600' 010546
(5) 000602' 012745 000014'
(4) 000606' 010045
(3) 000610' 004767 000000G
(3) 000614' 012605
925 000616'          CALL BADMEM IN <R0,#WB.TBL+2,WB.TMP,(R1)>
(3) 000616' 010546
(7) 000620' 011145
(6) 000622' 016745 177164
(5) 000626' 012745 000016'
(4) 000632' 010045
(3) 000634' 004767 000000G
(3) 000640' 012605
926 000642'          INLINE <TST      (R1)+>
(2) 000642' 005721
927 000644'          ENDRTN
(3) 000644'
(3) 000644'
(2) 000644' 000207
928
929
930          000001          .END

```

```

ERROR:
TST      -(R1)
MOV      R1,WB.TBL
MOV      R5,-(SP)
MOV      #WB.TBL,-(R5)
MOV      R0,-(R5)
JSR      PC,GPA
MOV      (SP)+,R5
MOV      R5,-(SP)
MOV      (R1),-(R5)
MOV      WB.TMP,-(R5)
MOV      #WB.TBL+2,-(R5)
MOV      R0,-(R5)
JSR      PC,BADMEM
MOV      (SP)+,R5
TST      (R1)+
50000$:
50001$:
RTS      PC

```

ACSR = 000102	CSRC = 000102	EOPBIT= 000001	MEMPAS= 040000	RANNUM= 000054
ACTBIT= 004000	CTRLC = 000003	ERROR 000572R	MODEXH= 004000	RBUFEA= 000130
ADDR22= 001000	CTRL0 = 000017	ERRTYP= 000106	MODHOL= 002000	RBUFPA= 000126
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	MODSEL= 001000	RBUFSZ= 000132
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	MSGCKD= 000010	RBUFVA= 000124
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	MSGCKS= 000011	RDSERV= 000101
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	MSGDER= 000005	RDWHMI= 000022
ASSEMB= 000010	DROPMO= 100000	EVNTRE= 000201	MSGDRP= 000017	RELERR= 000020
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGGECH= 177777	RELMOD= 020000
AUTO = 000010	DT.ADD= 000042	GENPAT 000322R	MSGGEOP= 000013	RELTIM= 010000
AUTOST= 020000	DT.AP = 000100	GPA = ***** G	MSGHDR= 000004	RESREG= ***** G
AWAS = 000110	DT.APK= 000076	HRDCNT= 000044	MSGHNG= 000022	RES1 = 000056
BADMEM= ***** G	DT.BLS= 000034	HRDPAS= 000050	MSGHRD= 000007	RES2 = 000060
BIT0 = 000001	DT.CFO= 000014	ICONT = 000036	MSGMAP= 000021	RICHAR= 031060
BIT00 = 000001	DT.CF1= 000016	ICOUNT= 000040	MSGNUL= 177775	RPTDAT= 002000
BIT01 = 000002	DT.ERR= 000020	IDNUM = 000122	MSGPOP= 000002	RSTRT = 000112
BIT02 = 000004	DT.ESI= 000044	IE = 000100	MSGPRM= 177776	RUBOUT= 000177
BIT03 = 000010	DT.EVN= 000000	INDPAR= 000040	MSGRES= 000001	RUNMOD= 100000
BIT04 = 000020	DT.EXS= 000060	INHDRP= 040000	MSGSFT= 000006	RSVALU= 001740
BIT05 = 000040	DT.FCH= 000037	INHEPR= 020000	MSGSKE= 000003	SAM = 075464
BIT06 = 000100	DT.FCN= 000036	INHREL= 001000	MSGSMB= 000015	SAVREG= ***** G
BIT07 = 000200	DT.HMX= 000104	INHRR= 000400	MSGSMH= 000014	SBADR = 000102
BIT08 = 000400	DT.KBE= 000024	INIT = 000030	MSGSMS= 000016	SBKMOD= 000000
BIT09 = 001000	DT.KBP= 000026	INTR = 000120	MSGSTD= 000000	SBKSEL= 010000
BIT1 = 000002	DT.KBR= 000022	IOMOD = 100000	MSGSYS= 000012	SC.ADR= 000006
BIT10 = 002000	DT.KBU= 000030	IOMODP= 102000	MSGVEC= 000020	SC.ALC= 000014
BIT11 = 004000	DT.MLS= 000032	IOMODR= 112000	NBKMOD= 001000	SC.APC= 000016
BIT12 = 010000	DT.MTI= 000110	IOMODX= 110000	NCPUOP= 000020	SC.CKL= 000002
BIT13 = 020000	DT.OFF= 000070	JACK = 035060	NOPTY= 000002	SC.CKP= 000004
BIT14 = 040000	DT.PAS= 000074	KIPAR0= 172340	NULL = 000000	SC.CLO= 000000
BIT15 = 100000	DT.PC = 000002	KIPAR1= 172342	OWEN = 024020	SC.HLD= 000010
BIT2 = 000004	DT.PFL= 000062	KIPAR2= 172344	PAERR = 000010	SC.SCA= 000012
BIT3 = 000010	DT.PSW= 000004	KIPAR3= 172346	PARPRE= 002000	SENDLS= 177777
BIT4 = 000020	DT.PTA= 000064	KIPAR4= 172350	PARSTA= 000100	SOFcnt= 000042
BIT5 = 000040	DT.RCS= 000102	KIPAR5= 172352	PASCNT= 000034	SOPPAS= 000046
BIT6 = 000100	DT.REL= 000040	KIPAR6= 172354	PAT1 000432R	SPACE = 000040
BIT7 = 000200	DT.SCT= 000066	KIPAR7= 172356	PAT2 000502R	SPOINT= 000032
BIT8 = 000400	DT.SMX= 000106	KIPDR0= 172300	PDPLSI= 020000	SPVALU= 002200
BIT9 = 001000	DT.SP = 000006	KIPDR1= 172302	PDP60 = 004000	SRO = 177572
BKDEF = 000002	DT.SSI= 000046	KIPDR2= 172304	PDP70 = 010000	SR1 = 177574
BKMOD = 000020	DT.STO= 000010	KIPDR3= 172306	PRI0 = 000000	SR2 = 177576
BKMODE= 040000	DT.ST1= 000012	KIPDR4= 172310	PRI1 = 000040	SR3 = 172516
BKSLSH= 000134	DT.SWR= 000056	KIPDR5= 172312	PRI4 = 000200	STAT = 000026
CAPRES= 000004	DT.SYP= 000072	KIPDR6= 172314	PRI5 = 000240	STATBI= 064757
CASTAT= 000004	DT.WBU= 000050	KIPDR7= 172316	PRI6 = 000300	STAT1 = 000027
CDERCT= 000146	DT.WHL= 000054	KTERRO= 000040	PRI7 = 000340	SUSPND= 000001
CDWDCT= 000144	DT.WLL= 000052	KTPRES= 000400	PRO = 000000	SVR0 = 000062
CKTIM = 100000	DVID1 = 000014	KTSTAT= 000020	PR4 = 000200	SVR1 = 000064
CLKPRE= 000001	ECCMEM= 000100	KTXTND= 040000	PR5 = 000240	SVR2 = 000066
CONFIG= 000056	ECCSTA= 000010	LF = 000012	PR6 = 000300	SVR3 = 000070
CQOVF = 000001	ENBEOP= 010000	LPSTAT= 000001	PR7 = 000340	SVR4 = 000072
CR = 000015	ENBNUL= 000001	MAPSTA= 000200	PS = 177776	SVR5 = 000074
CSRA = 000100	ENDLST= 000000	MED = 076600	PSW = 177776	SVR6 = 000076

SYSCNT= 000052	WBSTAT= 000040	\$BGNLE= 177777	\$F\$WHI= 000120	\$TSK4 = 050006
SYSERR= 000100	WBUFEA= 000136	\$ERFLG= 000400	\$F\$YES= 000402	\$SARGC= 000000
TABL = 000000	WBUFPA= 000134	\$F\$AND= 000310	\$IFLEV= 177777	\$S\$BYTE= 000403
TMPIO = 000002	WBUFRO= 000140	\$F\$BAD= 000401	\$ISK0 = 000001	\$SCASE= 000000
TQOVF = 000002	WBUFSZ= 000142	\$F\$BLA= 000170	\$ISK1 = 000001	\$SDST = 000000
UIPAR0= 177640	WB.CNT 000004R	\$F\$CAS= 000150	\$LOCTA= 177777	\$SELOC= 000402
UIPAR1= 177642	WB.DON 000002R	\$F\$DEC= 000220	\$LSTIN= 000001	\$SERFL= 000000
UIPAR2= 177644	WB.MAX 000006R	\$F\$DD = 000340	\$LSTTA= 000001	\$FLAG= 000001
UIPAR3= 177646	WB.ROT 000010R	\$F\$FAL= 000405	\$NESTL= 177777	\$FROM= 000000
UIPAR4= 177650	WB.SAV 000000R	\$F\$G00= 000400	\$NSK0 = 000300	\$SLOC = 000554R
UIPAR5= 177652	WB.SD1 000016R	\$F\$IF = 000110	\$NSK1 = 000110	\$SLOCN= 000000
UIPAR6= 177654	WB.SD2 000020R	\$F\$INC= 000210	\$NSK2 = 000110	\$REG = 177777
UIPAR7= 177656	WB.TBL 000014R	\$F\$L00= 000200	\$NSK3 = 000120	\$RETI= 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	XDN = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:43  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1056

3 COMMON EQUATE MODULE  
521 KBDIV - COMMON DEFINITIONS AND REFERENCES  
526 000000' .PRINT ;SPMAC: VERSION 1.1  
581 KBINI - KEYBOARD I/O INITIATOR ROUTINE  
623 KBINI - KBINI ROUTINE  
662 KBINT - KEYBOARD INTERRUPT SERVICE ROUTINE  
710 KBINT - ROUTINE KBINT

```
508 .TITLE KBDRV - KEYBOARD DRIVER ROUTINE
509 .IDENT /V0.0/
510 ;++
511 ; MODULE PACKAGE NAME:
512 ; KBDRV
513 ;
514 ; FUNCTIONAL DECSRIPTION:
515 ; THIS MODULE CONTAINS THE TWO ROUTINES: KBINI AND KBINT, WHICH TOGETHER
516 ; MAKE UP THE KEYBOARD DRIVER.
517 ;
518 ;
519 ;--
```



```

521 .SBTTL KBDRV - COMMON DEFINITIONS AND REFERENCES
522
523
524
525 .MCALL STRUCT
526 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
527 000001 $LSTIN = 1
528 000001 $LSITAG = 1
529
530
531
532 ;
533 ;*****
534 ;
535 ; REFERENCED BY OTHER MODULES
536 ;
537 .GLOBL KB.INP
538 .GLOBL KB.RBP
539 .GLOBL KB.RBF
540 .GLOBL KB.XFLAG
541 .GLOBL KBINT
542 .GLOBL KBINI
543 ;
544 ;*****
545 ;
546 ; GLOBAL REFERENCES
547 ;
548 .GLOBL DTABLE ;DTABLE ADDRESS
549 .GLOBL CTRL0F ;ENABLE/DISABLE TERMINAL OUTPUT FLAG
550 .GLOBL OV.KBBUF ;KEYBOARD BUFFER OVERLAY
551 .GLOBL OV.KBSIZ ;KEYBOARD BUFFER SIZE
552 .GLOBL OV.HIKB ;HIGH BUFFER ADDRESS
553 .GLOBL PCTRLC ;PROCESS A CTRLC
554 .GLOBL DX.KFL
555 .GLOBL DX.R5 ;MONITOR R5
556 .GLOBL DX.SP ;MONITOR R6
557 .GLOBL MD.BSY ;MESSAGE BUSY FLAG
558 ;
559 ;*****
560 ;
561 ; DL11 DEVICE REGISTER DEFINITIONS
562 ;
563 .GLOBL RCSR ;RECEIVER CONTROL AND STATUS REGISTER
564 .GLOBL RBUF ;RECEIVER DATA BUFFER REGISTER
565 .GLOBL XCSR ;TRANSMITTER CONTROL AND STATUS REGISTER
566 ;
567 ;*****
568 ;
569 ; LOCAL STORAGE - PROGRAM IMPURE STORAGE
570 ;
571 000000' 000000 KB.XFLAG: .WORD 0 ;XOFF FLAG WORD INDICATOR
572 000002' 000000 KB.CNT: .WORD 0 ;COUNT OF PREVIOUSLY RUBBED OUT CHARACTERS
573 000004' 000000 KB.INP: .WORD 0 ;POINTER TO KEYBOARD INPUT BUFFER
574 000006' 000000 KB.RBP: .WORD 0 ;POINTER TO NEXT CHAR TO BE RUBBED OUT
575 000010' 000000 KB.RBF: .WORD 0 ;NUMBER OF RUBOUTS RECEIVED

```

576 000012' 000000G  
577 000014' 000000G  
578 000016' 000000G  
579

KB.KBSIZ: .WORD OV.KBSIZ ;KEYBOARD BUFFER SIZE  
KB.HIKB: .WORD OV.HIKB ;KEYBOARD BUFFER HIGH LIMIT  
KB.KBBUF: .WORD OV.KBBUF ;KEYBOARD BUFFER ADDRESS

```
581 .SBTTL KBINI - KEYBOARD I/O INITIATOR ROUTINE
582 ;++
583 ; ROUTINE NAME:
584 ; KBINI
585 ;
586 ; FUNCTIONAL DESCRIPTION:
587 ; THE I/O INITIATOR ROUTINE PERFORMS THE NECESSARY
588 ; SET-UP FUNCTIONS, SUCH AS CLEARING THE KEYBOARD BUFFER AND RESETTING
589 ; THE APPROPRIATE POINTERS TO ALLOW THE NEXT KEYBOARD COMMAND TO BE
590 ; ENTERED. A RETURN TO THE CALLING ROUTINE IS THEN EXECUTED.
591 ;
592 ; INPUTS:
593 ; 1. ADDRESS OF DATA TABLE
594 ;
595 ; IMPLICIT INPUTS:
596 ; 1. ADDRESS OF NEXT CHAR TO BE ECHOED (DATA TABLE + DT.KBECH)
597 ;
598 ; OUTPUTS:
599 ; NONE
600 ;
601 ; IMPLICIT OUTPUTS:
602 ; NONE
603 ;
604 ; PATHOLOGICAL CONNECTIONS:
605 ; KEYBOARD CSR
606 ;
607 ; SUBORDINATE MODULES CALLED:
608 ; NONE
609 ;
610 ; FUNCTIONAL SIDE EFFECTS:
611 ; NONE
612 ;
613 ; CALLING SEQUENCE:
614 ; CALL KBINI IN <DTABLE>
615 ;
616 ; VERSION:
617 ; 0.0
618 ;
619 ; EDIT BY DATE REASON
620 ;--
621
```

```

623          .SBTTL  KBINI - KBINI ROUTINE
624
625
626 000020'          ROUTINE KBINI <DTA>
(2) 000020'
627
628
629          ;+
630          ; DO THE NECESSARY SETUP STUFF, WHICH CONSISTS OF GETTING THE DATA TABLE
631          ; ADDRESS
632          ; -
633 000020'          PUSH    R0,R1
(2) 000020' 010046          MOV     R0,-(SP)
(3) 000022' 010146          MOV     R1,-(SP)
634 000024'          LET R1 := DTA(R5)
(4) 000024' 016501 000000          MOV     DTA(R5),R1
635
636          ;+
637          ; INITIALIZE THE INPUT, ECHO, AND RUBOUT POINTERS, AS
638          ; WELL AS THE RUBOUT FLAG
639          ; -
640 000030'          LET R0 := KB.KBBUF
(4) 000030' 016700 177762          MOV     KB.KBBUF,R0
641 000034'          LET KB.INP := R0
(4) 000034' 010067 177744          MOV     R0,KB.INP
642 000040'          LET DT.KBECH(R1) := R0
(4) 000040' 010061 000024          MOV     R0,DT.KBECH(R1)
643 000044'          LET KB.RBP := #0
(4) 000044' 005067 177736          CLR     KB.RBP
644 000050'          LET KB.RBF := #0
(4) 000050' 005067 177734          CLR     KB.RBF
645
646          ;+
647          ; ZERO THE KEYBOARD INPUT BUFFER
648          ; -
648 000054'          LET R1 := KB.KBSIZ
(4) 000054' 016701 177732          MOV     KB.KBSIZ,R1
649 000060'          WHILE R1 GT #0 DO
(4) 000060'
(6) 000060' 005701          50002$: TST     R1
(9) 000062' 003403          BLE     50003$
650 000064'          LET (R0)+ := #0
(4) 000064' 105020          CLRB   (R0)+
651 000066'          LET R1 := R1 - #1
(6) 000066' 005301          DEC     R1
652 000070'          ENDDO
(4) 000070' 000773          BR     50002$
(3) 000072'          50003$:
653
654          ;+
655          ; RESTORE THE REGISTERS, ENABLE INTERRUPTS
656          ; AND RETURN TO CALLER
657          ; -
658 000072'          POP     R1,R0
(2) 000072' 012601          MOV     (SP)+,R1
(3) 000074' 012600          MOV     (SP)+,R0
659 000076'          LET @RCSR := #<IE>

```

(4) 000076' 012777 000100 000000G  
660 000104'  
(3) 000104'  
(3) 000104'  
(2) 000104' 000207

ENDRTN

MOV #<IE>,@RCSR

50000\$:  
50001\$:

RTS PC

```
662 .SBTTL KBINT - KEYBOARD INTERRUPT SERVICE ROUTINE
663 ;++
664 ; ROUTINE NAME:
665 ; KBINT
666 ;
667 ; FUNCTIONAL DESCRIPTION:
668 ; THE INTERRUPT SERVICE ROUTINE IS ENTERED DIRECTLY
669 ; VIA THE HARDWARE INTERRUPT VECTOR AND PERFORMS THE NEC-
670 ; ESSARY FUNCTIONS TO HANDLE THE INTERRUPT. UPON RECEIVING
671 ; A LINE TERMINATOR EVENT, CONTROL IS PASSED TO THE APPROPRIATE
672 ; SERVICE ROUTINE, OR THE EVENT-PENDING FLAG IS SET.
673 ; ALSO, ALL LOWER CASE INPUT IS CONVERTED TO UPPER CASE
674 ;
675 ; INPUTS:
676 ; NONE
677 ;
678 ; IMPLICIT INPUTS:
679 ; NONE
680 ;
681 ; OUTPUTS:
682 ; EVENT CODE
683 ;
684 ; IMPLICIT OUTPUTS:
685 ; NONE
686 ;
687 ; PATHOLOGICAL CONNECTIONS:
688 ; 1. CTRLQF ;CONTROL-Q FLAG
689 ; 2. DX.KFL
690 ; 3. DTABLE
691 ;
692 ; SUBORDINATE ROUTINES CALLED:
693 ; NONE
694 ;
695 ; FUNCTIONAL SIDE EFFECTS:
696 ; IF XOFF (^S) IS GENERATED BY KEYBOARD, THE RESULT WILL
697 ; BE TO DISABLE TDRV UNTIL XON (^Q) IS GENERATED.
698 ;
699 ; CALLING SEQUENCE:
700 ; ENTERED DIRECTLY VIA THE HARDWARE INTERRUPT MECHANISM
701 ;
702 ; VERSION:
703 ; 0.0
704 ;
705 ; EDIT BY DATE REASON
706 ;--
707
708
```

```

710          .SBTTL  KBINT - ROUTINE KBINT
711
712
713          ;+
714          ; DISABLE INTERRUPTS, GET THE CHARACTER ONTO THE STACK
715          ; AND STRIP-OFF THE ASCII PARITY BIT
716          ; -
717
718          IROUTINE KBINT
719          (2) 000106'          KBINT:
720          (6) 000106' 042777 000100 000000G  LET @RCSR := @RCSR CLR.BY #<IE>          BIC    #<IE>,@RCSR
721          (4) 000114' 117746 000000G  LET -(SP) :B= @RBUF          MOVB  @RBUF,-(SP)
722          (6) 000120' 042716 177600  LET (SP) := (SP) CLR.BY #177600          BIC    #177600,(SP)
723
724          ;+
725          ; CONVERT LOWER CASE TO UPPER CASE
726          ; -
727          (6) 000124'          IFB (SP) GE #141 ANDB (SP) LE #172 THEN          CMPB  (SP),#141
728          (9) 000130' 002405          BLT    50002$
729          (6) 000132' 121627 000172          CMPB  (SP),#172
730          (9) 000136' 003002          BGT    50002$
731          (6) 000140'          LET (SP) :B= (SP) CLR.BY #BIT05          BICB  #BIT05,(SP)
732          (6) 000140' 142716 000040  ENDF
733          (4) 000144'          50002$:
734
735          ;+
736          ; IF XOFF(^S) THEN SET XFLAG
737          ; -
738          IFB (SP) EQ #<XOFF> THEN          CMPB  (SP),#<XOFF>
739          (6) 000144' 121627 000023          BNE   50003$
740          (9) 000150' 001005          LET KB.XFLAG := #1          MOV   #1,KB.XFLAG
741          (4) 000152' 012767 000001 177620  INLINE <JMP 2$>          JMP  2$
742          (2) 000160' 000167 000426  ENDF
743          (4) 000164'          50003$:
744
745          ;+
746          ; IF XON(^Q) THEN CLEAR XFLAG AD IF MESSAGE IN PROGRESS
747          ; SET TRANSMITTER INTERRUPT ENABLE
748          ; -
749          IFB (SP) EQ #<XON> THEN          CMPB  (SP),#<XON>
750          (6) 000164' 121627 000021          BNE   50004$
751          (9) 000170' 001012          LET KB.XFLAG := #0          CLR  KB.XFLAG
752          (4) 000172' 005067 177602          IF MD.BSY NE #0 THEN
753          (4) 000176'

```

```

(6) 000176' 005767 000000G
(9) 000202' 001403
748 000204'
(6) 000204' 052777 000100 000000G
749 000212'
(4) 000212'
750 000212'
(2) 000212' 000167 000374
751 000216'
(4) 000216'
752
753
754
755
756
757
758 000216'
(6) 000216' 121627 000003
(9) 000222' 001017
759 000224'
(2) 000224' 012746 000340
760 000230'
(2) 000230' 012746 000236'
761 000234'
(2) 000234' 000002
762 000236'
(2) 000236'
763 000236'
(4) 000236' 016705 000000G
764 000242'
(4) 000242' 016706 000000G
765 000246'
(3) 000246' 010546
(4) 000250' 012745 000000G
(3) 000254' 004767 000000G
(3) 000260' 012605
766 000262'
(4) 000262'
767
768
769
770
771 000262'
(6) 000262' 121627 000015
(8) 000266' 001403
(6) 000270' 121627 000012
(9) 000274' 001006
(6) 000276'
772
773
774
775
776
777 000276'
(6) 000276' 005767 177506
(9) 000302' 001402

```

```

          LET @XCSR := @XCSR SET.BY #<IE>
          ENDIF
          50005$:
          INLINE <JMP 2$>
          JMP 2$
          50004$:
          ;+
          ; IF THE CHARACTER JUST ENTERED IS A CONTROL-C, THEN RAISE THE
          ; PRIORITY TO 7, SWITCH TO MONITOR STACKS AND DISPATCH
          ; CONTROL TO PROCESS "CONTROL C" ROUTINE.
          ;-
          IFB (SP) EQ #<CTRLC> THEN
          CMPB (SP),#<CTRLC>
          BNE 50006$
          PUSH #PRI7
          MOV #PRI7,-(SP)
          PUSH #1$
          MOV #1$,-(SP)
          INLINE <RTI>
          RTI
          INLINE <1$:>
          1$:
          LET R5 := DX.R5
          MOV DX.R5,R5
          LET SP := DX.SP
          MOV DX.SP,SP
          CALL PCTRLC IN <#DTABLE>
          MOV R5,-(SP)
          MOV #DTABLE,-(R5)
          JSR PC,PCTRLC
          MOV (SP)+,R5
          ENDIF
          50006$:
          ;+
          ; IF THE CHARACTER JUST ENTERED IS A CARRIAGE RETURN OR A LINE FEED, THEN
          ;-
          IFB (SP) EQ #<CR> ORB (SP) EQ #<LF> THEN
          CMPB (SP),#<CR>
          BEQ 50007$
          CMPB (SP),#<LF>
          BNE 50010$
          50007$:
          ;+
          ; IF THERE HAVE BEEN RUBOUTS, THEN LOAD THE BUFFER WITH RUBOUT CHARACTER
          ; AND UPDATE INPUT BUFFER POINTER
          ;-
          IF KB.RBF NE #0 THEN
          TST KB.RBF
          BEQ 50011$

```



```

778 000304'          INLINE <JSR PC,22222$>
(2) 000304' 004767 000432          JSR PC,22222$
779 000310'          ENDIF
(4) 000310'          50011$:
780
781          ;+
782          ; GO LOAD BUFFER
783          ; -
784
785 000310'          INLINE <BR 11111$>
(2) 000310' 000546          BR 11111$
786 000312'          ENDIF
(4) 000312'          50010$:
787
788          ;+
789          ; IF THE CHARACTER IS A CONTROL-U, THEN SAVE IT IN THE
790          ; INPUT BUFFER, DEFER SERVICING AND THEN SET THE KEYBOARD FLAG.
791          ; -
792
793 000312'          IFB (SP) EQ #<CTRLU> THEN
(6) 000312' 121627 000025          CMPB (SP),#<CTRLU>
(9) 000316' 001010          BNE 50012$
794 000320'          IF KB.RBF NE #0 THEN
(6) 000320' 005767 177464          TST KB.RBF
(9) 000324' 001402          BEQ 50013$
795 000326'          INLINE <JSR PC,22222$>
(2) 000326' 004767 000410          JSR PC,22222$
796 000332'          ENDIF
(4) 000332'          50013$:
797
798          ;+
799          ; GO LOAD UP INPUT CHARACTER INTO BUFFER
800          ; -
801
802 000332'          INLINE <JSR PC,33333$>
(2) 000332' 004767 000430          JSR PC,33333$
803
804          ;+
805          ; NOW LOAD UP <CR> AND <LF> AND LEAVE
806          ; -
807
808 000336'          INLINE <BR 11111$>
(2) 000336' 000533          BR 11111$
809 000340'          ENDIF
(4) 000340'          50012$:
810
811          ;+
812          ; IF THE CHARACTER IS A CONTROL-O THEN SIMPLY TOGGLE THE
813          ; ENABLE/DISABLE FLAG AND UPDATE THE STACK
814          ; -
815 000340'          IFB (SP) EQ #<CTRL0> THEN
(6) 000340' 121627 000017          CMPB (SP),#<CTRL0>
(9) 000344' 001011          BNE 50014$
816 000346'          IF CTRL0F NE #0 THEN
(6) 000346' 005767 000000G          TST CTRL0F
(9) 000352' 001403          BEQ 50015$

```

```

817 000354'          LET CTRL0F := #0
(4) 000354' 005067 000000G          CLR      CTRL0F
818 000360'          ELSE
(4) 000360' 000402          BR        50015$
(3) 000362'          50015$:
819 000362'          LET CTRL0F := CTRL0F + #1          INC      CTRL0F
(6) 000362' 005267 000000G          50016$:
820 000366'          ENDIF
(4) 000366'          50016$:
821 000366'          ELSE
(4) 000366' 000511          BR        50017$
(3) 000370'          50014$:
822
823 ;+
824 ; IF THE CHARACTER IS A RUBOUT, DO THE NECESSARY STUFF
825 ;-
826 000370'          IFB (SP) EQ #<RUBOUT> THEN
(6) 000370' 121627 000177          CMPB    (SP),#<RUBOUT>
(9) 000374' 001075          BNE     50020$
827
828 ;+
829 ; MAKE SURE THERE ARE CHARACTERS TO RUBOUT!!!!!!!!!!
830 ;-
831
832 000376'          IF KB.INP GT KB.KBBUF THEN
(6) 000376' 026767 177402 177412          CMP     KB.INP,KB.KBBUF
(9) 000404' 003465          BLE     50021$
833 000406'          IF KB.RBF EQ #0 OR KB.RBP GE KB.KBBUFF THEN
(6) 000406' 005767 177376          TST     KB.RBF
(8) 000412' 001404          BEQ     50022$
(6) 000414' 026767 177366 177374          CMP     KB.RBP,KB.KBBUFF
(9) 000422' 002456          BLT     50023$
(6) 000424'          50022$:
834
835 ;+
836 ; IS IT THE FIRST RUBOUT?
837 ;-
837 000424'          IF KB.RBF EQ #0 THEN
(6) 000424' 005767 177360          TST     KB.RBF
(9) 000430' 001012          BNE     50024$
838
839 ;+
840 ; FOR THE FIRST RUBOUT, SET UP THE RUBOUT FLAG
841 ; AND RUBOUT POINTER, AND STORE THE RUBOUT CHAR
842 ; IN THE INPUT BUFFER
843 ;-
844 000432'          LET KB.RBF := KB.RBF + #1
(6) 000432' 005267 177352          INC     KB.RBF
845 000436'          INLINE <JSR PC,33333$>
(2) 000436' 004767 000324          JSR    PC,33333$
846 000442'          LET KB.RBP := KB.INP -- #2
(4) 000442' 016767 177336 177336          MOV     KB.INP,KB.RBP
(6) 000450' 162767 000002 177330          SUB     #2,KB.RBP
847 000456'          ENDIF
(4) 000456'          50024$:
848
849 ;+

```

```

850                                     ; DELETE THE EXISTING CHARACTER, BACK UP THE RUBOUT POINTER, AND IF
851                                     ; THERE ARE ANY PREVIOUSLY RUBBED OUT CHARACTERS IN THE BUFFER,
852                                     ; MOVE THE RUBOUT POINTER PAST THEM.
853                                     ;-
854 000456'                               LET @KB.INP :B= @KB.RBP
855 (4) 000456' 117777 177324 177320      MOVB   @KB.RBP,@KB.INP
856 000464'                               IF KB.INP LT KB.HIKB THEN
857 (6) 000464' 026767 177314 177322      CMP    KB.INP,KB.HIKB
858 (9) 000472' 002002                    BGE    50025$
859 000474'                               LET KB.INP := KB.INP + #1
860 (6) 000474' 005267 177304            INC    KB.INP
861 000500'                               ENDIF
862 (4) 000500'                               50025$:
863 000500'                               LET KB.RBP := KB.RBP - #1
864 (6) 000500' 005367 177302            DEC    KB.RBP
865 000504'                               IFB @KB.RBP EQ #RUBOUT THEN
866 (6) 000504' 127727 177276 000177      CMPB   @KB.RBP,#RUBOUT
867 (9) 000512' 001021                    BNE    50026$
868 000514'                               LET KB.CNT := #0
869 (4) 000514' 005067 177262            CLR    KB.CNT
870 000520'                               REPEAT
871 (3) 000520'                               50027$:
872 (6) 000520' 005367 177262            LET KB.RBP := KB.RBP - #1
873 000524'                               DEC    KB.RBP
874 (6) 000524' 005267 177252            LET KB.CNT := KB.CNT + #1
875 000530'                               INC    KB.CNT
876 (3) 000530' 127727 177252 000177      UNTILB @KB.RBP EQ #RUBOUT
877 (6) 000536' 001370                    CMPB   @KB.RBP,#RUBOUT
878 000540'                               BNE    50027$
879 (3) 000540'                               REPEAT
880 000540'                               50030$:
881 (6) 000540' 005367 177242            LET KB.RBP := KB.RBP - #1
882 000544'                               DEC    KB.RBP
883 (6) 000544' 005367 177232            LET KB.CNT := KB.CNT - #1
884 000550'                               DEC    KB.CNT
885 (3) 000550' 005767 177226            UNTIL KB.CNT EQ #0
886 (6) 000554' 001371                    TST    KB.CNT
887 000556'                               BNE    50030$
888 (4) 000556'                               ENDIF
889 000556'                               50026$:
890 (2) 000556' 000415                    INLINE <BR 2$>
891 000560'                               BR 2$
892 000560'                               ENDIF
893 (4) 000560'                               50023$:
894 000560'                               ENDIF
895 (4) 000560'                               50021$:
896 000560'                               ;+
897 (2) 000560' 004767 000202            ; THERE AREN'T ANY CHARACTERS TO RUBOUT, STORE A RUBOUT AND LOAD CRLF
898 000564'                               ; NOTE: THE LAST ENTRY IN KEYBOARD BUFFER IS OVERWRITTEN BY A <CR>
899 (2) 000564' 000420                    INLINE <JSR PC,33333$>
900 000564'                               JSR PC,33333$
901 (2) 000564' 000420                    INLINE <BR 11111$>
902 000564'                               BR 11111$

```

```

881
882           000566'           ELSE
(4) 000566' 000411
(3) 000570'
883
884
885           ;+
886           ; IF IT IS NOT THE FIRST RUBOUT CHARACTER THEN CLEAR THE RUBOUT FLAG
887           ; AND LOAD RUBOUT CHARACTER INTO BUFFER
888           ; -
889           000570'           IF KB.RBF NE #0 THEN
(6) 000570' 005767 177214           TST   KB.RBF
(9) 000574' 001404           BEQ   50032$
890           000576'           LET KB.RBF := #0
(4) 000576' 005067 177206           CLR   KB.RBF
891           000602'           INLINE <JSR PC,22222$>
(2) 000602' 004767 000134           JSR PC,22222$
892           000606'           ENDIF
(4) 000606'           50032$:
893
894           ;+
895           ; IT IS NOT THE FIRST NONRUBOUT CHAR, SO SIMPLY
896           ; STORE THE CHAR IN THE INPUT BUFFER, UPDATE THE
897           ; POINTER AND DO A NICE CLEAN RETURN
898           ; -
899           000606'           INLINE <JSR PC,33333$>
(2) 000606' 004767 000154           JSR PC,33333$
901           000612'           ENDIF
(4) 000612'           50031$:
902           000612'           ENDIF
(4) 000612'           50017$:
903
904           ;+
905           ; ENABLE INTERRUPTS AND RETURN
906           ; -
907           000612'           INLINE <2$:>
(2) 000612'           2$:
908           000612'           LET SP := SP + #2
(6) 000612' 062706 000002           ADD   #2,SP
909           000616'           LET @RCSR := #<IE>
(4) 000616' 012777 000100 000000G           MOV   #<IE>,@RCSR
910           000624'           ENDRTI
(3) 000624'           50000$:
(3) 000624'           50001$:
(2) 000624' 000002           RTI
911
912           ;+
913           ; LINE TERMINATOR EXIT
914           ; EITHER A <CR> OR <LF> TERMINATOR, OR A CONTROL-U HAS BEEN RECEIVED-
915           ; SET THE FLAG
916           ; AND LOAD <CR> AND <LF> INTO BUFFER, ALONG WITH UPDATING STACK
917           ; -
918           000626'           INLINE <11111$:>
(2) 000626'           11111$:

```

```

920 000626' LET DX.KFL := #1
(4) 000626' 012767 000001 000000G MOV #1,DX.KFL
921 000634' PUSH R0,R1
(2) 000634' 010046 MOV R0,-(SP)
(3) 000636' 010146 MOV R1,-(SP)
922 000640' IF 4(SP) EQ #<LF> THEN
(6) 000640' 026627 000004 000012 CMP 4(SP),#<LF>
(9) 000646' 001005 BNE 50033$
923 000650' LET R0 := #<LF>
(4) 000650' 012700 000012 MOV #<LF>,R0
924 000654' LET R1 := #<CR>
(4) 000654' 012701 000015 MOV #<CR>,R1
925 000660' ELSE
(4) 000660' 000404 BR 50034$
(3) 000662' 50033$:
926 000662' LET R0 := #<CR>
(4) 000662' 012700 000015 MOV #<CR>,R0
927 000666' LET R1 := #<LF>
(4) 000666' 012701 000012 MOV #<LF>,R1
928 000672' ENDIF
(4) 000672' 50034$:
929 000672' LET @KB.INP :B= R0
(4) 000672' 110077 177106 MOVB R0,@KB.INP
930 000676' IF KB.INP LT KB.HIKB THEN
(6) 000676' 026767 177102 177110 CMP KB.INP,KB.HIKB
(9) 000704' 002002 BGE 50035$
931 000706' LET KB.INP := KB.INP + #1
(6) 000706' 005267 177072 INC KB.INP
932 000712' ENDIF
(4) 000712' 50035$:
933 000712' LET @KB.INP :B= R1
(4) 000712' 110177 177066 MOVB R1,@KB.INP
934 000716' IF KB.INP LT KB.HIKB THEN
(6) 000716' 026767 177062 177070 CMP KB.INP,KB.HIKB
(9) 000724' 002002 BGE 50036$
935 000726' LET KB.INP := KB.INP + #1
(6) 000726' 005267 177052 INC KB.INP
936 000732' ENDIF
(4) 000732' 50036$:
937
938 ;+
939 ; NOW CLEAN UP STACK, RESTORING REGISTERS AND REMOVING INPUT CHARACTER.
940 ; -
941
942 000732' POP R1,R0
(2) 000732' 012601 MOV (SP)+,R1
(3) 000734' 012600 MOV (SP)+,R0
943 000736' INLINE <TST (SP)+>
(2) 000736' 005726 TST (SP)+
944 000740' INLINE <RTI>
(2) 000740' 000002 RTI
945
946
947 ;+
948 ; LOAD RUBOUT INTO BUFFER AND UPDATE POINTER
949 ; -

```

```

950
951 000742'          INLINE <22222$:>
(2) 000742'
952 000742'          LET @KB.INP :B= #<RUBOUT>
(4) 000742' 112777 000177 177034
953 000750'          IF KB.INP LT KB.HIKB THEN
(6) 000750' 026767 177030 177036
(9) 000756' 002002
954 000760'          LET KB.INP := KB.INP + #1
(6) 000760' 005267 177020
955 000764'          ENDIF
(4) 000764'
956 000764'          INLINE <RTS PC>
(2) 000764' 000207
957
958 ;+
959 ; LOAD CHARACTER INTO BUFFER
960 ; -
961
962 000766'          INLINE <33333$:>
(2) 000766'
963 000766'          LET @KB.INP :B= 2(SP)
(4) 000766' 116677 000002 177010
964 000774'          IF KB.INP LT KB.HIKB THEN
(6) 000774' 026767 177004 177012
(9) 001002' 002002
965 001004'          LET KB.INP := KB.INP + #1
(6) 001004' 005267 176774
966 001010'          ENDIF
(4) 001010'
967 001010'          INLINE <RTS PC>
(2) 001010' 000207
968 000001          .END

```

```

22222$:
MOVB #<RUBOUT>,@KB.IN
CMP KB.INP,KB.HIKB
BGE 50037$
INC KB.INP
50037$:
RTS PC
33333$:
MOVB 2(SP),@KB.INP
CMP KB.INP,KB.HIKB
BGE 50040$
INC KB.INP
50040$:
RTS PC

```

ACSR = 000102	CTRLC = 000003	ECCMEM= 000100	KIPDR4= 172310	PDP60 = 004000
ACTBIT= 004000	CTRLO = 000017	ECCSTA= 000010	KIPDR5= 172312	PDP70 = 010000
ADDR22= 001000	CTRL0F= ***** G	ENBEO= 010000	KIPDR6= 172314	PRI0 = 000000
ADR = 000006	CTRLU = 000025	ENBNUL= 000001	KIPDR7= 172316	PRI1 = 000040
APTFER= 000004	DCEVNT= 000011	ENDLST= 000000	KTERRO= 000040	PRI4 = 000200
APTPRE= 000200	DEFRTN= 000400	EOPBIT= 000001	KTPRES= 000400	PRI5 = 000240
ASB = 000106	DIAGMC= 000000	ERRTYP= 000106	KTSTAT= 000020	PRI6 = 000300
ASSEMB= 000010	DROPMD= 100000	EVNTBE= 000200	KTXTND= 040000	PRI7 = 000340
ASTAT = 000104	DSEVNT= 000014	EVNTHD= 000200	LF = 000012	PRO = 000000
AUTO = 000010	DTA = 000000	EVNTKT= 000203	LPSTAT= 000001	PR4 = 000200
AUTOST= 020000	DTABLE= ***** G	EVNTPE= 000202	MAPSTA= 000200	PR5 = 000240
AWAS = 000110	DT.ADD= 000042	EVNTRE= 000201	MD.BSY= ***** G	PR6 = 000300
BIT0 = 000001	DT.AP = 000100	FATERR= 100000	MED = 076600	PR7 = 000340
BIT00 = 000001	DT.APK= 000076	HRDCNT= 000044	MEMPAS= 040000	PS = 177776
BIT01 = 000002	DT.BLS= 000034	HRDPAS= 000050	MODEXH= 004000	PSW = 177776
BIT02 = 000004	DT.CFO= 000014	ICONT = 000036	MODHOL= 002000	RANNUM= 000054
BIT03 = 000010	DT.CF1= 000016	ICOUNT= 000040	MODSEL= 001000	RBUF = ***** G
BIT04 = 000020	DT.ERR= 000020	IDNUM = 000122	MSGCKD= 000010	RBUFEA= 000130
BIT05 = 000040	DT.ESI= 000044	IE = 000100	MSGCKS= 000011	RBUFPA= 000126
BIT06 = 000100	DT.EVN= 000000	INDPAR= 000040	MSGDER= 000005	RBUFSZ= 000132
BIT07 = 000200	DT.EXS= 000060	INHDRP= 040000	MSGDRP= 000017	RBUFVA= 000124
BIT08 = 000400	DT.FCH= 000037	INHEPR= 020000	MSGECH= 177777	RCSR = ***** G
BIT09 = 001000	DT.FCN= 000036	INHREL= 001000	MSGEOP= 000013	RDSERV= 000101
BIT1 = 000002	DT.HMX= 000104	INHRE= 000400	MSGHDR= 000004	RDWHMI= 000022
BIT10 = 002000	DT.KBE= 000024	INIT = 000030	MSGHNG= 000022	RELERR= 000020
BIT11 = 004000	DT.KBP= 000026	INTR = 000120	MSGHRD= 000007	RELMOD= 020000
BIT12 = 010000	DT.KBR= 000022	IOMOD = 100000	MSGMAP= 000021	RELTIM= 010000
BIT13 = 020000	DT.KBU= 000030	IOMODP= 102000	MSGNUL= 177775	RES1 = 000056
BIT14 = 040000	DT.MLS= 000032	IOMODR= 112000	MSGPOP= 000002	RES2 = 000060
BIT15 = 100000	DT.MTI= 000110	IOMODX= 110000	MSGPRM= 177776	RICHAR= 031060
BIT2 = 000004	DT.OFF= 000070	JACK = 035060	MSGRES= 000001	RPTDAT= 002000
BIT3 = 000010	DT.PAS= 000074	KBINI 000020RG	MSGSFT= 000006	RSTRT = 000112
BIT4 = 000020	DT.PC = 000002	KBINT 000106RG	MSGSKE= 000003	RUBOUT= 000177
BIT5 = 000040	DT.PFL= 000062	KB.CNT 000002R	MSGSMB= 000015	RUNMOD= 100000
BIT6 = 000100	DT.PSW= 000004	KB.HIK 000014R	MSGSMH= 000014	RSVALU= 001740
BIT7 = 000200	DT.PTA= 000064	KB.INP 000004RG	MSGSMS= 000016	SAM = 075464
BIT8 = 000400	DT.RCS= 000102	KB.KBB 000016R	MSGSTD= 000000	SBADR = 000102
BIT9 = 001000	DT.REL= 000040	KB.KBS 000012R	MSGSYS= 000012	SBKMOD= 000000
BKDEF = 000002	DT.SCT= 000066	KB.RBF 000010RG	MSGVEC= 000020	SBKSEL= 010000
BKMOD = 000020	DT.SMX= 000106	KB.RBP 000006RG	NBKMOD= 001000	SC.ADR= 000006
BKMODE= 040000	DT.SP = 000006	KB.XFL 000000RG	NCPUOP= 000020	SC.ALC= 000014
BKSLSH= 000134	DT.SSI= 000046	KIPAR0= 172340	NOAPTY= 000002	SC.APC= 000016
CAPRES= 000004	DT.STO= 000010	KIPAR1= 172342	NULL = 000000	SC.CKL= 000002
CASTAT= 000004	DT.ST1= 000012	KIPAR2= 172344	OV.HIK= ***** G	SC.CKP= 000004
CDERCT= 000146	DT.SWR= 000056	KIPAR3= 172346	OV.KBB= ***** G	SC.CLO= 000000
CDWDCT= 000144	DT.SYP= 000072	KIPAR4= 172350	OV.KBS= ***** G	SC.HLD= 000010
CKTIM = 100000	DT.WBU= 000050	KIPAR5= 172352	DWEN = 024020	SC.SCA= 000012
CLKPRE= 000001	DT.WHL= 000054	KIPAR6= 172354	PAERR = 000010	SENDLS= 177777
CONFIG= 000056	DT.WLL= 000052	KIPAR7= 172356	PARPRE= 002000	SOFCNT= 000042
CQOVF = 000001	DVID1 = 000014	KIPDR0= 172300	PARSTA= 000100	SOFPAS= 000046
CR = 000015	DX.KFL= ***** G	KIPDR1= 172302	PASCNT= 000034	SPACE = 000040
CSRA = 000100	DX.R5 = ***** G	KIPDR2= 172304	PCTRLC= ***** G	SPOINT= 000032
CSRC = 000102	DX.SP = ***** G	KIPDR3= 172306	PDPLSI= 020000	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\$LOD= 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	WBUFSSZ= 000142	\$F\$RTI= 000350	\$\$SAVLE= 177777	\$\$SRC = 000000
SYSERR= 000100	WDFR = 000116	\$F\$RTN= 000300	\$\$SSK0 = 050003	\$\$TGSV= 000000
TMPID = 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	\$ISKO = 000001	\$TSK4 = 050026	

. ABS. 000000 000  
001012 001

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

DSKZ:KBDV,DSKZ:KBDV=SPMAC/ML,EQUATE,KBDV  
RUN-TIME: 24 15 .4 SECONDS  
RUN-TIME RATIO: 60/39=1.5  
CORE USED: 14K (27 PAGES)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:44  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1074

3 COMMON EQUATE MODULE  
509 LPDRV STRUCTURED HEADER COVER SHEET  
535 MACRO-SPMAC ROUTINE (INTERRUPT SERVICE ROUTINE)  
540 COMMON DEFINITIONS AND REFERENCES  
542 000000' .PRINT ;SPMAC: VERSION 1.1  
582 LPINI (LINE PRINTER INITIATOR MODULE)  
636 LPINI (CODE)  
668 LPINT (LINE PRINTER INTERRUPT HANDLER)  
716 LPINT (CODE)

```
511 .TITLE LPDRV (LINE PRINTER OUTPUT DRIVER MODULE)
512 .IDENT /V0.0/
513
514 ;++
515 ; MODULE NAME:
516 ; LP
517 ;
518 ; FUNCTIONAL DESCRIPTION:
519 ; THIS MODULE CONTAINS TWO MODULES - LPINI AND LPINT.
520 ; THE LPINI IS THE INITIALIZATION ROUTINE FOR THE
521 ; OUTPUT INTERRUPT HANDLING ROUTINE - LPINT.
522 ; LPINT ACTUALLY PROCESSES ALL MESSAGES TO THE DEVICE.
523 ;
524 ; A DESCRIPTION IS GIVEN IN EACH MODULE.
525 ;
526 ; VERSION:
527 ; 0.0
528 ;
529 ; EDIT DATE BY REASON
530 ; -----
531 ;
532 ;--
533
```

```

542 000000' .PRINT ;SPMAC: VERSION 1.1
546 ;*****
547 ; GLOBAL REFERENCES
548 .GLOBL LPCSR ;DEVICE REGISTER - CONTROL & STATUS
549 .GLOBL LPBUF ;DEVICE REGISTER - DATA BUFFER
550 .GLOBL CTRLLOF ;FLAG - OUTPUT SUPPRESS
551 ;*****
552 ; REFERENCED BY OTHER MODULES
553 .GLOBL LPINT ;MODULE ENTRY POINT - LP HANDLER
554 .GLOBL LPINI ;MODULE ENTRY POINT - LP INITIATOR
555 ;*****
556 ; LOCAL EQUATES
557 000045 LP.PCNT = 45 ;CHARACTER - '%' PERCENT
558 000134 LP.BKSLSH = 134 ;CHARACTER - '\' BACKSLASH
559 ;*****
560 ; LOCAL STORAGE
561 000000' 000000 LP.FCH: 0 ;CHARACTER - FILL
562 000002' 000000 LP.PFC: 0 ;COUNT - FILL, PERMANENT
563 000004' 000000 LP.FCF: 0 ;COUNT, FLAG - FILL (FLAG SET NE 0)
564 000006' 000000 LP.EXT: 0 ;POINTER - LOCATION TO JUMP OUT INDIRECTLY
565 000010' 000000 LP.PTR: 0 ;POINTER - MESSAGE
566 000012' 000000 LP.TP: 0 ;POINTER - TEMPORARY BUFFER
567 000014' 000000 LP.COD: 0 ;CODE - MESSAGE
568 000016' LP.FLG: ;FLAGS - PERCENT AND CONTROL-U
569 000016' 000 LP.PCF: .BYTE 0 ;FLAG - PERCENT
570 000017' 000 LP.CUF: .BYTE 0 ;FLAG - CONTROL-U
571 000020' 015 LP.PCB: .BYTE 15 ;+BUFFER - PERCENT '%' <CR>+
572 000021' 012 .BYTE 12 ;+BUFFER ECHO <LF>+
573 000022' 001 .BYTE 1 ;+BUFFER - TERMINATOR <+1>+
574 000023' 136 LP.CUB: .BYTE 136 ;#BUFFER - CONTROL-U <^>#
575 000024' 125 .BYTE 125 ;#BUFFER ECHO <U>#
576 000025' 001 LP.BEN: .BYTE 1 ;#BUFFER - TERMINATOR <+1>#
577 ;FLAG - (LP.BEN) END SPECIAL BUFFER
578 .EVEN

```

```
584  
585  
586      ;++  
587      ; MODULE NAME:  
588      ;     LPINI  
589      ;  
590      ; FUNCTIONAL DESCRIPTION:  
591      ;     THE DEVICE OUTPUT STARTER MODULE (LPINI) SETS UP FOR  
592      ;     THE DEVICE OUTPUT INTERRUPT HANDLING MODULE (LPINT).  
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 LPINI 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      ;
```

```

638
639
640      ;+
641      ; INITIALIZE MESSAGE TYPE, MESSAGE
642      ; ADDRESS, FILL COUNT, AND FILL CHARACTER.
643      ; STUFF RETURN ADDRESS AT LP.EXT,
644      ; ENABLE INTERRUPTS AND RETURN.
645      ;-
646      ROUTINE LPINI <DTBL,MSGCOD,MSGADR,RETADR>
647      (2) 000026'          PUSH R0
648      (2) 000026' 010046          LET R0 := DTBL(R5)
649      (4) 000030' 016500 000000          LET LP.FCH := DT.FCH(R0)
650      (4) 000034' 116067 000037 177736          LET LP.PFC := DT.FCN(R0)
651      (4) 000042' 116067 000036 177732          LET LP.COD := MSGCOD(R5)
652      (4) 000050' 016567 000002 177736          LET LP.PTR := MSGADR(R5)
653      (4) 000056' 016567 000004 177724          LET LP.EXT := RETADR(R5)
654      (4) 000064' 016567 000006 177714          POP R0
655      (2) 000072' 012600          MOV R0,-(SP)
656
657
658      ;+
659      ; NOW FIRST CLEAR INTERRUPTS IN CASE THEY WERE STILL ENABLED (BY
660      ; THE LINE PRINTER OPTION MODULE OR SOME OTHER THING
661      ;-
662      (6) 000074' 042777 000100 000000G          BIC #IE,@LPCSR
663      (6) 000102' 052777 000100 000000G          BIS #IE,@LPCSR
664      (3) 000110'          ENDRTN
665      (3) 000110'          50000$:
666      (2) 000110' 000207          50001$:
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
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
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
  
```

670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712

```
;++  
; 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  
; -----  
; --
```

```

718 000112'          IROUTINE          LPINT
(2) 000112'
719                ;+
720                ;DISABLE INTERRUPTS.
721                ;+
722 000112'          LET @LPCSR := @LPCSR CLR.BY #IE
(6) 000112' 042777 000100 000000G          BIC          #IE,@LPCSR
723                ;+
724                ; IS FILL COUNT-FLAG NOT EQUAL TO ZERO ?
725                ; TRUE -          OUTPUT THE FILLER-CHARACTER
726                ;                DECREMENT FILL COUNT-FLAG
727                ;                SET INTERRUPTS, RETURN
728                ; FALSE -        CONTINUE.
729                ;+
730 000120'          IF LP.FCF NE #0 THEN
(6) 000120' 005767 177660          TST          LP.FCF
(9) 000124' 001406          BEQ          50002$
731 000126'          LET @LPBUF :B= #NULL
(4) 000126' 112777 000000 000000G          MOVB         #NULL,@LPBUF
732 000134'          LET LP.FCF := LP.FCF - #1
(6) 000134' 005367 177644          DEC          LP.FCF
733 000140'          ELSE
(4) 000140' 000524          BR          50003$
(3) 000142'
734                ;+
735                ; IS OUTPUT OF <^U> OR <%> BUFFER COMPLETED ?
736                ; TRUE -          RESET <^U> AND <%> FLAGS
737                ;                RESTORE MESSAGE POINTER
738                ; FALSE -        NO ACTION
739                ;+
740 000142'          IF LP.FLG NE #0 ANDB @LP.PTR EQ LP.BEN THEN
(6) 000142' 005767 177650          TST          LP.FLG
(9) 000146' 001411          BEQ          50004$
(6) 000150' 127767 177634 177647          CMPB         @LP.PTR,LP.BEN
(9) 000156' 001005          BNE          50004$
741 000160'          LET LP.FLG := #0
(4) 000160' 005067 177632          CLR          LP.FLG
742 000164'          LET LP.PTR := LP.TP
(4) 000164' 016767 177622 177616          MOV          LP.TP,LP.PTR
743 000172'          ENDIF
(4) 000172'          50004$:
744                ;+
745                ; IS INPUT CHARACTER A NULL ?
746                ; TRUE -          THIS IS THE END OF THE MESSAGE
747                ;                RETURN BY 'JMP @LP.EXT'.
748                ; FALSE -        NO ACTION
749                ;+
750 000172'          IFB @LP.PTR EQ #NULL THEN
(6) 000172' 127727 177612 000000          CMPB         @LP.PTR,#NULL
(9) 000200' 001007          BNE          50005$
751 000202'          IF LP.COD EQ #MSGPOP THEN
(6) 000202' 026727 177606 000002          CMP          LP.COD,#MSGPOP
(9) 000210' 001001          BNE          50006$
752 000212'          INLINE <CMP (SP)+,(SP)+>
(2) 000212' 022626          CMP (SP)+,(SP)+
753 000214'          ENDIF
    
```

```

(4) 000214'
754 000214'          50006$:
(2) 000214' 000177 177566      INLINE <JMP @LP.EXT;LEAVE,FINISHED MESSAGE>      JMP @LP.EXT;LEAVE,FINISH
755 000220'          ENDIF
(4) 000220'          50005$:
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 ; IS INPUT CHARACTER IS A <%> PERCENT ?
790 ; TRUE - SET %-FLAG
791 ; SAVE REST OF MESSAGE POINTER
792 ; POINT TO %-BUFFER
793 ; FALSE - NO ACTION
794 ;-
795 000314' IFB @LP.PTR EQ #LP.PCNT THEN
(6) 000314' 127727 177470 000045 CMPB @LP.PTR,#LP.PCNT
(9) 000322' 001013 BNE 50014$
796 000324' LET LP.PCF :B= #1
(4) 000324' 112767 000001 177464 MOVB #1,LP.PCF
797 000332' LET LP.TP := LP.PTR + #1
(4) 000332' 016767 177452 177452 MOV LP.PTR,LP.TP
(6) 000340' 005267 177446 INC LP.TP
798 000344' LET LP.PTR := #LP.PCB
(4) 000344' 012767 000020' 177436 MOV #LP.PCB,LP.PTR
799 000352' ENDF
(4) 000352' 50014$:
800 ;+
801 ; IS MESSAGE SURPRESS FLAG NOT SET ?
802 ; TRUE - OUTPUT CHARACTER
803 ; FALSE - NO ACTION
804 ;-
805 000352' IF CTRLQF NE #1 THEN
(6) 000352' 026727 000000G 000001 CMP CTRLQF,#1
(9) 000360' 001403 BEQ 50015$
806 000362' LET @LPBUF :B= @LP.PTR
(4) 000362' 117777 177422 000000G MOVB @LP.PTR,@LPBUF
807 000370' ENDF
(4) 000370' 50015$:
808 000370' ENDF
(4) 000370' 50013$:
809 ;+
810 ; IS INPUT CHARACTER THE "FILL-CHARACTER" ?
811 ; TRUE - PRESET FILL COUNT-FLAG
812 ; FALSE - NO ACTION
813 ;-
814 000370' IFB @LP.PTR EQ LP.FCH THEN
(6) 000370' 127767 177414 177402 CMPB @LP.PTR,LP.FCH
(9) 000376' 001003 BNE 50016$
815 000400' LET LP.FCF := LP.PFC
(4) 000400' 016767 177376 177376 MOV LP.PFC,LP.FCF
816 000406' ENDF
(4) 000406' 50016$:
817 ;+
818 ; INCREMENT MESSAGE CHARACTER POINTER
819 ;-
820 000406' LET LP.PTR := LP.PTR + #1
(6) 000406' 005267 177376 INC LP.PTR
821 000412' ENDF
(4) 000412' 50003$:
822 ;+
823 ;ENABLE INTERRUPTS. BUT FIRST RAISE PRIORITY TO 4
824 ;-
825
826 000412' PUSH #PRI4
  
```

(2) 000412' 012746 000200  
B27 000416' PUSH #1000\$  
(2) 000416' 012746 000424'  
B28 000422' INLINE <RTI>  
(2) 000422' 000002  
B29 000424' INLINE <1000\$:>  
(2) 000424'  
B30 000424' LET @LPCSR := @LPCSR SET.BY #IE  
(6) 000424' 052777 000100 000000G  
B31 000432' ENDRTI  
(3) 000432'  
(3) 000432'  
(2) 000432' 000002  
B32 000001 .END

MOV #PRI4,-(SP)  
MOV #1000\$,-(SP)  
RTI  
1000\$:  
BIS #IE,@LPCSR

50000\$:  
50001\$:

RTI

ACSR = 000102	CTRLC = 000003	ENDLST= 000000	LP.BEN 000025R	PASCNT= 000034
ACTBIT= 004000	CTRLO = 000017	EOPBIT= 000001	LP.BKS= 000134	PDPLSI= 020000
ADDR22= 001000	CTRLOF= ***** G	ERRTYP= 000106	LP.COD 000014R	PDP60 = 004000
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	LP.CUB 000023R	PDP70 = 010000
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	LP.CUF 000017R	PRI0 = 000000
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	LP.EXT 000006R	PRI1 = 000040
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	LP.FCF 000004R	PRI4 = 000200
ASSEMB= 000010	DROPMO= 100000	EVNTRE= 000201	LP.FCH 000000R	PRI5 = 000240
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	LP.FLG 000016R	PRI6 = 000300
AUTD = 000010	DTBL = 000000	HRDCNT= 000044	LP.PCB 000020R	PRI7 = 000340
AUTDST= 020000	DT.ADD= 000042	HRDPAS= 000050	LP.PCF 000016R	PRO = 000000
AWAS = 000110	DT.AP = 000100	ICONT = 000036	LP.PCN= 000045	PR4 = 000200
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	LP.PFC 000002R	PR5 = 000240
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	LP.PTR 000010R	PR6 = 000300
BIT01 = 000002	DT.CFO= 000014	IE = 000100	LP.TP 000012R	PR7 = 000340
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MAPSTA= 000200	PS = 177776
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MED = 076600	PSW = 177776
BIT04 = 000020	DT.ESI= 000044	INHEPR= 020000	MEMPAS= 040000	RANNUM= 000054
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MODEXH= 004000	RBUFEA= 000130
BIT06 = 000100	DT.EXS= 000060	INHRR= 000400	MODHOL= 002000	RBUFPA= 000126
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MODSEL= 001000	RBUFSZ= 000132
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGADR= 000004	RBUFVA= 000124
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGCKD= 000010	RDSERV= 000101
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGCKS= 000011	RDWHMI= 000022
BIT10 = 002000	DT.KBP= 000026	ICMODR= 112000	MSGCOD= 000002	RELERR= 000020
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	MSGDER= 000005	RELMOD= 020000
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	MSGDRP= 000017	RELTIM= 010000
BIT13 = 020000	DT.MLS= 000032	KIPAR0= 172340	MSGECH= 177777	RES1 = 000056
BIT14 = 040000	DT.MTI= 000110	KIPAR1= 172342	MSGEOP= 000013	RES2 = 000060
BIT15 = 100000	DT.OFF= 000070	KIPAR2= 172344	MSGHDR= 000004	RETADR= 000006
BIT2 = 000004	DT.PAS= 000074	KIPAR3= 172346	MSGHNG= 000022	RICHAR= 031060
BIT3 = 000010	DT.PC = 000002	KIPAR4= 172350	MSGHRD= 000007	RPTDAT= 002000
BIT4 = 000020	DT.PFL= 000062	KIPAR5= 172352	MSGMAP= 000021	RSTRT = 000112
BIT5 = 000040	DT.PSW= 000004	KIPAR6= 172354	MSGNUL= 177775	RUBOUT= 000177
BIT6 = 000100	DT.PTA= 000064	KIPAR7= 172356	MSGPOP= 000002	RUNMOD= 100000
BIT7 = 000200	DT.RCS= 000102	KIPDR0= 172300	MSGPRM= 177776	REVALU= 001740
BIT8 = 000400	DT.REL= 000040	KIPDR1= 172302	MSGRES= 000001	SAM = 075464
BIT9 = 001000	DT.SCT= 000066	KIPDR2= 172304	MSGSFT= 000006	SBADR = 000102
BKDEF = 000002	DT.SMX= 000106	KIPDR3= 172306	MSGSKE= 000003	SBKMOD= 000000
BKMOD = 000020	DT.SP = 000006	KIPDR4= 172310	MSGSMB= 000015	SBKSEL= 010000
BKMODE= 040000	DT.SSI= 000046	KIPDR5= 172312	MSGSMH= 000014	SC.ADR= 000006
BKSLSH= 000134	DT.STO= 000010	KIPDR6= 172314	MSGSMS= 000016	SC.ALC= 000014
CAPRES= 000004	DT.ST1= 000012	KIPDR7= 172316	MSGSTD= 000000	SC.APC= 000016
CASTAT= 000004	DT.SWR= 000056	KTERR0= 000040	MSGSYS= 000012	SC.CKL= 000002
CDERCT= 000146	DT.SYP= 000072	KTPRES= 000400	MSGVEC= 000020	SC.CKP= 000004
CDWDCT= 000144	DT.WBU= 000050	KTSTAT= 000020	NBKMOD= 001000	SC.CLO= 000000
CKTIM = 100000	DT.WHL= 000054	KTXTND= 040000	NCPUOP= 000020	SC.HLD= 000010
CLKPRE= 000001	DT.WLL= 000052	LF = 000012	NOAPTY= 000002	SC.SCA= 000012
CONFIG= 000056	DVID1 = 000014	LPBUF = ***** G	NULL = 000000	SENDLS= 177777
CQOVF = 000001	ECCMEM= 000100	LPCSR = ***** G	OWEN = 024020	SOFcnt= 000042
CR = 000015	ECCSTA= 000010	LPINI 000026RG	PAERR = 000010	SOPPAS= 000046
CSRA = 000100	ENBEDP= 010000	LPINT 000112RG	PARPRE= 002000	SPACE = 000040
CSRC = 000102	ENBNUL= 000001	LPSTAT= 000001	PARSTA= 000100	SPOINT= 000032

SPVALU= 002200	UIPAR3= 177646	XFLAG = 000005	\$FSTRU= 000404	\$\$ARGC= 000000
SR0 = 177572	UIPAR4= 177650	XOFF = 000023	\$FSUNT= 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	\$ISKO = 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	\$\$RETN= 000000
SVR3 = 000070	UIPDR7= 177616	\$F\$GDO= 000400	\$NSKO = 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\$LDO= 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
TMPID = 000002	WBUFSZ= 000142	\$F\$OR = 000320	\$TAGNU= 050017	\$\$TD = 000000
TQOVF = 000002	WDFR = 000116	\$F\$RTI= 000350	\$TEMP = 000350	\$\$\$TAG= 050000
UIPAR0= 177640	WDTO = 000114	\$F\$RTN= 000300	\$TSKO = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:44  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1086

3 COMMON EQUATE MODULE  
509 TTDRV STRUCTURED HEADER COVER SHEET  
535 MACRO-SPMAC ROUTINE (INTERRUPT SERVICE ROUTINE)  
540 COMMON DEFINITIONS AND REFERENCES  
542 000000' .PRINT ;SPMAC: VERSION 1.1  
583 TTINI (TERMINAL INITIATOR MODULE)  
637 TTINI (CODE)  
661 TTINT (TERMINAL INTERRUPT HANDLER)  
712 TTINT (CODE)

TDRV (TERMINAL DRIVER MODULE)  
TDRV.MAC 22-AUG-78 08:18

MACY11 30A(1052) 20-SEP-78 18:44 PAGE 19  
TDRV STRUCTURED HEADER COVER SHEET

SEQ 1087

```
511 .TITLE TDRV (TERMINAL DRIVER MODULE)
512 .IDENT /V0.0/
513
514 ;++
515 ; MODULE NAME:
516 ;     TDRV
517 ;
518 ; FUNCTIONAL DESCRIPTION:
519 ;     THIS MODULE CONTAINS TWO MODULES - TTINI AND TTINT.
520 ;     THE TTINI IS THE INITIALIZATION ROUTINE FOR THE
521 ;     TERMINAL INTERRUPT HANDLING ROUTINE - TTINT.
522 ;     TTINT ACTUALLY PROCESSES ALL MESSAGES TO THE DEVICE.
523 ;
524 ;     A DESCRIPTION IS GIVEN IN EACH MODULE.
525 ;
526 ; VERSION:
527 ;     0.0
528 ;
529 ;     EDIT      DATE      BY      REASON
530 ;     -----
531 ;
532 ;
533 ;
```

```

542 000000'
546
547
548
549
550
551
552
553
554
555
556
557
558      000045
559      000134
560
561
562 000000' 000000
563 000002' 000000
564 000004' 000000
565 000006' 000000
566 000010' 000000
567 000012' 000000
568 000014' 000000
569 000016'
570 000016'      000
571 000017'      000
572 000020'      015
573 000021'      012
574 000022'      001
575 000023'      136
576 000024'      125
577 000025'      001
578
579

.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 CTRLQF ;FLAG - OUTPUT SUPPRESS
;*****
; REFERENCED BY OTHER MODULES
.GLOBL TTINT ;MODULE ENTRY POINT - TT HANDLER
.GLOBL TTINI ;MODULE ENTRY POINT - TT INITIATOR
;*****
; LOCAL EQUATES
TT.PCNT = 45 ;CHARACTER - '%' PERCENT
TT.BKSLSH = 134 ;CHARACTER - '\' BACKSLASH
;*****
; LOCAL STORAGE
TT.FCH: 0 ;CHARACTER - FILL
TT.PFC: 0 ;COUNT - FILL, PERMANENT
TT.FCF: 0 ;COUNT, FLAG - FILL (FLAG SET NE 0)
TT.EXT: 0 ;POINTER - LOCATION TO JUMP OUT INDIRECTLY
TT.PTR: 0 ;POINTER - MESSAGE
TT.TP: 0 ;POINTER - TEMPORARY BUFFER
TT.COD: 0 ;CODE - MESSAGE
TT.FLG: ;FLAGS - PERCENT AND CONTROL-U
TT.PCF: .BYTE 0 ;FLAG - PERCENT
TT.CUF: .BYTE 0 ;FLAG - CONTROL-U
TT.PCB: .BYTE 15 ;+BUFFER - PERCENT '%' <CR>+
        .BYTE 12 ;+BUFFER ECHO <LF>+
        .BYTE 1 ;+BUFFER - TERMINATOR <+1>+
TT.CUB: .BYTE 136 ;#BUFFER - CONTROL-U <^>#
        .BYTE 125 ;#BUFFER ECHO <U>#
TT.BEN: .BYTE 1 ;#BUFFER - TERMINATOR <+1>#
;FLAG - (TT.BEN) END SPECIAL BUFFER

.EVEN

```





639  
640  
641  
642  
643  
644  
645  
646  
647 000026'  
(2) 000026'  
648 000026' 010046  
(2) 000026' 010046  
649 000030'  
(4) 000030' 016500 000000  
650 000034'  
(4) 000034' 116067 000037 177736  
651 000042'  
(4) 000042' 116067 000036 177732  
652 000050'  
(4) 000050' 016567 000002 177736  
653 000056'  
(4) 000056' 016567 000004 177724  
654 000064'  
(4) 000064' 016567 000006 177714  
655 000072'  
(2) 000072' 012600  
656 000074'  
(6) 000074' 052777 000100 000000G  
657 000102'  
(3) 000102'  
(3) 000102'  
(2) 000102' 000207

;+  
; 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  
MOVB (SP)+,R0  
BIS #IE,@XCSR  
50000\$:  
50001\$:  
RTS PC



```

714 000104'      IROUTINE      TTINT
(2) 000104'
715              ;+
716              ;DISABLE INTERRUPTS.
717              ;-
718 000104'      LET @XCSR := @XCSR CLR.BY #IE
(6) 000104' 042777 000100 000000G      BIC      #IE,@XCSR
719
720              ;+
721              ; IF XOFF FLAG SET
722              ; TRUE - RTI
723              ; ELSE - CONTINUE
724              ;-
725
726 000112'      IF KB.XFLAG NE #0 THEN
(6) 000112' 005767 000000G      TST      KB.XFLAG
(9) 000116' 001401      BEQ      50002$
727              INLINE <RTI>
(2) 000120' 000002      RTI
728 000122'      ENDIF
(4) 000122'      50002$:
729
730              ;+
731              ; IS FILL COUNT-FLAG NOT EQUAL TO ZERO ?
732              ; TRUE - OUTPUT THE FILLER-CHARACTER
733              ; DECREMENT FILL COUNT-FLAG
734              ; SET INTERRUPTS, RETURN
735              ; FALSE - CONTINUE.
736              ;-
737 000122'      IF TT.FCF NE #0 THEN
(6) 000122' 005767 177656      TST      TT.FCF
(9) 000126' 001406      BEQ      50003$
738              LET @XBUF :B= #NULL
(4) 000130' 112777 000000 000000G      MOVB     #NULL,@XBUF
739 000136'      LET TT.FCF := TT.FCF - #1
(6) 000136' 005367 177642      DEC      TT.FCF
740 000142'      ELSE
(4) 000142' 000524      BR      50004$
(3) 000144'      50003$:
741              ;+
742              ; IS OUTPUT OF <^U> OR <%> BUFFER COMPLETED ?
743              ; TRUE - RESET <^U> AND <%> FLAGS
744              ; RESTORE MESSAGE POINTER
745              ; FALSE - NO ACTION
746              ;-
747 000144'      IF TT.FLG NE #0 ANDB @TT.PTR EQ TT.BEN THEN
(6) 000144' 005767 177646      TST      TT.FLG
(9) 000150' 001411      BEQ      50005$
(6) 000152' 127767 177632 177645      CMPB     @TT.PTR,TT.BEN
(9) 000160' 001005      BNE      50005$
748 000162'      LET TT.FLG := #0
(4) 000162' 005067 177630      CLR      TT.FLG
749 000166'      LET TT.PTR := TT.TP
(4) 000166' 016767 177620 177614      MOV      TT.TP,TT.PTR
750 000174'      ENDIF
(4) 000174'      50005$:

```

```

751      ;+
752      ; IS INPUT CHARACTER A NULL ?
753      ; TRUE -      THIS IS THE END OF THE MESSAGE
754      ;           RETURN BY 'JMP @TT.EXT'.
755      ; FALSE -     NO ACTION
756      ; -
757      IFB @TT.PTR EQ #NULL THEN
758      (6) 000174' 127727 177610 000000
759      (9) 000202' 001007
760      50007$:
761      IF TT.COD EQ #MSGPOP THEN
762      (6) 000204' 026727 177604 000002
763      (9) 000212' 001001
764      CMP      TT.COD,#MSGPOP
765      BNE      50007$
766      (2) 000214' 022626
767      CMP      (SP)+,(SP)+
768      50007$:
769      ENDIF
770      (4) 000216'
771      (2) 000216' 000177 177564
772      JMP @TT.EXT;LEAVE,FINISHED MESSAGE>
773      50006$:
774      (4) 000222'
775      ENDIF
776      ;+
777      ; NON-STANDARD MESSAGE CODE HANDLING
778      ; SPECIAL CONDITIONS- CONTROL U,RUBOUT
779      ; -
780      IF TT.COD LT #0 THEN
781      (6) 000222' 005767 177566
782      (9) 000226' 002033
783      TST      TT.COD
784      BGE      50010$
785      ;+
786      ; IS INPUT CHARACTER IS A <^U> CONTROL U ?
787      ; TRUE -      SET ^U-FLAG
788      ;           SAVE REST OF MESSAGE POINTER
789      ;           POINT TO ^U-BUFFER
790      ; FALSE -     NO ACTION
791      ; -
792      IFB @TT.PTR EQ #CTRLU THEN
793      (6) 000230' 127727 177554 000025
794      (9) 000236' 001013
795      CMPB     @TT.PTR,#CTRLU
796      BNE      50011$
797      (4) 000240' 112767 000001 177551
798      LET TT.CUF :B= #1
799      MOV      #1,TT.CUF
800      (4) 000246' 016767 177536 177536
801      LET TT.TP := TT.PTR + #1
802      MOV      TT.PTR,TT.TP
803      (6) 000254' 005267 177532
804      INC      TT.TP
805      (4) 000260' 012767 000023' 177522
806      LET TT.PTR := #TT.CUB
807      MOV      #TT.CUB,TT.PTR
808      50011$:
809      ENDIF
810      ;+
811      ; IS INPUT CHARACTER IS A RUBOUT ?
812      ; TRUE -      OUTPUT BACKSLASH
813      ; FALSE -     OUTPUT CHARACTER
814      ; -
815      IFB @TT.PTR EQ #RUBOUT THEN
816      (6) 000266' 127727 177516 000177
817      (9) 000274' 001004
818      CMPB     @TT.PTR,#RUBOUT
819      BNE      50012$
820      (4) 000276' 112777 000134 000000G
821      LET @XBUF :B= #TT.BKSLSH
822      MOV      #TT.BKSLSH,@XBUF

```

```
787 000304' ELSE
(4) 000304' 000403 BR 50013$
(3) 000306' 50012$:
788 000306' LET @XBUF :B= @TT.PTR
(4) 000306' 117777 177476 000000G MOVB @TT.PTR,@XBUF
789 000314' ENDIF 50013$:
(4) 000314'
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$:
807 ;+
808 ; IS MESSAGE SURPRESS FLAG NOT SET ?
809 ; TRUE - OUTPUT CHARACTER
810 ; FALSE - NO ACTION
811 ;-
812 000354' IF CTRLQF NE #1 THEN
(6) 000354' 026727 000000G 000001 CMP CTRLQF,#1
(9) 000362' 001403 BEQ 50016$
813 000364' LET @XBUF :B= @TT.PTR
(4) 000364' 117777 177420 000000G MOVB @TT.PTR,@XBUF
814 000372' ENDIF
(4) 000372' 50016$:
815 000372' ENDIF 50014$:
(4) 000372'
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
```

TDRV (TERMINAL DRIVER MODULE)  
TDRV.MAC 22-AUG-78 08:18

MACY11 30A(1052) 20-SEP-78 18:44 PAGE 24-3  
TTINT (CODE)

SEQ 1095

```
(4) 000402' 016767 177374 177374
823 000410'          ENDIF
(4) 000410'          50017$:
824          ;+
825          ; INCREMENT MESSAGE CHARACTER POINTER
826          ; -
827 000410'          LET TT.PTR := TT.PTR + #1
(6) 000410' 005267 177374          INC      TT.PTR
828 000414'          ENDIF          50004$:
(4) 000414'
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	CTRLC = 000003	ENDLST= 000000	MODEXH= 004000	RBUFEA= 000130
ACTBIT= 004000	CTRLO = 000017	EOPBIT= 000001	MODHOL= 002000	RBUFPA= 000126
ADDR22= 001000	CTRLQF= ***** G	ERRTYP= 000106	MODSEL= 001000	RBUFSZ= 000132
ADR = 000006	CTRLU = 000025	EVNTBE= 000200	MSGADR= 000004	RBUFVA= 000124
APTFER= 000004	DCEVNT= 000011	EVNTHD= 000200	MSGCKD= 000010	RDSERV= 000101
APTPRE= 000200	DEFRTN= 000400	EVNTKT= 000203	MSGCKS= 000011	RDWHMI= 000022
ASB = 000106	DIAGMC= 000000	EVNTPE= 000202	MSGCOD= 000002	RELERR= 000020
ASSEMB= 000010	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	RELMOD= 020000
ASTAT = 000104	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	RELTIM= 010000
AUTD = 000010	DTBL = 000000	HRDCNT= 000044	MSGECH= 177777	RES1 = 000056
AUTDST= 020000	DT.ADD= 000042	HRDPAS= 000050	MSGEOP= 000013	RES2 = 000060
AWAS = 000110	DT.AP = 000100	ICDNT = 000036	MSGHDR= 000004	RETADR= 000006
BIT0 = 000001	DT.APK= 000076	ICOUNT= 000040	MSGHNG= 000022	RICHAR= 031060
BIT00 = 000001	DT.BLS= 000034	IDNUM = 000122	MSGHRD= 000007	RPTDAT= 002000
BIT01 = 000002	DT.CFO= 000014	IE = 000100	MSGMAP= 000021	RSTRT = 000112
BIT02 = 000004	DT.CF1= 000016	INDPAR= 000040	MSGNUL= 177775	RUBOUT= 000177
BIT03 = 000010	DT.ERR= 000020	INHDRP= 040000	MSGPOP= 000002	RUNMOD= 100000
BIT04 = 000020	DT.ESI= 000044	INHEPR= 020000	MSGPRM= 177776	R5VALU= 001740
BIT05 = 000040	DT.EVN= 000000	INHREL= 001000	MSGRES= 000001	SAM = 075464
BIT06 = 000100	DT.EXS= 000060	INHRR= 000400	MSGSFT= 000006	SBADR = 000102
BIT07 = 000200	DT.FCH= 000037	INIT = 000030	MSGSKE= 000003	SBKMOD= 000000
BIT08 = 000400	DT.FCN= 000036	INTR = 000120	MSGSMB= 000015	SBKSEL= 010000
BIT09 = 001000	DT.HMX= 000104	IOMOD = 100000	MSGSMH= 000014	SC.ADR= 000006
BIT1 = 000002	DT.KBE= 000024	IOMODP= 102000	MSGSMS= 000016	SC.ALC= 000014
BIT10 = 002000	DT.KBP= 000026	IOMODR= 112000	MSGSTD= 000000	SC.APC= 000016
BIT11 = 004000	DT.KBR= 000022	IOMODX= 110000	MSGSYS= 000012	SC.CKL= 000002
BIT12 = 010000	DT.KBU= 000030	JACK = 035060	MSGVEC= 000020	SC.CKP= 000004
BIT13 = 020000	DT.MLS= 000032	KB.XFL= ***** G	NBKMOD= 001000	SC.CLD= 000000
BIT14 = 040000	DT.MTI= 000110	KIPAR0= 172340	NCPUOP= 000020	SC.HLD= 000010
BIT15 = 100000	DT.OFF= 000070	KIPAR1= 172342	NDAPTY= 000002	SC.SCA= 000012
BIT2 = 000004	DT.PAS= 000074	KIPAR2= 172344	NULL = 000000	SENDLS= 177777
BIT3 = 000010	DT.PC = 000002	KIPAR3= 172346	OWEN = 024020	SQFCNT= 000042
BIT4 = 000020	DT.PFL= 000062	KIPAR4= 172350	PAERR = 000010	SQFPAS= 000046
BIT5 = 000040	DT.PSW= 000004	KIPAR5= 172352	PARPRE= 002000	SPACE = 000040
BIT6 = 000100	DT.PTA= 000064	KIPAR6= 172354	PARSTA= 000100	SPOINT= 000032
BIT7 = 000200	DT.RCS= 000102	KIPAR7= 172356	PASCNT= 000034	SPVALU= 002200
BIT8 = 000400	DT.REL= 000040	KIPDR0= 172300	PDPLSI= 020000	SR0 = 177572
BIT9 = 001000	DT.SCT= 000066	KIPDR1= 172302	PDP60 = 004000	SR1 = 177574
BKDEF = 000002	DT.SMX= 000106	KIPDR2= 172304	PDP70 = 010000	SR2 = 177576
BKMOD = 000020	DT.SP = 000006	KIPDR3= 172306	PRI0 = 000000	SR3 = 172516
BKMODE= 040000	DT.SSI= 000046	KIPDR4= 172310	PRI1 = 000040	STAT = 000026
BKSLSH= 000134	DT.STO= 000010	KIPDR5= 172312	PRI4 = 000200	STATBI= 064757
CAPRES= 000004	DT.ST1= 000012	KIPDR6= 172314	PRI5 = 000240	STAT1 = 000027
CASTAT= 000004	DT.SWR= 000056	KIPDR7= 172316	PRI6 = 000300	SUSPND= 000001
CDERCT= 000146	DT.SYP= 000072	KTERRO= 000040	PRI7 = 000340	SVR0 = 000062
CDWDCT= 000144	DT.WBU= 000050	KTPRES= 000400	PR0 = 000000	SVR1 = 000064
CKTIM = 100000	DT.WHL= 000054	KTSTAT= 000020	PR4 = 000200	SVR2 = 000066
CLKPRE= 000001	DT.WLL= 000052	KTXTND= 040000	PR5 = 000240	SVR3 = 000070
CONFIG= 000056	DVID1 = 000014	LF = 000012	PR6 = 000300	SVR4 = 000072
CQQVF = 000001	ECCMEM= 000100	LPSTAT= 000001	PR7 = 000340	SVR5 = 000074
CR = 000015	ECCSTA= 000010	MAPSTA= 000200	PS = 177776	SVR6 = 000076
CSRA = 000100	ENBEOP= 010000	MED = 076600	PSW = 177776	SYSCNT= 000052
CSRC = 000102	ENBNUL= 000001	MEMPAS= 040000	RANNUM= 000054	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	\$ISKO = 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	\$\$RETN= 000000
TT.PCB 000020R	WBSTAT= 000040	\$F\$GOD= 000400	\$NSKO = 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\$LDD= 000200	\$NSK3 = 000110	\$\$TGSV= 000000
TT.PTR 000010R	WBUFSZ= 000142	\$F\$NAM= 000160	\$SAVLE= 177777	\$\$TGS1= 000000
TT.TP 000012R	WDFR = 000116	\$F\$ND = 000403	\$TAGLE= 177777	\$\$TGS2= 000000
UIPAR0= 177640	WDTO = 000114	\$F\$OR = 000320	\$TAGNU= 050020	\$\$TD = 000000
UIPAR1= 177642	WTINRE= 000352	\$F\$RTI= 000350	\$TEMP = 000350	\$\$\$TAG= 050000
UIPAR2= 177644	WTWHMI= 000222	\$F\$RTN= 000300	\$TSKO = 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)



.MAIN. MACY11 30A(1052) 20-SEP-78 18:45  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1098

3 COMMON EQUATE MODULE  
560 COMMON DEFINITIONS AND REFERENCES  
563 000000' .PRINT ;SPMAC: VERSION 1.1  
605 KERNEL - DETERMINE MEMORY SIZE  
670 KERNEL - WRITE GOOD PARITY  
719 KERNEL - SORT MODULE LIST

```
508 .TITLE SIZPLA - SIZE AND POLL SYSTEM - NO SUPPORT OF ANY CPU OPTIONS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; SIZPLA
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE WILL SIZE MEMORY TO 28K AND WRITE
517 ; GOOD PARITY IN THAT AREA. THIS ROUTINE WILL NOT CHECK FOR
518 ; ANY CPU OPTIONS AND WILL INDICATE THERE ARE NONE. THE "MODQ" LIST WILL
519 ; BE SORTED TO DEVELOP A BACKGROUND MODULE LIST (QUEUE).
520 ;
521 ; INPUTS:
522 ; DTABLE ADDRESS
523 ;
524 ; IMPLICIT INPUTS:
525 ; CONFIGURATION WORD 0
526 ; EXERCISER SIZE
527 ; SYSTEM SIZE
528 ; MODULE LIST
529 ; BACKGROUND LIST
530 ;
531 ; OUTPUTS:
532 ; NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ; SYSTEM SIZE
536 ; EXERCISE SIZE
537 ; CONFIGURATION WORD 0
538 ;
539 ; PATHOLOGICAL CONNECTIONS:
540 ;
541 ; SUBORDINATE ROUTINES CALLED:
542 ; SAVREG - SAVE REGISTERS
543 ; RESREG - RESTORE REGISTERS
544 ; HRDADRCHK - ILLEGAL ADDRESS CHECK
545 ;
546 ; FUNCTIONAL SIDE EFFECTS:
547 ; NONE
548 ;
549 ; CALLING SEQUENCE:
550 ; CALL SIZPOL IN <A>
551 ; A - DTABLE ADDRESS
552 ;
553 ; VERSION:
554 ; 0.0
555 ;
556 ; EDIT DATE BY REASON
557 ;--
```

```
559
560          .SBTTL COMMON DEFINITIONS AND REFERENCES
561
562          .MCALL STRUCT
563 000000'   STRUCT
(1) 000000'   .PRINT ;SPMAC: VERSION 1.1
564          000001   $LSTIN = 1
565          000001   $LSTTAG = 1
566
567
568          ;*****
569          ; GLOBAL REFERENCES
570          ;
571          .GLOBL SAVREG          ;SARE REGISTERS
572          .GLOBL RESREG         ;RESTORE REGISTERS
573          .GLOBL HRDADRCHK      ;ILLEGAL ADDRESS CHECK
574
575
576          ;*****
577          ; LOCAL EQUATES
578          ;
579          004000   SS.0KC = 4000          ;ONE K CHUNK
580          000034   SS.28 = ^D28        ;28 DECIMAL
581          160000   SS.RFS = 160000     ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE
582
583          ;*****
584          ; REFERENCED BY OTHER MODULES
585          ;
586          .GLOBL SIZPOL          ;MODULE ENTRY POINT
587
588
589
590
591          ;+++++
592          ; OVERLAY REGION GLOBALS
593          ;+++++
594          .GLOBL OV.KBBUF
595          .GLOBL OV.HIKB
596          .GLOBL OV.KBSIZ
597          .GLOBL OV.CQ
598          .GLOBL OV.HICQ
599          .GLOBL OV.CQSIZ
600          .GLOBL OV.TQ
601          .GLOBL OV.HITQ
602          .GLOBL OV.TQSIZ
```

```

604
605          .SBTTL KERNEL - DETERMINE MEMORY SIZE
606
607
608 000000'          ROUTINE SIZPOL <DTABLE>
(2) 000000'
609
610          ;+
611          ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
612          ; R1
613          ; -
614 000000'          CALL SAVREG
(3) 000000' 004767 000000G          JSR          PC, SAVREG
615 000004'          LET R2 := DTABLE(R5)
(4) 000004' 016502 000000          MOV          DTABLE(R5), R2
616 000010'          LET R1 := #0
(4) 000010' 005001          CLR          R1
617
618
619          ;+
620          ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
621          ; AND LOAD IT INTO EXERCISER SIZE WORD
622          ; -
623
624 000012'          LET DT.ESIZ(R2) := 0
(4) 000012' 016762 000000 000044          MOV          0, DT.ESIZ(R2)
625
626          ;+
627          ; DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
628          ; THAN 28K.
629          ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
630          ; -
631
632 000020'          LET R0 := #0
(4) 000020' 005000          CLR          R0
633 000022'          LET R3 := #0
(4) 000022' 005003          CLR          R3
634
635          ;+
636          ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K IS FOUND
637          ; -
638
639 000024'          REPEAT
(3) 000024'          50002$:
640 000024'          LET R0 := R0 + #1
(6) 000024' 005200          INC          R0
641 000026'          LET R3 := R3 + #SS.0KC
(6) 000026' 062703 004000          ADD          #SS.0KC, R3
642 000032'          CALL HRDADRCHK IN <R3>
(3) 000032' 010546          MOV          R5, -(SP)
(4) 000034' 010345          MOV          R3, -(R5)
(3) 000036' 004767 000000G          JSR          PC, HRDADRCHK
(3) 000042' 012605          MOV          (SP)+, R5
643 000044'          IF.ERROR THEN
(6) 000044' 103001          BCC          50003$
644 000046'          INLINE <BR 1$>
    
```



```

670          .SBTTL KERNEL - WRITE GOOD PARITY
671          ;+
672          ; SET UP POINTER TO POINT TO LOC. 0 AND SAVE BUSERR VECTOR
673          ; -
674
675 000144'          LET R0 := #0
676 (4) 000144' 005000          PUSH @#4
677 (2) 000146' 013746 000004          LET @#4 := #2$
678 (4) 000152' 012737 000164' 000004          MOV #2$,@#4
679
680          ;+
681          ; NOW WRITE GOOD PARITY
682          ; -
683 000160'          INLINE<3$:>
684 (2) 000160'          LET (R0)+ := (R0)
685 (4) 000160' 011020          MOV (R0),(R0)+
686 (2) 000162' 000776          BR 3$
687
688          ;+
689          ; WILL TRAP HERE WHEN TOP OF MEMORY IS FOUND
690          ; CLEAN UP STACK AND RESTORE BUSERR
691          ; -
692
693 000164'          INLINE <2$:>
694 (2) 000164'          INLINE <CMP (SP)+,(SP)+>
695 (2) 000164' 022626          CMP (SP)+,(SP)+
696 (2) 000166' 012637 000004          POP @#4
697
698          ;+
699          ; LOAD NCPUGP BIT INTO R1
700          ; -
701 000172'          LET R1 := R1 SET.BY #NCPUOP
702 (6) 000172' 052701 000020          BIS #NCPUOP,R1
703
704          ;+
705          ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD 0
706          ; -
707
708 000176'          IF @#42 NE #0 THEN
709 (6) 000176' 005737 000042          TST @#42
710 (9) 000202' 001402          BEQ 50006$
711 (6) 000204' 052701 000010          LET R1 := R1 SET.BY #AUTO
712 (6) 000210'          BIS #AUTO,R1
713 (4) 000210'          ENDIF
714
715          50006$:
    
```

```
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  
717  
BIS R1,DT.CF0(R2)
```

```

719          .SBTTL KERNEL - SORT MODULE LIST
720
721
722          ;+
723          ; COUNT THE MODULES
724          ; -
725
726          000214'          LET R0 := #0
727          (4) 000214' 005000          CLR          R0
728          000216'          LET R1 := DT.MLST(R2)
729          (4) 000216' 016201 000032          MOV          DT.MLST(R2),R1
730          000222'          WHILE (R1) NE #ENDLST DO
731          (4) 000222'          50007$:
732          (6) 000222' 021127 000000          CMP          (R1),#ENDLST
733          (9) 000226' 001404          BEQ          50010$
734          000230'          LET R0 := R0 + #1
735          (6) 000230' 005200          INC          R0
736          000232'          LET R1 := R1 + #2
737          (6) 000232' 062701 000002          ADD          #2,R1
738          000236'          ENDDO
739          (4) 000236' 000771          BR          50007$
740          (3) 000240'          50010$:
741
742          ;+
743          ; START THE SORT OPERATION, AND SORT "BKMODS" TO BOTTOM OF MODQ LIST
744          ; DEVELOPING A "BKMOD QUEUE".
745          ; -
746
747          000240'          LET R1 := DT.MLST(R2)
748          (4) 000240' 016201 000032          MOV          DT.MLST(R2),R1
749
750          ;+
751          ; WORK THRU THE LIST UNTIL THE END IS REACHED
752          ; -
753
754          000244'          WHILE R0 GT #0 DO
755          (4) 000244'          50011$:
756          (6) 000244' 005700          TST          R0
757          (9) 000246' 003432          BLE          50012$
758          000250'          REPEAT
759          (3) 000250'          50013$:
760
761          ;+
762          ; IS THE CURRENT ENTRY A BKMOD MODULE?
763          ; -
764
765          000250'          LET R4 := (R1)
766          (4) 000250' 011104          MOV          (R1),R4
767          000252'          IF #BKMOD SETIN STAT(R4) THEN
768          (6) 000252' 032764 000020 000026          BIT          #BKMOD,STAT(R4)
769          (9) 000260' 001414          BEQ          50014$
770
771          ;+
772          ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
773          ; -
774
775
776
777

```



```

758 000262'          LET R3 := 2(R1)
(4) 000262' 016103 000002          MOV      2(R1),R3
759
760          ;+
761          ; MAKE SURE ITS NOT THE END OF THE LIST
762          ; -
763
764 000266'          IF R3 NE #ENDLST THEN
(6) 000266' 020327 000000          CMP      R3,#ENDLST
(9) 000272' 001407          BEQ      50015$
765
766          ;+
767          ; IF ITS NOT A BKMOD, SWAP THEM
768          ; -
769
770 000274'          IF #BKMOD NOTSETIN STAT(R3) THEN
(6) 000274' 032763 000020 000026          BIT      #BKMOD,STAT(R3)
(9) 000302' 001003          BNE      50016$
771 000304'          LET 2(R1) := (R1)
(4) 000304' 011161 000002          MOV      (R1),2(R1)
772 000310'          LET (R1) := R3
(4) 000310' 010311          MOV      R3,(R1)
773 000312'          ENDIF
(4) 000312'          50016$:
774 000312'          ENDIF          50015$:
(4) 000312'          ENDIF          50014$:
775 000312'
(4) 000312'
776
777          ;+
778          ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
779          ; -
780
781 000312'          LET R1 := R1 + #2
(6) 000312' 062701 000002          ADD      #2,R1
782 000316'          UNTIL (R1) EQ #ENDLST
(3) 000316' 021127 000000          CMP      (R1),#ENDLST
(6) 000322' 001352          BNE      50013$
783
784          ;+
785          ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
786          ; -
787
788 000324'          LET R0 := R0 - #1
(6) 000324' 005300          DEC      R0
789 000326'          LET R1 := DT.MLST(R2)
(4) 000326' 016201 000032          MOV      DT.MLST(R2),R1
790 000332'          ENDDO
(4) 000332' 000744          BR      50011$
(3) 000334'          50012$:
791
792          ;+
793          ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
794          ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
795          ; A BACKGROUND LIST) INTO THE DATA TABLE
796          ; -

```

```

797
798 000334'          LET R1 := DT.MLST(R2)
(4) 000334' 016201 000032          MOV      DT.MLST(R2),R1
799
800          ;+
801          ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
802          ; -
803
804 000340'          LET DT.BLST(R2) := #0
(4) 000340' 005062 000034          CLR      DT.BLST(R2)
805 000344'          LET R3 := #-1
(4) 000344' 012703 177777          MOV      #-1,R3
806 000350'          WHILE (R1) NE #ENDLST AND R3 NE #0 DO
(4) 000350'          50017$:
(6) 000350' 021127 000000          CMP      (R1),#ENDLST
(9) 000354' 001420          BEQ      50020$
(6) 000356' 005703          TST      R3
(9) 000360' 001416          BEQ      50020$
807 000362'          LET R4 := (R1)
(4) 000362' 011104          MOV      (R1),R4
808 000364'          IF #BKMOD SETIN STAT(R4) THEN
(6) 000364' 032764 000020 000026  BIT      #BKMOD,STAT(R4)
(9) 000372' 001406          BEQ      50021$
809
810          ;+
811          ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
812          ; -
813
814 000374'          LET DT.BLST(R2) := R1
(4) 000374' 010162 000034          MOV      R1,DT.BLST(R2)
815 000400'          LET R1 := #ENDLST
(4) 000400' 012701 000000          MOV      #ENDLST,R1
816 000404'          LET R3 := #0
(4) 000404' 005003          CLR      R3
817 000406'          ELSE
(4) 000406' 000402          BR      50022$
(3) 000410'          50021$:
818
819          ;+
820          ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
821          ; -
822
823 000410'          LET R1 := R1 + #2
(6) 000410' 062701 000002          ADD      #2,R1
824 000414'          ENDIF
(4) 000414'          50022$:
825 000414'          ENDDO
(4) 000414' 000755          BR      50017$
(3) 000416'          50020$:
826
827          ;+
828          ; NOW RETURN TO THE CALLER
829          ; -
830
831 000416'          CALL RESREG
(3) 000416' 004767 000000G          JSR      PC,RESREG
    
```

```

832 000422'          ENDRTN
(3) 000422'          50000$:
(3) 000422'          50001$:
(2) 000422' 000207          RTS      PC
833
834          ;+
835          ; LABEL TYPE QUEUE HIGH LIMIT
836          ; -
837
838          ;+
839          ; ALLOW STORAGE FOR TQ
840          ; -
841
842 000424' 000120          .BLKB  ^D<80>
843 000544'          INLINE <OV.TQEND:>
(2) 000544'          OV.TQEND:
844
845
846
847
848
849          ;*****
850          ; NOTE: THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
851          ; (THAT IS, AFTER SIZPOL AND TQ.END LABELS)
852
853          ;+++++
854          ; OVERLAY REGIONS EQUATES
855          ;+++++
856          000000'          OV.KBBUF = SIZPOL          ;KEYBOARD BUFFER STARTING ADDRESS
857          000122'          OV.CQ= OV.KBBUF + ^D<82>          ;CONTROL QUEUE STARTING ADDRESS
858          000244'          OV.TQ = OV.CQ + ^D<82>          ;TYPE QUEUE STARTING ADDRESS
859          000120'          OV.HIKB = OV.CQ - 2          ;KEYBOARD BUFFER HIGH LIMIT
860          000122'          OV.KBSIZ = OV.CQ - OV.KBBUF          ;KEYBOARD BUFFER SIZE
861          000242'          OV.HICQ = OV.TQ - 2          ;CONTROL QUEUE HIGH LIMIT
862          000024'          OV.CQSIZ = <OV.TQ - OV.CQ>/4          ;CONTROL QUEUE SIZE
863          000300'          OV.A = OV.TQEND - OV.TQ          ;GET THE TQ SIZE
864          000000'          OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
865          000544'          OV.HITQ = OV.TQEND - OV.REM          ;TYPE QUEUE HIGH LIMIT
866          000030'          OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10          ;TYPE QUEUE SIZE
867
868          000001          .END
    
```

ACSR = 000102	CTRLC = 000003	EOPBIT= 000001	MODHOL= 002000	PRI7 = 000340
ACTBIT= 004000	CTRLD = 000017	ERRTYP= 000106	MODSEL= 001000	PRO = 000000
ADDR22= 001000	CTRLU = 000025	EVNTBE= 000200	MSGCKD= 000010	PR4 = 000200
ADR = 000006	DCEVNT= 000011	EVNTHD= 000200	MSGCKS= 000011	PR5 = 000240
APTFER= 000004	DEFRTN= 000400	EVNTKT= 000203	MSGDER= 000005	PR6 = 000300
APTPRE= 000200	DIAGMC= 000000	EVNTPE= 000202	MSGDRP= 000017	PR7 = 000340
ASB = 000106	DROPMO= 100000	EVNTRE= 000201	MSGECH= 177777	PS = 177776
ASSEMB= 000010	DSEVNT= 000014	FATERR= 100000	MSGEDP= 000013	PSW = 177776
ASTAT = 000104	DTABLE= 000000	HRDADR= ***** G	MSGHDR= 000004	RANNUM= 000054
AUTO = 000010	DT.ADD= 000042	HRDCNT= 000044	MSGHNG= 000022	RBUFEA= 000130
AUTOST= 020000	DT.AP = 000100	HRDPAS= 000050	MSGHRD= 000007	RBUFPA= 000126
AWAS = 000110	DT.APK= 000076	ICONT = 000036	MSGMAP= 000021	RBUFSZ= 000132
BIT0 = 000001	DT.BLS= 000034	ICOUNT= 000040	MSGNUL= 177775	RBUFVA= 000124
BIT00 = 000001	DT.CFO= 000014	IDNUM = 000122	MSGPQP= 000002	RDSERV= 000101
BIT01 = 000002	DT.CF1= 000016	IE = 000100	MSGPRM= 177776	RDWHMI= 000022
BIT02 = 000004	DT.ERR= 000020	INDPAR= 000040	MSGRES= 000001	RELERR= 000020
BIT03 = 000010	DT.ESI= 000044	INHDRP= 040000	MSGSFT= 000006	RELMOD= 020000
BIT04 = 000020	DT.EVN= 000000	INHEPR= 020000	MSGSKE= 000003	RELTIM= 010000
BIT05 = 000040	DT.EXS= 000060	INHREL= 001000	MSGSMB= 000015	RESREG= ***** G
BIT06 = 000100	DT.FCH= 000037	INHRR= 000400	MSGSMH= 000014	RES1 = 000056
BIT07 = 000200	DT.FCN= 000036	INIT = 000030	MSGSMS= 000016	RES2 = 000060
BIT08 = 000400	DT.HMX= 000104	INTR = 000120	MSGSTD= 000000	RICHAR= 031060
BIT09 = 001000	DT.KBE= 000024	IOMOD = 100000	MSGSYS= 000012	RPTDAT= 002000
BIT1 = 000002	DT.KBP= 000026	IOMODP= 102000	MSGVEC= 000020	RSTRT = 000112
BIT10 = 002000	DT.KBR= 000022	IOMODR= 112000	NBKMOD= 001000	RUBOUT= 000177
BIT11 = 004000	DT.KBU= 000030	IOMODX= 110000	NCPUQP= 000020	RUNMOD= 100000
BIT12 = 010000	DT.MLS= 000032	JACK = 035060	NDAPTY= 000002	R5VALU= 001740
BIT13 = 020000	DT.MTI= 000110	KIPARO= 172340	NULL = 000000	SAM = 075464
BIT14 = 040000	DT.OFF= 000070	KIPAR1= 172342	OV.A = 000300	SAVREG= ***** G
BIT15 = 100000	DT.PAS= 000074	KIPAR2= 172344	OV.CQ = 000122RG	SBADR = 000102
BIT2 = 000004	DT.PC = 000002	KIPAR3= 172346	OV.CQS= 000024 G	SBKMOD= 000000
BIT3 = 000010	DT.PFL= 000062	KIPAR4= 172350	OV.HIC= 000242RG	SBKSEL= 010000
BIT4 = 000020	DT.PSW= 000004	KIPAR5= 172352	OV.HIK= 000120RG	SC.ADR= 000006
BIT5 = 000040	DT.PTA= 000064	KIPAR6= 172354	OV.HIT= 000544RG	SC.ALC= 000014
BIT6 = 000100	DT.RCS= 000102	KIPAR7= 172356	OV.KBB= 000000RG	SC.APC= 000016
BIT7 = 000200	DT.REL= 000040	KIPDR0= 172300	OV.KBS= 000122 G	SC.CKL= 000002
BIT8 = 000400	DT.SCT= 000066	KIPDR1= 172302	OV.REM= 000000	SC.CKP= 000004
BIT9 = 001000	DT.SMX= 000106	KIPDR2= 172304	OV.TQ = 000244RG	SC.CLO= 000000
BKDEF = 000002	DT.SP = 000006	KIPDR3= 172306	OV.TQE 000544R	SC.HLD= 000010
BKMOD = 000020	DT.SSI= 000046	KIPDR4= 172310	OV.TQS= 000030 G	SC.SCA= 000012
BKMODE= 040000	DT.ST0= 000010	KIPDR5= 172312	OWEN = 024020	SENDLS= 177777
BKSLSH= 000134	DT.ST1= 000012	KIPDR6= 172314	PAERR = 000010	SIZPOL 000000RG
CAPRES= 000004	DT.SWR= 000056	KIPDR7= 172316	PARPRE= 002000	SOFCNT= 000042
CASTAT= 000004	DT.SYP= 000072	KTERRO= 000040	PARSTA= 000100	SOFPAS= 000046
CDERCT= 000146	DT.WBU= 000050	KTPRES= 000400	PASCNT= 000034	SPACE = 000040
CDWDCT= 000144	DT.WHL= 000054	KTSTAT= 000020	PDPLSI= 020000	SPOINT= 000032
CKTIM = 100000	DT.WLL= 000052	KTXTND= 040000	PDP60 = 004000	SPVALU= 002200
CLKPRE= 000001	DVID1 = 000014	LF = 000012	PDP70 = 010000	SR0 = 177572
CONFIG= 000056	ECCMEM= 000100	LPSTAT= 000001	PRI0 = 000000	SR1 = 177574
CQOVF = 000001	ECCSTA= 000010	MAPSTA= 000200	PRI1 = 000040	SR2 = 177576
CR = 000015	ENBEOP= 010000	MED = 076600	PRI4 = 000200	SR3 = 172516
CSRA = 000100	ENBNUL= 000001	MEMPAS= 040000	PRI5 = 000240	SS.RFS= 160000
CSRC = 000102	ENDLST= 000000	MODEXH= 004000	PRI6 = 000300	SS.OKC= 004000

SS.28 = 000034	UIPAR7= 177656	\$ERFLG= 000400	\$IFLEV= 177777	\$TSK4 = 050015
STAT = 000026	UIPDR0= 177600	\$F\$AND= 000310	\$ISKO = 000001	\$TSK5 = 050016
STATBI= 064757	UIPDR1= 177602	\$F\$BAD= 000401	\$ISK1 = 000001	\$SARGC= 000002
STAT1 = 000027	UIPDR2= 177604	\$F\$BLA= 000170	\$ISK2 = 000001	\$B\$YTE= 000403
SUSPND= 000001	UIPDR3= 177606	\$F\$CAS= 000150	\$LDOCTA= 177777	\$C\$ASE= 000000
SVR0 = 000062	UIPDR4= 177610	\$F\$DEC= 000220	\$LSTIN= 000001	\$D\$ST = 000000
SVR1 = 000064	UIPDR5= 177612	\$F\$DO = 000340	\$LSTTA= 000001	\$E\$LOC= 000402
SVR2 = 000066	UIPDR6= 177614	\$F\$FAL= 000405	\$NESTL= 177777	\$E\$RFL= 000000
SVR3 = 000070	UIPDR7= 177616	\$F\$GOD= 000400	\$NSKO = 000300	\$F\$LAG= 000001
SVR4 = 000072	WASADR= 000104	\$F\$IF = 000110	\$NSK1 = 000120	\$F\$FROM= 000000
SVR5 = 000074	WBSTAT= 000040	\$F\$INC= 000210	\$NSK2 = 000110	\$S\$LOC = 000372R
SVR6 = 000076	WBUFEA= 000136	\$F\$LDO= 000200	\$NSK3 = 000110	\$L\$OCN= 000000
SYSCNT= 000052	WBUFPA= 000134	\$F\$NAM= 000160	\$NSK4 = 000110	\$R\$EG = 177777
YSERR= 000100	WBUFRQ= 000140	\$F\$ND = 000403	\$NSK5 = 000110	\$R\$ETU= 000000
TMPIO = 000002	WBUFSZ= 000142	\$F\$OR = 000320	\$SAVLE= 177777	\$R\$TN1= 050000
TQOVF = 000002	WDFR = 000116	\$F\$RTI= 000350	\$SSKO = 050020	\$R\$TN2= 050001
UIPAR0= 177640	WDTO = 000114	\$F\$RTN= 000300	\$TAGLE= 177777	\$S\$RC = 000000
UIPAR1= 177642	WTINRE= 000352	\$F\$SEL= 000140	\$TAGNU= 050023	\$T\$G\$SV= 000000
UIPAR2= 177644	WTWHMI= 000222	\$F\$THE= 000330	\$TEMP = 000300	\$T\$GS1= 000000
UIPAR3= 177646	XFLAG = 000005	\$F\$TRU= 000404	\$TSKO = 050017	\$T\$GS2= 000000
UIPAR4= 177650	XOFF = 000023	\$F\$UNT= 000130	\$TSK1 = 050020	\$T\$O = 000000
UIPAR5= 177652	XON = 000021	\$F\$WHI= 000120	\$TSK2 = 050022	\$T\$TAG= 050000
UIPAR6= 177654	\$BGNLE= 177777	\$F\$YES= 000402	\$TSK3 = 050014	. = 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:46

EQUATE.MAC 13-SEP-78 16:13

TABLE OF CONTENTS

SEQ 1111

3 COMMON EQUATE MODULE  
561 COMMON DEFINITIONS AND REFERENCES  
564 000000' .PRINT ;SPMAC: VERSION 1.1  
611 KERNEL - DETERMINE PROCESSOR TYPE  
683 KERNEL - DETERMINE IF CACHE OPTION AVAILABLE  
700 KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT  
744 KERNEL - DETERMINE SYSTEM SIZE  
797 KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT  
868 KERNEL - SORT MODULE LIST

```
508 .TITLE SIZPLB SIZE AND POLL SYSTEM - FOR NON-KT MACHINES
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     SIZPLB
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE WILL SIZE THE SYSTEM AND DETERMINE IF PARITY
517 ;     OR ECC IS PRESENT AND ALSO SORT THE
518 ;     MODQ LIST TO GENERATE A BACKGROUND LIST(QUEUE).
519 ;
520 ; INPUTS:
521 ;     DTABLE ADDRESS
522 ;
523 ; IMPLICIT INPUTS:
524 ;     CONFIGURATION WORD 0
525 ;     EXERCISER SIZE
526 ;     SYSTEM SIZE
527 ;     PARITY TABLE ADDRESS
528 ;     MODULE LIST
529 ;     BACKGROUND LIST
530 ;
531 ; OUTPUTS:
532 ;     NONE
533 ;
534 ; IMPLICIT OUTPUTS:
535 ;     PARITY TABLE ADDRESS
536 ;     SYSTEM SIZE
537 ;     EXERCISE SIZE
538 ;     CONFIGURATION WORD 0
539 ;
540 ; PATHOLOGICAL CONNECTIONS:
541 ;
542 ; SUBORDINATE ROUTINES CALLED:
543 ;     SAVREG - SAVE REGISTERS
544 ;     RESREG - RESTORE REGISTERS
545 ;     HRDADRCHK - ILLEGAL ADDRESS CHECK
546 ;
547 ; FUNCTIONAL SIDE EFFECTS:
548 ;     NONE
549 ;
550 ; CALLING SEQUENCE:
551 ;     CALL SIZPOL IN <A>
552 ;             A - DTABLE ADDRESS
553 ;
554 ; VERSION:
555 ;     0.0
556 ;
557 ;     EDIT             DATE             BY             REASON
558 ;--
```

```

560
561          .SBTTL COMMON DEFINITIONS AND REFERENCES
562
563          .MCALL STRUCT
564 000000'   STRUCT
565 (1) 000000' .PRINT ;SPMAC: VERSION 1.1
566          000001 $LSTIN = 1
567          000001 $LSTTAG = 1
568
569          ;*****
570          ; GLOBAL REFERENCES
571          ;
572          .GLOBL SAVREG          ;SARE REGISTERS
573          .GLOBL RESREG         ;RESTORE REGISTERS
574          .GLOBL HTPAER         ;PARITY VECTOR
575          .GLOBL HRDADRCHK      ;ILLEGAL ADDRESS CHECK
576          .GLOBL CPUCPE         ;CPU ERROR REGISTER
577          .GLOBL CCNTRL         ;CACHE CONTROL REGISTER
578          .GLOBL KONTRL         ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER
579
580          ;*****
581          ; LOCAL EQUATES
582          ;
583          172100 SS.PCSR = 172100 ;START OF PARITY CSRS
584          172136 SS.PCLS = 172136 ;LAST PARITY CSR
585          004000 SS.0KC = 4000    ;ONE K CHUNK
586          000034 SS.28 = 0D28    ;28 DECIMAL
587          177600 SS.IOP = 177600 ;I/O PAGE
588          160000 SS.RFS = 160000 ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE
589
590          ;*****
591          ; REFERENCED BY OTHER MODULES
592          ;
593          .GLOBL SIZPOL          ;MODULE ENTRY POINT
594
595
596
597
598          ;+++++
599          ; OVERLAY REGION GLOBALS
600          ;+++++
601          .GLOBL OV.KBBUF
602          .GLOBL OV.HIKB
603          .GLOBL OV.KBSIZ
604          .GLOBL OV.CQ
605          .GLOBL OV.HICQ
606          .GLOBL OV.CQSIZ
607          .GLOBL OV.TQ
608          .GLOBL OV.HITQ
609          .GLOBL OV.TQSIZ
  
```





```

650 000070'          INLINE <1$:>
(2) 000070'
651
652
653          ;+
654          ; SET RETURN TO 2$ TO RESTORE PSW
655          ; -
656 000070'          LET (SP) := #2$
(4) 000070' 012716 000076'
657 000074'          INLINE <RTI>
(2) 000074' 000002
658
659          ;+
660          ; MUST BE AN 11/70, BECAUSE OF TRAP
661          ; -
662
663 000076'          INLINE <2$:>
(2) 000076'
664 000076'          LET R1 := R1 SET.BY #PDP70
(6) 000076' 052701 010000
665
666          ;+
667          ; RESTORE RESERVED INSTRUCTION TRAP
668          ; -
669
670 000102'          INLINE <3$:>
(2) 000102'
671 000102'          POP @#10
(2) 000102' 012637 000010
672 000106'          ELSE
(4) 000106' 000411
(3) 000110'
673
674          ;+
675          ; CHECK TO SEE IF LSI11
676          ; -
677
678 000110'          CALL HRDADRCHK IN <#PSW>
(3) 000110' 010546
(4) 000112' 012745 177776
(3) 000116' 004767 000000G
(3) 000122' 012605
679 000124'          IF.ERROR THEN
(6) 000124' 103002
680 000126'          LET R1 := R1 SET.BY #PDPLSI
(6) 000126' 052701 020000
681
682 000132'          ENDIF
(4) 000132'
683 000132'          ENDIF
(4) 000132'
684
    
```

```

1$:
MOV #2$, (SP)
RTI
2$:
BIS #PDP70, R1
3$:
MOV (SP)+, @#10
BR 50003$
50002$:
MOV R5, -(SP)
MOV #PSW, -(R5)
JSR PC, HRDADRCHK
MOV (SP)+, R5
BCC 50004$
BIS #PDPLSI, R1
50004$:
50003$:
    
```



```

699
700      .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT
701
702      ;+
703      ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
704      ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
705      ; IN PARITY TABLE.
706      ;-
707
708      000154'      LET R0 := DT.PTA(R2)
(4) 000154' 016200 000064      MOV      DT.PTA(R2),R0
709
710      000160'      LET R3 := #SS.PCSR
(4) 000160' 012703 172100      MOV      #SS.PCSR,R3
711      000164'      WHILE R3 LE #SS.PCLS DD
(4) 000164'
(6) 000164' 020327 172136      50006$:  CMP      R3,#SS.PCLS
(9) 000170' 003035      BGT      50007$
712      000172'      CALL HRDADRCHK IN <R3>
(3) 000172' 010546      MOV      R5,-(SP)
(4) 000174' 010345      MOV      R3,-(R5)
(3) 000176' 004767 000000G    JSR      PC,HRDADRCHK
(3) 000202' 012605      MOV      (SP)+,R5
713      000204'      IF.NO.ERROR THEN
(6) 000204' 103414      BCS      50010$
714      000206'      LET (R0)+ := R3
(4) 000206' 010320      MOV      R3,(R0)+
715
716      ;+
717      ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
718      ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
719      ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
720      ;-
721      000210'      LET (R3) := (R3) SET.BY #BIT01
(6) 000210' 052713 000002      BIS      #BIT01,(R3)
722      000214'      IF #BIT01 SETIN (R3) THEN
(6) 000214' 032713 000002      BIT      #BIT01,(R3)
(9) 000220' 001403      BEQ      50011$
723      000222'      LET R1 := R1 SET.BY #ECCMEM
(6) 000222' 052701 000100      BIS      #ECCMEM,R1
724      000226'      ELSE
(4) 000226' 000402      BR      50012$
(3) 000230'      50011$:
725      000230'      LET R1 := R1 SET.BY #PARPRES
(6) 000230' 052701 002000      BIS      #PARPRES,R1
726      000234'      ENDIF
(4) 000234'      50012$:
727      000234'      LET (R3) := #0
(4) 000234' 005013      CLR      (R3)
728      000236'      ENDIF
(4) 000236'      50010$:
729
730      ;+
731      ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
732      ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
733      ; PARITY CSR + 2

```

```
734                                     ;-
735 000236'                               IF #PDP70 SETIN R1 THEN
(6) 000236' 032701 010000
(9) 000242' 001405
736 000244'                               LET R1 := R1 SET.BY #PARPRES
(6) 000244' 052701 002000
737 000250'                               LET R3 := R3 + #4
(6) 000250' 062703 000004
738 000254'                               ELSE
(4) 000254' 000402
(3) 000256'
739 000256'                               LET R3 := R3 + #2
(6) 000256' 062703 000002
740 000262'                               ENDIF
(4) 000262'
741 000262'                               ENDDO
(4) 000262' 000740
(3) 000264'
742
```

	BIT	#PDP70,R1
	BEQ	50013\$
	BIS	#PARPRES,R1
	ADD	#4,R3
	BR	50014\$
50013\$:		
	ADD	#2,R3
50014\$:		
	BR	50006\$
50007\$:		



782	000352'		IF.NO.ERROR THEN		MOV	R5,-(SP)
783	000354'		LET R3 := R3 + #10000		MOV	#SS.RFS,-(R5)
784	000354'	062703	010000		JSR	PC,HRDADRCHK
785	000360'		ENDIF		MOV	(SP)+,R5
786	000360'		ENDIF		BCS	50020\$
787	000360'				ADD	#10000,R3
788	000360'					
789	000360'					
790	000360'					
791	000360'					
792	000360'					
793	000360'		LET CARRY := #0			
794	000362'		LET DT.SSIZ(R2) := R3 ROTATE -6		CLC	
795	000362'	010362	000046		MOV	R3,DT.SSIZ(R2)
	000366'	006062	000046		ROR	DT.SSIZ(R2)
	000372'	006062	000046		ROR	DT.SSIZ(R2)
	000376'	006062	000046		ROR	DT.SSIZ(R2)
	000402'	006062	000046		ROR	DT.SSIZ(R2)
	000406'	006062	000046		ROR	DT.SSIZ(R2)
	000412'	006062	000046		ROR	DT.SSIZ(R2)

```

;+
; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
; SIZE WORD.
;-

```

```
797 .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT
798
799 ;+
800 ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
801 ; -
802
803 000416' IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
(6) 000416' 032701 002000 BIT #PARPRES,R1
(8) 000422' 001003 BNE 50021$
(6) 000424' 032701 000100 BIT #ECCMEM,R1
(9) 000430' 001424 BEQ 50022$
(6) 000432' 50021$:
804
805 ;+
806 ; IF CACHE PRESENT, SET BIT 0 AND 1 IN CACHE CONTROL REGISTER
807 ; THIS WILL TURN CACHE OFF
808 ; -
809
810 000432' IF #CAPRES SETIN R1 THEN
(6) 000432' 032701 000004 BIT #CAPRES,R1
(9) 000436' 001406 BEQ 50023$
811 000440' LET KONTRL := #3
(4) 000440' 012767 000003 000000G MOV #3,KONTRL
812 000446' LET @CCNTRL := KONTRL
(4) 000446' 016777 000000G 000000G MOV KONTRL,@CCNTRL
813 000454' ENDIF
(4) 000454' 50023$:
814
815 ;+
816 ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
817 ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
818 ; OF MEMORY
819 ; -
820
821 ;+
822 ; SET UP POINTER AND SAVE BUSERR VECTOR
823 ; -
824
825 000454' LET R0 := #0
(4) 000454' 005000 CLR R0
826 000456' PUSH @#4
(2) 000456' 013746 000004 MOV @#4,-(SP)
827 000462' LET @#4 := #6$
(4) 000462' 012737 000474' 000004 MOV #6$,@#4
828
829 ;+
830 ; NOW WRITE GOOD PARITY
831 ; -
832
833 000470' INLINE<5$:>
(2) 000470' 5$:
834 000470' LET (R0)+ := (R0)
(4) 000470' 011020 MOV (R0),(R0)+
835 000472' INLINE <BR 5$>
```







```

907 000572'          LET R3 := 2(R1)
(4) 000572' 016103 000002          MOV      2(R1),R3
908
909          ;+
910          ; MAKE SURE ITS NOT THE END OF THE LIST
911          ; -
912
913 000576'          IF R3 NE #ENDLST THEN
(6) 000576' 020327 000000          CMP      R3,#ENDLST
(9) 000602' 001407          BEQ      50033$
914
915          ;+
916          ; IF ITS NOT A BKMOD, SWAP THEM
917          ; -
918
919 000604'          IF #BKMOD NOTSETIN STAT(R3) THEN
(6) 000604' 032763 000020 000026          BIT      #BKMOD,STAT(R3)
(9) 000612' 001003          BNE      50034$
920 000614'          LET 2(R1) := (R1)
(4) 000614' 011161 000002          MOV      (R1),2(R1)
921 000620'          LET (R1) := R3
(4) 000620' 010311          MOV      R3,(R1)
922 000622'          ENDIF
(4) 000622'          50034$:
923 000622'          ENDIF
(4) 000622'          50033$:
924 000622'          ENDIF
(4) 000622'          50032$:
925
926          ;+
927          ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
928          ; -
929
930 000622'          LET R1 := R1 + #2
(6) 000622' 062701 000002          ADD      #2,R1
931 000626'          UNTIL (R1) EQ #ENDLST
(3) 000626' 021127 000000          CMP      (R1),#ENDLST
(6) 000632' 001352          BNE      50031$
932
933          ;+
934          ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
935          ; -
936
937 000634'          LET R0 := R0 - #1
(6) 000634' 005300          DEC      R0
938 000636'          LET R1 := DT.MLST(R2)
(4) 000636' 016201 000032          MOV      DT.MLST(R2),R1
939 000642'          ENDDO
(4) 000642' 000744          BR      50027$
(3) 000644'          50030$:
940
941          ;+
942          ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
943          ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
944          ; A BACKGROUND LIST) INTO THE DATA TABLE
945          ; -
    
```

```

946
947 000644'          LET R1 := DT.MLST(R2)
(4) 000644' 016201 000032
948
949
950
951
952
953 000650'          LET DT.BLST(R2) := #0
(4) 000650' 005062 000034
954 000654'          LET R3 := #-1
(4) 000654' 012703 177777
955 000660'          WHILE (R1) NE #ENDLST AND R3 NE #0 DO
(4) 000660'
(6) 000660' 021127 000000
(9) 000664' 001420
(6) 000666' 005703
(9) 000670' 001416
956 000672'          LET R4 := (R1)
(4) 000672' 011104
957 000674'          IF #BKMOD SETIN STAT(R4) THEN
(6) 000674' 032764 000020 000026
(9) 000702' 001406
958
959
960
961
962
963 000704'          LET DT.BLST(R2) := R1
(4) 000704' 010162 000034
964 000710'          LET R1 := #ENDLST
(4) 000710' 012701 000000
965 000714'          LET R3 := #0
(4) 000714' 005003
966 000716'          ELSE
(4) 000716' 000402
(3) 000720'
967
968
969
970
971
972 000720'          LET R1 := R1 + #2
(6) 000720' 062701 000002
973 000724'          ENDIF
(4) 000724'
974 000724'          ENDDO
(4) 000724' 000755
(3) 000726'
975
976
977
978
979
980 000726'          CALL RESREG
(3) 000726' 004767 000000G

```

```

;+
; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
;-

```

50035\$:

MOV DT.MLST(R2),R1

CLR DT.BLST(R2)

MOV #-1,R3

CMP (R1),#ENDLST

BEQ 50036\$

TST R3

BEQ 50036\$

MOV (R1),R4

BIT #BKMOD,STAT(R4)

BEQ 50037\$

```

;+
; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
;-

```

50037\$:

MOV R1,DT.BLST(R2)

MOV #ENDLST,R1

CLR R3

BR 50040\$

50040\$:

ADD #2,R1

BR 50035\$

50036\$:

```

;+
; NOW RETURN TO THE CALLER
;-

```

JSR PC,RESREG



ACSR = 000102	CSRA = 000100	ENBNUL= 000001	MAPSTA= 000200	PRI1 = 000040
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	MED = 075600	PRI4 = 000200
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	MEMPAS= 040000	PRI5 = 000240
ADR = 000006	CTRL0 = 000017	ERRTYP= 000106	MODEXH= 004000	PRI6 = 000300
APTFER= 000004	CTRLU = 000025	EVNTBE= 000200	MODH0L= 002000	PRI7 = 000340
APTPRE= 000200	DCEVNT= 000011	EVNTHD= 000200	MODSEL= 001000	PR0 = 000000
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MSGCKD= 000010	PR4 = 000200
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MSGCKS= 000011	PR5 = 000240
ASTAT = 000104	DROPMO= 100000	EVNTRE= 000201	MSGDER= 000005	PR6 = 000300
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGDRP= 000017	PR7 = 000340
AUTOST= 020000	DTABLE= 000000	HRDADR= ***** G	MSGECH= 177777	PS = 177776
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGEOP= 000013	PSW = 177776
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGHDR= 000004	RANNUM= 000054
BIT00 = 000001	DT.APK= 000076	HTPAER= ***** G	MSGHNG= 000022	RBUFEA= 000130
BIT01 = 000002	DT.BLS= 000034	ICONT = 000036	MSGHRD= 000007	RBUFPA= 000126
BIT02 = 000004	DT.CFO= 000014	ICOUNT= 000040	MSGMAP= 000021	RBUFSZ= 000132
BIT03 = 000010	DT.CF1= 000016	IDNUM = 000122	MSGNUL= 177775	RBUFVA= 000124
BIT04 = 000020	DT.ERR= 000020	IE = 000100	MSGPOP= 000002	RDSERV= 000101
BIT05 = 000040	DT.ESI= 000044	INDPAR= 000040	MSGPRM= 177776	RDWHMI= 000022
BIT06 = 000100	DT.EVN= 000000	INHDRP= 040000	MSGRES= 000001	RELERR= 000020
BIT07 = 000200	DT.EXS= 000060	INHPRP= 020000	MSGSFT= 000006	RELMOD= 020000
BIT08 = 000400	DT.FCH= 000037	INHREL= 001000	MSGSKE= 000003	RELTIM= 010000
BIT09 = 001000	DT.FCN= 000036	INHRRP= 000400	MSGSMB= 000015	RESREG= ***** G
BIT1 = 000002	DT.HMX= 000104	INIT = 000030	MSGSMH= 000014	RES1 = 000056
BIT10 = 002000	DT.KBF= 000024	INTR = 000120	MSGSMS= 000016	RES2 = 000060
BIT11 = 004000	DT.KBP= 000026	IOMOD = 100000	MSGSTD= 000000	RICHAR= 031060
BIT12 = 010000	DT.KBR= 000022	IOMODP= 102000	MSGSYS= 000012	RPTDAT= 002000
BIT13 = 020000	DT.KBU= 000030	IOMODR= 112000	MSGVEC= 000020	RSTRT = 000112
BIT14 = 040000	DT.MLS= 000032	IOMODX= 110000	NBKMOD= 001000	RUBOUT= 000177
BIT15 = 100000	DT.MTI= 000110	JACK = 035060	NCPUQP= 000020	RUNMOD= 100000
BIT2 = 000004	DT.OFF= 000070	KIPAR0= 172340	NDAPTY= 000002	RSVALU= 001740
BIT3 = 000010	DT.PAS= 000074	KIPAR1= 172342	NULL = 000000	SAM = 075464
BIT4 = 000020	DT.PC = 000002	KIPAR2= 172344	OV.A = 000420	SAVREG= ***** G
BIT5 = 000040	DT.PFL= 000062	KIPAR3= 172346	OV.CQ = 000122RG	SBADR = 000102
BIT6 = 000100	DT.PSW= 000004	KIPAR4= 172350	OV.CQS= 000036 G	SBKMOD= 000000
BIT7 = 000200	DT.PTA= 000064	KIPAR5= 172352	OV.HIC= 000312RG	SBKSEL= 010000
BIT8 = 000400	DT.RCS= 000102	KIPAR6= 172354	OV.HIK= 000120RG	SC.ADR= 000006
BIT9 = 001000	DT.REL= 000040	KIPAR7= 172356	OV.HIT= 000734RG	SC.ALC= 000014
BKDEF = 000002	DT.SCT= 000066	KIPDR0= 172300	OV.KBB= 000000RG	SC.APC= 000016
BKMOD = 000020	DT.SMX= 000106	KIPDR1= 172302	OV.KBS= 000122 G	SC.CKL= 000002
BKMODE= 040000	DT.SP = 000006	KIPDR2= 172304	OV.REM= 000000	SC.CKP= 000004
BKSLSH= 000134	DT.SSI= 000046	KIPDR3= 172306	OV.TQ = 000314RG	SC.CLO= 000000
CAPRES= 000004	DT.ST0= 000010	KIPDR4= 172310	OV.TQE 000734R	SC.HLD= 000010
CASAT= 000004	DT.ST1= 000012	KIPDR5= 172312	OV.TQS= 000042 G	SC.SCA= 000012
CCNTRL= ***** G	DT.SWR= 000056	KIPDR6= 172314	OWEN = 024020	SENDLS= 177777
CDERCT= 000146	DT.SYP= 000072	KIPDR7= 172316	PAERR = 000010	SIZPOL 000000RG
CDWDCT= 000144	DT.WBU= 000050	KONTRL= ***** G	PARPRE= 002000	SOFCNT= 000042
CKTIM = 100000	DT.WHL= 000054	KTERR0= 000040	PARSTA= 000100	SOPPAS= 000046
CLKPRE= 000001	DT.WLL= 000052	KTPRES= 000400	PASCNT= 000034	SPACE = 000040
CONFIG= 000056	DVID1 = 000014	KTSTAT= 000020	PDPLSI= 020000	SPOINT= 000032
CPUCPE= ***** G	ECCMEM= 000100	KTXTND= 040000	PDP60 = 004000	SPVALU= 002200
CQOVF = 000001	ECCSTA= 000010	LF = 000012	PDP70 = 010000	SRO = 177572
CR = 000015	ENBEQP= 010000	LPSTAT= 000001	PRI0 = 000000	SR1 = 177574

SR2 = 177576	UIPAR2= 177644	XOFF = 000023	\$F\$YES= 000402	\$TSK5 = 050034
SR3 = 172516	UIPAR3= 177646	XON = 000021	\$IFLEV= 177777	\$SARGC= 000002
SS.IOP= 177600	UIPAR4= 177650	\$BGNLE= 177777	\$ISK0 = 000001	\$S\$BYTE= 000403
SS.PCL= 172136	UIPAR5= 177652	\$ERFLG= 000400	\$ISK1 = 000001	\$SCASE= 000000
SS.PCS= 172100	UIPAR6= 177654	\$F\$AND= 000310	\$ISK2 = 000001	\$SDST = 000000
SS.RFS= 160000	UIPAR7= 177656	\$F\$BAD= 000401	\$LOCTA= 177777	\$ELOC= 000402
SS.OKC= 004000	UIPDR0= 177600	\$F\$BLA= 000170	\$LSTIN= 000001	\$SERFL= 000000
SS.28 = 000034	UIPDR1= 177602	\$F\$CAS= 000150	\$LSTTA= 000001	\$FLAG= 000001
STAT = 000026	UIPDR2= 177604	\$F\$DEC= 000220	\$NESTL= 177777	\$FROM= 000000
STATBI= 064757	UIPDR3= 177606	\$F\$DO = 000340	\$NSK0 = 000300	\$LOC = 000702R
STAT1 = 000027	UIPDR4= 177610	\$F\$FAL= 000405	\$NSK1 = 000120	\$LOCN= 000000
SUSPND= 000001	UIPDR5= 177612	\$F\$G00= 000400	\$NSK2 = 000110	\$REG = 177777
SVR0 = 000062	UIPDR6= 177614	\$F\$IF = 000110	\$NSK3 = 000110	\$REU= 000000
SVR1 = 000064	UIPDR7= 177616	\$F\$INC= 000210	\$NSK4 = 000110	\$RTN1= 050000
SVR2 = 000066	WASADR= 000104	\$F\$L00= 000200	\$NSK5 = 000110	\$RTN2= 050001
SVR3 = 000070	WBSTAT= 000040	\$F\$NAM= 000160	\$SAVLE= 177777	\$SRC = 000000
SVR4 = 000072	WBUFEA= 000136	\$F\$ND = 000403	\$SSK0 = 050036	\$TGSV= 000000
SVR5 = 000074	WBUFPA= 000134	\$F\$OR = 000320	\$TAGLE= 177777	\$TGS1= 000000
SVR6 = 000076	WBUFQ= 000140	\$F\$RTI= 000350	\$TAGNU= 050041	\$TGS2= 000000
SYSCNT= 000052	WBUFSZ= 000142	\$F\$RTN= 000300	\$TEMP = 000300	\$TO = 000000
SYSERR= 000100	WDFR = 000116	\$F\$SEL= 000140	\$TSK0 = 050035	\$TAG= 050000
TMPIO = 000002	WDTO = 000114	\$F\$THE= 000330	\$TSK1 = 050036	. = 000734R
TQOVF = 000002	WTINRE= 000352	\$F\$TRU= 000404	\$TSK2 = 050040	
UIPAR0= 177640	WTWHMI= 000222	\$F\$UNT= 000130	\$TSK3 = 050032	
UIPAR1= 177642	XFLAG = 000005	\$F\$WHI= 000120	\$TSK4 = 050033	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:47  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1129

3 COMMON EQUATE MODULE  
564 COMMON DEFINITIONS AND REFERENCES  
567 000000' .PRINT ;SPMAC: VERSION 1.1  
618 KERNEL - DETERMINE PROCESSOR TYPE  
695 KERNEL - DETERMINE CPU OPTIONS  
726 KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT  
771 KERNEL - DETERMINE SYSTEM SIZE  
862 KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT  
987 KERNEL - SORT MODULE LIST



```
508 .TITLE SIZPLC SIZE AND POLL SYSTEM - KT OPTIONS
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     SIZPLC
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
517 ;     OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
518 ;     WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
519 ;     A BACKGROUND MODULE LIST (QUEUE)
520 ;     NOTE: IT WILL NOT SIZE FOR 22-BIT ADDRESSING
521 ;
522 ; INPUTS:
523 ;     DTABLE ADDRESS
524 ;
525 ; IMPLICIT INPUTS:
526 ;     CONFIGURATION WORD 0
527 ;     EXERCISER SIZE
528 ;     SYSTEM SIZE
529 ;     PARITY TABLE ADDRESS
530 ;     MODULE LIST
531 ;     BACKGROUND LIST
532 ;
533 ; OUTPUTS:
534 ;     NONE
535 ;
536 ; IMPLICIT OUTPUTS:
537 ;     PARITY TABLE ADDRESS
538 ;     SYSTEM SIZE
539 ;     EXERCISE SIZE
540 ;     CONFIGURATION WORD 0
541 ;
542 ; PATHOLOGICAL CONNECTIONS:
543 ;
544 ; SUBORDINATE ROUTINES CALLED:
545 ;     SAVREG - SAVE REGISTERS
546 ;     RESREG - RESTORE REGISTERS
547 ;     KTSET - SET UP KT REGISTERS
548 ;     HRDADRCHK - ILLEGAL ADDRESS CHECK
549 ;
550 ; FUNCTIONAL SIDE EFFECTS:
551 ;     NONE
552 ;
553 ; CALLING SEQUENCE:
554 ;     CALL SIZPOL IN <A>
555 ;         A - DTABLE ADDRESS
556 ;
557 ; VERSION:
558 ;     0.0
559 ;
560 ;--- EDIT DATE BY REASON
561
```

```
563  
564 .SBTTL COMMON DEFINITIONS AND REFERENCES  
565  
566 .MCALL STRUCT  
567 000000' STRUCT  
(1) 000000' .PRINT ;SPMAC: VERSION 1.1  
568 000001 $LSTIN = 1  
569 000001 $LSTTAG = 1  
570  
571  
572 ;*****  
573 ; GLOBAL REFERENCES  
574 ;  
575 .GLOBL HTPAER ;PARITY VECTOR  
576 .GLOBL HTKT ;KT VECTOR  
577 .GLOBL SAVREG ;SARE REGISTERS  
578 .GLOBL RESREG ;RESTORE REGISTERS  
579 .GLOBL KTSET ;SET UP KT  
580 .GLOBL HRDADRCHK ;ILLEGAL ADDRESS CHECK  
581 .GLOBL CPUCPE ;11/70 CPU ERROR REGISTER  
582 .GLOBL CCNTRL ;CACHE CONTROL REGISTER  
583 .GLOBL KONTRL ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER  
584  
585 ;*****  
586 ; LOCAL EQUATES  
587 ;  
588 172100 SS.PCSR = 172100 ;START OF PARITY CSRS  
589 172136 SS.PCLS = 172136 ;LAST PARITY CSR  
590 004000 SS.OKC = 4000 ;ONE K CHUNK  
591 000034 SS.28 = ^D28 ;28 DECIMAL  
592 000040 SS.1KP = 40 ;1K IN PAR FORMAT  
593 120000 SS.120K = 120000 ;FIRST ADDRESS IN PAR5 MAP  
594 177600 SS.IOP = 177600 ;I/O PAGE  
595 160000 SS.RFS = 160000 ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE  
596  
597 ;*****  
598 ; REFERENCED BY OTHER MODULES  
599 ;  
600 .GLOBL SIZPOL ;MODULE ENTRY POINT  
601  
602  
603  
604  
605 ;+++++  
606 ; OVERLAY REGION GLOBALS  
607 ;+++++  
608 .GLOBL OV.KBBUF  
609 .GLOBL OV.HIKB  
610 .GLOBL OV.KBSIZ  
611 .GLOBL OV.CQ  
612 .GLOBL OV.HICQ  
613 .GLOBL OV.CQSIZ  
614 .GLOBL OV.TQ  
615 .GLOBL OV.HITQ  
616 .GLOBL OV.TQSIZ
```

```
618 .SBTTL KERNEL - DETERMINE PROCESSOR TYPE
619
620
621
622 000000' ROUTINE SIZPOL <DTABLE>
(2) 000000' SIZPOL:
623
624
625 000000' LET @#114 := #HTPAER
(4) 000000' 012737 000000G 000114 MOV #HTPAER,@#114
626 000006' LET @#116 := #PRI7
(4) 000006' 012737 000340 000116 MOV #PRI7,@#116
627
628 000014' LET @#250 := #HTKT
(4) 000014' 012737 000000G 000250 MOV #HTKT,@#250
629 000022' LET @#252 := #PRI7
(4) 000022' 012737 000340 000252 MOV #PRI7,@#252
630
631 ;+
632 ; STUFF PARITY AND KT VECTORS
633 ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
634 ; R1
635 ;--
636
637 000030' CALL SAVREG
(3) 000030' 004767 000000G JSR PC,SAVREG
638 000034' LET R2 := DTABLE(R5)
(4) 000034' 016502 000000 MOV DTABLE(R5),R2
639 000040' LET R1 := #0
(4) 000040' 005001 CLR R1
640
641 ;+
642 ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
643 ;--
644
645 000042' CALL HRDADRCHK IN <CPUCPE>
(3) 000042' 010546 MOV R5,-(SP)
(4) 000044' 016745 000000G MOV CPUCPE,-(R5)
(3) 000050' 004767 000000G JSR PC,HRDADRCHK
(3) 000054' 012605 MOV (SP)+,R5
646
647 000056' IF.NO.ERROR THEN
(6) 000056' 103422 BCS 50002$
648 ;+
649 ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
650 ; 11/60
651 ;--
652
653 000060' PUSH @#10
(2) 000060' 013746 000010 MOV @#10,-(SP)
654 000064' LET @#10 := #1$
(4) 000064' 012737 000104' 000010 MOV #1$,@#10
655 000072' INLINE <MED>
(2) 000072' 076600 MED
656 000074' INLINE <RDSERV>
(2) 000074' 000101 RDSERV
```

657	000076'			LET R1 := R1 SET.BY #PDP60		
(6)	000076'	052701	004000		BIS	#PDP60,R1
658	000102'			INLINE <BR 3\$>		
(2)	000102'	000405			BR	3\$
659						
660	000104'			INLINE <1\$:>		1\$:
(2)	000104'					
661						
662				:+		
663				; SET RETURN TO 2\$ TO RESTORE PSW		
664				;-		
665						
666	000104'			LET (SP) := #2\$		
(4)	000104'	012716	000112'		MOV	#2\$,(SP)
667	000110'			INLINE <RTI>		
(2)	000110'	000002			RTI	
668						
669				:+		
670				; MUST BE AN 11/70, BECAUSE OF TRAP		
671				;-		
672						
673	000112'			INLINE <2\$:>		2\$:
(2)	000112'					
674	000112'			LET R1 := R1 SET.BY #PDP70		
(6)	000112'	052701	010000		BIS	#PDP70,R1
675						
676				:+		
677				; RESTORE RESERVED INSTRUCTION TRAP		
678				;-		
679						
680	000116'			INLINE <3\$:>		3\$:
(2)	000116'					
681	000116'			POP @#10		
(2)	000116'	012637	000010		MOV	(SP)+,@#10
682	000122'			ELSE		
(4)	000122'	000411			BR	50003\$
(3)	000124'				50002\$:	
683						
684				:+		
685				; CHECK TO SEE IF LSI11		
686				;-		
687						
688	000124'			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
689	000140'			IF.ERROR THEN		
(6)	000140'	103002			BCC	50004\$
690	000142'			LET R1 := R1 SET.BY #PDPLSI		
(6)	000142'	052701	020000		BIS	#PDPLSI,R1
691						
692	000146'			ENDIF		
(4)	000146'				50004\$:	
693	000146'			ENDIF		
(4)	000146'				50003\$:	

```
695          .SBTTL KERNEL - DETERMINE CPU OPTIONS
696
697          ;+
698          ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
699          ; SEE IF EXTENDED KT" IS PRESENT
700          ;-
701
702 000146'    CALL HRDADRCHK IN <#SRO>
(3) 000146' 010546
(4) 000150' 012745 177572
(3) 000154' 004767 000000G
(3) 000160' 012605
703 000162'
(6) 000162' 103413
704 000164'
(6) 000164' 052701 000400
705 000170'
(3) 000170' 010546
(4) 000172' 012745 172516
(3) 000176' 004767 000000G
(3) 000202' 012605
706 000204'
(6) 000204' 103402
707 000206'
(6) 000206' 052701 040000
708
709 000212'
(4) 000212'
710 000212'
(4) 000212'
711
712          ;+
713          ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
714          ; CACHE CONTROL REGISTER
715          ;-
716
717
718
719
720 000212'    CALL HRDADRCHK IN <CCNTRL>
(3) 000212' 010546
(4) 000214' 016745 000000G
(3) 000220' 004767 000000G
(3) 000224' 012605
721 000226'
(5) 000226' 103402
722 000230'
(6) 000230' 052701 000004
723 000234'
(4) 000234'
724
```

MOV R5,-(SP)  
MOV #SRO,-(R5)  
JSR PC,HRDADRCHK  
MOV (SP)+,R5

BCS 50005\$

BIS #KTPRES,R1

MOV R5,-(SP)  
MOV #SR3,-(R5)  
JSR PC,HRDADRCHK  
MOV (SP)+,R5

BCS 50006\$

BIS #KTXTND,R1

50006\$:

50005\$:

MOV R5,-(SP)  
MOV CCNTRL,-(R5)  
JSR PC,HRDADRCHK  
MOV (SP)+,R5

BCS 50007\$

BIS #CAPRES,R1

50007\$:

```

726      .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT
727
728      ;+
729      ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
730      ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
731      ; IN PARITY TABLE.
732      ;-
733
734      000234'      LET R0 := DT.PTA(R2)
(4)      000234' 016200 000064      MOV      DT.PTA(R2),R0
735
736      000240'      LET R3 := #SS.PCSR
(4)      000240' 012703 172100      MOV      #SS.PCSR,R3
737      000244'      WHILE R3 LE #SS.PCLS DO
(4)      000244'
(6)      000244' 020327 172136      50010$:  CMP      R3,#SS.PCLS
(9)      000250' 003035      BGT      50011$
738      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
739      000264'      IF.NO.ERROR THEN
(6)      000264' 103414      BCS      50012$
740      000266'      LET (R0)+ := R3
(4)      000266' 010320      MOV      R3,(R0)+
741
742      ;+
743      ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
744      ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
745      ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
746      ;-
747      000270'      LET (R3) := (R3) SET.BY #BIT01
(6)      000270' 052713 000002      BIS      #BIT01,(R3)
748      000274'      IF #BIT01 SETIN (R3) THEN
(6)      000274' 032713 000002      BIT      #BIT01,(R3)
(9)      000300' 001403      BEQ      50013$
749      000302'      LET R1 := R1 SET.BY #ECCMEM
(6)      000302' 052701 000100      BIS      #ECCMEM,R1
750      000306'      ELSE
(4)      000306' 000402      BR      50014$
(3)      000310'
751      000310'      LET R1 := R1 SET.BY #PARPRES
(6)      000310' 052701 002000      50013$:  BIS      #PARPRES,R1
752      000314'      ENDIF
(4)      000314'
753      000314'      LET (R3) := #0
(4)      000314' 005013      50014$:  CLR      (R3)
754      000316'      ENDIF
(4)      000316'      50012$:
755
756      ;+
757      ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
758      ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
759      ; PARITY CSR + 2
760      ;-

```

761	000316'		IF #PDP70 SETIN R1 THEN		
(6)	000316'	032701			BIT #PDP70,R1
(9)	000322'	001405			50015\$
762	000324'		LET R1 := R1 SET.BY #PARPRES		
(6)	000324'	052701			BIS #PARPRES,R1
763	000330'		LET R3 := R3 + #4		
(6)	000330'	062703			ADD #4,R3
764	000334'		ELSE		
(4)	000334'	000402			BR 50016\$
(3)	000336'			50015\$:	
765	000336'		LET R3 := R3 + #2		
(6)	000336'	062703			ADD #2,R3
766	000342'		ENDIF		
(4)	000342'			50016\$:	
767	000342'		ENDDO		
(4)	000342'	000740			BR 50010\$
(3)	000344'			50011\$:	
768					
769					

```

771      .SBTTL KERNEL - DETERMINE SYSTEM SIZE
772
773      ;+
774      ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
775      ; AND LOAD IT INTO EXERCISER SIZE WORD
776      ; -
777
778      000344'      LET DT.ESIZ(R2) := 0
779      (4) 000344' 016762 000000 000044      MOV      0,DT.ESIZ(R2)
780
781      ;+
782      ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
783      ; THAN 28K.
784      ; -
785
786      000352'      IF #KTPRES NOTSETIN R1 THEN
787      (6) 000352' 032701 000400      BIT      #KTPRES,R1
788      (9) 000356' 001053      BNE      50017$
789
790      ;+
791      ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
792      ; -
793
794      000360'      LET R0 := #0
795      (4) 000360' 005000      CLR      R0
796
797      000362'      LET R3 := #0
798      (4) 000362' 005003      CLR      R3
799
800      ;+
801      ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
802      ; -
803
804      REPEAT
805
806      000364'      LET R0 := R0 + #1
807      (3) 000364'      50020$:      INC      R0
808      (6) 000364' 005200
809
810      000366'      LET R3 := R3 + #SS.0KC
811      (6) 000366' 062703 004000      ADD      #SS.0KC,R3
812
813      000372'      CALL HRDADRCHK IN <R3>
814      (3) 000372' 010546      MOV      R5,-(SP)
815      (4) 000374' 010345      MOV      R3,-(R5)
816      (3) 000376' 004767 000000G      JSR      PC,HRDADRCHK
817      (3) 000402' 012605      MOV      (SP)+,R5
818
819      000404'      IF.ERROR THEN
820      (6) 000404' 103001      BCC      50021$
821
822      000406'      INLINE <BR 4$>
823      (2) 000406' 000403      BR      4$
824
825      000410'      ENDIF
826      (4) 000410'      50021$:
827
828      000410'      UNTIL R0 EQ #SS.28
829      (3) 000410' 020027 000034      CMP      R0,#SS.28
830      (6) 000414' 001363      BNE      50020$
831
832      ;+
833      ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE

```



```

810      ; -
811
812      000416'      INLINE <4$:>
(2)      000416'
813      000416'      IF R0 EQ #SS.28 THEN
(6)      000416' 020027 000034      CMP      R0,#SS.28
(9)      000422' 001011      BNE      50022$
814      000424'      CALL HRDADRCHK IN <#SS.RFS>
(3)      000424' 010546      MOV      R5,-(SP)
(4)      000426' 012745 160000      MOV      #SS.RFS,-(R5)
(3)      000432' 004767 000000G      JSR      PC,HRDADRCHK
(3)      000436' 012605      MOV      (SP)+,R5
815      000440'      IF.ND.ERROR THEN
(6)      000440' 103402      BCS      50023$
816      000442'      LET R3 := R3 + #10000
(6)      000442' 062703 010000      ADD      #10000,R3
817      000446'      ENDIF
(4)      000446'      50023$:
818      000446'      ENDIF
(4)      000446'      50022$:
819
820      ; +
821      ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
822      ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
823      ; SIZE WORD.
824      ; -
825
826      000446'      LET CARRY := #0
(4)      000446' 000241      CLC
827      000450'      LET DT.SSIZ(R2) := R3 ROTATE -6
(4)      000450' 010362 000046      MOV      R3,DT.SSIZ(R2)
(7)      000454' 006062 000046      ROR      DT.SSIZ(R2)
(7)      000460' 006062 000046      ROR      DT.SSIZ(R2)
(7)      000464' 006062 000046      ROR      DT.SSIZ(R2)
(7)      000470' 006062 000046      ROR      DT.SSIZ(R2)
(7)      000474' 006062 000046      ROR      DT.SSIZ(R2)
(7)      000500' 006062 000046      ROR      DT.SSIZ(R2)
828      000504'      ELSE
(4)      000504' 000435      BR      50024$
(3)      000506'      50017$:
829
830      ; +
831      ; KT IS PRESENT, FIRST GO MAP APRS'
832      ; -
833      000506'      CALL KTSET IN <R2>
(3)      000506' 010546      MOV      R5,-(SP)
(4)      000510' 010245      MOV      R2,-(R5)
(3)      000512' 004767 000000G      JSR      PC,KTSET
(3)      000516' 012605      MOV      (SP)+,R5
834
835      ; +
836      ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
837      ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
838      ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
839      ; 1K STEPS
840

```

```
841 000520'          LET @#KIPAR5 := #SS.1KP
(4) 000520' 012737 000040 172352          MOV      #SS.1KP,@#KIPAR5
842
843 000526'          LET @#SRO := @#SRO SET.BY #BIT00
(6) 000526' 052737 000001 177572          BIS      #BIT00,@#SRO
844
845 000534'          REPEAT
(3) 000534'          50025$:
846 000534'          LET @#KIPAR5 := @#KIPAR5 + #SS.1KP
(6) 000534' 062737 000040 172352          ADD      #SS.1KP,@#KIPAR5
847 000542'          CALL HRDADRCHK IN <#SS.120K>
(3) 000542' 010546          MOV      R5,-(SP)
(4) 000544' 012745 120000          MOV      #SS.120K,-(R5)
(3) 000550' 004767 000000G          JSR      PC,HRDADRCHK
(3) 000554' 012605          MOV      (SP)+,R5
848 000556'          IF.ERROR THEN
(6) 000556' 103001          BCC      50026$
849 000560'          INLINE <BR 5$>
(2) 000560' 000404          BR      5$
850 000562'          ENDIF
(4) 000562'          50026$:
851 000562'          UNTIL @#KIPAR5 EQ #SS.IOP
(3) 000562' 023727 172352 177600          CMP      @#KIPAR5,#SS.IOP
(6) 000570' 001361          BNE      50025$
852
853          ;+
854          ; LOAD SYSTEM SIZE WORD WITH CONTENTS OF KIPAR5
855          ;--
856
857 000572'          INLINE <5$:>
(2) 000572'          5$:
858 000572'          LET DT.SSIZ(R2) := @#KIPAR5
(4) 000572' 013762 172352 000046          MOV      @#KIPAR5,DT.SSIZ
859 000600'          ENDIF
(4) 000600'          50024$:
860
```

```

862      .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT
863
864      ;+
865      ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
866      ; -
867
868      000600'      IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
(6)      000600' 032701 002000
(8)      000604' 001003
(6)      000606' 032701 000100
(9)      000612' 001500
(6)      000614'
869
870      ;+
871      ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER
872      ; THIS WILL TURN CACHE OFF
873      ; -
874
875      000614'      IF #CAPRES SETIN R1 THEN
(6)      000614' 032701 000004
(9)      000620' 001406
876      000622'      LET KONTRL := #3
(4)      000622' 012767 000003 000000G
877      000630'      LET @CCNTRL := KONTRL
(4)      000630' 016777 000000G 000000G
878      000636'      ENDIF
(4)      000636'
879
880      ;+
881      ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
882      ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
883      ; OF MEMORY, FIRST DETERMINE IF KT IS PRESENT THEN SET
884      ; UP KT REGISTERS.
885      ; -
886
887      000636'      IF #KTPRES SETIN R1 THEN
(6)      000636' 032701 000400
(9)      000642' 001426
888      000644'      CALL KTSET
(3)      000644' 004767 000000G
889
890      ;+
891      ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
892      ; SET UP KIPAR5 TO MAP TO 0
893      ; -
894
895      000650'      LET @#KIPDR6 := #0
(4)      000650' 005037 172314
896      000654'      LET @#KIPAR5 := #0
(4)      000654' 005037 172352
897
898      ;+
899      ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
900      ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
901      ; SET UP R0 TO USE PAR5
902      ; -

```

```

BIT      #PARPRES,R1
BNE      50027$
BIT      #ECCMEM,R1
BEQ      50030$
50027$:
BIT      #CAPRES,R1
BEQ      50031$
MOV      #3,KONTRL
MOV      KONTRL,@CCNTRL
50031$:

```

```

903
904 000660'          PUSH @#4
(2) 000660' 013746 000004          MOV    @#4,-(SP)
905 000664'          LET @#4 := #8$
(4) 000664' 012737 000760' 000004          MOV    #8$,@#4
906 000672'          PUSH @#250
(2) 000672' 013746 000250          MOV    @#250,-(SP)
907 000676'          LET @#250 := #7$
(4) 000676' 012737 000740' 000250          MOV    #7$,@#250
908 000704'          LET @#SRO := @#SRO SET.BY #BIT00
(6) 000704' 052737 000001 177572          BIS    #BIT00,@#SRO
909 000712'          LET R0 := #SS.120K
(4) 000712' 012700 120000          MOV    #SS.120K,R0
910 000716'          ELSE
(4) 000716' 000406          BR     50033$
(3) 000720'          50032$:
911
912          ;+
913          ; IF KT NOT PRESENT THEN SET UP POINTER AND SAVE BUSERR VECTOR
914          ; -
915
916 000720'          LET R0 := #0
(4) 000720' 005000          CLR    R0
917 000722'          PUSH @#4
(2) 000722' 013746 000004          MOV    @#4,-(SP)
918 000726'          LET @#4 := #10$
(4) 000726' 012737 001006' 000004          MOV    #10$,@#4
919 000734'          ENDIF
(4) 000734'          50033$:
920
921          ;+
922          ; NOW WRITE GOOD PARITY
923          ; -
924 000734'          INLINE<6$:>
(2) 000734'          6$:
925 000734'          LET (R0)+ := (R0)
(4) 000734' 011020          MOV    (R0),(R0)+
926 000736'          INLINE <BR 6$>
(2) 000736' 000776          BR    6$
927
928          ;+
929          ; COME HERE BY WAY OF KT TRAP.
930          ; UPDATE PARS TO NEXT 4K BLOCK AND RESET R0 TO START
931          ; OF BLOCK. LOAD RETURN PC WITH 6$ AND RTI
932          ; -
933
934 000740'          INLINE <7$:>
(2) 000740'          7$:
935 000740'          LET @#KIPARS := @#KIPARS + #200
(6) 000740' 062737 000200 172352          ADD    #200,@#KIPARS
936 000746'          LET R0 := #SS.120K
(4) 000746' 012700 120000          MOV    #SS.120K,R0
937 000752'          LET (SP) := #6$
(4) 000752' 012716 000734'          MOV    #6$,(SP)
938 000756'          INLINE <RTI>
(2) 000756' 000002          RTI

```

```

939
940      ;+
941      ; COME HERE BY WAY OF ILLEGAL ADDRESS TRAP.
942      ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
943      ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
944      ; -
945
946      000760'      INLINE <8$:>
(2) 000760'
947      000760'      LET @#SRO := @#SRO CLR.BY #BIT00
(6) 000760' 042737 000001 177572      BIC      #BIT00,@#SRO
948
949      000766'      LET (SP) := #9$
(4) 000766' 012716 000774'      MOV      #9$,(SP)
950      000772'      INLINE <RTI>
(2) 000772' 000002      RTI
951
952      000774'      INLINE <9$:>
(2) 000774'
953      000774'      POP @#250
(2) 000774' 012637 000250      MOV      (SP)+,@#250
954      001000'      POP @#4
(2) 001000' 012637 000004      MOV      (SP)+,@#4
955      001004'      INLINE <BR 11$>
(2) 001004' 000403      BR 11$
956
957      ;+
958      ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
959      ; STACK AND RESTORE BUSERR
960      ; -
961
962      001006'      INLINE <10$:>
(2) 001006'
963      001006'      INLINE <CMP (SP)+,(SP)+>
(2) 001006' 022626      CMP (SP)+,(SP)+
964      001010'      POP @#4
(2) 001010' 012637 000004      MOV      (SP)+,@#4
965
966      001014'      ENDIF
(4) 001014'      50030$:
967
968
969      ;+
970      ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD0
971      ; -
972
973      001014'      INLINE <11$:>
(2) 001014'
974      001014'      IF @#42 NE #0 THEN
(6) 001014' 005737 000042      TST      @#42
(9) 001020' 001402      BEQ      50034$
975      001022'      LET R1 := R1 SET.BY #AUTO
(6) 001022' 052701 000010      EIS      #AUTO,R1
976      001026'      ENDIF
(4) 001026'      50034$:
977

```

SIZPLC SIZE AND POLL SYSTEM - KT OPTIONS  
SIZPLC.MAC 08-SEP-78 09:35

MACY11 30A(1052) 20-SEP-78 18:47 PAGE 21-11  
KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT

SEQ 1143

```
978          ;+
979          ; CLEAN UP R1 BEFORE LOADING CONFIGURATION WORD 0
980          ; LOAD UP CONFIGURATION WORD 0
981          ; -
982
983 001026'   LET R1 := R1 CLR.BY #JACK
(6) 001026' 042701 035060
984 001032'
(6) 001032' 050162 000014
985
```

BIC #JACK,R1  
BIS R1,DT.CF0(R2)

```
.SBTTL KERNEL - SORT MODULE LIST

987
988
989
990
991      ;+
992      ; COUNT THE MODULES
993      ; -
994      001036'          LET R0 := #0
995      (4) 001036' 005000
996      001040'          LET R1 := DT.MLST(R2)
997      (4) 001040' 016201 000032
998      001044'          WHILE (R1) NE #ENDLST DO
999      (4) 001044'
1000     (6) 001044' 021127 000000          50035$:
1001     (9) 001050' 001404          CMP      (R1),#ENDLST
1002     001052'          LET R0 := R0 + #1          BEQ      50036$
1003     (6) 001052' 005200          INC      R0
1004     001054'          LET R1 := R1 + #2          INC      R0
1005     (6) 001054' 062701 000002          ADD      #2,R1
1006     001060'          ENDDO
1007     (4) 001060' 000771          BR       50035$
1008     (3) 001062'          50036$:
1009
1010     ;+
1011     ; START THE SORT OPERATION, AND SORT "BKMODS" TO BOTTOM OF MODQ
1012     ; LIST, DEVELOPING A "BKMOD QUEUE".
1013     ; -
1014     001062'          LET R1 := DT.MLST(R2)
1015     (4) 001062' 016201 000032          MOV      DT.MLST(R2),R1
1016
1017     ;+
1018     ; WORK THRU THE LIST UNTIL THE END IS REACHED
1019     ; -
1020     001066'          WHILE R0 GT #0 DO
1021     (4) 001066'          50037$:
1022     (6) 001066' 005700          TST      R0
1023     (9) 001070' 003432          BLE      50040$
1024     001072'          REPEAT
1025     (3) 001072'          50041$:
1026
1027     ;+
1028     ; IS THE CURRENT ENTRY A BKMOD MODULE?
1029     ; -
1030     001072'          LET R4 := (R1)
1031     (4) 001072' 011104          MOV      (R1),R4
1032     001074'          IF #BKMOD SETIN STAT(R4) THEN
1033     (6) 001074' 032764 000020 000026          BIT      #BKMOD,STAT(R4)
1034     (9) 001102' 001414          BEQ      50042$
1035
1036     ;+
1037     ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
1038     ; -
1039
1040
1041
1042
```

```

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          ; -

```



```

1065
1066 001156'          LET R1 := DT.MLST(R2)
(4) 001156' 016201 000032                                MOV    DT.MLST(R2),R1
1067
1068                ;+
1069                ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
1070                ; -
1071
1072 001162'          LET DT.BLST(R2) := #0
(4) 001162' 005062 000034                                CLR    DT.BLST(R2)
1073 001166'          LET R3 := #-1
(4) 001166' 012703 177777                                MOV    #-1,R3
1074 001172'          WHILE (R1) NE #ENDLST AND R3 NE #0 DO
(4) 001172'
(6) 001172' 021127 000000                                50045$: CMP    (R1),#ENDLST
(9) 001176' 001420                                BEQ    50046$
(6) 001200' 005703                                TST    R3
(9) 001202' 001416                                BEQ    50046$
1075 001204'          LET R4 := (R1)
(4) 001204' 011104                                MOV    (R1),R4
1076 001206'          IF #BKMOD SETIN STAT(R4) THEN
(6) 001206' 032764 000020 000026                                BIT    #BKMOD,STAT(R4)
(9) 001214' 001406                                BEQ    50047$
1077
1078                ;+
1079                ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
1080                ; -
1081
1082 001216'          LET DT.BLST(R2) := R1
(4) 001216' 010162 000034                                MOV    R1,DT.BLST(R2)
1083 001222'          LET R1 := #ENDLST
(4) 001222' 012701 000000                                MOV    #ENDLST,R1
1084 001226'          LET R3 := #0
(4) 001226' 005003                                CLR    R3
1085 001230'          ELSE
(4) 001230' 000402                                BR     50050$
(3) 001232'                                50047$:
1086
1087                ;+
1088                ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
1089                ; -
1090
1091 001232'          LET R1 := R1 + #2
(6) 001232' 062701 000002                                ADD    #2,R1
1092 001236'          ENDIF
(4) 001236'                                50050$:
1093 001236'          ENDDO
(4) 001236' 000755                                BR     50045$
(3) 001240'                                50046$:
1094
1095                ;+
1096                ; NOW RETURN TO THE CALLER
1097                ; -
1098
1099 001240'          CALL RESREG
(3) 001240' 004767 000000G                                JSR    PC,RESREG

```

```
1100 001244'          ENDRTN
(3) 001244'          50000$:
(3) 001244'          50001$:
(2) 001244' 000207          RTS      PC
1101
1102                ;+
1103                ; LABEL TYPE QUEUE HIGH LIMIT
1104                ; -
1105
1106 001246'          INLINE <OV.TQEND:>          OV.TQEND:
(2) 001246'
1107
1108
1109
1110
1111
1112                ;*****
1113                ; NOTE: THESE OVERLAY EQUATES MUST BE LEFT IN THIS POSITION
1114                ; (THAT IS, AFTER SIZPOL AND TQ.END LABELS)
1115
1116                ;+++++
1117                ; OVERLAY REGIONS EQUATES
1118                ;+++++
1119 000000'          OV.KBBUF = SIZPOL          ;KEYBOARD BUFFER STARTING ADDRESS
1120 000122'          OV.CQ= OV.KBBUF + ^D<82>          ;CONTROL QUEUE STARTING ADDRESS
1121 000364'          OV.TQ = OV.CQ + ^D<162>          ;TYPE QUEUE STARTING ADDRESS
1122 000120'          OV.HIKB = OV.CQ - 2          ;KEYBOARD BUFFER HIGH LIMIT
1123 000122'          OV.KBSIZ = OV.CQ - OV.KBBUF          ;KEYBOARD BUFFER SIZE
1124 000362'          OV.HICQ = OV.TQ - 2          ;CONTROL QUEUE HIGH LIMIT
1125 000050'          OV.CQSIZ = <OV.TQ - OV.CQ>/4          ;CONTROL QUEUE SIZE
1126 000662'          OV.A = OV.TQEND - OV.TQ          ;GET THE TQ SIZE
1127 000002'          OV.REM = OV.A - <<OV.A/10>*10>          ;GET ANY REMAINDER
1128 001244'          OV.HITQ = OV.TQEND - OV.REM          ;TYPE QUEUE HIGH LIMIT
1129 000066'          OV.TQSIZ = <OV.TQEND - OV.TQ - OV.REM> /10          ;TYPE QUEUE SIZE
1130
1131 000001'          .END
```

ACSR = 000102	CSRA = 000100	ENBNUL= 000001	LF = 000012	PDP70 = 010000
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	LPSTAT= 000001	PRI0 = 000000
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	MAPSTA= 000200	PRI1 = 000040
ADR = 000006	CTRLD = 000017	ERRTPY= 000106	MED = 076600	PRI4 = 000200
APTFER= 000004	CTRLU = 000025	EVNTBE= 000200	MEMPAS= 040000	PRI5 = 000240
APTPRE= 000200	DCEVNT= 000011	EVNTHD= 000200	MODEXH= 004000	PRI6 = 000300
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MODHQL= 002000	PRI7 = 000340
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MODSEL= 001000	PRO = 000000
ASTAT = 000104	DROPMO= 100000	EVNTRE= 000201	MSGCKD= 000010	PR4 = 000200
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGCKS= 000011	PR5 = 000240
AUTDST= 020000	DTABLE= 000000	HRDADR= ***** G	MSGDER= 000005	PR6 = 000300
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGDRP= 000017	PR7 = 000340
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGECH= 177777	PS = 177776
BIT00 = 000001	DT.APK= 000076	HTKT = ***** G	MSGEOP= 000013	PSW = 177776
BIT01 = 000002	DT.BLS= 000034	HTPAER= ***** G	MSGHDR= 000004	RANNUM= 000054
BIT02 = 000004	DT.CFO= 000014	ICONT = 000036	MSGHNG= 000022	RBUFEA= 000130
BIT03 = 000010	DT.CF1= 000016	ICOUNT= 000040	MSGHRD= 000007	RBUFPA= 000126
BIT04 = 000020	DT.ERR= 000020	IDNUM = 000122	MSGMAP= 000021	RBUFSZ= 000132
BIT05 = 000040	DT.ESI= 000044	IE = 000100	MSGNUL= 177775	RBUFVA= 000124
BIT06 = 000100	DT.EVN= 000000	INDPAR= 000040	MSGPOP= 000002	RDSERV= 000101
BIT07 = 000200	DT.EXS= 000060	INHDRP= 040000	MSGPRM= 177776	RDWHMI= 000022
BIT08 = 000400	DT.FCH= 000037	INHEPR= 020000	MSGRES= 000001	RELERR= 000020
BIT09 = 001000	DT.FCN= 000036	INHREL= 001000	MSGSFT= 000006	RELMOD= 020000
BIT1 = 000002	DT.HMX= 000104	INHRRE= 000400	MSGSKE= 000003	RELTIM= 010000
BIT10 = 002000	DT.KBE= 000024	INIT = 000030	MSGSMB= 000015	RESREG= ***** G
BIT11 = 004000	DT.KBP= 000026	INTR = 000120	MSGSMH= 000014	RES1 = 000056
BIT12 = 010000	DT.KBR= 000022	IOMOD = 100000	MSGSMS= 000016	RES2 = 000060
BIT13 = 020000	DT.KBU= 000030	IOMODP= 102000	MSGSTD= 000000	RICHAR= 031060
BIT14 = 040000	DT.MLS= 000032	IOMODR= 112000	MSGSYS= 000012	RPTDAT= 002000
BIT15 = 100000	DT.MTI= 000110	IOMODX= 110000	MSGVEC= 000020	RSTRT = 000112
BIT2 = 000004	DT.OFF= 000070	JACK = 035060	NBKM0D= 001000	RUBOUT= 000177
BIT3 = 000010	DT.PAS= 000074	KIPAR0= 172340	NCPUOP= 000020	RUNMOD= 100000
BIT4 = 000020	DT.PC = 000002	KIPAR1= 172342	NDAPTY= 000002	RVALU= 001740
BIT5 = 000040	DT.PFL= 000062	KIPAR2= 172344	NULL = 000000	SAM = 075464
BIT6 = 000100	DT.PSW= 000004	KIPAR3= 172346	OV.A = 000662	SAVREG= ***** G
BIT7 = 000200	DT.PTA= 000064	KIPAR4= 172350	OV.CQ = 000122RG	SBADR = 000102
BIT8 = 000400	DT.RCS= 000102	KIPAR5= 172352	OV.CQS= 000050 G	SBKMOD= 000000
BIT9 = 001000	DT.REL= 000040	KIPAR6= 172354	OV.HIC= 000362RG	SBKSEL= 010000
BKDEF = 000002	DT.SCT= 000066	KIPAR7= 172356	OV.HIK= 000120RG	SC.ADR= 000006
BKMOD = 000020	DT.SMX= 000106	KIPDR0= 172300	OV.HIT= 001244RG	SC.ALC= 000014
BKMODE= 040000	DT.SP = 000006	KIPDR1= 172302	OV.KBB= 000000RG	SC.APC= 000016
BKSLSH= 000134	DT.SSI= 000046	KIPDR2= 172304	OV.KBS= 000122 G	SC.CKL= 000002
CAPRES= 000004	DT.STO= 000010	KIPDR3= 172306	OV.REM= 000002	SC.CKP= 000004
CASAT= 000004	DT.ST1= 000012	KIPDR4= 172310	OV.TQ = 000364RG	SC.CLO= 000000
CCNTRL= ***** G	DT.SWR= 000056	KIPDR5= 172312	OV.TQE 001246R	SC.HLD= 000010
CDERCT= 000146	DT.SYP= 000072	KIPDR6= 172314	OV.TQS= 000066 G	SC.SCA= 000012
CDWDCT= 000144	DT.WBU= 000050	KIPDR7= 172316	OWEN = 024020	SENDLS= 177777
CKTIM = 100000	DT.WHL= 000054	KONTRL= ***** G	PAERR = 000010	SIZPOL 000000RG
CLKPRE= 000001	DT.WLL= 000052	KTERR0= 000040	PARPRE= 002000	SOFCNT= 000042
CONFIG= 000056	DVID1 = 000014	KTPRES= 000400	PARSTA= 000100	SOPPAS= 000046
CPUCPE= ***** G	ECCMEM= 000100	KTSET = ***** G	PASCNT= 000034	SPACE = 000040
CQOVF = 000001	ECCSTA= 000010	KTSTAT= 000020	PDPLSI= 020000	SPPOINT= 000032
CR = 000015	ENBEOP= 010000	KTXTND= 040000	PDP60 = 004000	SPVALU= 002200

SR0 = 177572	TQOVF = 000002	WTWHMI= 000222	\$F\$WHI= 000120	\$TSK5 = 050044
SR1 = 177574	UIPAR0= 177640	XFLAG = 000005	\$F\$YES= 000402	\$SARGC= 000002
SR2 = 177576	UIPAR1= 177642	XOFF = 000023	\$IFLEV= 177777	\$S\$BYTE= 000403
SR3 = 172516	UIPAR2= 177644	XQN = 000021	\$ISKO = 000001	\$SCASE= 000000
SS.IDP= 177600	UIPAR3= 177646	\$BGNLE= 177777	\$ISK1 = 000001	\$SDST = 000000
SS.PCL= 172136	UIPAR4= 177650	\$ERFLG= 000400	\$ISK2 = 000001	\$SELDC= 000402
SS.PCS= 172100	UIPAR5= 177652	\$F\$AND= 000310	\$LOCTA= 177777	\$SERFL= 000000
SS.RFS= 160000	UIPAR6= 177654	\$F\$BAD= 000401	\$LSTIN= 000001	\$FLAG= 000001
SS.OKC= 004000	UIPAR7= 177656	\$F\$BLA= 000170	\$LSTTA= 000001	\$FROM= 000000
SS.1KP= 000040	UIPDR0= 177600	\$F\$CAS= 000150	\$NESTL= 177777	\$LLOC = 001214R
SS.120= 120000	UIPDR1= 177602	\$F\$DEC= 000220	\$NSKO = 000300	\$LLOCN= 000000
SS.28 = 000034	UIPDR2= 177604	\$F\$DO = 000340	\$NSK1 = 000120	\$REG = 177777
STAT = 000026	UIPDR3= 177606	\$F\$FAL= 000405	\$NSK2 = 000110	\$REU= 000000
STATBI= 064757	UIPDR4= 177610	\$F\$G00= 000400	\$NSK3 = 000110	\$RTN1= 050000
STAT1 = 000027	UIPDR5= 177612	\$F\$IF = 000110	\$NSK4 = 000110	\$RTN2= 050001
SUSPND= 000001	UIPDR6= 177614	\$F\$INC= 000210	\$NSK5 = 000110	\$SRC = 000000
SVR0 = 000062	UIPDR7= 177616	\$F\$L00= 000200	\$SAVLE= 177777	\$TGSV= 000000
SVR1 = 000064	WASADR= 000104	\$F\$NAM= 000160	\$SSKO = 050046	\$TGS1= 000000
SVR2 = 000066	WBSTAT= 000040	\$F\$NO = 000403	\$TAGLE= 177777	\$TGS2= 000000
SVR3 = 000070	WBUFEA= 000136	\$F\$OR = 000320	\$TAGNU= 050051	\$TO = 000000
SVR4 = 000072	WBUFPA= 000134	\$F\$RTI= 000350	\$TEMP = 000300	\$TAG= 050000
SVR5 = 000074	WBUFRQ= 000140	\$F\$RTN= 000300	\$TSKO = 050045	. = 001246R
SVR6 = 000076	WBUFSZ= 000142	\$F\$SEL= 000140	\$TSK1 = 050046	
SYS CNT= 000052	WDFR = 000116	\$F\$THE= 000330	\$TSK2 = 050050	
SYSERR= 000100	WDTO = 000114	\$F\$TRU= 000404	\$TSK3 = 050042	
TMPIO = 000002	WTINRE= 000352	\$F\$UNT= 000130	\$TSK4 = 050043	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:49  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1150

3 COMMON EQUATE MODULE  
564 COMMON DEFINITIONS AND REFERENCES  
567 000000' .PRINT ;SPMAC: VERSION 1.1  
618 KERNEL - DETERMINE PROCESSOR TYPE  
696 KERNEL - DETERMINE CPU OPTIONS  
727 KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT  
772 KERNEL - DETERMINE SYSTEM SIZE  
863 KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT  
987 KERNEL - SORT MODULE LIST

```
508 .TITLE SIZPLD SIZE AND POLL SYSTEM - KT OPTIONS AND 11/60 SUPPORT
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ; SIZPLD
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ; THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
517 ; OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
518 ; WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
519 ; A BACKGROUND MODULE LIST (QUEUE)
520 ; NOTE: IT WILL NOT SIZE FOR 22-BIT ADDRESSING BUT WILL SIZE FOR 11/60 CPU
521 ;
522 ; INPUTS:
523 ; DTABLE ADDRESS
524 ;
525 ; IMPLICIT INPUTS:
526 ; CONFIGURATION WORD 0
527 ; EXERCISER SIZE
528 ; SYSTEM SIZE
529 ; PARITY TABLE ADDRESS
530 ; MODULE LIST
531 ; BACKGROUND LIST
532 ;
533 ; OUTPUTS:
534 ; NONE
535 ;
536 ; IMPLICIT OUTPUTS:
537 ; PARITY TABLE ADDRESS
538 ; SYSTEM SIZE
539 ; EXERCISE SIZE
540 ; CONFIGURATION WORD 0
541 ;
542 ; PATHOLOGICAL CONNECTIONS:
543 ;
544 ; SUBORDINATE ROUTINES CALLED:
545 ; SAVREG - SAVE REGISTERS
546 ; RESREG - RESTORE REGISTERS
547 ; KTSET - SET UP KT REGISTERS
548 ; HRDADRCHK - ILLEGAL ADDRESS CHECK
549 ;
550 ; FUNCTIONAL SIDE EFFECTS:
551 ; NONE
552 ;
553 ; CALLING SEQUENCE:
554 ; CALL SIZPOL IN <A>
555 ; A - DTABLE ADDRESS
556 ;
557 ; VERSION:
558 ; 0.0
559 ;
560 ; EDIT DATE BY REASON
561 ;--
```

```
563
564 .SBTTL COMMON DEFINITIONS AND REFERENCES
565
566 .MCALL STRUCT
567 000000' STRUCT
(1) 000000' .PRINT ;SPMAC: VERSION 1.1
568 000001 $LSTIN = 1
569 000001 $LSTTAG = 1
570
571
572 ;*****
573 ; GLOBAL REFERENCES
574 ;
575 .GLOBL HTKT
576 .GLOBL HTPAER
577 .GLOBL SAVREG ;SARE REGISTERS
578 .GLOBL RESREG ;RESTORE REGISTERS
579 .GLOBL KTSET ;SET UP KT
580 .GLOBL HRDADRCHK ;ILLEGAL ADDRESS CHECK
581 .GLOBL CPUCPE ;11/70 CPU ERROR REGISTER
582 .GLOBL CCNTRL ;CACHE CONTROL REGISTER
583 .GLOBL KONTRL ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER
584
585 ;*****
586 ; LOCAL EQUATES
587 ;
588 172100 SS.PCSR = 172100 ;START OF PARITY CSRS
589 172136 SS.PCLS = 172136 ;LAST PARITY CSR
590 004000 SS.OKC = 4000 ;ONE K CHUNK
591 000034 SS.28 = ^D28 ;28 DECIMAL
592 000040 SS.1KP = 40 ;1K IN PAR FORMAT
593 120000 SS.120K = 120000 ;FIRST ADDRESS IN PAR5 MAP
594 177600 SS.IOP = 177600 ;I/O PAGE
595 160000 SS.RFS = 160000 ;REVERSED FOR DIAGNOSTIC USE IN I/O PAGE
596
597 ;*****
598 ; REFERENCED BY OTHER MODULES
599 ;
600 .GLOBL SIZPOL ;MODULE ENTRY POINT
601
602
603
604
605 ;+++++
606 ; OVERLAY REGION GLOBALS
607 ;+++++
608 .GLOBL OV.KBBUF
609 .GLOBL OV.HIKB
610 .GLOBL OV.KBSIZ
611 .GLOBL OV.CQ
612 .GLOBL OV.HICQ
613 .GLOBL OV.CQSIZ
614 .GLOBL OV.TQ
615 .GLOBL OV.HITQ
616 .GLOBL OV.TQSIZ
```

```

618          .SBTTL KERNEL - DETERMINE PROCESSOR TYPE
619
620
621
622 000000'          ROUTINE SIZPOL <DTABLE>
(2) 000000'
623
624          ;+
625          ; STUFF PARITY AND KT VECTORS
626          ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
627          ; R1
628          ; -
629
630
631 000000'          LET @#114 := #HTPAER
(4) 000000' 012737 000000G 000114          MOV          #HTPAER,@#114
632 000006'          LET @#116 := #PRI7
(4) 000006' 012737 000340 000116          MOV          #PRI7,@#116
633
634 000014'          LET @#250 := #HTKT
(4) 000014' 012737 000000G 000250          MOV          #HTKT,@#250
635 000022'          LET @#252 := PRI7
(4) 000022' 016737 000340 000252          MOV          PRI7,@#252
636
637 000030'          CALL SAVREG
(3) 000030' 004767 000000G          JSR          PC,SAVREG
638 000034'          LET R2 := DTABLE(R5)
(4) 000034' 016502 000000          MOV          DTABLE(R5),R2
639 000040'          LET R1 := #0
(4) 000040' 005001          CLR          R1
640
641
642          ;+
643          ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
644          ; -
645 000042'          CALL HRDADRCHK IN <CPUCPE>
(3) 000042' 010546          MOV          R5,-(SP)
(4) 000044' 016745 000000G          MOV          CPUCPE,-(R5)
(3) 000050' 004767 000000G          JSR          PC,HRDADRCHK
(3) 000054' 012605          MOV          (SP)+,R5
646
647 000056'          IF.NO.ERROR THEN
(6) 000056' 103422          BCS          50002$
648
649          ;+
650          ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
651          ; 11/60
652          ; -
653 000060'          PUSH @#10
(2) 000060' 013746 000010          MOV          @#10,-(SP)
654 000064'          LET @#10 := #1$
(4) 000064' 012737 000104' 000010          MOV          #1$,@#10
655 000072'          INLINE <MED>
(2) 000072' 076600          MED
656 000074'          INLINE <RDSERV>
(2) 000074' 00010i          RDSERV
    
```



```

657 000076'          LET R1 := R1 SET.BY #PDP60
(6) 000076' 052701 004000          BIS      #PDP60,R1
658 000102'          INLINE <BR 3$>
(2) 000102' 000405          BR      3$
659
660 000104'          INLINE <1$:>
(2) 000104'          1$:
661
662          ;+
663          ; SET RETURN TO 2$ TO RESTORE PSW
664          ; -
665
666 000104'          LET (SP) := #2$
(4) 000104' 012716 000112'          MOV      #2$, (SP)
667 000110'          INLINE <RTI>
(2) 000110' 000002          RTI
668
669          ;+
670          ; MUST BE AN 11/70
671          ; -
672
673 000112'          INLINE <2$:>
(2) 000112'          2$:
674 000112'          LET R1 := R1 SET.BY #PDP70
(6) 000112' 052701 010000          BIS      #PDP70,R1
675
676          ;+
677          ; RESTORE RESERVED INSTRUCTION TRAP
678          ; -
679
680 000116'          INLINE <3$:>
(2) 000116'          3$:
681 000116'          POP @#10
(2) 000116' 012637 000010          MOV      (SP)+,@#10
682 000122'          ELSE
(4) 000122' 000411          BR      50003$
(3) 000124'          50002$:
683
684          ;+
685          ; CHECK TO SEE IF LSI11
686          ; -
687
688 000124'          CALL HRDADRCHK IN <#PSW>
(3) 000124' 010546          MOV      R5,-(SP)
(4) 000126' 012745 177776          MOV      #PSW,-(R5)
(3) 000132' 004767 000000G        JSR      PC,HRDADRCHK
(3) 000136' 012605          MOV      (SP)+,R5
689 000140'          IF.ERROR THEN
(6) 000140' 103002          BCC      50004$
690 000142'          LET R1 := R1 SET.BY #PDPLSI
(6) 000142' 052701 020000          BIS      #PDPLSI,R1
691
692 000146'          ENDF
(4) 000146'          50004$:
693 000146'          ENDF
(4) 000146'          50003$:
    
```

SIZPLD SIZE AND POLL SYSTEM - KT OPTIONS AND 11/60 SUPPORT  
SIZPLD.MAC 08-SEP-78 09:35

MACY11 30A(1052)

20-SEP-78 18:49 PAGE 21-2

KERNEL - DETERMINE PROCESSOR TYPE

SEQ 1155

694

```

696          .SBTTL KERNEL - DETERMINE CPU OPTIONS
697
698          ;+
699          ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
700          ; SEE IF EXTENDED KT" IS PRESENT
701          ; -
702
703          CALL HRDADRCHK IN <#SRO>
704          (3) 000146' 010546          MOV      R5,-(SP)
705          (4) 000150' 012745 177572    MOV      #SRO,-(R5)
706          (3) 000154' 004767 000000G   JSR      PC,HRDADRCHK
707          (3) 000160' 012605          MOV      (SP)+,R5
708
709          IF.NO.ERROR THEN
710          (6) 000162' 103413          BCS     50005$
711          (3) 000164' 052701 000400    LET R1 := R1 SET.BY #KTPRES
712          (6) 000164' 052701 000400    BIS     #KTPRES,R1
713          (3) 000170' 010546          CALL HRDADRCHK IN <#SR3>
714          (4) 000172' 012745 172516    MOV      R5,-(SP)
715          (3) 000176' 004767 000000G   MOV      #SR3,-(R5)
716          (3) 000202' 012605          JSR      PC,HRDADRCHK
717          (3) 000204' 012605          MOV      (SP)+,R5
718
719          IF.NO.ERROR THEN
720          (6) 000204' 103402          BCS     50006$
721          (3) 000206' 052701 040000    LET R1 := R1 SET.BY #KTXTND
722          (6) 000206' 052701 040000    BIS     #KTXTND,R1
723
724          ENDIF
725          (4) 000212'          50006$:
726          (4) 000212'          ENDIF          50005$:
727
728          ;+
729          ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
730          ; CACHE CONTROL REGISTER
731          ; -
732
733          CALL HRDADRCHK IN <CCNTRL>
734          (3) 000212' 010546          MOV      R5,-(SP)
735          (4) 000214' 016745 000000G   MOV      CCNTRL,-(R5)
736          (3) 000220' 004767 000000G   JSR      PC,HRDADRCHK
737          (3) 000224' 012605          MOV      (SP)+,R5
738
739          IF.NO.ERROR THEN
740          (6) 000226' 103402          BCS     50007$
741          (3) 000230' 052701 000004    LET R1 := R1 SET.BY #CAPRES
742          (6) 000230' 052701 000004    BIS     #CAPRES,R1
743          (3) 000234'          ENDIF          50007$:
744          (4) 000234'
745

```

```

727      .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT
728
729      ;+
730      ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
731      ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
732      ; IN PARITY TABLE.
733      ;-
734
735      000234'      LET R0 := DT.PTA(R2)
736      (4) 000234' 016200 000064      MOV      DT.PTA(R2),R0
737
738      000240'      LET R3 := #SS.PCSR
739      (4) 000240' 012703 172100      MOV      #SS.PCSR,R3
740
741      000244'      WHILE R3 LE #SS.PCLS DO
742      (4) 000244'      50010$:
743      (6) 000244' 020327 172136      CMP      R3,#SS.PCLS
744      (9) 000250' 003035      BGT      50011$
745
746      000252'      CALL HRDADRCHK IN <R3>
747      (3) 000252' 010546      MOV      R5,-(SP)
748      (4) 000254' 010345      MOV      R3,-(R5)
749      (3) 000256' 004767 000000G    JSR      PC,HRDADRCHK
750      (3) 000262' 012605      MOV      (SP)+,R5
751
752      000264'      IF.NO.ERROR THEN
753      (6) 000264' 103414      BCS      50012$
754
755      000266'      LET (R0)+ := R3
756      (4) 000266' 010320      MOV      R3,(R0)+
757
758      ;+
759      ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
760      ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
761      ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
762      ;-
763
764      000270'      LET (R3) := (R3) SET.BY #BIT01
765      (6) 000270' 052713 000002      BIS      #BIT01,(R3)
766
767      000274'      IF #BIT01 SET IN (R3) THEN
768      (6) 000274' 032713 000002      BIT      #BIT01,(R3)
769      (9) 000300' 001403      BEQ      50013$
770
771      000302'      LET R1 := R1 SET.BY #ECCMEM
772      (6) 000302' 052701 000100      BIS      #ECCMEM,R1
773
774      000306'      ELSE
775      (4) 000306' 000402      BR      50014$
776
777      000310'      LET R1 := R1 SET.BY #PARPRES
778      (3) 000310'      50013$:
779      (6) 000310' 052701 002000      BIS      #PARPRES,R1
780
781      000314'      ENDIF
782      (4) 000314'      50014$:
783
784      000314'      LET (R3) := #0
785      (4) 000314' 005013      CLR      (R3)
786
787      000316'      ENDIF
788      (4) 000316'      50012$:
789
790      ;+
791      ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
792      ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
793      ; PARITY CSR + 2
794      ;-
    
```

```
762 000316'          IF #PDP70 SETIN R1 THEN
(6) 000316' 032701 010000
(9) 000322' 001405
763 000324'          LET R1 := R1 SET.BY #PARPRES
(6) 000324' 052701 002000
764 000330'          LET R3 := R3 + #4
(6) 000330' 062703 000004
765 000334'          ELSE
(4) 000334' 000402
(3) 000336'          50015$: BR 50016$
766 000336'          LET R3 := R3 + #2
(6) 000336' 062703 000002
767 000342'          ENDIF
(4) 000342'          50016$:
768 000342'          ENDDO
(4) 000342' 000740
(3) 000344'          50011$: BR 50010$
769
770
```

```

772      .SBTTL KERNEL - DETERMINE SYSTEM SIZE
773
774      ;+
775      ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
776      ; AND LOAD IT INTO EXERCISER SIZE WORD
777      ; -
778
779      000344'      LET DT.ESIZ(R2) := 0
(4)      000344' 016762 000000 000044      MOV      0,DT.ESIZ(R2)
780
781
782      ;+
783      ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
784      ; THAN 28K.
785      ; -
786
787      000352'      IF #KTPRES NOTSETIN R1 THEN
(6)      000352' 032701 000400      BIT      #KTPRES,R1
(9)      000356' 001053      BNE     50017$
788
789      ;+
790      ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
791      ; -
792
793      000360'      LET R0 := #0
(4)      000360' 005000      CLR     R0
794      000362'      LET R3 := #0
(4)      000362' 005003      CLR     R3
795
796      ;+
797      ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
798      ; -
799
800      000364'      REPEAT
(3)      000364'      50020$:
801      000364'      LET R0 := R0 + #1
(6)      000364' 005200      INC     R0
802      000366'      LET R3 := R3 + #SS.0KC
(6)      000366' 062703 004000      ADD     #SS.0KC,R3
803      000372'      CALL HRDADRCHK IN <R3>
(3)      000372' 010546      MOV     R5,-(SP)
(4)      000374' 010345      MOV     R3,-(R5)
(3)      000376' 004767 000000G      JSR     PC,HRDADRCHK
(3)      004402' 012605      MOV     (SP)+,R5
804      000404'      IF.ERROR THEN
(6)      000404' 103001      BCC     50021$
805      000406'      INLINE <BR 4$>
(2)      000406' 000403      BR     4$
806      000410'      ENDIF
(4)      000410'      50021$:
807      000410'      UNTIL R0 EQ #SS.28
(3)      000410' 020027 000034      CMP     R0,#SS.28
(6)      000414' 001363      BNE     50020$
808
809      ;+
810      ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
    
```

```

811          ; -
812
813 000416'    INLINE <4$:>
(2) 000416'
814 000416'    IF R0 EQ #SS.28 THEN
(6) 000416' 020027 000034
(9) 000422' 001011
815 000424'    CALL HRDADRCHK IN <#SS.RFS>
(3) 000424' 010546
(4) 000426' 012745 160000
(3) 000432' 004767 000000G
(3) 000436' 012605
816 000440'    IF.NO.ERROR THEN
(6) 000440' 103402
817 000442'    LET R3 := R3 + #10000
(6) 000442' 062703 010000
818 000446'    ENDIF
(4) 000446'
819 000446'    ENDIF
(4) 000446'
820
821          ; +
822          ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
823          ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
824          ; SIZE WORD.
825          ; -
826
827 000446'    LET CARRY := #0
(4) 000446' 000241
828 000450'    LET DT.SSIZ(R2) := R3 ROTATE -6
(4) 000450' 010362 000046
(7) 000454' 006062 000046
(7) 000460' 006062 000046
(7) 000464' 006062 000046
(7) 000470' 006062 000046
(7) 000474' 006062 000046
(7) 000500' 006062 000046
829 000504'    ELSE
(4) 000504' 000435
(3) 000506'
830
831          ; +
832          ; KT IS PRESENT, FIRST GO MAP PAR'S AND PDR'S
833          ; -
834 000506'    CALL KTSET IN <R2>
(3) 000506' 010546
(4) 000510' 010245
(3) 000512' 004767 000000G
(3) 000516' 012605
835
836          ; +
837          ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
838          ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
839          ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
840          ; 1K STEPS
841

```

4\$:

CMP R0,#SS.28  
 BNE 50022\$

MOV R5,-(SP)  
 MOV #SS.RFS,-(R5)  
 JSR PC,HRDADRCHK  
 MOV (SP)+,R5

BCS 50023\$

ADD #10000,R3

50023\$:

50022\$:

CLC

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)

BR 50024\$

50017\$:

MOV R5,-(SP)  
 MOV R2,-(R5)  
 JSR PC,KTSET  
 MOV (SP)+,R5

```

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'
847 000534'          LET @#KIPAR5 := @#KIPAR5 + #SS.1KP          50025$:
(6) 000534' 062737 000040 172352          ADD      #SS.1KP,@#KIPAR5
848 000542'          CALL HRDADRCHK IN <#SS.120K>
(3) 000542' 010546          MOV      R5,-(SP)
(4) 000544' 012745 120000          MOV      #SS.120K,-(R5)
(3) 000550' 004767 000000G        JSR      PC,HRDADRCHK
(3) 000554' 012605          MOV      (SP)+,R5
849 000556'          IF.ERROR THEN
(6) 000556' 103001          BCC      50026$
850 000560'          INLINE <BR 5$>
(2) 000560' 000404          BR      5$
851 000562'          ENDIF
(4) 000562'
852 000562'          UNTIL @#KIPAR5 EQ #SS.IOP          50026$:
(3) 000562' 023727 172352 177600        CMP      @#KIPAR5,#SS.IOP
(6) 000570' 001361          BNE      50025$
853
854          ;+
855          ; LOAD SYSTEM SIZE WORD WITH CONTENTS OF KIPAR5
856          ; -
857
858 000572'          INLINE <5$:>
(2) 000572'
859 000572'          LET DT.SSIZ(R2) := @#KIPAR5          5$:
(4) 000572' 013762 172352 000046        MOV      @#KIPAR5,DT.SSIZ
860 000600'          ENDIF
(4) 000600'          50024$:
861

```



```

863 .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT
864
865 ;+
866 ; DETERMINE IF PARITY OF ECC MEMORY IS PRESENT.
867 ; -
868
869 000600' IF #PARPRES SETIN R1 OR #ECCMEM SETIN R1 THEN
(6) 000600' 032701 002000 BIT #PARPRES,R1
(8) 000604' 001003 BNE 50027$
(6) 000606' 032701 000100 BIT #ECCMEM,R1
(9) 000612' 001500 BEQ 50030$
(6) 000614' 50027$:
870
871 ;+
872 ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER
873 ; THIS WILL TURN CACHE OFF
874 ; -
875
876 000614' IF #CAPRES SETIN R1 THEN
(6) 000614' 032701 000004 BIT #CAPRES,R1
(9) 000620' 001406 BEQ 50031$
877 000622' LET KONTRL := #3
(4) 000622' 012767 000003 000000G MOV #3,KONTRL
878 000630' LET @CCNTRL := KONTRL
(4) 000630' 016777 000000G 000000G MOV KONTRL,@CCNTRL
879 000636' ENDIF
(4) 000636' 50031$:
880
881 ;+
882 ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
883 ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
884 ; OF MEMORY, FIRST KT PRESENT THEN SET
885 ; UP KT REGISTERS.
886 ; -
887
888 000636' IF #KTPRES SETIN R1 THEN
(6) 000636' 032701 000400 BIT #KTPRES,R1
(9) 000642' 001426 BEQ 50032$
889 000644' CALL KTSET
(3) 000644' 004767 000000G JSR PC,KTSET
890
891 ;+
892 ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
893 ; SET UP KIPAR5 TO MAP TO 0
894 ; -
895
896 000650' LET @#KIPDR6 := #0
(4) 000650' 005037 172314 CLR @#KIPDR6
897 000654' LET @#KIPAR5 := #0
(4) 000654' 005037 172352 CLR @#KIPAR5
898
899 ;+
900 ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
901 ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
902 ; SET UP R0 TO USE PAR5
903 ; -

```

```

904
905 000660'          PUSH @#4
906 (2) 000660' 013746 000004          MOV    @#4,-(SP)
907 000664'          LET @#4 := #8$
908 (4) 000664' 012737 000760' 000004          MOV    #8$,@#4
909 000672'          PUSH @#250
910 (2) 000672' 013746 000250          MOV    @#250,-(SP)
911 000676'          LET @#250 := #7$
912 (4) 000676' 012737 000740' 000250          MOV    #7$,@#250
913 000704'          LET @#SR0 := @#SR0 SET.BY #BIT00
914 (6) 000704' 052737 000001 177572          BIS    #BIT00,@#SR0
915 000712'          LET R0 := #SS.120K
916 (4) 000712' 012700 120000          MOV    #SS.120K,R0
917 000716'          ELSE
918 (4) 000716' 000406          BR     50033$
919 (3) 000720'          50032$:
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935 000720'          LET R0 := #0
936 (4) 000720' 005000          CLR    R0
937 000722'          PUSH @#4
938 (2) 000722' 013746 000004          MOV    @#4,-(SP)
939 000726'          LET @#4 := #10$
940 (4) 000726' 012737 001006' 000004          MOV    #10$,@#4
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

940
941
942          ;+
943          ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
944          ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
945          ;-
946 000760'          INLINE <8$:>
(2) 000760'
947 000760'          LET @#SRO := @#SRO CLR.BY #BIT00
(6) 000760' 042737 000001 177572          BIC      #BIT00,@#SRO
948
949 000766'          LET (SP) := #9$
(4) 000766' 012716 000774'          MOV      #9$, (SP)
950 000772'          INLINE <RTI>
(2) 000772' 000002          RTI
951
952 000774'          INLINE <9$:>
(2) 000774'
953 000774'          POP @#250
(2) 000774' 012637 000250          MOV      (SP)+,@#250
954 001000'          POP @#4
(2) 001000' 012637 000004          MOV      (SP)+,@#4
955 001004'          INLINE <BR 11$>
(2) 001004' 000403          BR 11$
956
957          ;+
958          ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
959          ; STACK AND RESTORE BUSERR
960          ;-
961
962 001006'          INLINE <10$:>
(2) 001006'
963 001006'          INLINE <CMP (SP)+,(SP)+>
(2) 001006' 022626          CMP (SP)+,(SP)+
964 001010'          POP @#4
(2) 001010' 012637 000004          MOV      (SP)+,@#4
965
966 001014'          ENDIF
(4) 001014'          50030$:
967
968
969          ;+
970          ; IF SCRIPTING, SET AUTO BIT IN CONFIGURATION WORD0
971          ;-
972
973 001014'          INLINE <11$:>
(2) 001014'
974 001014'          IF @#42 NE #0 THEN
(6) 001014' 005737 000042          TST      @#42
(9) 001020' 001402          BEQ      50034$
975 001022'          LET R1 := R1 SET.BY #AUTO
(6) 001022' 052701 000010          BIS      #AUTO,R1
976 001026'          ENDIF
(4) 001026'          50034$:
977
978          ;+
    
```



```

987          .SBTTL KERNEL - SORT MODULE LIST
988
989
990          ;+
991          ; COUNT THE MODULES
992          ; -
993
994          001036'          LET R0 := #0
995          (4) 001036' 005000
996          001040'          LET R1 := DT.MLST(R2)
997          (4) 001040' 016201 000032
998          001044'          WHILE (R1) NE #ENDLST DO
999          (4) 001044'
1000          (6) 001044' 021127 000000
1001          (9) 001050' 001404
1002          001052'          LET R0 := R0 + #1
1003          (6) 001052' 005200
1004          001054'          LET R1 := R1 + #2
1005          (6) 001054' 062701 000002
1006          001060'          ENDDO
1007          (4) 001060' 000771
1008          (3) 001062'
1009
1010          ;+
1011          ; START THE SORT OPERATION.
1012          ; -
1013
1014          001062'          LET R1 := DT.MLST(R2)
1015          (4) 001062' 016201 000032
1016
1017          ;+
1018          ; WORK THRU THE LIST UNTIL THE END IS REACHED
1019          ; -
1020
1021          001066'          WHILE R0 GT #0 DO
1022          (4) 001066'
1023          (6) 001066' 005700
1024          (9) 001070' 003432
1025          001072'          REPEAT
1026          (3) 001072'
1027
1028          ;+
1029          ; IS THE CURRENT ENTRY A BKMOD MODULE?
1030          ; -
1031
1032          001072'          LET R4 := (R1)
1033          (4) 001072' 011104
1034          001074'          IF #BKMOD SETIN STAT(R4) THEN
1035          (6) 001074' 032764 000020 000026
1036          (9) 001102' 001414
1037
1038          ;+
1039          ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
1040          ; -
1041
1042          001104'          LET R3 := 2(R1)
    
```

50035\$:

CLR R0  
 MOV DT.MLST(R2),R1  
 CMP (R1),#ENDLST  
 BEQ 50036\$  
 INC R0  
 ADD #2,R1  
 BR 50035\$

50036\$:

MOV DT.MLST(R2),R1

50037\$:

TST R0  
 BLE 50040\$

50041\$:

MOV (R1),R4  
 BIT #BKMOD,STAT(R4)  
 BEQ 50042\$

```

(4) 001104' 016103 000002                                MOV      2(R1),R3
1026
1027
1028
1029
1030
1031
(6) 001110' 020327 000000                                IF R3 NE #ENDLST THEN
(9) 001114' 001407                                CMP      R3,#ENDLST
                                                BEQ     50043$
1032
1033
1034
1035
1036
1037 001116'
(6) 001116' 032763 000020 000026                                IF #BKMOD NOTSETIN STAT(R3) THEN
(9) 001124' 001003                                BIT      #BKMOD,STAT(R3)
1038 001126'
(4) 001126' 011161 000002                                LET 2(R1) := (R1)
1039 001132'
(4) 001132' 010311                                MOV      (R1),2(R1)
1040 001134'
(4) 001134'                                LET (R1) := R3
1041 001134'
(4) 001134'                                MOV      R3,(R1)
1042 001134'
(4) 001134'                                ENDIF
                                                50044$:
                                                ENDIF
                                                50043$:
                                                ENDIF
                                                50042$:
1043
1044
1045
1046
1047
1048 001134'
(6) 001134' 062701 000002                                LET R1 := R1 + #2
1049 001140'
(3) 001140' 021127 000000                                UNTIL (R1) EQ #ENDLST
(6) 001144' 001352                                ADD      #2,R1
1050
1051
1052
1053
1054
1055 001146'
(6) 001146' 005300                                CMP      (R1),#ENDLST
1056 001150'
(4) 001150' 016201 000032                                BNE     50041$
1057 001154'
(4) 001154' 000744                                LET R0 := R0 - #1
(3) 001156'                                LET R1 := DT.MLST(R2)
                                                MOV      DT.MLST(R2),R1
                                                BR       50037$
                                                50040$:
1058
1059
1060
1061
1062
1063
1064

```

```

1065 001156'          LET R1 := DT.MLST(R2)
(4) 001156' 016201 000032
1066
1067
1068          ;+
1069          ; WORK THRU THE MODULE LIST UNTIL THE FIRST BACKGROUND MODULE IS FOUND
1070          ; -
1071 001162'          LET DT.BLST(R2) := #0
(4) 001162' 005062 000034
1072 001166'          LET R3 := #-1
(4) 001166' 012703 177777
1073 001172'          WHILE (R1) NE #ENDLST AND R3 NE #0 DO
(4) 001172'
(6) 001172' 021127 000000
(9) 001176' 001420
(6) 001200' 005703
(9) 001202' 001416
1074 001204'          LET R4 := (R1)
(4) 001204' 011104
1075 001206'          IF #BKMOD SETIN STAT(R4) THEN
(6) 001206' 032764 000020 000026
(9) 001214' 001406
1076
1077          ;+
1078          ; A BACKGROUND MODULE HAS BEEN FOUND - SO THIS IS THE START OF THE BACKGROUND LIST
1079          ; -
1080
1081 001216'          LET DT.BLST(R2) := R1
(4) 001216' 010162 000034
1082 001222'          LET R1 := #ENDLST
(4) 001222' 012701 000000
1083 001226'          LET R3 := #0
(4) 001226' 005003
1084 001230'          ELSE
(4) 001230' 000402
(3) 001232'
1085
1086          ;+
1087          ; UPDATE THE POINTER AND CONTINUE TO SEARCH THRU THE MODULE LIST
1088          ; -
1089
1090 001232'          LET R1 := R1 + #2
(6) 001232' 062701 000002
1091 001236'          ENDIF
(4) 001236'
1092 001236'          ENDDO
(4) 001236' 000755
(3) 001240'
1093
1094          ;+
1095          ; NOW RETURN TO THE CALLER
1096          ; -
1097
1098 001240'          CALL RESREG
(3) 001240' 004767 000000G
1099 001244'          ENDRTN

```

50045\$:

MOV DT.MLST(R2),R1

CLR DT.BLST(R2)

MOV #-1,R3

CMP (R1),#ENDLST

BEQ 50046\$

TST R3

BEQ 50046\$

MOV (R1),R4

BIT #BKMOD,STAT(R4)

BEQ 50047\$

50047\$:

MOV R1,DT.BLST(R2)

MOV #ENDLST,R1

CLR R3

BR 50050\$

50050\$:

ADD #2,R1

50046\$:

BR 50045\$

JSR PC,RESREG

```

(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:>
(2) 001246' OV.TQEND:
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	CSRA = 000100	ENBNUL= 000001	LF = 000012	PDP70 = 010000
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	LPSTAT= 000001	PRI0 = 000000
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	MAPSTA= 000200	PRI1 = 000040
ADR = 000006	CTRLD = 000017	ERRTYP= 000106	MED = 076600	PRI4 = 000200
APTFER= 000004	CTRLU = 000025	EVNTBE= 000200	MEMPAS= 040000	PRI5 = 000240
APTPRE= 000200	DCEVNT= 000011	EVNTHD= 000200	MODEXH= 004000	PRI6 = 000300
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MODHOL= 002000	PRI7 = 000340
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MODSEL= 001000	PRO = 000000
ASTAT = 000104	DROPMQ= 100000	EVNTRE= 000201	MSGCKD= 000010	PR4 = 000200
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGCKS= 000011	PR5 = 000240
AUTOST= 020000	DTABLE= 000000	HRDADR= ***** G	MSGDER= 000005	PR6 = 000300
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGDRP= 000017	PR7 = 000340
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGGECH= 177777	PS = 177776
BIT00 = 000001	DT.APK= 000076	HTKT = ***** G	MSGGEP= 000013	PSW = 177776
BIT01 = 000002	DT.BLS= 000034	HTPAER= ***** G	MSGHDR= 000004	RANNUM= 000054
BIT02 = 000004	DT.CFO= 000014	ICONT = 000036	MSGHNG= 000022	RBUFEA= 000130
BIT03 = 000010	DT.CF1= 000016	ICOUNT= 000040	MSGHRD= 000007	RBUFPA= 000126
BIT04 = 000020	DT.ERR= 000020	IDNUM = 000122	MSGMAP= 000021	RBUFSZ= 000132
BIT05 = 000040	DT.ESI= 000044	IE = 000100	MSGNUL= 177775	RBUFVA= 000124
BIT06 = 000100	DT.EVN= 000000	INDPAR= 000040	MSGPOP= 000002	RDSERV= 000101
BIT07 = 000200	DT.EXS= 000060	INHDRP= 040000	MSGPRM= 177776	RDWHMI= 000022
BIT08 = 000400	DT.FCH= 000037	INHPR= 020000	MSGRES= 000001	RELERR= 000020
BIT09 = 001000	DT.FCN= 000036	INHREL= 001000	MSGSFT= 000006	RELMOD= 020000
BIT1 = 000002	DT.HMX= 000104	INHRR= 000400	MSGSKE= 000003	RELTIM= 010000
BIT10 = 002000	DT.KBE= 000024	INIT = 000030	MSGSMB= 000015	RESREG= ***** G
BIT11 = 004000	DT.KBP= 000026	INTR = 000120	MSGSMH= 000014	RES1 = 000056
BIT12 = 010000	DT.KBR= 000022	IOMOD = 100000	MSGSMS= 000016	RES2 = 000060
BIT13 = 020000	DT.KBU= 000030	IOMODP= 102000	MSGSTD= 000000	RICHAR= 031060
BIT14 = 040000	DT.MLS= 000032	IOMODR= 112000	MSGSYS= 000012	RPTDAT= 002000
BIT15 = 100000	DT.MTI= 000110	IOMODX= 110000	MSGVEC= 000020	RSTRT = 000112
BIT2 = 000004	DT.OFF= 000070	JACK = 035060	NBKMOD= 001000	RUBOUT= 000177
BIT3 = 000010	DT.PAS= 000074	KIPAR0= 172340	NCPUOP= 000020	RUNMOD= 100000
BIT4 = 000020	DT.PC = 000002	KIPAR1= 172342	NDAPTY= 000002	RVALU= 001740
BIT5 = 000040	DT.PFL= 000062	KIPAR2= 172344	NULL = 000000	SAM = 075464
BIT6 = 000100	DT.PSW= 000004	KIPAR3= 172346	OV.A = 000662	SAVREG= ***** G
BIT7 = 000200	DT.PTA= 000064	KIPAR4= 172350	OV.CQ = 000122RG	SBADR = 000102
BIT8 = 000400	DT.RCS= 000102	KIPAR5= 172352	OV.CQS= 000050 G	SBKMOD= 000000
BIT9 = 001000	DT.REL= 000040	KIPAR6= 172354	OV.HIC= 000362RG	SBKSEL= 010000
BKDEF = 000002	DT.SCT= 000066	KIPAR7= 172356	OV.HIK= 000120RG	SC.ADR= 000006
BKMOD = 000020	DT.SMX= 000106	KIPDR0= 172300	OV.HIT= 001244RG	SC.ALC= 000014
BKMODE= 040000	DT.SP = 000006	KIPDR1= 172302	OV.KBB= 000000RG	SC.APC= 000016
BKSLSH= 000134	DT.SSI= 000046	KIPDR2= 172304	OV.KBS= 000122 G	SC.CKL= 000002
CAPRES= 000004	DT.STO= 000010	KIPDR3= 172306	OV.REM= 000002	SC.CKP= 000004
CASTAT= 000004	DT.ST1= 000012	KIPDR4= 172310	OV.TQ = 000364RG	SC.CLO= 000000
CCNTRL= ***** G	DT.SWR= 000056	KIPDR5= 172312	OV.TQE 001246R	SC.HLD= 000010
CDERCT= 000146	DT.SYP= 000072	KIPDR6= 172314	OV.TQS= 000066 G	SC.SCA= 000012
CDWDCT= 000144	DT.WBU= 000050	KIPDR7= 172316	OWEN = 024020	SENDLS= 177777
CKTIM = 100000	DT.WHL= 000054	KONTRL= ***** G	PARERR= 000010	SIZPOL 000000RG
CLKPRE= 000001	DT.WLL= 000052	KTERRO= 000040	PARPRE= 002000	SOFCNT= 000042
CONFIG= 000056	DVID1 = 000014	KTPRES= 000400	PARSTA= 000100	SOFPAS= 000046
CPUCPE= ***** G	ECCMEM= 000100	KTSET = ***** G	PASCNT= 000034	SPACE = 000040
CQOVF = 000001	ECCSTA= 000010	KTSTAT= 000020	PDPLSI= 020000	SPOINT= 000032
CR = 000015	ENBEQP= 010000	KTXTND= 040000	PDP60 = 004000	SPVALU= 002200

SR0 = 177572	TQOVF = 000002	WTWHMI= 000222	\$F\$WHI= 000120	\$TSK5 = 050044
SR1 = 177574	UIPAR0= 177640	XFLAG = 000005	\$F\$YES= 000402	\$\$ARGC= 000002
SR2 = 177576	UIPAR1= 177642	XOFF = 000023	\$IFLEV= 177777	\$\$BYTE= 000403
SR3 = 172516	UIPAR2= 177644	XON = 000021	\$ISK0 = 000001	\$\$CASE= 000000
SS.IOP= 177600	UIPAR3= 177646	\$BGNLE= 177777	\$ISK1 = 000001	\$\$DST = 000000
SS.PCL= 172136	UIPAR4= 177650	\$ERFLG= 000400	\$ISK2 = 000001	\$\$ELDC= 000402
SS.PCS= 172100	UIPAR5= 177652	\$F\$AND= 000310	\$LOCTA= 177777	\$\$ERFL= 000000
SS.RFS= 160000	UIPAR6= 177654	\$F\$BAD= 000401	\$LSTIN= 000001	\$\$FLAG= 000001
SS.OKC= 004000	UIPAR7= 177656	\$F\$BLA= 000170	\$LSTTA= 000001	\$\$FROM= 000000
SS.1KP= 000040	UIPDR0= 177600	\$F\$CAS= 000150	\$NESTL= 177777	\$\$LOC = 001214R
SS.120= 120000	UIPDR1= 177602	\$F\$DEC= 000220	\$NSKO = 000300	\$\$LOCN= 000000
SS.28 = 000034	UIPDR2= 177604	\$F\$DO = 000340	\$NSK1 = 000120	\$\$REG = 177777
STAT = 000026	UIPDR3= 177606	\$F\$FAL= 000405	\$NSK2 = 000110	\$\$RETN= 000000
STATBI= 064757	UIPDR4= 177610	\$F\$GDO= 000400	\$NSK3 = 000110	\$\$RTN1= 050000
STAT1 = 000027	UIPDR5= 177612	\$F\$IF = 000110	\$NSK4 = 000110	\$\$RTN2= 050001
SUSPND= 000001	UIPDR6= 177614	\$F\$INC= 000210	\$NSK5 = 000110	\$\$SRC = 000000
SVR0 = 000062	UIPDR7= 177616	\$F\$LDD= 000200	\$SAVLE= 177777	\$\$TGSV= 000000
SVR1 = 000064	WASADR= 000104	\$F\$NAM= 000160	\$SSKO = 050046	\$\$TGS1= 000000
SVR2 = 000066	WBSTAT= 000040	\$F\$NO = 000403	\$TAGLE= 177777	\$\$TGS2= 000000
SVR3 = 000070	WBUFEA= 000136	\$F\$OR = 000320	\$TAGNU= 050051	\$\$TO = 000000
SVR4 = 000072	WBUFPA= 000134	\$F\$RTI= 000350	\$TEMP = 000300	\$\$\$TAG= 050000
SVR5 = 000074	WBUFQ= 000140	\$F\$RTN= 000300	\$TSKO = 050045	. = 001246R
SVR6 = 000076	WBUFsz= 000142	\$F\$SEL= 000140	\$TSK1 = 050046	
SYSCNT= 000052	WDFR = 000116	\$F\$THE= 000330	\$TSK2 = 050050	
SYSERR= 000100	WDT0 = 000114	\$F\$TRU= 000404	\$TSK3 = 050042	
TMPID = 000002	WTINRE= 000352	\$F\$UNT= 000130	\$TSK4 = 050043	

. 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)

.MAIN. MACY11 30A(1052) 20-SEP-78 18:54  
EQUATE.MAC 13-SEP-78 16:13 TABLE OF CONTENTS

SEQ 1172

3 COMMON EQUATE MODULE  
565 COMMON DEFINITIONS AND REFERENCES  
568 000000' .PRINT ;SPMAC: VERSION 1.1  
620 KERNEL - DETERMINE PROCESSOR TYPE  
698 KERNEL - DETERMINE CPU OPTIONS  
738 KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT  
791 KERNEL - DETERMINE SYSTEM SIZE  
883 KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT  
1016 KERNEL - SORT MODULE LIST

```
508 .TITLE SIZPLE SIZE AND POLL SYSTEM - 11/70 SUPPORT ALONG WITH 22-BIT ADDRESSING
509 .IDENT /V0.0/
510
511 ;++
512 ; MODULE NAME:
513 ;     SIZPLE
514 ;
515 ; FUNCTIONAL DESCRIPTION:
516 ;     THIS MODULE WILL SIZE THE SYSTEM FOR CPU TYPE AND PROCESSOR
517 ;     OPTIONS. IT WILL ALSO SIZE MEMORY AND IF "PARITY" OR "ECC" MEMORY
518 ;     WILL WRITE GOOD PARITY. THE "MODQ" LIST WILL BE SORTED TO DEVELOP
519 ;     A BACKGROUND MODULE LIST (QUEUE)
520 ;     NOTE: SUPPORT 11/70 AND 22-BIT ADDRESSING
521 ;
522 ; INPUTS:
523 ;     DTABLE ADDRESS
524 ;
525 ; IMPLICIT INPUTS:
526 ;     CONFIGURATION WORD 0
527 ;     EXERCISER SIZE
528 ;     SYSTEM SIZE
529 ;     PARITY TABLE ADDRESS
530 ;     MODULE LIST
531 ;     BACKGROUND LIST
532 ;
533 ; OUTPUTS:
534 ;     NONE
535 ;
536 ; IMPLICIT OUTPUTS:
537 ;     PARITY TABLE ADDRESS
538 ;     SYSTEM SIZE
539 ;     EXERCISE SIZE
540 ;     CONFIGURATION WORD 0
541 ;
542 ; PATHOLOGICAL CONNECTIONS:
543 ;
544 ; SUBORDINATE ROUTINES CALLED:
545 ;     SAVREG - SAVE REGISTERS
546 ;     RESREG - RESTORE REGISTERS
547 ;     KTSET - SET UP KT REGISTERS
548 ;     HRDADRCHK - ILLEGAL ADDRESS CHECK
549 ;     ICSRSZ - INDIRECT ECC CSR SIZING
550 ;
551 ; FUNCTIONAL SIDE EFFECTS:
552 ;     NONE
553 ;
554 ; CALLING SEQUENCE:
555 ;     CALL SIZPOL IN <A>
556 ;             A - DTABLE ADDRESS
557 ;
558 ; VERSION:
559 ;     0.0
560 ;
561 ;     EDIT                DATE                BY                REASON
562 ;--
```

```

564
565          .SBTTL COMMON DEFINITIONS AND REFERENCES
566
567          .MCALL STRUCT
568 000000'   STRUCT
569 (1) 000000'   .PRINT ;SPMAC: VERSION 1.1
570          000001   $LSTIN = 1
571          000001   $LSTTAG = 1
572
573          ;*****
574          ; GLOBAL REFERENCES
575          ;
576          .GLOBL HTPAER
577          .GLOBL HTKT
578          .GLOBL SAVREG          ;SARE REGISTERS
579          .GLOBL RESREG          ;RESTORE REGISTERS
580          .GLOBL KTSET          ;SET UP KT
581          .GLOBL HRDADRCHK       ;ILLEGAL ADDRESS CHECK
582          .GLOBL ICSRSZ         ;INDIRECT ECC CSR SIZING
583          .GLOBL CPUCPE         ;11/70 CPU ERROR REGISTER
584          .GLOBL CCNTRL         ;CACHE CONTROL REGISTER
585          .GLOBL KONTRL         ;TEMPORARY STORAGE FOR CACHE CONTROL REGISTER
586
587          ;*****
588          ; LOCAL EQUATES
589          ;
590          172100   SS.PCSR = 172100          ;START OF PARITY CSRS
591          172136   SS.PCLS = 172136        ;LAST PARITY CSR
592          004000   SS.OKC = 4000          ;ONE K CHUNK
593          000034   SS.28 = ^D28          ;28 DECIMAL
594          000040   SS.1KP = 40           ;1K IN PAR FORMAT
595          120000   SS.120K = 120000       ;FIRST ADDRESS IN PAR5 MAP
596          177600   SS.IOP = 177600       ;I/O PAGE
597          160000   SS.RFS = 160000       ;RESERVED FOR DIAGNOSTIC USE IN I/O PAGE
598
599          ;*****
600          ; REFERENCED BY OTHER MODULES
601          ;
602          .GLOBL SIZPOL          ;MODULE ENTRY POINT
603
604
605
606
607          ;+++++
608          ; OVERLAY REGION GLOBALS
609          ;+++++
610          .GLOBL OV.KBBUF
611          .GLOBL OV.HIKB
612          .GLOBL OV.KBSIZ
613          .GLOBL OV.CQ
614          .GLOBL OV.HICQ
615          .GLOBL OV.CQSIZ
616          .GLOBL OV.TQ
617          .GLOBL OV.HITQ
618          .GLOBL OV.TQSIZ
    
```

```

620          .SBTTL KERNEL - DETERMINE PROCESSOR TYPE
621
622
623
624 000000'          ROUTINE SIZPOL <DTABLE>
(2) 000000'
625
626          ;+
627          ; STUFF PARITY AND KT VECTORS
628          ; SAVE REGISTERS, GET DTABLE ADDRESS FROM R5 STACK AND CLEAR
629          ; R1
630          ; -
631
632
633 000000'          LET @#114 := #HTPAER
(4) 000000' 012737 000000G 000114          MOV          #HTPAER,@#114
634 000006'          LET @#116 := #PRI7
(4) 000006' 012737 000340 000116          MOV          #PRI7,@#116
635
636 000014'          LET @#250 := #HTKT
(4) 000014' 012737 000000G 000250          MOV          #HTKT,@#250
637 000022'          LET @#252 := #PRI7
(4) 000022' 012737 000340 000252          MOV          #PRI7,@#252
638
639 000030'          CALL SAVREG
(3) 000030' 004767 000000G          JSR          PC,SAVREG
640 000034'          LET R2 := DTABLE(R5)
(4) 000034' 016502 000000          MOV          DTABLE(R5),R2
641 000040'          LET R1 := #0
(4) 000040' 005001          CLR          R1
642
643          ;+
644          ; DETERMINE PROCESSOR TYPE; 11/70, 11/60 OR LSI11
645          ; -
646
647 000042'          CALL HRDADRCHK IN <CPUCPE>
(3) 000042' 010546          MOV          R5,-(SP)
(4) 000044' 016745 000000G          MOV          CPUCPE,-(R5)
(3) 000050' 004767 000000G          JSR          PC,HRDADRCHK
(3) 000054' 012605          MOV          (SP)+,R5
648
649 000056'          IF.NO.ERROR THEN
(6) 000056' 103422          BCS          50002$
650
651          ;+
652          ; SET UP ILLEGAL INSTRUCTION TRAP TO FIELD TRAP IF NOT
653          ; 11/60
654          ; -
655          PUSH @#10
(2) 000060' 013746 000010          MOV          @#10,-(SP)
656 000064'          LET @#10 := #1$
(4) 000064' 012737 000104' 000010          MOV          #1$,@#10
657 000072'          INLINE <MED>
(2) 000072' 076600          MED
658 000074'          INLINE <RDSERV>
(2) 000074' 000101          RDSERV
    
```

```

659 000076'          LET R1 := R1 SET.BY #PDP60
(6) 000076' 052701 004000          BIS      #PDP60,R1
660 000102'          INLINE <BR 3$>
(2) 000102' 000405          BR 3$
661
662 000104'          INLINE <1$:>
(2) 000104'          1$:
663
664          ;+
665          ; SET RETURN TO 2$ TO RESTORE PSW
666          ; -
667
668 000104'          LET (SP) := #2$
(4) 000104' 012716 000112'          MOV      #2$, (SP)
669 000110'          INLINE <RTI>
(2) 000110' 000002          RTI
670
671          ;+
672          ; MUST BE AN 11/70, BECAUSE OF TRAP
673          ; -
674
675 000112'          INLINE <2$:>
(2) 000112'          2$:
676 000112'          LET R1 := R1 SET.BY #PDP70
(6) 000112' 052701 010000          BIS      #PDP70,R1
677
678          ;+
679          ; RESTORE RESERVED INSTRUCTION TRAP
680          ; -
681
682 000116'          INLINE <3$:>
(2) 000116'          3$:
683 000116'          POP @#10
(2) 000116' 012637 000010          MOV      (SP)+, @#10
684 000122'          ELSE
(4) 000122' 000411          BR 50003$
(3) 000124'          50002$:
685
686          ;+
687          ; CHECK TO SEE IF LSI11
688          ; -
689
690 000124'          CALL HRDADRCHK IN <#PSW>
(3) 000124' 010546          MOV      R5, -(SP)
(4) 000126' 012745 177776          MOV      #PSW, -(R5)
(3) 000132' 004767 000000G        JSR      PC, HRDADRCHK
(3) 000136' 012605          MOV      (SP)+, R5
691 000140'          IF.ERROR THEN
(6) 000140' 103002          BCC      50004$
692 000142'          LET R1 := R1 SET.BY #PDPLSI
(6) 000142' 052701 020000          BIS      #PDPLSI,R1
693
694          ENDIF
(4) 000146'          50004$:
695 000146'          ENDIF
(4) 000146'          50003$:

```

SIZPLE SIZE AND POLL SYSTEM - 11/70 SUPPORT ALONG WITH 22-BIT ADDRESSING  
SIZPLE.MAC 08-SEP-78 09:33 KERNEL - DETERMINE PROCESSOR TYPE

MACY11 30A(1052) 20-SEP-78 18:54 PAGE 21-2  
SEQ 1177

696



```

698          .SBTTL KERNEL - DETERMINE CPU OPTIONS
699
700          ;+
701          ; DETERMINE IF "KT" IS PRESENT AND IF IT IS CHECK TO
702          ; SEE IF EXTENDED KT" IS PRESENT
703          ; -
704
705          CALL HRDADRCHK IN <#SR0>
706          (3) 000146' 010546
707          (4) 000150' 012745 177572
708          (3) 000154' 004767 000000G
709          (3) 000160' 012605
710
711          IF.NO.ERROR THEN
712
713          LET R1 := R1 SET.BY #KTPRES
714          (6) 000164' 052701 000400
715          CALL HRDADRCHK IN <#SR3>
716          (3) 000170' 010546
717          (4) 000172' 012745 172516
718          (3) 000176' 004767 000000G
719          (3) 000202' 012605
720          (6) 000204' 103424
721          (6) 000204' 103413
722          (6) 000206' 052701 040000
723          (6) 000206' 052701 040000
724
725          ;+
726          ; DETERMINE IF 22-ADDRESSING BY LOADING PAR5 WITH 16 BIT
727          ; WORD AND READ PAR5 TO SEE IF WORD READ IS SAME AS WRITTEN
728          ; -
729
730          LET @#KIPAR5 := #100000
731          (4) 000212' 012737 100000 172352
732          (4) 000220' 023727 172352 100000
733          (6) 000220' 023727 172352 100000
734          (9) 000226' 001002
735          (6) 000230' 052701 001000
736          (6) 000230' 052701 001000
737          (4) 000234'
738          (4) 000234'
739          (4) 000234'
740          (4) 000234'
741          (4) 000234'
742          (4) 000234'
743          (4) 000234'
744
745          ;+
746          ; NOW DETERMINE IF CACHE MEMORY IS PRESENT BY CHECKING
747          ; CACHE CONTROL REGISTER
748          ; -
749
750          CALL HRDADRCHK IN <CCNTRL>
751          (3) 000234' 010546
752          (4) 000236' 016745 000000G
  
```

MOV R5,-(SP)  
 MOV #SR0,-(R5)  
 JSR PC,HRDADRCHK  
 MOV (SP)+,R5

BCS 50005\$

BIS #KTPRES,R1

MOV R5,-(SP)  
 MOV #SR3,-(R5)  
 JSR PC,HRDADRCHK  
 MOV (SP)+,R5

BCS 50006\$

BIS #KTXTND,R1

MOV #100000,@#KIPAR5

CMP @#KIPAR5,#100000  
 BNE 50007\$

BIS #ADDR22,R1

50007\$:

50006\$:

50005\$:

(3) 000242' 004767 000000G  
(3) 000246' 012605  
733 000250'  
(6) 000250' 103402  
734 000252'  
(6) 000252' 052701 000004  
735 000256'  
(4) 000256'  
736

IF.NO.ERROR THEN  
LET R1 := R1 SET.BY #CAPRES  
ENDIF

JSR PC,HRDADRCHK  
MOV (SP)+,R5  
BCS 50010\$  
BIS #CAPRES,R1

50010\$:

```

738 .SBTTL KERNEL - DETERMINE IF PARITY OR ECC MEMORY PRESENT
739
740 ;+
741 ; GET STARTING ADDRESS OF PARITY TABLE, THEN DETERMINE IF
742 ; PARITY CSR REGISTERS ARE AVAILABLE IF SO PLACE CSR ADDRESS
743 ; IN PARITY TABLE.
744 ; -
745
746 000256' LET R0 := DT.PTA(R2)
(4) 000256' 016200 000064 MOV DT.PTA(R2),R0
747
748 000262' LET R3 := #SS.PCSR
(4) 000262' 012703 172100 MOV #SS.PCSR,R3
749 000266' WHILE R3 LE #SS.PCLS DO
(4) 000266' 50011$:
(6) 000266' 020327 172136 CMP R3,#SS.PCLS
(9) 000272' 003035 BGT 50012$
750 000274' CALL HRDADRCHK IN <R3>
(3) 000274' 010546 MOV R5,-(SP)
(4) 000276' 010345 MOV R3,-(R5)
(3) 000300' 004767 000000G JSR PC,HRDADRCHK
(3) 000304' 012605 MOV (SP)+,R5
751 000306' IF.NO.ERROR THEN
(6) 000306' 103414 BCS 50013$
752 000310' LET (R0)+ := R3
(4) 000310' 010320 MOV R3,(R0)+
753
754 ;+
755 ; DETERMINE IF ECC MEMORY, IF IT IS SET ECC MEMORY BIT IN
756 ; CONFIGURATION WORD 0 ELSE SET PARITY PRESENT BIT IN
757 ; CONFIGURATION WORD 0. CLEAR THE PARITY CSR'S
758 ; -
759 000312' LET (R3) := (R3) SET.BY #BIT01
(6) 000312' 052713 000002 BIS #BIT01,(R3)
760 000316' IF #BIT01 SETIN (R3) THEN
(6) 000316' 032713 000002 BIT #BIT01,(R3)
(9) 000322' 001403 BEQ 50014$
761 000324' LET R1 := R1 SET.BY #ECCMEM
(6) 000324' 052701 000100 BIS #ECCMEM,R1
762 000330' ELSE
(4) 000330' 000402 BR 50015$
(3) 000332' 50014$:
763 000332' LET R1 := R1 SET.BY #PARPRES
(6) 000332' 052701 002000 BIS #PARPRES,R1
764 000336' ENDIF
(4) 000336' 50015$:
765 000336' LET (R3) := #0
(4) 000336' 005013 CLR (R3)
766 000340' ENDIF
(4) 000340' 50013$:
767
768 ;+
769 ; IF 11/70 PROCESSOR THEN SET PARITY PRESENT IN CONFIGURATION
770 ; WORD 0 AND POINT TO PARITY CSR + 4, ELSE POINT TO
771 ; PARITY CSR + 2
772 ; -
    
```

```

773 000340'          IF #PDP70 SETIN R1 THEN
(6) 000340' 032701 010000
(9) 000344' 001405
774 000346'          LET R1 := R1 SET.BY #PARPRES
(6) 000346' 052701 002000
775 000352'          LET R3 := R3 + #4
(6) 000352' 062703 000004
776 000356'          ELSE
(4) 000356' 000402
(3) 000360'          50016$:
777 000360'          LET R3 := R3 + #2
(6) 000360' 062703 000002
778 000364'          ENDIF
(4) 000364'          50017$:
779 000364'          ENDDO
(4) 000364' 000740
(3) 000366'          50012$:
780
781
782 ;+
783 ; DETERMINE IF WE SHOULD CHECK FOR INDIRECT CSRS ON THE 11/70
784 ; -
785 000366'          IF #PDP70 SETIN R1 AND @DT.PTA(R2) EQ #0 THEN
(6) 000366' 032701 010000
(9) 000372' 001410
(6) 000374' 005772 000064
(9) 000400' 001005
786 000402'          CALL ICSRSZ IN <R2>
(3) 000402' 010546
(4) 000404' 010245
(3) 000406' 004767 000000G
(3) 000412' 012605
787 000414'          ENDIF
(4) 000414'          50020$:
788
789
    
```

```

BIT #PDP70,R1
BEQ 50016$
BIS #PARPRES,R1
ADD #4,R3
BR 50017$
ADD #2,R3
BR 50011$
BIT #PDP70,R1
BEQ 50020$
TST @DT.PTA(R2)
BNE 50020$
MOV R5,-(SP)
MOV R2,-(R5)
JSR PC,ICRSZ
MOV (SP)+,R5
    
```

```

791          .SBTTL KERNEL - DETERMINE SYSTEM SIZE
792
793          ;+
794          ; NOW TAKE THE CONTENTS OF LOC. 0 (EXERCISER SIZE)
795          ; AND LOAD IT INTO EXERCISER SIZE WORD
796          ; -
797
798          000414'          LET DT.ESIZ(R2) := 0
(4) 000414' 016762 000000 000044          MOV          0,DT.ESIZ(R2)
799
800          ;+
801          ; IF NO "KT" DETERMINE SYSTEM SIZE, WHICH CAN NOT BE GREATER
802          ; THAN 28K.
803          ; -
804
805          000422'          IF #KTPRES NOTSETIN R1 THEN
(6) 000422' 032701 000400          BIT          #KTPRES,R1
(9) 000426' 001053          BNE          50021$
806
807          ;+
808          ; SET UP COUNTER OF 1K CHUNKS AND 1K BLOCK POINTER
809          ; -
810
811          000430'          LET R0 := #0
(4) 000430' 005000          CLR          R0
812          000432'          LET R3 := #0
(4) 000432' 005003          CLR          R3
813
814          ;+
815          ; REPEAT UNTIL NON-EXISTENT MEMORY OR 28K
816          ; -
817
818          REPEAT
(3) 000434'          50022$:
819          000434'          LET R0 := R0 + #1
(6) 000434' 005200          INC          R0
820          000436'          LET R3 := R3 + #SS.0KC
(6) 000436' 062703 004000          ADD          #SS.0KC,R3
821          000442'          CALL HRDADRCHK IN <R3>
(3) 000442' 010546          MOV          R5,-($P)
(4) 000444' 010345          MOV          R3,-(R5)
(3) 000446' 004767 000000G          JSR          PC,HRDADRCHK
(3) 000452' 012605          MOV          (SP)+,R5
822          000454'          IF.ERROR THEN
(6) 000454' 103001          BCC          50023$
823          000456'          INLINE <BR 4$>
(2) 000456' 000403          BR          4$
824          000460'          ENDIF
(4) 000460'          50023$:
825          000460'          UNTIL R0 EQ #SS.28
(3) 000460' 020027 000034          CMP          R0,#SS.28
(6) 000464' 001363          BNE          50022$
826
827          ;+
828          ; DETERMINE IF SYSTEM HAS 30K BY LOOKING AT FIRST ADDRESS IN I/O PAGE
829          ; -
    
```

```

830
831 000466'          INLINE <4$:>
(2) 000466'
832 000466'          IF R0 EQ #SS.28 THEN
(6) 000466' 020027 000034          CMP      R0,#SS.28
(9) 000472' 001011          BNE      50024$
833 000474'          CALL HRDADRCHK IN <#SS.RFS>
(3) 000474' 010546          MOV      R5,-(SP)
(4) 000476' 012745 160000          MOV      #SS.RFS,-(R5)
(3) 000502' 004767 000000G        JSR      PC,HRDADRCHK
(3) 000506' 012605          MOV      (SP)+,R5
834 000510'          IF.NO.ERROR THEN
(6) 000510' 103402          BCS      50025$
835 000512'          LET R3 := R3 + #10000
(6) 000512' 062703 010000          ADD      #10000,R3
836 000516'          ENDIF
(4) 000516'          50025$:
837 000516'          ENDIF          50024$:
(4) 000516'
838
839 ;+
840 ; NOW GET MEMORY SIZE IN "PAR" FORMAT BY SHIFTING R3, 6 TIMES
841 ; TO THE RIGHT, ONCE THIS IS DONE LOAD INTO SYSTEM
842 ; SIZE WORD.
843 ;-
844
845 000516'          LET CARRY := #0
(4) 000516' 000241          CLC
846 000520'          LET DT.SSIZ(R2) := R3 ROTATE -6
(4) 000520' 010362 000046          MOV      R3,DT.SSIZ(R2)
(7) 000524' 006062 000046          ROR      DT.SSIZ(R2)
(7) 000530' 006062 000046          ROR      DT.SSIZ(R2)
(7) 000534' 006062 000046          ROR      DT.SSIZ(R2)
(7) 000540' 006062 000046          ROR      DT.SSIZ(R2)
(7) 000544' 006062 000046          ROR      DT.SSIZ(R2)
(7) 000550' 006062 000046          ROR      DT.SSIZ(R2)
847 000554'          ELSE
(4) 000554' 000443          BR       50026$
(3) 000556'          50021$:
848
849 ;+
850 ; KT IS PRESENT, FIRST GO MAP APRS
851 ;-
852 000556'          CALL KTSET IN <R2>
(3) 000556' 010546          MOV      R5,-(SP)
(4) 000560' 010245          MOV      R2,-(R5)
(3) 000562' 004767 000000G        JSR      PC,KTSET
(3) 000566' 012605          MOV      (SP)+,R5
853
854 ;+
855 ; WILL USE KIPAR5 TO SIZE MEMORY. TURN ON KT AND THEN
856 ; SIZE UNTIL NON-EXISTENT MEMORY IS FOUND OR THE
857 ; I/O PAGE IS FOUND. SIZING WILL BE DONE IN
858 ; 1K STEPS
859
860 000570'          LET @#KIPAR5 := #SS.1KP
    
```



```

883          .SBTTL KERNEL - WRITE GOOD PARITY IF ECC OR PARITY PRESENT
884
885          ;+
886          ; DETERMINE IF PARITY OR ECC MEMORY IS PRESENT.
887          ; -
888
889          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$:
890
891          ;+
892          ; IF CACHE PRESENT, SET BIT 0,1 IN CACHE CONTROL REGISTER.
893          ; THIS WILL TURN CACHE OFF
894          ; -
895
896          000700'          IF #CAPRES SETIN R1 THEN
(6)          000700' 032701 000004          BIT          #CAPRES,R1
(9)          000704' 001406          BEQ          50034$
897          000706'          LET KONTRL := #3
(4)          000706' 012767 000003 000000G          MOV          #3,KONTRL
898          000714'          LET @CCNTRL := KONTRL
(4)          000714' 016777 000000G 000000G          MOV          KONTRL,@CCNTRL
899          000722'          ENDIF
(4)          000722'          50034$:
900
901          ;+
902          ; SINCE PARITY OR ECC IS PRESENT AND TURNED OFF
903          ; WILL NOW SET UP TO WRITE GOOD PARITY INTO ALL
904          ; OF MEMORY, FIRST DETERMINE IF KT IS PRESENT THEN SET
905          ; UP KT REGISTERS.
906          ; -
907
908          000722'          IF #KTPRES SETIN R1 THEN
(6)          000722' 032701 000400          BIT          #KTPRES,R1
(9)          000726' 001434          BEQ          50035$
909          000730'          CALL KTSET
(3)          000730' 004767 000000G          JSR          PC,KTSET
910
911          ;+
912          ; SET UP KIPDR6 TO TRAP WHEN ENTERING PAGE. AND
913          ; SET UP KIPAR5 TO MAP TO 0
914          ; -
915
916          000734'          LET @#KIPDR6 := #0
(4)          000734' 005037 172314          CLR          @#KIPDR6
917          000740'          LET @#KIPAR5 := #0
(4)          000740' 005037 172352          CLR          @#KIPAR5
918
919          ;+
920          ; SAVE KT ERROR VECTOR AND SET UP KT ERROR VECTOR TO
921          ; GO TO 7$. THEN TURN ON KT AND SET UP BUSERR VECTOR AT 8$
922          ; SET UP R0 TO USE PAR5
923          ; -

```



```
924
925 000744' PUSH @#4
(2) 000744' 013746 000004 MOV @#4,-(SP)
926 000750' LET @#4 := #8$
(4) 000750' 012737 001060' 000004 MOV #8$,@#4
927 000756' PUSH @#250
(2) 000756' 013746 000250 MOV @#250,-(SP)
928 000762' LET @#250 := #7$
(4) 000762' 012737 001040' 000250 MOV #7$,@#250
929 000770' LET @#SR0 := @#SR0 SET.BY #BIT00
(6) 000770' 052737 000001 177572 BIS #BIT00,@#SR0
930 000776' IF #ADDR22 SETIN R1 THEN
(6) 000776' 032701 001000 BIT #ADDR22,R1
(9) 001002' 001403 BEQ 50036$
931
932
933
934
935 001004' LET @#SR3 := @#SR3 SET.BY #BIT04
(6) 001004' 052737 000020 172516 BIS #BIT04,@#SR3
936 001012' ENDIF
(4) 001012' 50036$:
937 001012' LET R0 := #SS.120K'
(4) 001012' 012700 120000 MOV #SS.120K,R0
938 001016' ELSE
(4) 001016' 000406 BR 50037$
(3) 001020' 50035$:
939
940 ;+
941 ; IF KT NOT PRESENT THEN SET UP POINTER AND SAVE BUSERR VECTOR
942 ;-
943
944 001020' LET R0 := #0
(4) 001020' 005000 CLR R0
945 001022' PUSH @#4
(2) 001022' 013746 000004 MOV @#4,-(SP)
946 001026' LET @#4 := #10$
(4) 001026' 012737 001106' 000004 MOV #10$,@#4
947
948 001034' ENDIF
(4) 001034' 50037$:
949
950 ;+
951 ; NOW WRITE GOOD PARITY
952 ;-
953 001034' INLINE<6$:>
(2) 001034' 6$:
954 001034' LET (R0)+ := (R0)
(4) 001034' 011020 MOV (R0),(R0)+
955 001036' INLINE <BR 6$>
(2) 001036' 000776 BR 6$
956
957 ;+
958 ; COME HERE BY WAY OF KT TRAP
959 ; UPDATE PARS TO NEXT 4K BLOCK AND RESET R0 TO START
960 ; OF BLOCK. LOAD RETURN PC WITH 6$ AND RTI
```

```

961          ; -
962
963          INLINE <7$:>
(2) 001040'
964 001040'          LET @#KIPAR5 := @#KIPAR5 + #200          7$:
(6) 001040' 062737 000200 172352          ADD          #200,@#KIPAR5
965 001046'          LET R0 := #SS.120K
(4) 001046' 012700 120000          MOV          #SS.120K,R0
966 001052'          LET (SP) := #6$
(4) 001052' 012716 001034'          MOV          #6$,(SP)
967 001056'          INLINE <RTI>
(2) 001056' 000002          RTI
968
969
970          ; +
971          ; COME HERE BY WAY OF ILLEGAL ADDRESS TRAP
972          ; TOP OF MEMORY WAS FOUND, TURN OFF MEMORY MANAGEMENT,
973          ; REMOVE TRAP PC AND PSW FROM STACK AND RESTORE BUSERR AND MEMGMT VECTORS
974          ; -
975 001060'          INLINE <8$:>
(2) 001060'
976 001060'          LET @#SRO := @#SRO CLR.BY #BIT00          8$:
(6) 001060' 042737 000001 177572          BIC          #BIT00,@#SRO
977
978 001066'          LET (SP) := #9$
(4) 001066' 012716 001074'          MOV          #9$,(SP)
979 001072'          INLINE <RTI>
(2) 001072' 000002          RTI
980
981 001074'          INLINE <9$:>
(2) 001074'
982 001074'          POP @#250          9$:
(2) 001074' 012637 000250          MOV          (SP)+,@#250
983 001100'          POP @#4
(2) 001100' 012637 000004          MOV          (SP)+,@#4
984 001104'          INLINE <BR 11$>
(2) 001104' 000403          BR 11$
985
986          ; +
987          ; WILL TRAP HERE IF NO MEMORY MANAGEMENT, CLEAN UP
988          ; STACK AND RESTORE BUSERR
989          ; -
990
991 001106'          INLINE <10$:>
(2) 001106'
992 001106'          INLINE <CMP (SP)+,(SP)+>          10$:
(2) 001106' 022626          CMP (SP)+,(SP)+
993 001110'          POP @#4
(2) 001110' 012637 000004          MOV          (SP)+,@#4
994
995 001114'          ENDF
(4) 001114'          50033$:
996
997
998          ; +
999          ; IF AUTOMATIC MODE(ACT,SLIDE,XXDP,ETC.), SET AUTO BIT IN CONFIGURATION WORD0
    
```

```

1000                                   ;-
1001
1002 001114'                    INLINE <11$:>
(2) 001114'
1003 001114'                    IF @#42 NE #0 THEN                                   11$:
(6) 001114' 005737 000042                                                           TST       @#42
(9) 001120' 001402                                                                 BEQ       50040$
1004 001122'                                    LET R1 := R1 SET.BY #AUTO                                   BIS       #AUTO,R1
(6) 001122' 052701 000010
1005 001126'                    ENDIF
(4) 001126'                                                                           50040$:
1006
1007                                   ;+
1008                                   ; LOAD UP CONFIGURATION WORD 0
1009                                   ; CLEAN UP R1 BEFORE LOADING DT.CFO
1010                                   ;-
1011
1012 001126'                    LET R1 := R1 CLR.BY #OWEN                                   BIC       #OWEN,R1
(6) 001126' 042701 024020
1013 001132'                    LET DT.CFO(R2) := DT.CFO(R2) SET.BY R1                                   BIS       R1,DT.CFO(R2)
(6) 001132' 050162 000014
1014

```

```

1016 .SBTTL KERNEL - SORT MODULE LIST
1017
1018
1019 ;+
1020 ; COUNT THE MODULES
1021 ; -
1022
1023 001136' LET R0 := #0
(4) 001136' 005000 CLR R0
1024 001140' LET R1 := DT.MLST(R2)
(4) 001140' 016201 000032 MOV DT.MLST(R2),R1
1025 001144' WHILE (R1) NE #ENDLST DO
(4) 001144' 50041$:
(6) 001144' 021127 000000 CMP (R1),#ENDLST
(9) 001150' 001404 BEQ 50042$
1026 001152' LET R0 := R0 + #1
(6) 001152' 005200 INC R0
1027 001154' LET R1 := R1 + #2
(6) 001154' 062701 000002 ADD #2,R1
1028 001160' ENDDO
(4) 001160' 000771 BR 50041$
(3) 001162' 50042$:
1029
1030 ;+
1031 ; START THE SORT OPERATION, AND SORT "BKMOD" TO BOTTOM OF MODQ LIST,
1032 ; DEVELOPING A "BKMOD QUEUE"
1033 ; -
1034
1035 001162' LET R1 := DT.MLST(R2)
(4) 001162' 016201 000032 MOV DT.MLST(R2),R1
1036
1037 ;+
1038 ; WORK THRU THE LIST UNTIL THE END IS REACHED
1039 ; -
1040
1041 001166' WHILE R0 GT #0 DO
(4) 001166' 50043$:
(6) 001166' 005700 TST R0
(9) 001170' 003432 BLE 50044$
1042 001172' REPEAT
(3) 001172' 50045$:
1043
1044 ;+
1045 ; IS THE CURRENT ENTRY A BKMOD MODULE?
1046 ; -
1047
1048 001172' LET R4 := (R1)
(4) 001172' 011104 MOV (R1),R4
1049 001174' IF #BKMOD SETIN STAT(R4) THEN
(6) 001174' 032764 000020 000026 BIT #BKMOD,STAT(R4)
(9) 001202' 001414 BEQ 50046$
1050
1051 ;+
1052 ; THE CURRENT ENTRY IS A BKMOD, WHAT ABOUT THE NEXT ENTRY?
1053 ; -
1054

```

```

1055 001204'          LET R3 := 2(R1)
(4) 001204' 016103 000002          MOV      2(R1),R3
1056
1057          ;+
1058          ; MAKE SURE ITS NOT THE END OF THE LIST
1059          ; -
1060
1061 001210'          IF R3 NE #ENDLST THEN
(6) 001210' 020327 000000          CMP      R3,#ENDLST
(9) 001214' 001407          BEQ      50047$
1062
1063          ;+
1064          ; IF ITS NOT A BKMOD, SWAP THEM
1065          ; -
1066
1067 001216'          IF #BKMOD NOTSETIN STAT(R3) THEN
(6) 001216' 032763 000020 000026          BIT      #BKMOD,STAT(R3)
(9) 001224' 001003          BNE      50050$
1068 001226'          LET 2(R1) := (R1)
(4) 001226' 011161 000002          MOV      (R1),2(R1)
1069 001232'          LET (R1) := R3
(4) 001232' 010311          MOV      R3,(R1)
1070 001234'          ENDIF
(4) 001234'          50050$:
1071 001234'          ENDIF
(4) 001234'          50047$:
1072 001234'          ENDIF
(4) 001234'          50046$:
1073
1074          ;+
1075          ; UPDATE THE POINTER AND CONTINUE SEARCHING THRU THE LIST
1076          ; -
1077
1078 001234'          LET R1 := R1 + #2
(6) 001234' 062701 000002          ADD      #2,R1
1079 001240'          UNTIL (R1) EQ #ENDLST
(3) 001240' 021127 000000          CMP      (R1),#ENDLST
(6) 001244' 001352          BNE      50045$
1080
1081          ;+
1082          ; THE END OF THE LIST HAS BEEN REACHED, SO UPDATE THE MAIN POINTER & CONTINUE
1083          ; -
1084
1085 001246'          LET R0 := R0 - #1
(6) 001246' 005300          DEC      R0
1086 001250'          LET R1 := DT.MLST(R2)
(4) 001250' 016201 000032          MOV      DT.MLST(R2),R1
1087          ENDDO
(4) 001254' 000744          BR      50043$
(3) 001256'          50044$:
1088
1089          ;+
1090          ; NOW THAT THE BACKGROUND MODULES HAVE BEEN GROUPED TOGETHER AT THE END
1091          ; OF THE MODULE LIST, PUT THE ADDRESS OF THIS BACKGROUND LIST (IF THERE IS
1092          ; A BACKGROUND LIST) INTO THE DATA TABLE
1093          ; -

```



```
1129 001344'          ENDRTN
(3) 001344'
(3) 001344'          50000$:
(2) 001344' 000207          50001$:
                               RTS      PC
1130
1131          ;+
1132          ; LABEL TYPE QUEUE HIGH LIMIT
1133          ; -
1134
1135 001346'          INLINE <OV.TQEND:>
(2) 001346'          OV.TQEND:
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	CSRA = 000100	ENBNUL= 000001	KTXTND= 040000	PDP60 = 004000
ACTBIT= 004000	CSRC = 000102	ENDLST= 000000	LF = 000012	PDP70 = 010000
ADDR22= 001000	CTRLC = 000003	EOPBIT= 000001	LPSTAT= 000001	PRI0 = 000000
ADR = 000006	CTRLD = 000017	ERRTYP= 000106	MAPSTA= 000200	PRI1 = 000040
APTFER= 000004	CTRLU = 000025	EVNTBE= 000200	MED = 076600	PRI4 = 000200
APTPRE= 000200	DCEVNT= 000011	EVNTHD= 000200	MEMPAS= 040000	PRI5 = 000240
ASB = 000106	DEFRTN= 000400	EVNTKT= 000203	MODEXH= 004000	PRI6 = 000300
ASSEMB= 000010	DIAGMC= 000000	EVNTPE= 000202	MDDHOL= 002000	PRI7 = 000340
ASTAT = 000104	DROPMD= 100000	EVNTRE= 000201	MODSEL= 001000	PRO = 000000
AUTO = 000010	DSEVNT= 000014	FATERR= 100000	MSGCKD= 000010	PR4 = 000200
AUTOST= 020000	DTABLE= 000000	HRDADR= ***** G	MSGCKS= 000011	PR5 = 000240
AWAS = 000110	DT.ADD= 000042	HRDCNT= 000044	MSGDER= 000005	PR6 = 000300
BIT0 = 000001	DT.AP = 000100	HRDPAS= 000050	MSGDRP= 000017	PR7 = 000340
BIT00 = 000001	DT.APK= 000076	HTKT = ***** G	MSGECH= 177777	PS = 177776
BIT01 = 000002	DT.BLS= 000034	HTPAER= ***** G	MSGEOP= 000013	PSW = 177776
BIT02 = 000004	DT.CFO= 000014	ICDNT = 000036	MSGHDR= 000004	RANNUM= 000054
BIT03 = 000010	DT.CF1= 000016	ICQUNT= 000040	MSGHNG= 000022	RBUFEA= 000130
BIT04 = 000020	DT.ERR= 000020	ICRSRZ= ***** G	MSGHRD= 000007	RBUFPA= 000126
BIT05 = 000040	DT.ESI= 000044	IDNUM = 000122	MSGMAP= 000021	RBUFSZ= 000132
BIT06 = 000100	DT.EVN= 000000	IE = 000100	MSGNUL= 177775	RBUFVA= 000124
BIT07 = 000200	DT.EXS= 000060	INDPAR= 000040	MSGPOP= 000002	RDSERV= 000101
BIT08 = 000400	DT.FCH= 000037	INHDRP= 040000	MSGPRM= 177776	RDWHMI= 000022
BIT09 = 001000	DT.FCN= 000036	INHEPR= 020000	MSGRES= 000001	RELERR= 000020
BIT1 = 000002	DT.HMX= 000104	INHREL= 001000	MSGSFT= 000006	RELMOD= 020000
BIT10 = 002000	DT.KBE= 000024	INHRRE= 000400	MSGSKE= 000003	RELTIM= 010000
BIT11 = 004000	DT.KBP= 000026	INIT = 000030	MSGSMB= 000015	RESREG= ***** G
BIT12 = 010000	DT.KBR= 000022	INTR = 000120	MSGSMH= 000014	RES1 = 000056
BIT13 = 020000	DT.KBU= 000030	IOMOD = 100000	MSGSMS= 000016	RES2 = 000060
BIT14 = 040000	DT.MLS= 000032	IOMODP= 102000	MSGSTD= 000000	RICHAR= 031060
BIT15 = 100000	DT.MTI= 000110	IOMODR= 112000	MSGSYS= 000012	RPTDAT= 002000
BIT2 = 000004	DT.OFF= 000070	IOMODX= 110000	MSGVEC= 000020	RSTRT = 000112
BIT3 = 000010	DT.PAS= 000074	JACK = 035060	NBKM0D= 001000	RUBOUT= 000177
BIT4 = 000020	DT.PC = 000002	KIPAR0= 172340	NCPUOP= 000020	RUNMOD= 100000
BIT5 = 000040	DT.PFL= 000062	KIPAR1= 172342	NDAPTY= 000002	R5VALU= 001740
BIT6 = 000100	DT.PSW= 000004	KIPAR2= 172344	NULL = 000000	SAM = 075464
BIT7 = 000200	DT.PTA= 000064	KIPAR3= 172346	OV.A = 000762	SAVREG= ***** G
BIT8 = 000400	DT.RCS= 000102	KIPAR4= 172350	OV.CQ = 000122RG	SBADR = 000102
BIT9 = 001000	DT.REL= 000040	KIPAR5= 172352	OV.CQS= 000050 G	SBKMOD= 000000
BKDEF = 000002	DT.SCT= 000066	KIPAR6= 172354	OV.HIC= 000362RG	SBKSEL= 010000
BKMOD = 000020	DT.SMX= 000106	KIPAR7= 172356	OV.HIK= 000120RG	SC.ADR= 000006
BKMODE= 040000	DT.SP = 000006	KIPDR0= 172300	OV.HIT= 001344RG	SC.ALC= 000014
BKSLSH= 000134	DT.SSI= 000046	KIPDR1= 172302	OV.KBB= 000000RG	SC.APC= 000016
CAPRES= 000004	DT.STO= 000010	KIPDR2= 172304	OV.KBS= 000122 G	SC.CKL= 000002
CASTAT= 000004	DT.ST1= 000012	KIPDR3= 172306	OV.REM= 000002	SC.CKP= 000004
CCNTRL= ***** G	DT.SWR= 000056	KIPDR4= 172310	OV.TQ = 000364RG	SC.CLD= 000000
CDERCT= 000146	DT.SYP= 000072	KIPDR5= 172312	OV.TQE 001346R	SC.HLD= 000010
CDWDCT= 000144	DT.WBU= 000050	KIPDR6= 172314	OV.TQS= 000076 G	SC.SCA= 000012
CKTIM = 100000	DT.WHL= 000054	KIPDR7= 172316	OWEN = 024020	SENDLS= 177777
CLKPRE= 000001	DT.WLL= 000052	KONTRL= ***** G	PAERR = 000010	SIZPOL 000000RG
CONFIG= 000056	DVID1 = 000014	KTERRO= 000040	PARPRE= 002000	SQFCNT= 000042
CPUCPE= ***** G	ECCMEM= 000100	KTPRES= 000400	PARSTA= 000100	SQFPAS= 000046
CQOVF = 000001	ECCSTA= 000010	KTSET = ***** G	PASCNT= 000034	SPACE = 000040
CR = 000015	ENBEOP= 010000	KTSTAT= 000020	PDPLSI= 020000	SPOINT= 000032



SPVALU= 002200	TMPID = 000002	WTINRE= 000352	\$FSUNT= 000130	\$TSK4 = 050047
SR0 = 177572	TQOVF = 000002	WTWHMI= 000222	\$F\$WHI= 000120	\$TSK5 = 050050
SR1 = 177574	UIPAR0= 177640	XFLAG = 000005	\$F\$YES= 000402	\$\$ARGC= 000002
SR2 = 177576	UIPAR1= 177642	XOFF = 000023	\$IFLEV= 177777	\$\$BYTE= 000403
SR3 = 172516	UIPAR2= 177644	XON = 000021	\$ISKO = 000001	\$\$CASE= 000000
SS.IOP= 177600	UIPAR3= 177646	\$BGNLE= 177777	\$ISK1 = 000001	\$\$DST = 000000
SS.PCL= 172136	UIPAR4= 177650	\$ERFLG= 000400	\$ISK2 = 000001	\$\$ELOC= 000402
SS.PCS= 172100	UIPAR5= 177652	\$F\$AND= 000310	\$LOCTA= 177777	\$\$ERFL= 000000
SS.RFS= 160000	UIPAR6= 177654	\$F\$BAD= 000401	\$LSTIN= 000001	\$\$FLAG= 000001
SS.OKC= 004000	UIPAR7= 177656	\$F\$BLA= 000170	\$LSTTA= 000001	\$\$FROM= 000000
SS.1KP= 000040	UIPDR0= 177600	\$F\$CAS= 000150	\$NESTL= 177777	\$\$LOC = 001314R
SS.120= 120000	UIPDR1= 177602	\$F\$DEC= 000220	\$NSKO = 000300	\$\$LOCN= 000000
SS.28 = 000034	UIPDR2= 177604	\$F\$DO = 000340	\$NSK1 = 000120	\$\$REG = 177777
STAT = 000026	UIPDR3= 177606	\$F\$FAL= 000405	\$NSK2 = 000110	\$\$RETU= 000000
STATBI= 064757	UIPDR4= 177610	\$F\$G00= 000400	\$NSK3 = 000110	\$\$RTN1= 050000
STAT1 = 000027	UIPDR5= 177612	\$F\$IF = 000110	\$NSK4 = 000110	\$\$RTN2= 050001
SUSPND= 000001	UIPDR6= 177614	\$F\$INC= 000210	\$NSK5 = 000110	\$\$SRC = 000000
SVR0 = 000062	UIPDR7= 177616	\$F\$L00= 000200	\$\$SAVLE= 177777	\$\$TGSV= 000000
SVR1 = 000064	WASADR= 000104	\$F\$NAM= 000160	\$\$SSKO = 050052	\$\$TGS1= 000000
SVR2 = 000066	WBSTAT= 000040	\$F\$NO = 000403	\$TAGLE= 177777	\$\$TGS2= 000000
SVR3 = 000070	WBUFEA= 000136	\$F\$OR = 000320	\$TAGNU= 050055	\$\$TO = 000000
SVR4 = 000072	WBUFPA= 000134	\$F\$RTI= 000350	\$TEMP = 000300	\$\$\$TAG= 050000
SVR5 = 000074	WBUFRQ= 000140	\$F\$RTN= 000300	\$TSKO = 050051	. = 001346R
SVR6 = 000076	WBUFSZ= 000142	\$F\$SEL= 000140	\$TSK1 = 050052	
SYSCNT= 000052	WDFR = 000116	\$F\$THE= 000330	\$TSK2 = 050054	
SYSERR= 000100	WDTD = 000114	\$F\$TRU= 000404	\$TSK3 = 050046	

. ABS. 000000 000  
001346 001

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

DSKZ:SIZEPLE,DSKZ:SIZEPLE=SPMAC/ML,EQUATE,SIZEPLE  
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 [XERLSU XERSPL version 102(2263)/3(61) running on MTA061, 25-Sep-78 12:27:16]  
12:27:16 LPDAT [XERSUS 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\*\*