

IDENTIFICATION

PRODUCT CODE: AC-S942A-MC
PRODUCT NAME: CNKXAAO KXT-11 PROC DIAG
DATE CREATED: MARCH 1982
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: G. PASQUANTONIO

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 DIGITAL'S 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) 1982 DIGITAL EQUIPMENT CORPORATION

1.0 GENERAL INFORMATION

THIS DIAGNOSTIC IS A RAM-BASED VERSION OF "CIKXAA0" AND IS INTENDED FOR FIELD SUPPORT OF THE KXT-11 SBC (M8063-AA).

1.1 ABSTRACT

THE KXT11-AA IS A T11 BASED SINGLE BOARD COMPUTER (SBC-11). IT INCLUDES A T11 CPU, 2KW RAM, SOCKETS FOR OPTIONAL ROM, TWO 8 BIT SERIAL I/O PORTS, AND ONE 8 BIT PARALLEL I/O PORT. THIS PROGRAM IS AN ASSORTMENT OF 147 (OCTAL) TESTS DESIGNED TO VERIFY THE INTEGRITY AND OPERABILITY OF THOSE COMPONENTS.

THE PROGRAM RESIDES IN Q-BUS RAM AND EXERCISES THE VARIOUS LOGICAL SEGMENTS OF THE SBC BOARD IN THE FOLLOWING SEQUENCE:

1. T11 CPU INSTRUCTION TESTS.
2. T11 CPU TRAPS AND INTERRUPTS TESTS.
3. LOCAL 2KW RAM TESTS.
4. LOCAL 1KW ROM ADDRESS TEST.
5. SERIAL LINE 1 TESTS.
6. SERIAL LINE 2 TESTS.
7. PARALLEL I/O PORT TESTS.

THIS PROGRAM IS DESIGNED TO RUN UNDER THE XXDP+ MONITOR. IT DOES NOT REQUIRE THE SERVICES OF THE DIAGNOSTIC SUPERVISOR. ALL TERMINAL I/O (ERROR REPORTS ETC.) ARE HANDLED BY DIRECT CALLS (EMT'S) TO THE MONITOR.

 *** THIS PROGRAM IS NOT ACT/APT COMPATABLE, HOWEVER ***
 *** IT IS CHAINABLE DIRECTLY UNDER THE XXDP+ MONITOR ***

1.2 HARDWARE REQUIRED

KXT11-AA (M8063-AA) MODULE UNDER TEST, WITH 1KW MACRO-ODT ROM.
 KXT11-LP (H3275) DUAL LOOP-BACK CONNECTOR.
 16KW (OR MORE) Q-BUS MEMORY (MSV11-D).
 XXDP+ LOAD DEVICE AND MEDIA (RX01, RX02, OR TU58).
 CONSOLE TERMINAL (VT52, VT100, ETC...)
 Q-BUS EXERCISER (G5281) MODULE (OPTIONAL).

1.3 RELATED DOCUMENTS AND STANDARDS

KXT11-AA ENGINEERING SPEC.
XXDP+ DIAGNOSTIC SYSTEM USERS GUIDE.

2.0 OPERATING PROCEDURE

THE PROGRAM IS LOADED AND STARTED USING STANDARD XXDP+ LOAD PROCEDURES. THE START/RESTART ADDRESS IS 200.

2.1 DEFAULT PROGRAM PARAMETERS

THE PROGRAM REQUIRES THAT THE KXT-11 BE CONFIGURED AS FOLLOWS:

START ADDRESS	172000		(POWER-UP START)
RESTART ADDRESS	172004		(TIME-OUT RESTART)
LOCAL ROM ADDRESS/SIZE	170000	1KW	(MACRO-ODT)
LOCAL RAM ADDRESS/SIZE	160010	2KW-4	
BEVNT VECTOR	000100	PRI6	
BHALT VECTOR	000140	PRI7	(MASKABLE)
SERIAL LINE 1 ADDRESS	177560		(CONSOLE TERMINAL)
RCVR VECTOR	000060	PRI4	
XMTR VECTOR	000064	PRI4	
SERIAL LINE 2 ADDRESS	176540		(WITH LOOP-BACK...)
RCVR VECTOR	000120	PRI5	(...CLOCK DISABLED)
XMTR VECTOR	000124	PRI5	
PARALLEL PORT ADDRESS	176200		(MODE 1, A IN, B OUT...)
OUTPUT (B) VECTOR	000130	PRI5	(...WITH LOOP-BACK)
INPUT (A) VECTOR	000134	PRI5	
Q-BUS RAM ADDRESS/SIZE	000000	16KW	(XXDP+ MINIMUM)
Q-BUS EXERCISER CSR1	174020		(OPTIONAL)

2.2 RUN TIME OPTIONS

THE FOLLOWING SOFT SWITCH-REGISTER OPTIONS ARE SUPPORTED:

SWR<15> 100000 HALT ON ERROR.
SWR<13> 020000 INHIBIT ERROR MESSAGES.
SWR<12> 010000 PRINT ERROR SUMMARY AT END-PASS.
SWR<09> 001000 LOOP ON ERROR.
SWR<02:00> N RUN SEGMENT N (1 - 7) ONLY.

THE PROGRAM WILL ASK FOR AN INITIAL SWITCH SETTING AT START OR RESTART TIME. THEREAFTER, YOU MAY REQUEST A CHANGE BY TYPING CONTROL G <^G>.

3.0 PERFORMANCE AND PROGRESS REPORTS

THE PROGRAM REQUIRES ABOUT 6 SECONDS PER PASS, AND WILL REPORT THE END OF EACH PASS ON THE CONSOLE TERMINAL, INCLUDING A TOTAL (CUMULATIVE) ERROR COUNT. AN OPTIONAL ERROR SUMMARY MAY BE INCLUDED IF ENABLED VIA SWR<12>.

4.0 ERROR REPORTING AND RECOVERY

BY DEFAULT, THE PROGRAM WILL REPORT AND LOG ALL ERRORS AS THEY OCCUR AND PROCEED (FOREVER OR UNTIL THE OPERATOR INTERVENES).

ALTERNATE ACTION AND/OR RECOVERY MAY BE SELECTED VIA THE SOFT SWITCH REGISTER AS DESCRIBED IN 2.2 ABOVE.

4.1 DYNAMIC ERROR LOGGING

THE ERROR LOGGING FACILITY PROVIDES FOR THE ACCUMULATION OF ERROR STATISTICS OVER AN EXTENDED PERIOD OF TIME. ALL ERRORS ARE LOGGED, WHETHER REPORTED OR NOT.

THE ERROR DATA IS ACCUMULATED IN Q-BUS SCRATCH RAM STARTING AT LOCATION 004000, IN THE FOLLOWING FORMAT:

004000	XXXXXX	; TOTAL ERROR COUNT.
004002	YYYYYY	; TOTAL PASS COUNT.
004004	ZZZZZZ	; CONSECUTIVE ERROR-FREE PASS COUNT.
004006	PCPCPC	; 1ST ERROR PC...
004010	N	; ...AND NUMBER OF OCCURANCES.
004012	PCPCPC	; 2ND...
004014	N	
:	:	; ...AND ROOM FOR A TOTAL OF 510.
:	:	
007776	177777	; TABLE TERMINATOR.

AN ERROR SUMMARY REPORT MAY BE ENABLED VIA THE SWITCH REGISTER SWR<12> AS DESCRIBED IN 2.2 ABOVE.

5.0 HARDWARE TEST DESCRIPTION

THIS SECTION PROVIDES A BRIEF DESCRIPTION OF THE FUNCTIONALITY OF EACH OF THE SEVEN MAJOR PROGRAM SEGMENTS. REFER TO THE PROGRAM LISTING (6.0) FOR FURTHER DETAILS.

5.1 SEGMENT 1 -- T11 CPU BASIC INSTRUCTION TESTS.

VERIFY THAT T11 CAN EXECUTE ALL INSTRUCTIONS (EXCEPT 'WAIT' AND 'HALT') IN ALL ADDRESS MODES. ALSO CHECKS FOR EXPECTED SIDE EFFECTS (CC BITS, AUTO-INCR/DECR, ETC...).

5.2 SEGMENT 2 -- T11 CPU TRAPS AND INTERRUPTS TEST.

VERIFY THAT SOFTWARE TRAPS AND INTERRUPTS EXECUTE CORRECTLY, AND THAT PROPER STACK POINTER AND PSW CONTEXT IS MAINTAINED.

IF A Q-BUS EXERCISOR MODULE IS INSTALLED (AT 174020) POWER-FAIL, BEVNT, AND Q-BUS INTERRUPTS WILL ALSO BE TESTED HERE.

5.3 SEGMENT 3 -- LOCAL 2KW RAM TESTS.

VERIFY READ/WRITE ACCESS TO THE ON-BOARD 2KW RAM USING VARIOUS ADDRESS AND DATA PATTERNS. IF A Q-BUS-EXERCISER MODULE IS INSTALLED (AT 174020), TEST READ/WRITE DMA ACCESS AS WELL.

NOTE: MACRO-ODT UTILIZES A PORTION OF THE LOCAL RAM. IF A BREAK-AND-PROCEED SEQUENCE IS SERVICED DURING THIS SEGMENT ERRORS MAY OCCUR IF/WHEN RAM BACKGROUND DATA PATTERNS ARE DESTROYED (BY ODT ITSELF).

5.4 SEGMENT 4 -- LOCAL 1KW ROM ADDRESS TEST.

VERIFY THAT THE ROM ADDRESS SPACE IS CORRECTLY CONFIGURED. THIS IS AN ADDRESS RESPONSE TEST ONLY. ROM DATA VERIFICATION IS NOT ATTEMPTED.

5.5 SECTION 5 -- SERIAL LINE 1 TESTS (DEC DC319 DLART).

VERIFY THAT THE "CONSOLE" SERIAL I/O PORT FUNCTIONS CORRECTLY, INCLUDING THE DETECTION OF FRAMING AND OVERRUN ERRORS, AND THAT INTERRUPTS OCCUR THRU VECTORS 60 AND 64 AT PRIORITY 4.
VERIFY THAT THE PROGRAMMABLE BAUD-RATE FEATURE WORKS CORRECTLY.
VERIFY THAT "BREAK" INVOKES A MASKABLE LEVEL 7 HALT INTERRUPT THRU VECTOR 140.

5.6 SECTION 6 -- SERIAL LINE 2 TESTS (DEC DC319 DLART).

SAME AS SECTION 5 WITH THE FOLLOWING EXCEPTIONS:

1. "BREAK" DOES NOT INVOKES A HALT REQUEST.
2. INTERRUPTS VECTOR THRU 120 AND 124 AT PRIORITY 5.

NOTE: THIS SEGMENT REQUIRES THAT THE SERIAL LOOP-BACK SWITCH ON THE DUAL LOOP-BACK CARD BE CORRECTLY SET TO THE "LOOP" POSITION.

5.7 SECTION 7 -- PARALLEL I/O PORT TESTS (INTEL 8255A).

VERIFY THAT THE PARALLEL I/O PORT FUNCTIONS CORRECTLY.
CHECK THAT DATA CAN BE TRANSMITTED FROM PORT B, THRU THE PARALLEL LOOP-BACK CONNECTOR, AND RECEIVED IN PORT A.
CHECK THAT INTERRUPTS OCCUR THRU VECTORS 130 (PORT B) AND 134 (PORT A) AT PRIORITY 5.

NOTE: THE RED LED ON THE TOP EDGE OF THE KXT-11 BOARD SHOULD LIGHT AT THE BEGINNING OF THIS SEGMENT, AND REMAIN ON FOR ABOUT 1 SECOND.

5.8 END PASS

AT THIS POINT AN END-PASS MESSAGE (AND OPTIONAL ERROR SUMMARY) WILL BE PRINTED ON THE CONSOLE TERMINAL.

6.0 PROGRAM LISTING.

THE PROGRAM LISTING FOLLOWS:

NOTE THAT THE TABLE OF CONTENTS INCLUDES A SEQUENTIAL LIST OF ALL ERROR HALTS AND AN INDICATION OF THE NATURE OF THE ERROR ENCOUNTERED AT THAT LOCATION.

20	SWITCH REGISTER OPTIONS
97	KXT-11 (FALCON) DEFAULT CONFIGURATION
198	INITIAL START AND CLEAR VECTORS.
238	
239	RESTART
252	QUICK-VERIFY EMT AND TRAP.
275	PC 10204 = EMT OR TRAP OPCODE (IN R0) FAILED.
284	PC 10216 = RTI AFTER EMT OR TRAP FAILED.
304	
306	SECTION 1 -- T11 BASIC INSTRUCTION TESTS.
311	T1 -- BRANCHES, FORWARD, BACKWARD, AND CONDITIONAL
314	PC 10354 = FORWARD BRANCH FAILED.
318	PC 10370 = BACKWARD BRANCH FAILED.
334	PC 10430 = BRANCH FAILURE OR CC NOT 0000.
345	PC 10456 = BRANCH FAILURE OR CC NOT 1000.
358	PC 10510 = BRANCH FAILURE OR CC NOT 1010.
370	PC 10540 = BRANCH FAILURE OR CC NOT 1011.
384	PC 10574 = BRANCH FAILURE OR CC NOT 1111.
387	T2 -- SET AND CLEAR ALL CONDITION CODES
390	PC 10610 = CC BITS WRONG AFTER SCC.
394	PC 10620 = CC BITS WRONG AFTER CCC.
396	T3 -- TEST REGISTER SELECTION
411	PC 10702 = REGISTER SELECTION FAILURE.
414	T4 -- JMP INSTRUCTION -- MODE 1
418	PC 10722 = MODE 1 JUMP FAILED.
420	PC 10730 = CC BITS ALTERED ON JMP.
423	PC 10742 = R0 INCORRECT AFTER JUMP.
426	T5 -- JMP INSTRUCTION -- MODES 2 AND 3
430	PC 10762 = MODE 2 JUMP FAILED.
432	PC 10770 = CC BITS ALTERED AFTER JMP.
435	PC 11002 = MODE 2 FAILED ON JUMP.
442	PC 11032 = MODE 3 JUMP FAILED.
445	PC 11046 = MODE 3 JUMP FAILED.
448	T6 -- JMP INSTRUCTION -- MODES 4 AND 5
452	PC 11066 = DIDN'T JUMP AT ALL.
454	PC 11074 = MODE 4 JUMP FAILED..
457	PC 11106 = R0 INCORRECT AFTER JUMP.
463	PC 11134 = MODE 5 JUMP DIDN'T.
464	PC 11140 = AUTO-DECR FAILED ON MODE 5 JUMP.
467	PC 11152 = R0 INCORRECT AFTER JMP5.
470	T7 -- JMP INSTRUCTION -- MODES 6 AND 7
473	PC 11172 = MODE 6 JUMP FAILED.
476	PC 11204 = R3 WRONG AFTER JMP6.
478	PC 11214 = JUMP FAILED.
482	PC 11230 = JUMP FAILED.
487	PC 11250 = MODE 7 JUMP FAILED.
491	PC 11270 = JUMP FAILED.
495	PC 11312 = MODE 7 JUMP FAILED..
498	T10 -- JSR AND RTS INSTRUCTIONS
502	PC 11334 = JSR INSTRUCTION FAILED.
504	PC 11342 = CC BITS CHANGED ON JSR.
507	PC 11354 = STACK WRONG AFTER JSR.
510	PC 11366 = SP DID NOT HAVE CORRECT RETURN ADDRESS.
514	PC 11400 = RTS INSTRUCTION FAILED.
517	PC 11412 = SP WAS NOT RESTORED BY RTS INSTRUCTION.
520	

TABLE OF CONTENTS

521	DESTINATION MODE 0
523	T11 -- TSTB, CLRB, AND MOVB
528	PC 11434 = CLRB FAILED.
531	PC 11444 = TSTB FAILED..
534	PC 11456 = INCORRECT CC BITS.
537	PC 11466 = INCORRECT CC BITS.
539	T12 -- CMPB AND BISB
543	PC 11506 = INCORRECT CC BITS.
546	PC 11520 = BISB OR CMPB INSTRUCTION FAILED.
550	PC 11534 = CMPB INSTRUCTION FAILED (WRONG CC).
553	PC 11544 = CMPB INSTRUCTION FAILED (WRONG CC).
557	PC 11562 = CMPB BECAME CMP INSTRUCTION.
561	PC 11600 = MOVB OR CMPB FAILED.
564	T13 -- BICB AND BITB
570	PC 11626 = INCORRECT CC BITS.
573	PC 11636 = BICB OR BITB INSTRUCTION FAILED.
576	PC 11650 = INCORRECT CC BITS.
579	PC 11660 = BISB INSTRUCTION FAILED.
582	PC 11672 = INCORRECT CC BITS.
585	PC 11704 = INCORRECT CC BITS.
587	T14 -- INCB AND DECB
592	PC 11726 = INCORRECT CC BITS.
596	PC 11742 = INCORRECT CC BITS.
599	PC 11752 = INCORRECT CC BITS.
602	PC 11762 = INCORRECT CC BITS.
605	PC 11774 = INCB INSTRUCTION FAILED.
609	PC 12006 = INCORRECT CC BITS.
612	PC 12016 = INCORRECT CC BITS.
616	PC 12032 = INCORRECT CC BITS.
619	PC 12042 = INCORRECT CC BITS.
621	T15 -- COMB AND NEGB
626	PC 12064 = INCORRECT CC BITS.
629	PC 12076 = COMB INSTRUCTION FAILED.
633	PC 12110 = INCORRECT CC BITS.
636	PC 12122 = COMB INSTRUCTION FAILED.
641	PC 12140 = INCORRECT CC BITS.
646	PC 12154 = INCORRECT CC BITS.
649	PC 12166 = NEGB INSTRUCTION FAILED.
653	PC 12202 = INCORRECT CC BITS.
656	PC 12214 = NEGB FAILED.
659	T16 -- ROLB AND RORB
665	PC 12240 = INCORRECT CC BITS.
668	PC 12252 = ROLB INSTRUCTION FAILED.
671	PC 12262 = INCORRECT CC BITS.
675	PC 12276 = ROLB FAILED..
683	PC 12322 = RORB INSTRUCTION FAILED.
686	PC 12332 = INCORRECT CC BITS.
689	PC 12342 = INCORRECT CC BITS.
692	PC 12354 = RORB FAILED.
695	T17 -- ASLB AND ASRB
701	PC 12400 = INCORRECT CC BITS.
704	PC 12412 = ASLB INSTRUCTION FAILED.
707	PC 12422 = INCORRECT CC BITS.
710	PC 12432 = INCORRECT CC BITS.
718	PC 12456 = ASRB INSTRUCTION FAILED.
721	PC 12466 = INCORRECT CC BITS.

724	PC 12476 = INCORRECT CC BITS.
729	PC 12514 = INCORRECT CC BITS.
732	PC 12526 = ASRB FAILED.
735	T20 -- ADCB AND SBCB
740	PC 12546 = INCORRECT CC BITS.
746	PC 12564 = INCORRECT CC BITS.
749	PC 12576 = ADCB INSTRUCTION FAILED.
754	PC 12614 = INCORRECT CC BITS.
757	PC 12626 = ADCB INSTRUCTION FAILED.
762	PC 12644 = INCORRECT CC BITS.
768	PC 12662 = INCORRECT CC BITS.
771	PC 12674 = SBCB INSTRUCTION FAILED.
777	PC 12712 = INCORRECT CC BITS.
780	PC 12724 = SBCB INSTRUCTION FAILED.
784	PC 12736 = INCORRECT CC BITS.
788	PC 12750 = INCORRECT CC BITS.
791	PC 12762 = SBCB INSTRUCTION FAILED.
796	PC 13000 = INCORRECT CC BITS.
798	T21 -- TST, CLR, AND MOV
802	PC 13016 = INCORRECT CC BITS.
805	PC 13026 = INCORRECT CC BITS.
809	PC 13042 = INCORRECT CC BITS.
812	PC 13052 = INCORRECT CC BITS.
815	PC 13062 = R1 WRONG AFTER MOV.
819	PC 13074 = INCORRECT CC BITS.
821	T22 -- CMP AND BIS
826	PC 13116 = INCORRECT CC BITS.
829	PC 13126 = BIS OR CMP INSTRUCTION FAILED.
832	PC 13140 = CMP INSTRUCTION FAILED (WRONG CC).
835	PC 13152 = BIC OR CMP FAILED.
838	T23 -- BIC AND BIT
845	PC 13204 = INCORRECT CC BITS.
848	PC 13214 = BIC OR BIT INSTRUCTION FAILED.
851	PC 13226 = INCORRECT CC BITS.
854	PC 13240 = BIT OR BIS INSTRUCTION FAILED.
857	PC 13252 = INCORRECT CC BITS.
861	PC 13266 = INCORRECT CC BITS.
865	PC 13300 = INCORRECT CC BITS.
868	PC 13310 = BIC FAILED.
871	T24 -- INC AND DEC
876	PC 13332 = INCORRECT CC BITS.
880	PC 13346 = INCORRECT CC BITS.
883	PC 13356 = INCORRECT CC BITS.
886	PC 13366 = INCORRECT CC BITS.
889	PC 13400 = INC INSTRUCTION FAILED.
893	PC 13412 = INCORRECT CC BITS.
896	PC 13422 = INCORRECT CC BITS.
900	PC 13436 = INCORRECT CC BITS.
903	PC 13446 = INCORRECT CC BITS.
905	T25 -- COM AND NEG
910	PC 13470 = INCORRECT CC BITS.
913	PC 13502 = COM INSTRUCTION FAILED.
917	PC 13514 = INCORRECT CC BITS.
920	PC 13526 = COM INSTRUCTION FAILED.
925	PC 13544 = INCORRECT CC BITS.
930	PC 13560 = INCORRECT CC BITS.

TABLE OF CONTENTS

933		PC 13572 = NEG INSTRUCTION FAILED.
937		PC 13606 = INCORRECT CC BITS.
940		PC 13620 = NEG FAILED.
943	T26	-- ROL AND ROR
949		PC 13644 = INCORRECT CC BITS.
952		PC 13656 = ROL INSTRUCTION FAILED.
955		PC 13666 = INCORRECT CC BITS.
959		PC 13702 = ROL FAILED.
967		PC 13726 = ROR INSTRUCTION FAILED.
970		PC 13736 = INCORRECT CC BITS.
973		PC 13746 = INCORRECT CC BITS.
976		PC 13760 = ROR FAILED.
979	T27	-- ASL AND ASR
985		PC 14004 = INCORRECT CC BITS.
988		PC 14016 = ASL INSTRUCTION FAILED.
991		PC 14026 = INCORRECT CC BITS.
994		PC 14036 = INCORRECT CC BITS.
1002		PC 14062 = ASR INSTRUCTION FAILED.
1005		PC 14072 = INCORRECT CC BITS.
1008		PC 14102 = INCORRECT CC BITS.
1013		PC 14120 = INCORRECT CC BITS.
1016		PC 14132 = ASR FAILED.
1019	T30	-- ADC AND SBC
1024		PC 14152 = INCORRECT CC BITS.
1030		PC 14170 = INCORRECT CC BITS.
1033		PC 14202 = ADC INSTRUCTION FAILED.
1038		PC 14220 = INCORRECT CC BITS.
1041		PC 14232 = ADC INSTRUCTION FAILED.
1046		PC 14250 = INCORRECT CC BITS.
1052		PC 14266 = INCORRECT CC BITS.
1055		PC 14300 = SBC INSTRUCTION FAILED.
1061		PC 14316 = INCORRECT CC BITS.
1064		PC 14330 = SBC INSTRUCTION FAILED.
1068		PC 14342 = INCORRECT CC BITS.
1072		PC 14354 = INCORRECT CC BITS.
1075		PC 14366 = SBC INSTRUCTION FAILED.
1080		PC 14404 = INCORRECT CC BITS.
1082	T31	-- SXT, SWAB AND XOR
1088		PC 14426 = INCORRECT CC BITS.
1091		PC 14436 = SXT INSTRUCTION FAILED.
1095		PC 14450 = INCORRECT CC BITS.
1098		PC 14462 = SXT FAILED.
1105		PC 14502 = INCORRECT CC BITS.
1108		PC 14514 = SWAB INSTRUCTION FAILED.
1114		PC 14534 = INCORRECT CC BITS.
1117		PC 14546 = SWAB FAILED.
1124		PC 14570 = INCORRECT CC BITS.
1131		PC 14612 = INCORRECT CC BITS.
1137		PC 14634 = INCORRECT CC BITS.
1140		PC 14646 = XOR FAILED.
1143	T32	-- ADD AND SUB
1147		PC 14666 = INCORRECT CC BITS.
1150		PC 14700 = ADD INSTRUCTION FAILED.
1154		PC 14714 = INCORRECT CC BITS.
1157		PC 14726 = ADD INSTRUCTION FAILED.
1161		PC 14742 = INCORRECT CC BITS.

TABLE OF CONTENTS

1166	PC 14762 = ADD INSTRUCTION FAILED.
1171	PC 15000 = ADD FAILED.
1177	PC 15020 = INCORRECT CC BITS.
1180	PC 15032 = SUB INSTRUCTION FAILED.
1185	PC 15050 = SUB INSTRUCTION FAILED.
1190	PC 15070 = INCORRECT CC BITS.
1193	PC 15102 = SUB INSTRUCTION FAILED.
1197	PC 15116 = INCORRECT CC BITS.
1199	T33 -- MTPS AND MFPS
1204	PC 15140 = INCORRECT CC BITS.
1207	PC 15150 = MTPS OR MFPS INSTRUCTION FAILED.
1209	PC 15156 = INCORRECT CC BITS.
1213	PC 15172 = INCORRECT CC BITS.
1216	PC 15202 = INCORRECT CC BITS.
1219	PC 15214 = MTPS OR MFPS INSTRUCTION FAILED.
1222	
1223	NON-ZERO ADDRESS MODES
1225	T34 -- TEST MODES 0 AND 1 USING MOVB AND MOV
1231	PC 15242 = MOVB INSTRUCTION FAILED IN MODE 0.
1237	PC 15264 = MOV INSTRUCTION FAILED IN MODE 0.
1248	PC 15330 = MOV INSTRUCTION FAILED IN MODE 1.
1254	PC 15354 = MOV INSTRUCTION FAILED IN MODE 1.
1257	T35 -- TEST MODE 2 USING MOVB AND MOV
1272	PC 15436 = INSTRUCTIONS FAILED IN MODE 2.
1284	PC 15476 = INSTRUCTIONS FAILED IN MODE 2.
1287	T36 -- TEST MODE 3 USING MOVB AND MOV
1302	PC 15600 = INSTRUCTIONS FAILED IN MODE 3.
1315	PC 15654 = INSTRUCTIONS FAILED IN MODE 3.
1318	T37 -- TEST MODE 4 USING MOVB AND MOV
1326	PC 15714 = INSTRUCTIONS FAILED IN MODE 4.
1337	PC 15750 = INSTRUCTIONS FAILED IN MODE 4.
1344	PC 15776 = INSTRUCTIONS FAILED IN MODE 4.
1353	PC 16032 = INSTRUCTIONS FAILED IN MODE 4.
1368	PC 16102 = INSTRUCTIONS FAILED IN MODE 4.
1371	T40 -- TEST MODE 5 USING MOVB AND MOV
138	PC 16200 = INSTRUCTION FAILED IN MODE 5.
1395	PC 16226 = INSTRUCTIONS FAILED IN MODE 5.
1413	PC 16312 = INSTRUCTIONS FAILED IN MODE 5.
1416	T41 -- TEST MODE 6 USING MOVB AND MOV
1429	PC 16412 = INSTRUCTIONS FAILED IN MODE 6.
1436	PC 16446 = INSTRUCTIONS FAILED IN MODE 6.
1439	T42 -- TEST MODE 7 USING MOVB AND MOV
1451	PC 16544 = MODE 7 IS FAILING.
1456	PC 16574 = INSTRUCTIONS FAILED IN MODE 7.
1459	T43 -- TSTB, CLRB, AND MOVB
1465	PC 16622 = INCORRECT CC BITS.
1468	PC 16632 = INCORRECT CC BITS.
1471	PC 16644 = INCORRECT CC BITS.
1474	PC 16654 = INCORRECT CC BITS.
1480	PC 16702 = MOVB INSTRUCTION FAILED.
1483	PC 16712 = MOVB INSTRUCTION FAILED.
1486	T44 -- CMPB AND BISB
1493	PC 16746 = INCORRECT CC BITS.
1496	PC 16756 = BISB OR CMPB INSTRUCTION FAILED.
1499	PC 16766 = CMPB INSTRUCTION FAILED (WRONG CC).
1502	PC 16776 = INSTR FAILED.

TABLE OF CONTENTS

1505	T45 -- RIBB AND BITB
1514	PC 17040 = INCORRECT CC BITS.
1517	PC 17052 = BICB OR BITB INSTRUCTION FAILED.
1520	PC 17064 = INCORRECT CC BITS.
1523	PC 17074 = BITB OR BISB INSTRUCTION FAILED.
1527	PC 17110 = INCORRECT CC BITS.
1530	PC 17122 = INCORRECT CC BITS.
1538	PC 17150 = INCORRECT CC BITS.
1541	PC 17162 = BICB OR CMP INSTRUCTION FAILED IN THE SPECIFIC MODE.
1544	PC 17172 = BICB INSTRUCTION FAILED.
1548	PC 17204 = INCORRECT CC BITS.
1550	T46 -- INCB AND DECB
1556	PC 17232 = INCORRECT CC BITS.
1560	PC 17246 = INCORRECT CC BITS.
1567	PC 17266 = INCORRECT CC BITS.
1572	PC 17304 = INCORRECT CC BITS.
1575	PC 17316 = INCB INSTRUCTION FAILED.
1579	PC 17330 = INCORRECT CC BITS.
1582	PC 17340 = INCORRECT CC BITS.
1586	PC 17356 = INCORRECT CC BITS.
1589	PC 17370 = INCORRECT CC BITS.
1592	PC 17404 = DECB INSTRUCTION FAILED.
1595	T47 -- COMB AND NEGB
1603	PC 17440 = INCORRECT CC BITS.
1606	PC 17452 = COMB INSTRUCTION FAILED.
1610	PC 17464 = INCORRECT CC BITS.
1614	PC 17500 = COMB INSTRUCTION FAILED.
1620	PC 17520 = INCORRECT CC BITS.
1626	PC 17540 = INCORRECT CC BITS.
1629	PC 17552 = NEGB INSTRUCTION FAILED.
1633	PC 17566 = INCORRECT CC BITS.
1636	PC 17600 = NEGB FAILED..
1639	T50 -- ROLB AND RORB
1646	PC 17630 = INCORRECT CC BITS.
1649	PC 17642 = ROLB INSTRUCTION FAILED.
1652	PC 17652 = INCORRECT CC BITS.
1656	PC 17666 = ROLB FAILED.
1665	PC 17716 = RORB INSTRUCTION FAILED.
1668	PC 17726 = INCORRECT CC BITS.
1671	PC 17736 = INCORRECT CC BITS.
1674	PC 17750 = RORB FAILED.
1677	T51 -- ASLB AND ASRB
1684	PC 20000 = INCORRECT CC BITS.
1687	PC 20012 = ASLB INSTRUCTION FAILED.
1690	PC 20022 = INCORRECT CC BITS.
1693	PC 20032 = INCORRECT CC BITS.
1703	PC 20066 = ASRB INSTRUCTION FAILED.
1706	PC 20076 = INCORRECT CC BITS.
1709	PC 20106 = INCORRECT CC BITS.
1714	PC 20124 = INCORRECT CC BITS.
1717	PC 20136 = ASRB FAILED.
1720	T52 -- ADCB AND SBCB
1726	PC 20162 = INCORRECT CC BITS.
1732	PC 20200 = INCORRECT CC BITS.
1735	PC 20212 = ADCB INSTRUCTION FAILED.
1740	PC 20230 = INCORRECT CC BITS.

TABLE OF CONTENTS

1743	PC 20242 = ADCB INSTRUCTION FAILED.
1748	PC 20260 = INCORRECT CC BITS.
1755	PC 20302 = INCORRECT CC BITS.
1758	PC 20314 = SBCB INSTRUCTION FAILED.
1764	PC 20332 = INCORRECT CC BITS.
1767	PC 20344 = SBCB INSTRUCTION FAILED.
1771	PC 20356 = INCORRECT CC BITS.
1775	PC 20370 = INCORRECT CC BITS.
1778	PC 20402 = SBCB INSTRUCTION FAILED.
1783	PC 20420 = INCORRECT CC BITS.
1785	T53 -- TST, CLR, AND MOV
1791	PC 20446 = INCORRECT CC BITS.
1794	PC 20456 = INCORRECT CC BITS.
1799	PC 20476 = INCORRECT CC BITS.
1802	PC 20506 = INCORRECT CC BITS.
1804	T54 -- CMP AND BIS
1811	PC 20540 = INCORRECT CC BITS.
1814	PC 20552 = CMP OR BIS INSTRUCTION FAILED.
1817	PC 20564 = NO AUTO INCREMENT.
1820	PC 20576 = INCORRECT CC BITS.
1823	PC 20610 = INCORRECT CC BITS.
1826	PC 20622 = INCORRECT CC BITS.
1837	PC 20672 = CMP OR BIS INSTRUCTIONS FAILED IN MODES OTHER THAN 0.
1840	PC 20704 = MODE 5 IS FAILING.
1847	PC 20734 = CMP OR BIS INSTRUCTIONS FAILED.
1850	T55 -- BIC AND BIT
1861	PC 21004 = INCORRECT CC BITS.
1864	PC 21014 = BIC OR BIT INSTRUCTION FAILED.
1867	PC 21026 = INCORRECT CC BITS.
1870	PC 21040 = BIT OR BIS INSTRUCTION FAILED.
1875	PC 21060 = INCORRECT CC BITS.
1878	PC 21072 = RD WRONG.
1883	PC 21106 = INCORRECT CC BITS.
1886	PC 21120 = INCORRECT CC BITS.
1892	PC 21146 = BIC FAILED IN MODE 6.
1899	PC 21170 = INCORRECT CC BITS.
1902	PC 21202 = STACK POINTER FOULED UP.
1905	T56 -- INC AND DEC
1911	PC 21230 = INCORRECT CC BITS.
1916	PC 21250 = INCORRECT CC BITS.
1919	PC 21260 = INCORRECT CC BITS.
1922	PC 21270 = INCORRECT CC BITS.
1925	PC 21304 = INC INSTRUCTION FAILED.
1929	PC 21316 = INCORRECT CC BITS.
1932	PC 21326 = INCORRECT CC BITS.
1935	PC 21342 = INCORRECT CC BITS.
1939	PC 21352 = INCORRECT CC BITS.
1941	T57 -- COM AND NEG
1947	PC 21402 = INCORRECT CC BITS.
1950	PC 21414 = COM INSTRUCTION FAILED.
1954	PC 21426 = INCORRECT CC BITS.
1957	PC 21440 = COM INSTRUCTION FAILED.
1963	PC 21460 = INCORRECT CC BITS.
1971	PC 21510 = INCORRECT CC BITS.
1974	PC 21522 = NEG INSTRUCTION FAILED.
1978	PC 21536 = INCORRECT CC BITS.

TABLE OF CONTENTS

1981	PC 21550 = NEG FAILED..
1984	T60 -- ROL AND ROR
1991	PC 21600 = INCORRECT CC BITS.
1994	PC 21612 = ROL INSTRUCTION FAILED.
1997	PC 21624 = INCORRECT CC BITS.
2002	PC 21642 = ROL FAILED..
2011	PC 21672 = ROR INSTRUCTION FAILED.
2014	PC 21702 = INCORRECT CC BITS.
2017	PC 21712 = INCORRECT CC BITS.
2020	PC 21724 = ROR FAILED..
2023	T61 -- ASL AND ASR
2030	PC 21754 = INCORRECT CC BITS.
2033	PC 21766 = ASL INSTRUCTION FAILED.
2036	PC 21776 = INCORRECT CC BITS.
2039	PC 22006 = INCORRECT CC BITS.
2049	PC 22042 = ASR INSTRUCTION FAILED.
2052	PC 22052 = INCORRECT CC BITS.
2055	PC 22062 = INCORRECT CC BITS.
2060	PC 22100 = INCORRECT CC BITS.
2063	PC 22112 = ASR FAILED.
2066	T62 -- ADC AND SBC
2072	PC 22136 = INCORRECT CC BITS.
2078	PC 22154 = INCORRECT CC BITS.
2081	PC 22166 = ADC INSTRUCTION FAILED.
2086	PC 22204 = INCORRECT CC BITS.
2089	PC 22216 = ADC INSTRUCTION FAILED.
2094	PC 22234 = INCORRECT CC BITS.
2101	PC 22256 = INCORRECT CC BITS.
2104	PC 22270 = SBC INSTRUCTION FAILED.
2110	PC 22306 = INCORRECT CC BITS.
2113	PC 22320 = SBC INSTRUCTION FAILED.
2117	PC 22332 = INCORRECT CC BITS.
2121	PC 22344 = INCORRECT CC BITS.
2124	PC 22356 = SBC INSTRUCTION FAILED.
2129	PC 22374 = INCORRECT CC BITS.
2131	T63 -- SXT, SWAB, AND XOR
2138	PC 22422 = INCORRECT CC BITS.
2141	PC 22432 = SXT INSTRUCTION FAILED.
2145	PC 22444 = INCORRECT CC BITS.
2148	PC 22456 = SXT FAILED.
2156	PC 22502 = INCORRECT CC BITS.
2159	PC 22514 = SWAB INSTRUCTION FAILED.
2165	PC 22536 = INCORRECT CC BITS.
2168	PC 22550 = SWAB FAILED..
2175	PC 22576 = INCORRECT CC BITS.
2182	PC 22624 = INCORRECT CC BITS.
2188	PC 22646 = INCORRECT CC BITS.
2191	PC 22662 = XOR FAILED..
2194	T64 -- ADD, SUB, AND SOB
2201	PC 22716 = INCORRECT CC BITS.
2204	PC 22732 = ADD INSTRUCTION FAILED.
2209	PC 22750 = ADD INSTRUCTION FAILED IN MODE 2.
2214	PC 22770 = INCORRECT CC BITS.
2217	PC 23004 = ADD INSTRUCTION FAILED.
2222	PC 23030 = INCORRECT CC BITS.
2229	PC 23064 = INCORRECT CC BITS.

TABLE OF CONTENTS

2234	PC 23104 = INCORRECT CC BITS.
2238	PC 23114 = ADD INSTRUCTION FAILED IN MODE 5.
2242	PC 23136 = INCORRECT CC BITS.
2245	PC 23152 = ADD FAILED.
2253	PC 23206 = INCORRECT CC BITS.
2256	PC 23222 = SUB INSTRUCTION FAILED.
2260	PC 23240 = SUB INSTRUCTION FAILED.
2265	PC 23264 = INCORRECT CC BITS.
2268	PC 23300 = SUB INSTRUCTION FAILED.
2272	PC 23314 = INCORRECT CC BITS.
2276	PC 23336 = INCORRECT CC BITS.
2279	PC 23350 = SUB INSTRUCTION FAILED.
2286	PC 23372 = SOB INSTRUCTION FAILED.
2290	PC 23404 = INCORRECT CC BITS.
2293	PC 23414 = SOB INSTRUCTION FAILED.
2296	PC 23426 = SOB INSTRUCTION FAILED.
2301	PC 23444 = SOB FAILED.
2304	T65 -- MTPS AND MFPS
2311	PC 23476 = INCORRECT CC BITS.
2314	PC 23506 = INCORRECT CC BITS.
2318	PC 23520 = MTPS OR MFPS INSTRUCTION FAILED.
2322	PC 23534 = INCORRECT CC BITS.
2325	PC 23546 = INCORRECT CC BITS.
2328	PC 23562 = MFPS INSTRUCTION FAILED IN MODE 6.
2331	T66 -- BYTE INSTRUCTIONS AFFECTING FULL WORDS
2337	PC 23604 = INCORRECT CC BITS.
2340	PC 23616 = SIGN WAS NOT EXTENDED IN R0.
2345	PC 23636 = INCORRECT CC BITS.
2348	PC 23646 = SIGN WAS NOT EXTENDED IN R0..
2355	PC 23700 = R6 DID NOT GET INCREMENTED.
2359	PC 23712 = BYTE INSTRUCTION IS FAILING WITH R6.
2363	PC 23724 = R6 WAS NOT DECREMENTED.
2369	PC 23750 = INCORRECT CC BITS.
2372	PC 23764 = TEMP FOULED UP.
2377	PC 24004 = INCORRECT CC BITS.
2380	PC 24020 = TEMP FOULED UP.
2384	
2386	SECTION 2 -- T11 TRAPS AND INTERRUPTS.
2390	T67 -- TEST AUTO INCREMENT/DECREMENT OF STACK POINTER (R6)
2396	PC 24062 = R6 DID NOT AUTO INCREMENT BY TWO.
2401	PC 24102 = R6 DID NOT AUTO DECREMENT BY 2.
2406	PC 24120 = WRONG AUTO INCREMENT OF R6.
2412	PC 24140 = WRONG INCREMENT OF R6.
2415	PC 24152 = WRONG INCREMENT OF R1.
2421	PC 24172 = WRONG INCREMENT OF R1.
2424	PC 24204 = WRONG INCREMENT OF R6.
2427	T70 -- TEST BYTE TRANSFERS USING STACK POINTER (R6)
2435	PC 24252 = FALSE TRANSFER OF .BYTE.
2444	PC 24314 = FALSE R6 .BYTE TRANSFER.
2453	PC 24356 = FALSE R6 .BYTE TRANSFER.
2462	PC 24420 = FALSE R6 .BYTE TRANSFER.
2471	PC 24462 = FAILED LOW OF 6 TO HIGH OF 1.
2474	T71 -- TEST BYTE COMPARES ON SEQUENTIAL ODD/EVEN ADDRESSES
2480	PC 24522 = LOW TO HIGH IN SAME WORD FAILS.
2483	PC 24536 = HIGH TO LOW IN SAME WORD FAILS.
2486	PC 24552 = HIGH TO LOW IN DIFFERENT WORDS FAILS.

TABLE OF CONTENTS

2489 PC 24566 = EVEN TO EVEN FAILED.
2492 PC 24602 = ODD TO ODD FAILED.
2495 PC 24616 = LOW TO HIGH IN SAME WORD FAILED.
2498 PC 24632 = HIGH TO HIGH IN SAME WORD FAILED.
2501 PC 24646 = EVEN TO ODD FAILED.
2504 T72 -- TEST THAT BUS TIME-OUT TRAPS TO THE 'RESTART' ADDRESS
2515 PC 24704 = BUS TIME-OUT DIDN'T TRAP AT ALL.
2521 PC 24726 = STACKED PC INCORRECT ON TIME-OUT TRAP.
2525 PC 24736 = STACKED PSW INCORRECT ON TIME-OUT TRAP.
2529 T73 -- TEST THAT AN 'ILLEGAL' INSTRUCTION TRAPS TO 4
2546 PC 25022 = ILLEGAL JMP/JSR DIDN'T TRAP.
2551 PC 25034 = PSW INCORRECT AFTER ILLEGAL INSTR TRAP TO 4.
2554 PC 25046 = STACK POINTER INCORRECT.
2559 PC 25066 = INCORRECT PC SAVED ON STACK.
2562 PC 25102 = INCORRECT PSW SAVED ON STACK.
2568 T74 -- TEST THAT A RESERVED (UNDEFINED) INSTRUCTION TRAPS TO 10
2575 PC 25154 = RESERVED OPCODE DIDN'T TRAP.
2580 PC 25166 = NEW PSW INCORRECT AFTER TRAP TO 10.
2583 PC 25200 = STACK POINTER INCORRECT.
2586 PC 25212 = INCORRECT PC SAVED ON STACK.
2589 PC 25226 = INCORRECT PSW SAVED ON STACK.
2593 T75 -- TEST THAT A 'BPT' INSTRUCTION TRAPS TO 14
2600 PC 25272 = BPT DIDN'T TRAP.
2605 PC 25304 = PSW INCORRECT AFTER BPT TRAP TO 14.
2608 PC 25316 = STACK POINTER INCORRECT.
2611 PC 25330 = INCORRECT PC SAVED ON STACK.
2614 PC 25344 = INCORRECT PSW SAVED ON STACK.
2618 T76 -- TEST THAT AN 'IOT' INSTRUCTION TRAPS TO 20
2625 PC 25410 = IOT DID NOT TRAP.
2630 PC 25422 = PSW INCORRECT AFTER IOT TRAP TO 20.
2633 PC 25434 = STACK POINTER INCORRECT.
2636 PC 25446 = INCORRECT PC SAVED ON STACK.
2639 PC 25462 = INCORRECT PSW SAVED ON STACK.
2643 T77 -- TEST THAT 'POWER-FAIL' TRAPS TO 24 (REQUIRES Q-BUS EXERCISER)
2666 PC 25556 = POWER-FAIL DID NOT TRAP.
2673 PC 25574 = PSW INCORRECT AFTER PWR-FAIL TRAP TO 24.
2676 PC 25606 = STACK POINTER INCORRECT.
2679 PC 25620 = INCORRECT PC SAVED ON STACK.
2682 PC 25634 = INCORRECT PSW SAVED ON STACK.
2687 T100 -- TEST THAT AN 'EMT' INSTRUCTION TRAPS TO 30
2706 PC 25720 = PSW INCORRECT AFTER EMT TRAP TO 30.
2709 PC 25732 = STACK POINTER INCORRECT.
2712 PC 25744 = INCORRECT PC SAVED ON STACK.
2715 PC 25760 = INCORRECT PSW SAVED ON STACK.
2718 T101 -- TEST THAT A 'TRAP' INSTRUCTION TRAPS TO 34
2736 PC 26036 = PSW INCORRECT AFTER TRAP TO 34.
2739 PC 26050 = STACK POINTER INCORRECT.
2742 PC 26062 = INCORRECT PC SAVED ON STACK.
2745 PC 26076 = INCORRECT PSW SAVED ON STACK.
2748 T102 -- TEST THAT THE 'T' BIT CAUSES A TRACE TRAP
2760 PC 26142 = TRACE BIT DID NOT TRAP.
2765 PC 26154 = NEW PSW INCORRECT AFTER TRACE TRAP TO 14.
2768 PC 26166 = TRACE TRAP HAPPENED AT WRONG TIME.
2771 PC 26202 = T BIT WASN'T SAVED IN STACKED PSW.
2781 PC 26232 = TRACE TRAP DIDN'T HAPPEN.
2786 PC 26244 = NEW PSW INCORRECT AFTER TRACE TRAP TO 14.

TABLE OF CONTENTS

2789 PC 26256 = RTT DIDN'T ALLOW NEXT INSTRUCTION TO EXECUTE.
2792 PC 26272 = T BIT WASN'T SAVED IN STACKED PSW.
2795 T103 -- TEST THAT TRACE TRAP ON A TRAP IS INHIBITED
2807 PC 26352 = NEITHER TRAP OCCURRED ???.
2812 PC 26364 = TRACE TRAP ON TRAP WASN'T INHIBITED.
2816 T104 -- TEST THAT RESET HAS NO EFFECT ON A TRACE TRAP
2827 PC 26442 = BUS-RESET DISABLED OR INHIBITED TRACE TRAP.
2832 T105 -- TEST THAT ODD ADDRESS TRAPS ARE NOT IMPLEMENTED
2837 PC 26504 = JUMP TO AN ODD ADDRESS DID NOTHING.
2839 PC 26514 = JMP TO AN ODD ADDRESS TIMED OUT.
2842 T106 -- TEST THE CPU TYPE INSTRUCTION (MFPT)
2851 PC 26562 = CC BITS CHANGED ON MFPT INSTRUCTION.
2854 PC 26574 = RETURNED VALUE INCORRECT ON MFPT.
2856 PC 26602 = MFPT TRAPPED TO 10.
2859 T107 -- TEST EXTERNAL Q-BUS INTERRUPT (REQUIRES Q-BUS EXERCISER)
2877 PC 26706 = Q-BUS INTERRUPTS AT WRONG LEVEL.
2882 PC 26726 = Q-BUS INTERRUPT NOT RECEIVED.
2888 T110 -- TEST THE 'BEVNT' INTERRUPT (REQUIRES Q-BUS EXERCISER)
2902 PC 27006 = BEVNT INTERRUPT LEVEL INCORRECT.
2907 PC 27030 = BEVNT INTERRUPT NOT RECEIVED.
2913
2915 SECTION 3 -- LOCAL 2K RAM TESTS.
2921 T111 -- LOCAL 2K RAM ADDRESS TEST
2940 PC 27142 = RAM DATA INCORRECT AT ADDRESS IN R0.
2955 PC 27176 = RAM DATA (SWAPPED) INCORRECT AT ADDRESS IN R0.
2959 PC 27206 = BUS TIME-OUT -- RAM ADDRESS IN R0.
2962 T112 -- LOCAL 2K RAM DATA TEST
2987 PC 27272 = INITIAL BACKGROUND INCORRECT AT ADDRESS IN R0.
2999 PC 27330 = FLOATING DATA INCORRECT AT ADDRESS IN R0.
3009 PC 27350 = BACKGROUND CHANGED AT ADDRESS IN R0.
3017 PC 27376 = BUS TIME-OUT -- RAM ADDRESS IN R0.
3020 T113 -- OPTIONAL DMA TEST (REQUIRES Q-BUS EXERCISER)
3044 PC 27510 = NXM ERROR IN QBX ON DATI (MEM TO QBX).
3047 PC 27522 = DATA INCORRECT IN QBX ON DATI (MEM TO QBX).
3059 PC 27576 = NXM ERROR IN QBX ON DATO (QBX TO MEM).
3062 PC 27610 = DATA INCORRECT IN TEMP2 ON DATO (QBX TO MEM).
3069 PC 27624 = BUS TIME-OUT -- QBX (R4) OR RAM (R1).
3074
3076 SECTION 4 -- LOCAL PROM/RAM TESTS (EMPTY SOCKETS).
3082 T114 -- LOCAL PROM/RAM ADDRESS TEST
3102 PC 27734 = WRITE TO ROM ALTERED DATA ?????.
3108 PC 27746 = BUS TIME-OUT -- ROM ADDRESS IN R0.
3113
3115 SECTIONS 5 AND 6 -- SERIAL LINES 1 AND 2 TESTS.
3150 T115 -- TEST SERIAL LINE REGISTER ADDRESSES
3174 PC 30152 = BUS TIME-OUT ON SLU1 ADDRESS.
3176 PC 30162 = BUS TIME-OUT ON SLU2 ADDRESS.
3178 PC 30172 = UNEXPECTED RCVR INTERRUPT..
3180 PC 30202 = UNEXPECTED XMTR INTERRUPT..
3191 T116 -- TEST THAT BUS-RESET CLEARS THE RIGHT BITS
3211 PC 30342 = BUS-RESET DIDN'T CLEAR RCV-IE BIT IN RCSR.
3214 PC 30354 = BUS-RESET DIDN'T CLEAR XIE, MAINT, AND/OR XBRK.
3217 PC 30366 = BUS-RESET CLEARED WRONG BITS IN XCSR.
3220 PC 30400 = BUS-RESET CLEARED BITS IN XBUF.
3223 T117 -- TEST THAT ALL UNDEFINED BITS ARE ZERO
3230 PC 30442 = UNDEFINED BITS SET IN RCVR STATUS.

TABLE OF CONTENTS

3233 PC 30454 = UNDEFINED BITS SET IN RCVR DATA BUFFER.
3236 PC 30466 = UNDEFINED BITS SET IN XMTR STATUS.
3239 PC 30500 = UNDEFINED BITS SET IN XMTR DATA BUFFER.
3242 PC 30512 = XMITTER NOT READY.
3246
3248

TRANSMITTERS

3250 T120 -- FLOAT 1 AND 0 THRU THE XMIT DATA BUFFER
3257 PC 30552 = FLOATING 1 OR 0 INCORRECT IN XBUF.
3261 T121 -- TEST THAT 'XMIT-BRK' CAN BE SET AND CLEARED
3271 PC 30636 = XMIT-BRK BIT FAILED TO SET.
3274 PC 30650 = XMIT-BRK BIT FAILED TO CLEAR.
3277 T122 -- TEST THAT 'XMIT-MAINT' CAN BE SET AND CLEARED
3285 PC 30720 = XMIT-MAINT BIT FAILED TO SET.
3288 PC 30732 = XMIT-MAINT BIT FAILED TO CLEAR.
3291 T123 -- TEST THAT 'XMIT-PBRE' CAN BE SET AND CLEARED
3299 PC 31002 = XMIT-PBRE BIT FAILED TO SET.
3302 PC 31014 = XMIT-PBRE BIT FAILED TO CLEAR.
3305 T124 -- TEST THAT 'XMIT-PBRO' CAN BE SET AND CLEARED
3313 PC 31064 = XMIT-PBRO BIT FAILED TO SET.
3316 PC 31076 = XMIT-PBRO BIT FAILED TO CLEAR.
3319 T125 -- TEST THAT 'XMIT-PBR1' CAN BE SET AND CLEARED
3327 PC 31146 = XMIT-PBR1 BIT FAILED TO SET.
3330 PC 31160 = XMIT-PBR1 BIT FAILED TO CLEAR.
3333 T126 -- TEST THAT 'XMIT-PBR2' CAN BE SET AND CLEARED
3341 PC 31230 = XMIT-PBR2 BIT FAILED TO SET.
3344 PC 31242 = XMIT-PBR2 BIT FAILED TO CLEAR.
3347 T127 -- TEST THAT 'XMIT-IE' CAN BE SET AND CLEARED
3357 PC 31322 = XMIT-IE BIT FAILED TO SET.
3360 PC 31334 = XMIT-IE BIT FAILED TO CLEAR.
3363 T130 -- TEST THAT 'XMIT-RDY' CAN BE SET AND CLEARED
3370 PC 31366 = XMIT-RDY BIT FAILED TO SET.
3375 PC 31412 = XMIT-RDY BIT FAILED TO CLEAR.
3379 PC 31426 = XMIT-RDY BIT FAILED TO SET AFTER XMITTING CHARACTER.
3382 T131 -- TEST THAT WE CAN INTERRUPT ON 'XMT-RDY'
3394 PC 31510 = XMT-RDY INTERRUPT AT WRONG LEVEL (TOO HIGH).
3400 PC 31534 = XMT-RDY INTERRUPT NOT RECEIVED AT CPU LEVEL 0.
3413 PC 31570 = XMT INTERRUPT ACK FAILED TO REMOVE THE REQUEST.
3418 T132 -- TEST THAT EACH XMIT BAUD RATE SETS A DIFFERENT SPEED
3472 PC 31736 = XMT-RDY NEVER SET AT PBR IN RO.
3489 PC 32000 = SPEED DIFFERENTIAL ERROR.
3493

RECEIVERS

3495
3497 T133 -- TEST THAT 'RCV-IE' CAN BE SET AND CLEARED
3505 PC 32052 = RCV-IE BIT FAILED TO SET.
3508 PC 32064 = RCV-IE BIT FAILED TO CLEAR.
3511 T134 -- TEST THAT 'RCV-ACT' AND 'RCV-DONE' CAN SET AND CLEAR
3533 PC 32156 = RCV-ACT AND/OR RCV-DUN ARE SET IN ERROR.
3541 PC 32206 = RCV-ACT FAILED TO SET.
3544 PC 32220 = RCV-DUN SET WHEN RCV-ACT WAS SET.
3549 PC 32240 = RCV-ACT SET BUT FAILED TO CLEAR.
3554 PC 32260 = RCV-DUN FAILED TO SET WHEN RCV-ACT CLEARED.
3559 PC 32302 = RCV-DUN FAILED TO CLEAR AFTER READING DATA.
3562 T135 -- TEST THE 'RCV-BRK', 'RCV-ERR', AND 'FR-ERR' BITS
3582 PC 32410 = NO RCV-DUN AFTER XMIT WITH XMT-BRK SET.
3591 PC 32440 = HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6.
3599 PC 32470 = DOUBLE HALT TRAP RECEIVED.

TABLE OF CONTENTS

3602 PC 32500 = HALT TRAP INTERRUPT NOT MASKED BY CPU LEVEL 7.
3610 PC 32522 = RCV-BRK BIT DIDN'T SET.
3615 PC 32542 = RCV-ERR AND/OR FR-ERR BITS DIDN'T SET.
3618 PC 32552 = RECEIVED DATA NON-ZERO ON 'BRK'.
3632 PC 32624 = RCV-ERR, FR-ERR, AND/OR BRK BITS DIDN'T CLEAR.
3635 PC 32636 = DATA INCORRECT AFTER 'BRK' SEQUENCE.
3638 T136 -- TEST THAT 'RCV-ERR' AND 'OR-ERR' BITS SET AND CLEAR
3654 PC 32736 = RCV-ERR AND/OR OR-ERR NOT SET ON FORCED OVER-RUN.
3662 PC 32770 = RCV-ERR AND/OR OR-ERR BITS FAILED TO CLEAR.
3665 T137 -- TEST THAT WE CAN INTERRUPT ON 'RCV-DUN'
3679 PC 33066 = RCV-DUN INTERRUPT AT WRONG LEVEL (TOO HIGH).
3685 PC 33112 = RCV-DUN INTERRUPT NOT RECEIVED AT.
3698 PC 33146 = INTERRUPT ACK FAILED TO REMOVE THE REQUEST.
3704 T140 -- FLOAT 1 AND 0 THRU THE 'MAINT' DATA LOOP (INTERNAL)
3728 PC 33274 = RCVD DATA INCORRECT, MAINT DATA PATH FAILURE.
3732 T141 -- FLOAT 1 AND 0 THRU THE LOOP-BACK (EXTERNAL, SLU2 ONLY)
3750 PC 33400 = NO RCV-DUN -- IS LOOP CONNECTOR INSTALLED ???
3762 PC 33442 = RCVD DATA INCORRECT, EXTERNAL DATA PATH FAILURE.
3774

SECTION 7 -- PARALLEL I/O PORT (8255) TESTS.

3776 T142 -- TEST PIO REGISTER ADDRESSES
3795 PC 33634 = BUS TIME-OUT ON PIO ADDRESS.
3820 PC 33644 = UNEXPECTED PORT B INTERRUPT.
3822 PC 33654 = UNEXPECTED PORT A INTERRUPT.
3824 T143 -- TEST THAT RESET INVOKES MODE 0 AND CLEARS THE CH1:
3827 PC 33714 = PORT A RESET STATE INCORRECT.
3840 T144 -- TEST MODE 1 I/O THRU THE LOOP CONNECTOR (FLAG MODE)
3843 PC 33772 = PORT C MODE 1 STATUS INCORRECT.
3853 PC 34024 = OUTRDY STILL SET AFTER OUTPUT XFER.
3862 PC 34044 = STATUS INCORRECT AFTER OUTPUT XFER.
3867 PC 34066 = STATUS INCORRECT AFTER INPUT XFER.
3872 PC 34076 = DATA INCORRECT AFTER OUT/IN EXCHANGE.
3875 T145 -- TEST THE PORT B (OUTPUT) INTERRUPT
3879 PC 34142 = PORT B INTERRUPT LEVEL INCORRECT.
3888 PC 34160 = STATUS INCORRECT WITH PORT B INTERRUPT PENDING.
3892 PC 34176 = PORT B INTERRUPT NOT RECEIVED.
3897 PC 34224 = DOUBLE PORT B INTERRUPT.
3906 PC 34262 = PORT B INT REQUEST DIDN'T CLEAR AFTER WRITE.
3915 T146 -- TEST THE PORT A (INPUT) INTERRUPT
3923 PC 34344 = STATUS INCORRECT.
3930 PC 34374 = PORT A INTERRUPT LEVEL INCORRECT.
3939 PC 34412 = STATUS INCORRECT WITH PORT A INTERRUPT PENDING.
3944 PC 34430 = PORT A INTERRUPT NOT RECEIVED.
3949 PC 34456 = DOUBLE PORT A INTERRUPT.
3958 PC 34512 = PORT A INT REQUEST DIDN'T CLEAR AFTER READ.
3967 T147 -- TEST THAT D PRIORITY IS HIGHER THAN A
3971 PC 34614 = NEITHER ONE INTERRUPTED -- WHAT THE HELL !!.
3986 PC 34624 = A INTERRUPTED BEFORE B -- BAD NEWS !!.
3989 PC 34650 = A INT NOT RECEIVED AFTER B INT.
3996

4008
4010 END OF PASS ROUTINES.
4110 TRAP HANDLER

```

1      .TITLE CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
2
3      .REV B -- FIX RAM ADDRESS TEST.
4              FIX SLU SPEED DIFFERENTIAL TEST.
5              FIX PIO RESET TEST.
6
7      .PART 1 -- TEST THE T11 INSTRUCTION SET.
8      .PART 2 -- TEST THE T11 TRAPS AND INTERRUPTS.
9      .PART 3 -- TEST THE LOCAL 2K RAM MEMORY.
10     .PART 4 -- TEST THE LOCAL ROM/RAM MEMORY (EMPTY SOCKETS).
11     .PART 5 -- TEST SERIAL LINE UNIT 1 (DEC DC319 DLART).
12     .PART 6 -- TEST SERIAL LINE UNIT 2 ( DITTO ).
13     .PART 7 -- TEST THE PARALLEL I/O INTERFACE (INTEL 8255).
14
15     .ABS
16     .NLIST MD,MC,CND,TUC
17     .LIST ME
18
19     000076 PRGSI= <^H<$LSTAD>>+1 ; PROGRAM SIZE IN 1/8 K UNITS.
20
21     000176 .SBTTL SWITCH REGISTER OPTIONS
22             SWR= 176 ; THE STANDARD SOFT SWITCH REGISTER ADDRESS.
23             SWR<15> 100000 HALT ON ERROR.
24             SWR<14>
25             SWR<13> 020000 INHIBIT ERROR MESSAGES.
26             SWR<12> 010000 PRINT ERROR SUMMARY AT END-PASS.
27             SWR<11>
28             SWR<10>
29             SWR<09> 001000 LOOP ON ERROR.
30             SWR<08>
31             SWR<02:00> N RUN SEGMENT N (1 THRU 7) ONLY.
32     000000 *****
33             TN$= 0 ; INIT TEST NUMBER SEQUENCE.
34
35     .
36     .T11 CPU EQUATES.
37
38     000006 R6= SP
39     000007 R7= PC
40     000004 ERRVEC= 4 ; ILLEGAL INSTRUCTION VECTOR.
41     000010 RESVEC= 10 ; RESERVED OPCODE VECTOR.
42     000014 BPTVEC= 14 ; BREAKPOINT VECTOR...
43     000014 TRTVEC= BPTVEC ; ...TRACE TRAP TOO.
44     000020 IOTVEC= 20 ; IOT VECTOR.
45     000024 PWRVEC= 24 ; POWER FAIL VECTOR.
46     000030 EMTVEC= 30 ; EMT VECTOR.
47     000034 TRPVEC= 34 ; TRAP VECTOR.
48     000340 PRI7= 340
49     000300 PRI6= 300
50     000240 PRI5= 240
51     000200 PRI4= 200
52     000140 PRI3= 140
53     000100 PRI2= 100
54     000040 PRI1= 040
55     000000 PRI0= 000
56
57     000010 RESVD= 000010 ; RESERVED (UNDEFINED) OPCODE.

```

82	000007	MFPT= 000007	; MFPT OPCODE (NEW INSTRUCTION)...
83	000004	KXT11= 4	;...SHOULD RETURN #4 IN R0.
84	000254	CLNZ= CLN!CLZ	
85	000263	SEVC= SEV!SEC	
86	000273	SENV= SEN!SEV!SEC	
87	000260	NOP1= 260	
88	000401	SKP1= BR+1	
89	000402	SKP2= BR+2	
90	000403	SKP3= BR+3	
91	104400	CHECKCC= TRAP	; CC BIT CHECK CALL.
92	104420	BEGINTEST= TRAP+20	
93	104421	CHECKSWR= TRAP+21	; CHECK FOR <^G>.
94	104422	GETSWR= TRAP+22	; GET NEW SWR.
95	104423	ERROR= TRAP+23	; ERROR CALL.
96			
97		.SBTTL KXT-11 (FALCON) DEFAULT CONFIGURATION	
98			
99	172000	START= ODT	; ON POWER-UP, START MACRO-ODT IN PROM...
100	172004	RESTART= ODTRE	;...TRAP TO 'RESTART' ON TIME-OUT OR HALT...
101	000000	EHALT= HALT	;...AND HALT ON PROGRAM ERROR.
102			
103	170000	ROMADR= 170000	; LOCAL ROM SOCKETS STRAPPED...
104	002000	ROMSIZ= 1024.	;...TO RESPOND AT 30K TO 31K...
105			;...MACRO-ODT PROM MUST BE INSTALLED.
106	172000	ODT= 172000	; IT STARTS HERE...
107	172004	ODTRE= 172004	;...AND RESTARTS HERE.
108	170000	ODTBRK= 170000	; 'BREAK' HALT IS SERVICED HERE...
109	000140	ODTIN= BHALT	;...AND THE VECTOR SERVES AS A HANDY FLAG.
110			
111	160010	RAMADR= 160010	; LOCAL 2K RAM STRAPPED...
112	003764	RAMSIZ= 2048.-4.-8.	;...TO RESPOND AT 28K(+4) TO 30K...
113			;...1ST 4 WORDS ARE NON-EX SO THAT...
114			;...CONVENTIONAL SIZERS WILL WORK RIGHT...
115			;...AND LAST 8 WORDS BELONG TO MACRO-ODT...
116	167776	TRAP4= 167776	;... (THE LAST BEING THE TRAP4 EMULATE FLAG).
117			
118	000100	BEVNT= 100	; 'LINE CLOCK' (ON SLU2) VECTOR.
119	000140	BHALT= 140	; 'BREAK' (ON SLU1) OR Q-BUS 'BHALT' VECTOR.
120			
121	177560	SLU1= 177560	; SLU (DLART) 1 (CONSOLE TTY)...
122	000060	SLU1V= 60	;...1ST VECTOR (OF 2)...
123	000200	SLU1P= PRI4	;...AND PRIORITY.
124	176540	SLU2= 176540	; SLU (DLART) 2...
125	000120	SLU2V= 120	;...1ST VECTOR (OF 2)...
126	000240	SLU2P= PRI5	;...AND PRIORITY.
127			
128	176200	PIOA= 176200	; PIO PORT A (INPUT)...
129	000134	PIOAV= 134	;...AND VECTOR.
130	176202	PIOB= 176202	; PIO PORT B (OUTPUT)...
131	000130	PIOBV= 130	;...AND VECTOR.
132	000240	PIOP= PRI5	; INTERRUPT PRIORITY (FOR BOTH).
133	176204	PIOC= 176204	; PIO PORT C (PORT CONTROL)...
134	176206	PIOM= 176206	;...AND MODE CONTROL.
135			
136	174020	QBXCsr= 174020	; Q-BUS EXERCISER CSR1 ADDRESS (OPTIONAL)...
137	000110	QBxVEC= 110	;...AND VECTOR (AT PRI4).

;; NOW, EQUATE SOME GENERAL REGISTERS.

ADR= 1000 ; INSTRUCTION TEST SCRATCH LOCATIONS.

ADR1= ADR+2

ADR2= ADR+4

TEMP= ADR+6

TEMP1= ADR+10

TEMP2= ADR+12

TEMP3= ADR+14

K0= ADR+16

K1= ADR+20

K2= ADR+22

K3= ADR+24

K4= ADR+26

K5= ADR+30

K6= ADR+32

K7= ADR+34

QBXA= ADR+36 ; Q-BUS EXER CSR ADDRESS (IF INSTALLED).

SPD0= K0 ; DLART, VARIABLE SPEED COUNTERS.

SPD1= K1

SPD2= K2

SPD3= K3

SPD4= K4

SPD5= K5

SPD6= K6

SPD7= K7

DFPBK= ADR+40 ; DEFAULT PBR BITS...
 DFSPD= ADR+42 ; ...AND THE CORRESPONDING SPEED COUNTER.

RCSR= ADR+44 ; DLART REGISTER POINTERS.

RBUF= ADR+46

XCSR= ADR+50

XBUF= ADR+52

RVEC= ADR+54

XVEC= ADR+56

PPA= ADR+60 ; PARALLEL I/O REGISTER POINTERS.

PPB= ADR+62

PPC= ADR+64

PPCW= ADR+66

PPAV= ADR+70

PPBV= ADR+72

;;
 ;; >> REMAINDER OF 1ST K IS OUR STACK.
 ;; //

STACK= 3776

;; NOW THE ENTIRE 2N. K OF RAM IS USED AS AN 'ERROR LOG'.
 ;; ALL ERRORS ARE LOGGED AS THEY OCCUR.

ERRCNT= 4000 ; THE 1ST WORD IS A TOTAL ERROR COUNT...

TPASS= 4002 ; ...THEN A 'TOTAL' PASS COUNT...

GPASS= 4004 ; ...AND A 'CONSECUTIVE-GOOD' PASS COUNT.

EPC1BL= 4006 ; THE REST FORMS A TABLE OF 2 WORD ENTRIES...

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).

CNKXAA.P11 17-MAR-82 14:02

KXT-11 (FALCON) DEFAULT CONFIGURATION

K 2

MACY11 306(1063) 17-MAR-82 14:03 PAGE 1-3
SEQ 0023

194
195
196

007776

LOGEND= 7776

....SHOWING ERROR PC (WD1) AND THE NUMBER...
....OF OCCURRENCES OF THAT ERROR (WD2).
; END OF ERROR LOG TABLE.

```

198 .SBTTL INITIAL START AND CLEAR VECTORS.
199
200 : ON INITIAL START, WE'LL ATTEMPT TO INITIALIZE THE Q-BUS MEMORY
201 : VECTOR SPACE WITH A TRAP CATCHER IN THE TRADITIONAL STYLE.
202 : LOCATIONS 0 THRU 776 WILL BE COATED WITH:
203
204 : .WORD .+2
205 : .WORD EHALT
206
207 : WE HAVE TO PRESUME THAT THE T11 IS CAPABLE OF EXECUTING THE
208 : FEW BASIC FUNCTIONS REQUIRED TO SET THE TRAP CATCHERS.
209 : IF NOT -- THEN HOW DID WE EVER GET STARTED IN THE FIRST PLACE ???
210
211 : IN THE ENGINEERING AND FIELD VERSIONS, WE HAVE A MACRO-ODT IN ROM
212 : WHICH PROVIDES 'BUS TIME-OUT' AND 'HALT' EMULATION.
213 : ON A RESTART TRAP, WE'LL REMAIN IN ODT IF THE STACKED PC POINTS
214 : TO A 'HALT' OPCODE. OTHERWISE, WE'LL VECTOR THRU LOC 4.
215
216 : 'BHALT' AND 'BREAK' VECTOR THRU 140 TO THE ODT BREAK HANDLER.
217
218 : .= 10000 ; PROGRAM STARTS AT 2K.
219 010000 010000
220 010000 000240
221 010002 013746 000030
222 010006 013746 000040
223 010012 013746 000042
224 010016 012700 000002
225 010022 005001
226 010024 012702 000200
227 010030 01002
228 010032 012721 000000
229 010036 022020
230 010040 077205
231 010042 012637 000042
232 010046 012637 000040
233 010052 012637 000030
234 010056 012737 000137 000200
235 010064 012737 010072 000202
236
237 : IF WE GOT THIS FAR WE'RE IN PRETTY GOOD SHAPE !.!.!
238 : .SBTTL
239 : .SBTTL RESTART
240
241 010072 012700 001000
242 010076 005020
243 010100 020027 010000
244 010104 103774
245 010106 012706 003776
246 010112 012737 170000 000140
247 010120 012737 000340 000142
248 010126 012737 000001 167776
249 010134 010132
250 010134 106427 000000

```

```

BEG:N:
INIT:  NOP ; RESET ???
      MOV 3#30,-(SP) ; SAVE XXDP EMT VECTOR...
      MOV 3#40,-(SP) ; ...LOAD DEV ID AND UNIT...
      MOV 3#42,-(SP) ; ...AND CHAIN RETURN.
      MOV #2,R0 ; COAT THE ENTIRE VECTOR AREA...
      CLR R1 ; ...WITH A TRADITIONAL TRAP CATCHER.
1$:   MOV #200,R2
      MOV R0,(R1)+ ;
      MOV #EHALT,(R1)+ ; +2
      CMP (R0)+,(R0)+ ; EHALT
      SOB R2,1$
      MOV (SP)+,3#42 ; RESTORE CHAIN RETURN...
      MOV (SP)+,3#40 ; ...LOAD DEV...
      MOV (SP)+,3#30 ; ...AND EMT VECTOR.
      MOV #137,3#200 ; SET A STANDARD 'RESTART' ADDRESS.
      MOV #INIT1,3#202
      ;
      ; IF WE GOT THIS FAR WE'RE IN PRETTY GOOD SHAPE !.!.!
      .SBTTL
      .SBTTL RESTART
      ;
INIT1: MOV #1000,R0 ; '200G' RESTARTS HERE.
1$:   CLR (R0)+ ; CLEAR ALL THE REST OF RAM UP TO...
      CMP R0,#BEGIN ; ...THE PROGRAM START POINT.
      BLO 1$
      MOV #STACK,SP ; SET A STACK POINTER.
      MOV #ODTBRK,3#BHALT ; ENABLE 'BREAKS' TO ODT...
      MOV #PRI7,3#BHALT+2 ; ...THRU VECTOR 140.
      MOV #1,3#TRAP4 ; ENABLE ODT'S 'TRAP 4' EMULATOR.
.PRINT -2 ; *** DEBUG, 10000 ***
MTPS #PRI0 ; AND LOWER THE CPU LEVEL.

```

```

252 .SBTTL QUICK-VERIFY EMT AND TRAP.
253
254 : NOW TRAPS (104400) AND EMTS (104000) ARE USED EXTENSIVELY FOR
255 : ERROR HANDLING, AND TTY SERVICE. THE EMT HAS TO BE FUNCTIONAL
256 : OTHERWISE, THE XXDP+ MONITOR WOULD HAVE CRASHED. TRAPS ARE
257 : UNKNOWN AT THIS POINT. SO TO BE SAFE, WE'LL DO A QUICK VERIFY
258 : OF BOTH THE TRAP AND THE EMT OPCODES HERE.
259 : ADDITIONAL TESTING WILL BE ATTEMPTED IN SEGMENT 2.
260
261 010140 016746 167664 EMTTRP: MOV EMTVEC, -(SP) ; SAVE MONITORS EMT VECTOR.
262 010144 012767 010204 167656 MOV #4$, EMTV C ; SET EMT...
263 010152 012767 000036 167654 MOV #TRPVEC+2, TRPVEC ; ...AND RESET TRAP VECTORS.
264 010160 012700 104000 MOV #EMT+0, R0 ; TEST EMT'S FIRST...
265 010164 000402 BR 2$
266 010166 012700 104400 1$: MOV #TRAP+0, R0 ; ...THEN TRAP'S.
267 010172 010067 000000 2$: MOV R0, 3$
268 010176 104000 3$: EMT+0 ; EXECUTE EMT/TRAP.
269 010200 000240 NOP ; DIDN'T WORK -- BIG TROUBLE...
270 010202 000000 HALT ; ... (R0) HOLDS THE OFFENDER !!!
276 010204 012716 010216 4$: MOV #5$, (SP) ; EMT/TRAP SEEMS OK, TWEAK RETURN PC...
277 010210 000002 RTI ; ...AND TEST THE 'RTI' TOO.
278 010212 000240 NOP ; W-T-F ???
279 010214 000000 HALT ; RTI DOESN'T WORK !!!
285 010216 105200 5$: INCB R0 ; WE'RE HERE IF EVERYTHING WORKED...
286 010220 001364 BNE 2$ ; ...BUMP THE OPCODE AND LOOP 'TIL DONE.
287 010222 020027 104400 CMP R0, #TRAP ; OPCODE = TRAP ??
288 010226 001407 BEQ 6$ ; WE'RE DONE IF SO.
289 010230 012767 000032 167572 MOV #EMTVEC+2, EMTVEC ; OTHERWISE, RESET EMT...
290 010236 012767 010204 167570 MOV #4$, TRPVEC ; ...AND SET TRAP VECTORS...
291 010244 000750 BR 1$ ; ...AND GO 'ROUND ONCE.
292
293 010246 012667 167556 6$: MOV (SP)+, EMTVEC ; DONE, RESTORE EMT FOR MONITOR CALLS...
294 010252 012767 035336 167554 MOV #TRAPHAN, TRPVEC ; ...AND SET-UP OUR OWN TRAP HANDLER.
295 010260 012700 010300 MOV #HELLO, R0
296 010264 104003 EMT+3 ; 'GREETINGS'.
297 010266 104422 GLTSWR ; GET INITIAL SWITCH SETTING...
298 010270 016700 025370 MOV INSEG, R0 ; SINGLE SEGMENT SELECTED ??
299 010274 001420 BEQ DOALL ; ...IF NOT, RUN 'EM ALL...
300 010276 010007 MOV R0, PC ; ...ELSE, GO THERE.
301
302 010300 005015 045516 040530 HELLO: .ASCIZ <15><12>'NKXAA0 KXT-11 DIAGNOSTIC'<15><12>
303 010336 .EVEN
304 .SBTTL
  
```

```
306 .SBTTL SECTION 1 -- T11 BASIC INSTRUCTION TESTS.
307
308 DOALL:
309 010336 012767 036202 025634 S1: MOV #SEG1,SEGN ; SEGMENT 1 TEXT.
310
311 ;*****
312 (2) .SBTTL T1 -- BRANCHES, FORWARD, BACKWARD, AND CONDITIONAL
313 (2) ;*****
314 (2) 010344 104420 000001 T1: BEGINTEST, 1 ;; CHECK SWITCH REGISTER OPTIONS.
315 010350 000405 BRANCH: BR 2$
316 010352 000405 1$: BR 3$
317 010354 104423 010344 4$: MOV #5$,PC ;; *** FORWARD BRANCH FAILED ***
318 010360 012707 010374 2$: BR 1$
319 010366 000774 3$: BR 4$
320 010370 104423 010344 5$: ERROR, T1 ;; *** BACKWARD BRANCH FAILED ***
321 010374 000257 NOBIT: CCC ; ZERO CONDITION CODES, NZVC=0000
322 010376 103414 BCS 1$
323 010400 102413 BVS 1$
324 010402 001412 BEQ 1$
325 010404 100411 BMI 1$
326 010406 000260 NOP1 ; CHECK NOP1 INSTRUCTION (OPCODE 260).
327 010410 103407 BCS 1$
328 010412 102406 BVS 1$
329 010414 001405 BEQ 1$
330 010416 100404 BMI 1$
331 010420 002403 BLT 1$
332 010422 003402 BLE 1$
333 010424 101401 BLOS 1$
334 010426 101002 BHI 2$
335 (2) 010430 104423 010344 1$: ERROR, T1 ;; *** BRANCH FAILURE OR CC NOT 0000 ***
336 010434 000270 2$: NBIT: SEN ; NBIT IS SET, NZVC=1000
337 010436 100007 BPL 1$
338 010440 001406 BEQ 1$
339 010442 002005 BGE 1$
340 010444 003004 BGT 1$
341 010446 103403 BCS 1$
342 010450 101402 BLOS 1$
343 010452 103401 BLO 1$
344 010454 003402 BLE 2$
345 (2) 010456 104423 010344 1$: ERROR, T1 ;; *** BRANCH FAILURE OR CC NOT 1000 ***
346 010462 000270 2$: VBIT: SEN ; SET N...
347 010464 000262 SEV ;...AND V, NZVC = 1010
348 010466 102010 BVC 1$
349 010470 001407 BEQ 1$
350 010472 100006 BPL 1$
351 010474 103405 BCS 1$
352 010476 002404 BLT 1$
353 010500 003403 BLE 1$
354 010502 101402 BLOS 1$
355 010504 103401 BLO 1$
```

PC 10456 = BRANCH FAILURE OR CC NOT 1000.

```

357 010506 003002          BGT      2$
358 010510                1$:      ERROR,  T1          ;; *** BRANCH FAILURE OR CC NOT 1010 ***
(2) 010510 104423 010344
359 010514                2$:
360 010514 000270          CBIT:    SEN              ; SET N...
361 010516 000262          SEV              ;...V...
362 010520 000261          SEC              ;...AND C, NZVC=1011
363 010522 001406          BEQ      1$
364 010524 100005          BPL      1$
365 010526 102004          BVC      1$
366 010530 002403          BLT      1$
367 010532 003402          BLE      1$
368 010534 101001          BHI      1$
369 010536 002002          BGE      2$
370 010540                1$:
(2) 010540 104423 010344          ERROR,  T1          ;; *** BRANCH FAILURE OR CC NOT 1011 ***
371 010544                2$:
372 010544 000270          ZBIT:    SEN              ; SET N...
373 010546 000262          SEV              ;...V...
374 010550 000261          SEC              ;...C...
375 010552 000264          SEZ              ;...AND Z, NZVC = 1111.
376 010554 001007          BNE      1$
377 010556 100006          BPL      1$
378 010560 102005          BVC      1$
379 010562 103004          BCC      1$
380 010564 002403          BLT      1$
381 010566 003002          BGT      1$
382 010570 101001          BHI      1$
383 010572 001402          BEQ      2$
384 010574                1$:
(2) 010574 104423 010344          ERROR,  T1          ;; *** BRANCH FAILURE OR CC NOT 1111 ***
385 010600                2$:
386
387 ;*****
(2) .SBTTL      T2 -- SET AND CLEAR ALL CONDITION CODES
(2) ;*****
(2) 010600 104420 000002          T2:    BEGINTEST, 2          ;; CHECK SWITCH REGISTER OPTIONS.
388 010604 000277          ALLCC:    SCC              ; NZVC = 1111
389 010606 104417          CHECKCC+17          ; CHECK ALL BITS SET.
390 010610 104423 010600          ERROR,  T2          ;; *** CC BITS WRONG AFTER SCC ***
391
392 010614 000257          CCC              ; NZVC = 0000
393 010616 104400          CHECKCC+0
394 010620 104423 010600          ERROR,  T2          ;; *** CC BITS WRONG AFTER CCC ***
395
396 ;*****
(2) .SBTTL      T3 -- TEST REGISTER SELECTION
(2) ;*****
(2) 010624 104420 000003          T3:    BEGINTEST, 3          ;; CHECK SWITCH REGISTER OPTIONS.
397 010630 012700 000001          REGS:    MOV      #1,R0          ; LOAD UP R0 THRU R4.
398 010634 012701 000004          MOV      #4,R1
399 010640 012702 000020          MOV      #20,R2
400 010644 012703 000100          MOV      #100,R3
401 010650 012704 000400          MOV      #400,R4
402 010654 012705 001000          MOV      #1000,R5
403 010660 060600          ADD      R6,R0          ; R0 + #STACK...

```

```

404 010662 060500          ADD    R5,R0          ;...+ 1000...
405 010664 060400          ADD    R4,R0          ;...+ 400...
406 010666 060300          ADD    R3,R0          ;...+ 100...
407 010670 060200          ADD    R2,R0          ;...+ 20...
408 010672 060100          ADD    R1,R0          ;...+ 4...
409 010674 020027 005523    CMP    R0,#STACK+1525 ;...SHOULD = THIS.
410 010700 001402          BEQ     1$             ; BR IF OK.
411 010702 104423 010624    ERROR,  T3           ;; *** REGISTER SELECTION FAILURE ***
412 010706
413
414
(2)
(2)
(2) 010706 104420 000004    1$:
415 010712 012700 010726    SBTTL    T4 -- JMP INSTRUCTION -- MODE 1
416 010716 000277          T4:  BEGINTEST, 4      ;; CHECK SWITCH REGISTER OPTIONS.
417 010720 000110          JMP1:  MOV    #1$,R0    ;; TEST JUMP INSTRUCTION MODE 1
418 010722 104423 010706    SCC
419 010726 104417          JMP    (R0)
420 010730 104423 010706    ERROR,  T4           ;; *** MODE 1 JUMP FAILED ***
421 010734 020027 010726    1$:  CHECKCC+17
422 010740 001402          ERROR,  T4           ;; *** CC BITS ALTERED ON JMP ***
423 010742 104423 010706    CMP    R0,#1$
424 010746          BEQ     2$             ; CONTINUE IF R0 IS OK
425
426          ERROR,  T4           ;; *** R0 INCORRECT AFTER JUMP ***
(2)
(2)
(2) 010746 104420 000005    SBTTL    T5 -- JMP INSTRUCTION -- MODES 2 AND 3
427 010752 012700 010766    T5:  BEGINTEST, 5      ;; CHECK SWITCH REGISTER OPTIONS.
428 010756 000277          JMP2:  MOV    #1$,R0    ;; TEST JUMP INSTRUCTION MODE 2
429 010760 000120          SCC
430 010762 104423 010746    JMP    (R0)+
431 010766 104417          ERROR,  T5           ;; *** MODE 2 JUMP FAILED ***
432 010770 104423 010746    1$:  CHECKCC+17
433 010774 020027 010770    ERROR,  T5           ;; *** CC BITS ALTERED AFTER JMP ***
434 011000 001402          CMP    R0,#1$+2      ; IS THERE AUTO INC.?
435 011002 104423 010746    B10    JMP3
436          ERROR,  T5           ;; *** MODE 2 FAILED ON JUMP ***
437 011006 012767 011036 167772 JMP3:  MOV    #3$,TEMP    ; TEST JUMP INSTRUCTION MODE 3
438 011014 012767 011052 167766    MOV    #4$,TEMP+2
439 011022 012700 001006    MOV    #TEMP,R0
440 011026 000277          SCC
441 011030 000130          JMP    3(R0)+
442 011032 104423 010746    ERROR,  T5           ;; *** MODE 3 JUMP FAILED ***
443 011036 027067 000000 000006 3$:  CMP    3(R0),4$    ; IS THERE AUTO INC.?
444 011044 001402          BEQ     4$
445 011046 104423 010746    ERROR,  T5           ;; *** MODE 3 JUMP FAILED ***
446 011052
447
448
(2)
(2)
(2) 011052 104420 000006    SBTTL    T6 -- JMP INSTRUCTION -- MODES 4 AND 5
449 011056 012700 011074    T6:  BEGINTEST, 6      ;; CHECK SWITCH REGISTER OPTIONS.
450 011062 000277          JMP4:  MOV    #3$,R0    ;; TEST JUMP INSTRUCTION MODE 4
          SCC

```

T6 -- JMP INSTRUCTION -- MODES 4 AND 5

```

451 011064 000140          JMP      -(R0)
452 011066 104423 011052    ERROR,   T6          ;; *** DIDN'T JUMP AT ALL ***
453 011072 000402          BR        4$          ;; JUMP SHOULD LAND HERE
454 011074          3$:
(2) 011074 104423 011052    ERROR,   T6          ;; *** MODE 4 JUMP FAILED. ***
455 011100 022700 011072    4$:      CMP      #3$,R0          ;; CHECK R0
456 011104 001402          BEQ      JMP5
457 011106 104423 011052    ERROR,   T6          ;; *** R0 INCORRECT AFTER JUMP ***
458
459 011112 012767 011140 167670 JMP5:  MOV      #3$,TEMP1          ; TEST JUMP INSTRUCTION MODE 5
460 011120 012700 001010          MOV      #TEMP1,R0
461 011124 012767 011144 167654          MOV      #4$,TEMP1-2
462 011132 000150          JMP      @-(R0)
463 011134 104423 011052    ERROR,   T6          ;; *** MODE 5 JUMP DIDN'T ***
464 011140          3$:
(2) 011140 104423 011052    ERROR,   T6          ;; *** AUTO-DECR FAILED ON MODE 5 JUMP ***
455 011144 022700 001006    4$:      CMP      #TEMP1-2,R0          ;; CHECK R0
466 011150 001402          BEQ      5$
467 011152 104423 011052    ERROR,   T6          ;; *** R0 INCORRECT AFTER JMP5 ***
468 011156          5$:

```

 .SBTTL T7 -- JMP INSTRUCTION -- MODES 6 AND 7

T7: BEGINTEST, 7 ;; CHECK SWITCH REGISTER OPTIONS.

```

JMP6:  MOV      #1$,R3          ;; *** MODE 6 JUMP FAILED ***
        JMP      -6(R3)          ;; CHECK R3
1$:    ERROR,   T7
        CMP      R3,#1$+6
        BEQ      2$
        ERROR,   T7          ;; *** R3 WRONG AFTER JMP6 ***
2$:    JMP      3$-4(PC)          ; TEST JUMP INSTRUCTION MODE 6
        ERROR,   T7          ;; *** JUMP FAILED ***
3$:    MOV      #JMP7,R3          ; JUMP SHOULD LAND HERE
        JMP      0(R3)
        ERROR,   T7          ;; *** JUMP FAILED ***

```

```

JMP7:  MOV      #TEMP,R3
        MOV      #1$,R3
        JMP      @R3
        ERROR,   T7          ;; *** MODE 7 JUMP FAILED ***
1$:    MOV      #3$,R3          ; TEST JUMP INSTRUCTION MODE 7
        MOV      #TEMP-4,R0
        JMP      @4(R0)
        ERROR,   T7          ;; *** JUMP FAILED ***
2$:    MOV      #4$,TEMP          ; CONTINUE
        MOV      #TEMP,R0
        JMP      @0(R0)
        ERROR,   T7          ;; *** MODE 7 JUMP FAILED. ***
3$:
4$:

```

 .SBTTL T10 -- JSR AND RTS INSTRUCTIONS

T10: BEGINTEST, 10 ;; CHECK SWITCH REGISTER OPTIONS.

(2) 011316 104420 000010

```
499 011322 012706 003776 JSRTST: MOV #STACK,SP ; SET UP STACK POINTER.
500 011326 000277 SCC
501 011330 004767 000004 JSR PC,2$
502 011334 104423 011316 1$: ERROR, T10 ;: *** JSR INSTRUCTION FAILED ***
(2) 011334 104417 011316 2$: CHECKCC+17 ;: *** CC BITS CHANGED ON JSR ***
503 011340 104423 011316 ERROR, T10 ;: *** CC BITS CHANGED ON JSR ***
504 011342 022706 003774 4$: CMP #STACK-2,SP ; BR IF STACK IS RIGHT.
505 011352 001402 011316 BEQ 5$ ;: *** STACK WRONG AFTER JSR ***
506 011354 104423 011334 5$: ERROR, T10 ; BR IF STACKED PC IS RIGHT.
507 011360 022716 011334 CMP #1$, (SP) ;: *** SP DID NOT HAVE CORRECT RETURN ADDRESS ***
508 011364 001402 BEQ 6$ ; BR IF STACKED PC IS RIGHT.
509 011366 104423 011316 ERROR, T10 ;: *** SP DID NOT HAVE CORRECT RETURN ADDRESS ***
510 011372 012716 011404 6$: MOV #7$, (SP) ; CHANGE RETURN PC.
511 011376 000207 RTS PC ;: *** RTS INSTRUCTION FAILED ***
512 011400 104423 011316 7$: ERROR, T10 ;: *** RTS INSTRUCTION FAILED ***
513 011404 022706 003776 CMP #STACK,SP ; BR IF STACK PROPERLY RESTORED.
514 011410 001402 BEQ 8$ ;: *** SP WAS NOT RESTORED BY RTS INSTRUCTION ***
515 011412 104423 011316 ERROR, T10 ;: *** SP WAS NOT RESTORED BY RTS INSTRUCTION ***
516 011416 104423 011316 8$:
517 011416 104423 011316
518 011416 104423 011316
519 011416 104423 011316
520 011416 104423 011316
521 011416 104423 011316
522 011416 104423 011316
523 011416 104423 011316
(2) 011416 104423 011316
(2) 011416 104423 011316
(2) 011416 104423 011316
524 011422 012706 003776 T11: BEGINTEST, 11 ;: CHECK SWITCH REGISTER OPTIONS.
525 011426 000277 TSTB0: MOV #STACK,SP ;: INIT THE STACK.
526 011430 105000 SCC
527 011432 104404 CLRB R0 ;: CLEAR THE REGISTER
528 011434 104423 011416 CHECKCC+4 ;: CHECK FOR CC = 4
529 011440 105700 ERROR, T11 ;: *** CLRB FAILED ***
530 011442 104404 TSTB R0 ;: CHECK IT
531 011444 104423 011416 CHECKCC+4 ;: CHECK FOR CC = 4
532 011450 112701 000377 ERROR, T11 ;: *** TSTB FAILED. ***
533 011454 104410 MOV #377,R1 ;: LOAD THE REGISTER
534 011456 104423 011416 CHECKCC+10 ;: CHECK FOR CC = 10
535 011462 105701 ERROR, T11 ;: *** INCORRECT CC BITS ***
536 011464 104410 TSTB R1 ;: CHECK IT
537 011466 104423 011416 CHECKCC+10 ;: CHECK FOR CC = 10
538 011466 104423 011416 ERROR, T11 ;: *** INCORRECT CC BITS ***
539 011466 104423 011416
540 011472 104420 000012 T12: BEGINTEST, 12 ;: CHECK SWITCH REGISTER OPTIONS.
541 011476 000277 CMPB0: SCC
542 011500 152702 000377 BISB #377,R2 ;: LOAD REGISTER
543 011504 104411 CHECKCC+11 ;: CHECK FOR CC = 11
544 011506 104423 011472 ERROR, T12 ;: *** INCORRECT CC BITS ***
545 011512 122702 000377 CMPB #377,R2 ;: CHECK COMPARE
546 011516 001402 BEQ 2$ ;: CONTINUE IF OK
547 011520 104423 011472 ERROR, T12 ;: *** BISB OR CMPB INSTRUCTION FAILED ***
548 011524 112700 000077 2$: MOV #77,R0
```

548	011530	120002		CMPB	R0,R2	:	CHECK IT AGAIN
549	011532	100002		BPL	3\$	:	CONTINUE IF OK
550	011534	104423	011472	ERROR,	T12	:	*** CMPB INSTRUCTION FAILED (WRONG CC) ***
551	011540	120200		3\$: CMPB	R2,R0	:	ONCE MORE
552	011542	100402		BMI	4\$	:	CONTINUE IF OK
553	011544	104423	011472	ERROR,	T12	:	*** CMPB INSTRUCTION FAILED (WRONG CC) ***
554	011550	112702	000377	4\$: MOVB	#377,R2	:	LOAD REGISTER, SIGN EXTEND
555	011554	122702	000377	CMPB	#377,R2	:	CHECK IF BYTE INSTRUCTION
556	011560	001402		BEQ	5\$	:	CONTINUE IF OK
557	011562	104423	011472	ERROR,	T12	:	*** CMPB BECAME CMP INSTRUCTION ***
558	011566	112702	000377	5\$: MOVB	#377,R2	:	LOAD REGISTER, SIGN EXTEND
559	011572	120227	000377	CMPB	R2,#377	:	CHECK IF BYTE INSTRUCTION
560	011576	001402		BEQ	6\$	:	CONTINUE IF OK
561	011600	104423	011472	ERROR,	T12	:	*** MOVB OR CMPB FAILED ***
562	011604			6\$:			

563				*****			
564				:SBTTL T13 -- BICB AND BITB			
(2)				*****			
(2)				:*****			
(2)	011604	104420	000013	T13:	BEGINTEST, 13	:	CHECK SWITCH REGISTER OPTIONS.
565	011610	112703	000377	BICB0:	MOVB #377,R3	:	LOAD REGISTER
566	011614	112700	000252		MOVB #252,R0	:	PLACE #252 IN R0
567	011620	000277			SCC	:	
568	011622	140003			BICB R0,R3	:	CLEAR EVERY OTHER BIT
569	011624	104401			CHECKCC+1	:	CHECK FOR CC = 1
570	011626	104423	011604	ERROR,	T13	:	*** INCORRECT CC BITS ***
571	011632	130003			BITB R0,R3	:	CHECK IT
572	011634	001402			BEQ 4\$	:	CONTINUE IF OK
573	011636	104423	011604	ERROR,	T13	:	*** BICB OR BITB INSTRUCTION FAILED ***
574	011642	132703	000125	4\$: BITB	#125,R3	:	CHECK IT
575	011646	104401			CHECKCC+1	:	CHECK FOR CC = 1
576	011650	104423	011604	ERROR,	T13	:	*** INCORRECT CC BITS ***
577	011654	150003			BISB R0,R3	:	SET THE BITS THAT WERE CLEARED
578	011656	100402			BMI 6\$	:	
579	011660	104423	011604	ERROR,	T13	:	*** BISB INSTRUCTION FAILED ***
580	011664	142703	000177	6\$: BICB	#177,R3	:	CLEAR ALL THE BITS EXCEPT FOR SIGN
581	011670	104411			CHECKCC+11	:	CHECK FOR CC = 11
582	011672	104423	011604	ERROR,	T13	:	*** INCORRECT CC BITS ***
583	011676	132703	000377		BITB #377,R3	:	CHECK IT
584	011702	104411			CHECKCC+11	:	CHECK FOR CC = 11
585	011704	104423	011604	ERROR,	T13	:	*** INCORRECT CC BITS ***

586				*****			
587				:SBTTL T14 -- INCB AND DECB			
(2)				*****			
(2)				:*****			
(2)	011710	104420	000014	T14:	BEGINTEST, 14	:	CHECK SWITCH REGISTER OPTIONS.
588	011714	112704	000177	INCB0:	MOVB #177,R4	:	R4 = 177
589	011720	000261			SEC	:	
590	011722	105204			INCB R4	:	ADD ONES INTO REG. 4
591	011724	104413			CHECKCC+13	:	CHECK FOR CC = 13
592	011726	104423	011710	ERROR,	T14	:	*** INCORRECT CC BITS ***
593	011732	112704	000376		MOVB #376,R4	:	
594	011736	105204			INCB R4	:	
595	011740	104411			CHECKCC+11	:	CHECK FOR CC = 11
596	011742	104423	011710	ERROR,	T14	:	*** INCORRECT CC BITS ***
597	011746	105204			INCB R4	:	

CNKXAA.P11 17-MAR-82 14:02

PC 11742 = INCORRECT CC BITS.

```

598 011750 104405          CHECKCC+5          ; CHECK FOR CC = 5
599 011752 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
600 011756 105204          INCB R4
601 011760 104401          CHECKCC+1          ; CHECK FOR CC = 1
602 011762 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
603 011766 122704 000001  CMPB #1,R4          ; CHECK IT
604 011772 001402          BEQ 2$             ; CONTINUE IF OK
605 011774 104423 011710  ERROR, T14          ;; *** INCB INSTRUCTION FAILED ***
606 012000 000261          2$: SEC
607 012002 105304          DECB R4             ; SUBTRACT ONES FROM REG. 4
608 012004 104405          CHECKCC+5          ; CHECK FOR CC = 5
609 012006 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
610 012012 105304          DECB R4
611 012014 104411          CHECKCC+11         ; CHECK FOR CC = 11
612 012016 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
613 012022 012704 000200  MOV #200,R4
614 012026 105304          DECB R4
615 012030 104403          CHECKCC+3          ; CHECK FOR CC = 3
616 012032 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
617 012036 105304          DECB R4
618 012040 104401          CHECKCC+1          ; CHECK FOR CC = 1
619 012042 104423 011710  ERROR, T14          ;; *** INCORRECT CC BITS ***
620
621 *****
(2) .SBTTL T15 -- COMB AND NEGB
(2) *****
(2) 012046 104420 000015  T15: BEGINTEST, 15    ; CHECK SWITCH REGISTER OPTIONS.
622 012052 112703 000252  COMB0: MOVB #252,R3    ; LOAD EVERY OTHER BIT
623 012056 000277          SCC
624 012060 105103          COMB R3              ; 1'S COMPLEMENT
625 012062 104401          CHECKCC+1          ; CHECK FOR CC = 1
626 012064 104423 012046  ERROR, T15          ;; *** INCORRECT CC BITS ***
627 012070 122703 000125  CMPB #125,R3        ; CHECK IT
628 012074 001402          BEQ 2$             ; CONTINUE IF OK
629 012076 104423 012046  ERROR, T15          ;; *** COMB INSTRUCTION FAILED ***
630 012102 000277          2$: SCC
631 012104 105103          COMB R3              ; COMPLEMENT BACK
632 012106 104411          CHECKCC+11         ; CHECK FOR CC = 11
633 012110 104423 012046  ERROR, T15          ;; *** INCORRECT CC BITS ***
634 012114 122703 000252  CMPB #252,R3        ; CHECK IT
635 012120 001402          BEQ 3$             ; CONTINUE IF OK
636 012122 104423 012046  ERROR, T15          ;; *** COMB INSTRUCTION FAILED ***
637 012126 012703 000377  3$: MOV #377,R3
638 012132 000277          SCC
639 012134 105103          COMB R3
640 012136 104405          CHECKCC+5          ; CHECK FOR CC = 5
641 012140 104423 012046  ERROR, T15          ;; *** INCORRECT CC BITS ***
642
643 012144 112700 000001  NEGB0: MOVB #1,R0      ; LOAD THE REGISTER
644 012150 105400          NEGB R0             ; 2'S COMPLEMENT
645 012152 104411          CHECKCC+11         ; CHECK FOR CC = 11
646 012154 104423 012144  ERROR, NEGB0        ;; *** INCORRECT CC BITS ***
647 012160 122700 000377  CMPB #377,R0        ; CHECK IT
648 012164 001402          BEQ 2$             ; CONTINUE IF OK
649 012166 104423 012144  ERROR, NEGB0        ;; *** NEGB INSTRUCTION FAILED ***
650 012172 012700 000200  2$: MOV #200,R0

```

CNKXAA.P11 17-MAR-82 14:02

PC 12166 = NEGB INSTRUCTION FAILED.

651 012176 105400
 652 012200 104413
 653 012202 104423 012144
 654 012206 122700 000200
 655 012212 001402
 656 012214 104423 012144
 657 012220

NEGB R0 ; 2'S COMPLEMENT
 CHECKCC+13 ; CHECK FOR CC = 13
 ERROR, NEGB0 ; *** INCORRECT CC BITS ***
 CMPB #200,R0 ; CHECK IT
 BEQ 3\$; CONTINUE IF OK
 ERROR, NEGB0 ; *** NEGB FAILED ***

3\$:

658
 659
 (2)
 (2)

 ;SBTTL T16 -- ROLB AND RORB

(2) 012220 104420 000016
 660 012224 112701 000040
 661 012230 000257
 662 012232 106101
 663 012234 106101
 664 012236 104412
 665 012240 104423 012220
 666 012244 122701 000200
 667 012250 001402
 668 012252 104423 012220
 669 012256 106101

T16: BEGINTEST, 16 ; CHECK SWITCH REGISTER OPTIONS.
 ROLB0: MOVB #40,R1 ; LOAD REGISTER
 CCC ; CLEAR FLAGS
 ROLB R1 ; SHIFT
 ROLB R1
 CHECKCC+12 ; CHECK FOR CC = 12
 ERROR, T16 ; *** INCORRECT CC BITS ***
 CMPB #200,R1 ; CHECK IT
 BEQ 1\$; CONTINUE IF OK
 ERROR, T16 ; *** ROLB INSTRUCTION FAILED ***

1\$:

670 012260 104407
 671 012262 104423 012220
 672 012266 106101
 673 012270 122701 000001
 674 012274 001402
 675 012276 104423 012220
 676 012302
 677 012302 112702 000004
 678 012306 000257
 679 012310 106002
 680 012312 106002
 681 012314 122702 000001
 682 012320 001402
 683 012322 104423 012302
 684 012326 106002
 685 012330 104407
 686 012332 104423 012302
 687 012336 106002
 688 012340 104412
 689 012342 104423 012302
 690 012346 122702 000200
 691 012352 001402
 692 012354 104423 012302
 693 012360

ROLB R1 ; SHIFT
 CHECKCC+7 ; CHECK FOR CC = 7
 ERROR, T16 ; *** INCORRECT CC BITS ***
 ROLB R1 ; SHIFT
 CMPB #1,R1 ; CHECK IT
 BEQ 2\$; CONTINUE IF OK
 ERROR, T16 ; *** ROLB FAILED. ***

2\$:

677 012302 112702 000004
 678 012306 000257
 679 012310 106002
 680 012312 106002
 681 012314 122702 000001
 682 012320 001402
 683 012322 104423 012302
 684 012326 106002
 685 012330 104407
 686 012332 104423 012302
 687 012336 106002
 688 012340 104412
 689 012342 104423 012302
 690 012346 122702 000200
 691 012352 001402
 692 012354 104423 012302
 693 012360
 694
 695
 (2)
 (2)

RORB0: MOVB #4,R2 ; LOAD REGISTER
 CCC ; CLEAR FLAGS
 RORB R2 ; SHIFT
 RORB R2
 CMPB #1,R2 ; CHECK IT
 BEQ 1\$; CONTINUE IF OK
 ERROR, RORB0 ; *** RORB INSTRUCTION FAILED ***

1\$:

684 012326 106002
 685 012330 104407
 686 012332 104423 012302
 687 012336 106002
 688 012340 104412
 689 012342 104423 012302
 690 012346 122702 000200
 691 012352 001402
 692 012354 104423 012302
 693 012360

RORB R2 ; SHIFT
 CHECKCC+7 ; CHECK FOR CC = 7
 ERROR, RORB0 ; *** INCORRECT CC BITS ***
 RORB R2 ; SHIFT
 CHECKCC+12 ; CHECK FOR CC = 12
 ERROR, RORB0 ; *** INCORRECT CC BITS ***
 CMPB #200,R2 ; CHECK IT
 BEQ 2\$; CONTINUE IF OK
 ERROR, RORB0 ; *** RORB FAILED ***

2\$:

694
 695
 (2)
 (2)

 ;SBTTL T17 -- ASLB AND ASRB

(2) 012360 104420 000017
 696 012364 112703 000040
 697 012370 000257
 698 012372 106303
 699 012374 106303
 700 012376 104412

T17: BEGINTEST, 17 ; CHECK SWITCH REGISTER OPTIONS.
 ASLB0: MOVB #40,R3 ; LOAD REGISTER
 CCC ; CLEAR FLAGS
 ASLB R3 ; SHIFT
 ASLB R3
 CHECKCC+12 ; CHECK FOR CC = 12

PC 12400 = INCORRECT CC BITS.

701	012400	104423	012360	ERROR, T17	:: *** INCORRECT CC BITS ***
702	012404	122703	000200	CMPB #200,R3	: CHECK IT
703	012410	001402		BEQ 4\$	: CONTINUE IF OK
704	012412	104423	012360	ERROR, T17	:: *** ASLB INSTRUCTION FAILED ***
705	012416	106303		4\$: ASLB R3	: SHIFT
706	012420	104407		CHECKCC+7	: CHECK FOR CC = 7
707	012422	104423	012360	ERROR, T17	:: *** INCORRECT CC BITS ***
708	012426	106303		ASLB R3	: SHIFT
709	012430	104404		CHECKCC+4	: CHECK FOR CC = 4
710	012432	104423	012360	ERROR, T17	:: *** INCORRECT CC BITS ***
711					
712	012436	112704	000004	ASRB0: MOVB #4,R4	: LOAD REGISTER
713	012442	000257		CCC	: CLEAR FLAGS
714	012444	106204		ASRB R4	: SHIFT
715	012446	106204		ASRB R4	:
716	012450	122704	000001	CMPB #1,R4	: CHECK IT
717	012454	001402		BEQ 2\$	: CONTINUE IF OK
718	012456	104423	012436	ERROR, ASRB0	:: *** ASRB INSTRUCTION FAILED ***
719	012462	106204		2\$: ASRB R4	: SHIFT
720	012464	104407		CHECKCC+7	: CHECK FOR CC = 7
721	012466	104423	012436	ERROR, ASRB0	:: *** INCORRECT CC BITS ***
722	012472	106204		ASRB R4	: SHIFT
723	012474	104404		CHECKCC+4	: CHECK FOR CC = 4
724	012476	104423	012436	ERROR, ASRB0	:: *** INCORRECT CC BITS ***
725	012502	112703	000202	MOVB #202,R3	: LOAD REGISTER
726	012506	106203		ASRB R3	: SHIFT
727	012510	106203		ASRB R3	:
728	012512	104411		CHECKCC+11	: CHECK FOR CC = 11
729	012514	104423	012436	ERROR, ASRB0	:: *** INCORRECT CC BITS ***
730	012520	122703	000340	CMPB #340,R3	: CHECK IT
731	012524	001402		BEQ 3\$	: CONTINUE IF OK
732	012526	104423	012436	ERROR, ASRB0	:: *** ASRB FAILED ***
733	012532			3\$:	
734					
735					
(2)				*****	
(2)				.SMT1L 120 -- ADCB AND SBCB	
(2)				*****	
(2)	012532	104420	000020	T20: BEGINTEST, 20	:: CHECK SWITCH REGISTER OPTIONS.
736	012536	105000		ADCB0: CLRB R0	: CLEAR THE REGISTER
737	012540	000257		CCC	: CLEAR FLAGS
738	012542	105500		ADCB R0	: ADD C BIT = 0
739	012544	104404		CHECKCC+4	: CHECK FOR CC = 4
740	012546	104423	012532	ERROR, T20	:: *** INCORRECT CC BITS ***
741	012552	000261		SEC	: C=1
742	012554	105500		ADCB R0	: ADD C BIT=1
743	012556	000261		SEC	: C=1
744	012560	105500		ADCB R0	: AGAIN
745	012562	104400		CHECKCC+0	: CHECK FOR CC = 0
746	012564	104423	012532	ERROR, T20	:: *** INCORRECT CC BITS ***
747	012570	122700	000002	CMPB #2,R0	: CHECK IT
748	012574	001402		BEQ 4\$	: CONTINUE IF OK
749	012576	104423	012532	ERROR, T20	:: *** ADCB INSTRUCTION FAILED ***
750	012602	112700	000177	4\$: MOVB #177,R0	: LOAD LARGEST POSITIVE NUMBER
751	012606	000261		SFC	: C=1
752	012610	105500		ADCB R0	: ADD C BIT=1
753	012612	104412		CHECKCC+12	: CHECK FOR CC = 12

PC 12614 = INCORRECT CC BITS.

754	012614	104423	012532	ERROR, T20	:: *** INCORRECT CC BITS ***
755	012620	122700	000200	CMPB #200,R0	: CHECK IT
756	012624	001402		BEQ 6\$	: CONTINUE IF OK
757	012626	104423	012532	ERROR, T20	:: *** ADCB INSTRUCTION FAILED ***
758	012632	112700	000377	6\$: MOVB #377,R0	: LOAD -1
759	012636	000261		SEC	: C=1
760	012640	105500		ADCB R0	: ADD C BIT=1
761	012642	104405		CHECKCC+5	: CHECK FOR CC = 5
762	012644	104423	012532	ERROR, T20	:: *** INCORRECT CC BITS ***
763					
764	012650	112701	000003	SBCB0: MOVB #3,R1	: LOAD REGISTER
765	012654	000257		CCC	: CLEAR FLAGS
766	012656	105601		SBCB R1	: SUBTRACT C BIT=0
767	012660	104400		CHECKCC+0	: CHECK FOR CC = 0
768	012662	104423	012650	ERROR, SBCB0	:: *** INCORRECT CC BITS ***
769	012666	122701	000003	CMPB #3,R1	: CHECK IT
770	012672	001402		BEQ 2\$	: CONTINUE IF OK
771	012674	104423	012650	ERROR, SBCB0	:: *** SBCB INSTRUCTION FAILED ***
772	012700	000261		2\$: SEC	: C=1
773	012702	105601		SBCB R1	: SUBTRACT C BIT=1
774	012704	000261		SEC	: C=1
775	012706	105601		SBCB R1	
776	012710	104400		CHECKCC+0	: CHECK FOR CC = 0
777	012712	104423	012650	ERROR, SBCB0	:: *** INCORRECT CC BITS ***
778	012716	122701	000001	CMPB #1,R1	: CHECK IT
779	012722	001402		BEQ 3\$	: CONTINUE IF OK
780	012724	104423	012650	ERROR, SBCB0	:: *** SBCB INSTRUCTION FAILED ***
781	012730	000261		3\$: SEC	: C=1
782	012732	105601		SBCB R1	: SUBTRACT C BIT=1
783	012734	104404		CHECKCC+4	: CHECK FOR CC = 4
784	012736	104423	012650	ERROR, SBCB0	:: *** INCORRECT CC BITS ***
785	012742	000261		SEC	: C=1
786	012744	105501		SBCB R1	: SUBTRACT C BIT = 1
787	012746	104411		CHECKCC+11	: CHECK FOR CC = 11
788	012750	104423	012650	ERROR, SBCB0	:: *** INCORRECT CC BITS ***
789	012754	122701	000377	CMPB #377,R1	: CHECK IT
790	012760	001402		BEQ 4\$	: CONTINUE IF OK
791	012762	104423	012650	ERROR, SBCB0	:: *** SBCB INSTRUCTION FAILED ***
792	012766	112701	000200	4\$: MOVB #200,R1	: LOAD R1
793	012772	000261		SEC	: C=1
794	012774	105601		SBCB R1	: SUBTRACT C BIT = 1
795	012776	104402		CHECKCC+2	: CHECK FOR CC = 2
796	013000	104423	012650	ERROR, SBCB0	:: *** INCORRECT CC BITS ***
797					
798					
(2)				*****	*****
(2)				.SHTTL T21 -- TST, CLR, AND MOV	
(2)				*****	*****
(2)	013004	104420	000021	T21: BEGINTEST, 21	:: CHECK SWITCH REGISTER OPTIONS.
799	013010	000277		MOV0: SCC	
800	013012	005000		CLR R0	: CLEAR THE REGISTER
801	013014	104404		CHECKCC+4	: CHECK FOR CC = 4
802	013016	104423	013004	ERROR, T21	:: *** INCORRECT CC BITS ***
803	013022	005700		TST R0	: CHECK IT
804	013024	104404		CHECKCC+4	: CHECK FOR CC = 4
805	013026	104423	013004	ERROR, T21	:: *** INCORRECT CC BITS ***
806	013032	012704	177777	MOV #177777,R4	: LOAD THE REGISTER

```

807 013036 010401      MOV      R4,R1
808 013040 104410      CHECKCC+10      ; CHECK FOR CC = 10
809 013042 104423 013004 ERROR, T21      ; *** INCORRECT CC BITS ***
810 013046 005701      TST      R1      ; CHECK IT
811 013050 104410      CHECKCC+10      ; CHECK FOR CC = 10
812 013052 104423 013004 ERROR, T21      ; *** INCORRECT CC BITS ***
813 013056 020401      CMP      R4,R1      ; CHECK R1 TO CONTAIN PROPER DATA
814 013060 001402      BEQ      2$
815 013062 104423 013004 ERROR, T21      ; *** R1 WRONG AFTER MOV ***
816 013066 000263      SEVC
817 013070 010000      MOV      R0,R0      ; SET V & C BITS
818 013072 104405      CHECKCC+5
819 013074 104423 013004 ERROR, T21      ; *** INCORRECT CC BITS ***
820
821
822 (2)
823 (2)
824 (2) 013100 104420 000022 ;*****
;SBTTL      T22 -- CMP AND BIS ;*****
825 (2) 013104 000277      T22: BEGINTEST, 22      ; CHECK SWITCH REGISTER OPTIONS.
826 013106 012700 177777      CMP0: SCC
827 013112 050002      MOV      #177777,R0      ; LOAD REGISTER
828 013114 104411      BIS      R0,R2      ; CHECK THE BIS INSTRUCTION
829 013116 104423 013100      CHECKCC+11      ; CHECK FOR CC = 11
830 013122 020002      ERROR, T22      ; *** INCORRECT CC BITS ***
831 013124 001402      CMP      R0,R2      ; CHECK COMPARE
832 013126 104423 013100      BEQ      2$      ; CONTINUE IF OK
833 013132 022702 000077      ERROR, T22      ; *** BIS OR CMP INSTRUCTION FAILED ***
834 013136 100002      2$: CMP      #77,R2      ; CHECK IT AGAIN
835 013140 104423 013100      BPL      3$      ; CONTINUE IF OK
836 013144 020227 000077      ERROR, T22      ; *** CMP INSTRUCTION FAILED (WRONG CC) ***
837 013150 100402      3$: CMP      R2,#77      ; ONCE MORE
838 013152 104423 013100      BMI      4$      ; CONTINUE IF OK
839 013156      ERROR, T22      ; *** BIC OR CMP FAILED ***
840
841 (2)
842 (2)
843 (2) 013156 104420 000023 ;*****
;SBTTL      T23 -- BIC AND BIT ;*****
844 013162 012703 177777      T23: BEGINTEST, 23      ; CHECK SWITCH REGISTER OPTIONS.
845 013166 012700 001006      BIC0: MOV      #177777,R3      ; LOAD REGISTER
846 013172 012710 125252      MOV      #TEMP,R0      ; PLACE THE ADDRESS OF TEMP IN R0
847 013176 000277      MOV      #125252,(R0)      ; SET (R0)
848 013200 041003      SCC
849 013202 104401      BIC      (R0),R3      ; CLEAR EVERY OTHER BIT
850 013204 104423 013156      CHECKCC+1      ; CHECK FOR CC = 1
851 013210 031003      ERROR, T23      ; *** INCORRECT CC BITS ***
852 013212 001402      BIT      (R0),R3      ; CHECK IT
853 013214 104423 013156      BEQ      1$      ; CONTINUE IF OK
854 013220 032703 052525      ERROR, T23      ; *** BIC OR BIT INSTRUCTION FAILED ***
855 013224 104401      1$: BIT      #52525,R3      ; CHECK IT
856 013226 104423 013156      CHECKCC+1      ; CHECK FOR CC = 1
857 013232 052703 125252      ERROR, T23      ; *** INCORRECT CC BITS ***
858 013236 100402      BIS      #125252,R3      ; SET THE BITS THAT WERE CLEARED
859 013240 104423 013156      BMI      2$      ; CONTINUE IF OK
860 013244 042703 077777      ERROR, T23      ; *** BIT OR BIS INSTRUCTION FAILED ***
861 013250 104411      2$: BIC      #77777,R3      ; CLEAR ALL THE BITS EXCEPT FOR SIGN
      CHECKCC+11      ; CHECK FOR CC = 11

```

CNKXAA.P11 17-MAR-82 14:02

PC 13252 = INCORRECT CC BITS.

857 013252 104423 013156
 858 013256 012700 177777
 859 013262 030003
 860 013264 104411
 861 013266 104423 013156
 862 013272 000263
 863 013274 040000
 864 013276 104405
 865 013300 104423 013156
 866 013304 005700
 867 013306 001402
 868 013310 104423 013156
 869 013314

ERROR, T23 ;: *** INCORRECT CC BITS ***
 MOV #177777,R0
 BIT R0,R3 ; CHECK IT
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, T23 ;: *** INCORRECT CC BITS ***
 SEVC ; SET V & C BITS
 BIC R0,R0
 CHECKCC+5 ; CHECK CC = 5
 ERROR, T23 ;: *** INCORRECT CC BITS ***
 IST R0 ; CHECK R0 TO CONTAIN 0
 BEQ 3\$
 ERROR, T23 ;: *** BIC FAILED ***

3\$:

870
 871
 (2)
 (2)

 SBTTL T24 -- INC AND DEC

(2) 013314 104420 000024
 872 013320 012704 077777

T24: BEGINTEST, 24 ;: CHECK SWITCH REGISTER OPTIONS.
 INCO: MOV #77777,R4 ; R4=77777

873 013324 000261
 874 013326 005204
 875 013330 104413
 876 013332 104423 013314
 877 013336 012704 177776

SEC
 INC R4 ; ADD ONES INTO REG. 4
 CHECKCC+13 ; CHECK FOR CC = 13
 ERROR, T24 ;: *** INCORRECT CC BITS ***
 MOV #177776,R4

878 013342 005204
 879 013344 104411
 880 013346 104423 013314

INC R4
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, T24 ;: *** INCORRECT CC BITS ***

881 013352 005204
 882 013354 104405
 883 013356 104423 013314

INC R4
 CHECKCC+5 ; CHECK FOR CC = 5
 ERROR, T24 ;: *** INCORRECT CC BITS ***

884 013362 005204
 885 013364 104401
 886 013366 104423 013314

INC R4
 CHECKCC+1 ; CHECK FOR CC = 1
 ERROR, T24 ;: *** INCORRECT CC BITS ***

887 013372 022704 000001
 888 013376 001402

CMP #1,R4 ; CHECK IT
 BEQ 4\$; FAILED

889 013400 104423 013314
 890 013404 000261

4\$:

ERROR, T24 ;: *** INC INSTRUCTION FAILED ***

891 013406 005304
 892 013410 104405
 893 013412 104423 013314

SEC
 DEC R4 ; SUBTRACT ONES FROM REG. 4
 CHECKCC+5 ; CHECK FOR CC = 5
 ERROR, T24 ;: *** INCORRECT CC BITS ***

894 013416 005304
 895 013420 104411
 896 013422 104423 013314

DEC R4
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, T24 ;: *** INCORRECT CC BITS ***

897 013426 012704 100000
 898 013432 005304
 899 013434 104403

MOV #100000,R4
 DEC R4
 CHECKCC+5 ; CHECK FOR CC = 3
 ERROR, T24 ;: *** INCORRECT CC BITS ***

900 013436 104423 013314
 901 013442 005304
 902 013444 104401

DEC R4
 CHECKCC+1 ; CHECK FOR CC = 1
 ERROR, T24 ;: *** INCORRECT CC BITS ***

903 013446 104423 013314
 904
 905
 (2)

 SBTTL T25 -- COM AND NEG

(2) 013452 104420 000025
 906 013456 012703 125252

T25: BEGINTEST, 25 ;: CHECK SWITCH REGISTER OPTIONS.
 COM0: MOV #125252,R3 ;: LOAD EVERY OTHER BIT

```

907 013462 000277 SCC
908 013464 005103 COM R3 ; 1'S COMPLEMENT
909 013466 104401 CHECKCC+1 ; CHECK FOR CC = 1
910 013470 104423 013452 ERROR, T25 ; *** INCORRECT CC BITS ***
911 013474 022703 052525 CMP #52525,R3 ; CHECK IT
912 013500 001402 BEQ 2$ ; CONTINUE IF OK
913 013502 104423 013452 ERROR, T25 ; *** COM INSTRUCTION FAILED ***
914 013506 000277 2$: SCC
915 013510 005103 COM R3 ; COMPLEMENT BACK
916 013512 104411 CHECKCC+1 ; CHECK FOR CC = 11
917 013514 104423 013452 ERROR, T25 ; *** INCORRECT CC BITS ***
918 013520 022703 125252 CMP #125252,R3 ; CHECK IT
919 013524 001402 BEQ 3$ ; CONTINUE IF OK
920 013526 104423 013452 ERROR, T25 ; *** COM INSTRUCTION FAILED ***
921 013532 012703 177777 3$: MOV #177777,R3
922 013536 000277 SCC
923 013540 005103 COM R3
924 013542 104405 CHECKCC+5 ; CHECK FOR CC = 5
925 013544 104423 013452 ERROR, T25 ; *** INCORRECT CC BITS ***
926
927 013550 012700 000001 NEG0: MOV #1,R0 ; LOAD THE REGISTER
928 013554 005400 NEG R0 ; 2'S COMPLEMENT
929 013556 104411 CHECKCC+1 ; CHECK FOR CC = 11
930 013560 104423 013550 ERROR, NEG0 ; *** INCORRECT CC BITS ***
931 013564 022700 177777 CMP #177777,R0 ; CHECK IT
932 013570 001402 BEQ 2$ ; CONTINUE IF OK
933 013572 104423 013550 ERROR, NEG0 ; *** NEG INSTRUCTION FAILED ***
934 013576 012700 100000 2$: MOV #100000,R0
935 013602 005400 NEG R0 ; 2'S COMPLEMENT
936 013604 104413 CHECKCC+13 ; CHECK FOR CC = 13
937 013606 104423 013550 ERROR, NEG0 ; *** INCORRECT CC BITS ***
938 013612 022700 100000 CMP #100000,R0 ; CHECK IT
939 013616 001402 BEQ 3$ ; CONTINUE IF OK
940 013620 104423 013550 ERROR, NEG0 ; *** NEG FAILED ***
941 013624 3$:
942
943
944 (2) ;*****
945 (2) ;SBTTL T26 -- ROL AND ROR
946 (2) ;*****
947 (2) 013624 104420 000026 T26: BEGINTEST, 26 ; CHECK SWITCH REGISTER OPTIONS.
948 013630 012701 020000 ROL0: MOV #20000,R1 ; LOAD REGISTER
949 013634 000257 CCC ; CLEAR FLAGS
950 013636 006101 ROL R1 ; SHIFT
951 013640 006101 ROL R1
952 013642 104412 CHECKCC+12 ; CHECK FOR CC = 12
953 013644 104423 013624 ERROR, T26 ; *** INCORRECT CC BITS ***
954 013650 022701 100000 CMP #100000,R1 ; CHECK IT
955 013654 001402 BEQ 1$ ; CONTINUE IF OK
956 013656 104423 013624 ERROR, T26 ; *** ROL INSTRUCTION FAILED ***
957 013662 006101 1$: ROL R1 ; SHIFT
958 013664 104407 CHECKCC+7 ; CHECK FOR CC = 7
959 013666 104423 013624 ERROR, T26 ; *** INCORRECT CC BITS ***
960 013672 006101 ROL R1 ; SHIFT
961 013674 022701 000001 CMP #1,R1 ; CHECK IT
962 013700 001402 BEQ 2$ ; CONTINUE IF OK
963 013702 104423 013624 ERROR, T26 ; *** ROL FAILED ***

```

960	013706			2\$:		
961	013706	012702	000004	ROR0:	MOV #4,R2	: LOAD REGISTER
962	013712	000257			CCC	: CLEAR FLAGS
963	013714	006002			ROR R2	: SHIFT
964	013716	006002			ROR R2	
965	013720	022702	000001		CMP #1,R2	: CHECK IT
966	013724	001402			BEQ 1\$	: CONTINUE IF OK
967	013726	104423	013706		ERROR, ROR0	: *** ROR INSTRUCTION FAILED ***
968	013732	006002		1\$:	ROR R2	: SHIFT
969	013734	104407			CHECKCC+7	: CHECK FOR CC = 7
970	013736	104423	013706		ERROR, ROR0	: *** INCORRECT CC BITS ***
971	013742	006002			ROR R2	: SHIFT
972	013744	104412			CHECKCC+12	: CHECK FOR CC = 12
973	013746	104423	013706		ERROR, ROR0	: *** INCORRECT CC BITS ***
974	013752	022702	100000		CMP #100000,R2	: CHECK IT
975	013756	001402			BEQ 2\$	: CONTINUE IF OK
976	013760	104423	013706		ERROR, ROR0	: *** ROR FAILED ***
977	013764			2\$:		
978						
979						
(2)					*****	
(2)					SBTTL T27 -- ASL AND ASR	
(2)					*****	
(2)	013764	104420	000027	T27:	BEGINTEST, 27	: CHECK SWITCH REGISTER OPTIONS.
980	013770	012703	020000	ASL0:	MOV #20000,R3	: LOAD REGISTER
981	013774	000257			CCC	: CLEAR FLAGS
982	013776	006303			ASL R3	: SHIFT
983	014000	006303			ASL R3	
984	014002	104412			CHECKCC+12	: CHECK FOR CC = 12
985	014004	104423	013764		ERROR, T27	: *** INCORRECT CC BITS ***
986	014010	022703	100000		CMP #100000,R3	: CHECK IT
987	014014	001402			BEQ 4\$	: CONTINUE IF OK
988	014016	104423	013764		ERROR, T27	: *** ASL INSTRUCTION FAILED ***
989	014022	006303		4\$:	ASL R3	: SHIFT
990	014024	104407			CHECKCC+7	: CHECK FOR CC = 7
991	014026	104423	013764		ERROR, T27	: *** INCORRECT CC BITS ***
992	014032	006303			ASL R3	: SHIFT
993	014034	104404			CHECKCC+4	: CHECK FOR CC = 4
994	014036	104423	013764		ERROR, T27	: *** INCORRECT CC BITS ***
995						
996	014042	012704	000004	ASR0:	MOV #4,R4	: LOAD REGISTER
997	014046	000257			CCC	: CLEAR FLAGS
998	014050	006204			ASR R4	: SHIFT
999	014052	006204			ASR R4	
1000	014054	022704	000001		CMP #1,R4	: CHECK IT
1001	014060	001402			BEQ 2\$	: CONTINUE IF OK
1002	014062	104423	014042		ERROR, ASR0	: *** ASR INSTRUCTION FAILED ***
1003	014066	006204		2\$:	ASR R4	: SHIFT
1004	014070	104407			CHECKCC+7	: CHECK FOR CC = 7
1005	014072	104423	014042		ERROR, ASR0	: *** INCORRECT CC BITS ***
1006	014076	006204			ASR R4	: SHIFT
1007	014100	104404			CHECKCC+4	: CHECK FOR CC = 4
1008	014102	104423	014042		ERROR, ASR0	: *** INCORRECT CC BITS ***
1009	014106	012703	100002		MOV #100002,R3	: LOAD REGISTER
1010	014112	006203			ASR R3	: SHIFT
1011	014114	006203			ASR R3	
1012	014116	104411			CHECKCC+11	: CHECK FOR CC = 11

PC 14120 = INCORRECT CC BITS.

1013 014120 104423 014042
 1014 014124 022703 160000
 1015 014130 001402
 1016 014132 104423 014042
 1017 014136

ERROR, ASR0
 CMP #160000,R3
 BEQ 3\$
 ERROR, ASR0
 : : *** INCORRECT CC BITS ***
 : : CHECK IT
 : : CONTINUE IF OK
 : : *** ASR FAILED ***

3\$:

1018
 1019
 (2)
 (2)

 :SBTTL T30 -- ADC AND SBC
 :*****

(2) 014136 104420 000030

T30: BEGINTST, 30 : : CHECK SWITCH REGISTER OPTIONS.

1020 014142 005000

ADC0: CLR R0 : : CLEAR THE REGISTER

1021 014144 000257

CCC : : CLEAR FLAGS

1022 014146 005500

ADC R0 : : ADD C BIT = 0

1023 014150 104404

CHECKCC+4 : : CHECK FOR CC = 4

1024 014152 104423 014136

ERROR, T30 : : *** INCORRECT CC BITS ***

1025 014156 000261

SEC : : C=1

1026 014160 005500

ADC R0 : : ADD C BIT=1

1027 014162 000261

SEC : : C=1

1028 014164 005500

ADC R0 : : AGAIN

1029 014166 104400

CHECKCC+0 : : CHECK FOR CC = 0

1030 014170 104423 014136

ERROR, T30 : : *** INCORRECT CC BITS ***

1031 014174 022700 000002

CMP #2,R0 : : CHECK IT

1032 014200 001402

BEQ 4\$: : CONTINUE IF OK

1033 014202 104423 014136

ERROR, T30 : : *** ADC INSTRUCTION FAILED ***

1034 014206 012700 077777

4\$: MOV #77777,R0 : : LOAD LARGEST POSITIVE NUMBER

1035 014212 000261

SEC : : C=1

1036 014214 005500

ADC R0 : : ADD C BIT=1

1037 014216 104412

CHECKCC+12 : : CHECK FOR CC = 12

1038 014220 104423 014136

ERROR, T30 : : *** INCORRECT CC BITS ***

1039 014224 022700 100000

CMP #100000,R0 : : CHECK IT

1040 014230 001402

BEQ 6\$: : FAILED

1041 014232 104423 014136

ERROR, T30 : : *** ADC INSTRUCTION FAILED ***

1042 014236 012700 177777

6\$: MOV #-1,R0 : : LOAD -1

1043 014242 000261

SEC : : C=1

1044 014244 005500

ADC R0 : : ADD C BIT=1

1045 014246 104405

CHECKCC+5 : : CHECK FOR CC = 5

1046 014250 104423 014136

ERROR, T30 : : *** INCORRECT CC BITS ***

1047

1048 014254 012701 000003

SBC0: MOV #3,R1 : : LOAD REGISTER

1049 014260 000257

CCC : : CLEAR FLAGS

1050 014262 005601

SBC R1 : : SUBTRACT C BIT=0

1051 014264 104400

CHECKCC+0 : : CHECK FOR CC = 0

1052 014266 104423 014254

ERROR, SBC0 : : *** INCORRECT CC BITS ***

1053 014272 022701 000003

CMP #3,R1 : : CHECK IT

1054 014276 001402

BEQ 2\$: : CONTINUE IF OK

1055 014300 104423 014254

ERROR, SBC0 : : *** SBC INSTRUCTION FAILED ***

1056 014304 000261

2\$: SEC : : C=1

1057 014306 005601

SBC R1 : : SUBTRACT C BIT=1

1058 014310 000261

SEC : : C=1

1059 014312 005601

SBC R1 : : CHECK FOR CC = 0

1060 014314 104400

CHECKCC+0 : : *** INCORRECT CC BITS ***

1061 014316 104423 014254

ERROR, SBC0 : : CHECK IT

1062 014322 022701 000001

CMP #1,R1 : : CONTINUE IF OK

1063 014326 001402

BEQ 3\$: : *** SBC INSTRUCTION FAILED ***

1064 014330 104423 014254

3\$: ERROR, SBC0

1065 014334 000261

SEC : : C=1

PC 14330 = SBC INSTRUCTION FAILED.

1066 014336 005601
 1067 014340 104404
 1068 014342 104423 014254
 1069 014346 000261
 1070 014350 005601
 1071 014352 104411
 1072 014354 104423 014254
 1073 014360 022701 177777
 1074 014364 001402
 1075 014366 104423 014254
 1076 014372 012701 100000
 1077 014376 000261
 1078 014400 005601
 1079 014402 104402
 1080 014404 104423 014254
 1081
 1082
 (2)
 (2)
 (2) 014410 104420 000031
 1083 014414 005002
 1084 014416 000277
 1085 014420 000254
 1086 014422 006702
 1087 014424 104405
 1088 014426 104423 014410
 1089 014432 005702
 1090 014434 001402
 1091 014436 104423 014410
 1092 014442 000273
 1093 014444 006702
 1094 014446 104411
 1095 014450 104423 014410
 1096 014454 022702 177777
 1097 014460 001402
 1098 014462 104423 014410
 1099
 1100 014466 012703 125125
 1101 014472 000277
 1102 014474 000250
 1103 014476 000303
 1104 014500 104410
 1105 014502 104423 014466
 1106 014506 022703 052652
 1107 014512 001402
 1108 014514 104423 014466
 1109 014520 012703 000377
 1110 014524 000277
 1111 014526 000244
 1112 014530 000303
 1113 014532 104404
 1114 014534 104423 014466
 1115 014540 022703 177400
 1116 014544 001402
 1117 014546 104423 014466
 1118

```

SBC      R1      ; SUBTRACT C BIT=1
CHECKCC+4 ; CHECK FOR CC = 4
ERROR,   SBC0    ; *** INCORRECT CC BITS ***
SEC      ; C=1
SBC      R1      ; SUBTRACT C BIT = 1
CHECKCC+11 ; CHECK FOR CC = 11
ERROR,   SBC0    ; *** INCORRECT CC BITS ***
CMP      #-1,R1  ; CHECK IT
BEQ      4$      ; CONTINUE IF F OK
ERROR,   SBC0    ; *** SBC INSTRUCTION FAILED ***
4$:      MOV      #100000,R1 ; LOAD R1
SEC      ; C=1
SBC      R1      ; SUBTRACT C BIT = 1
CHECKCC+2 ; CHECK FOR CC = 2
ERROR,   SBC0    ; *** INCORRECT CC BITS ***

;*****
.SBTTL    T31 -- SXT, SWAB AND XOR
;*****
T31:      BEGINTEST, 31 ; CHECK SWITCH REGISTER OPTIONS.
SXT0:     CLR      R2    ; CLEAR REGISTER
          SCC
          CLNZ
          SXT      R2    ; SIGN EXTEND
          CHECKCC+5 ; CHECK FOR CC = 5
          ERROR,   T31   ; *** INCORRECT CC BITS ***
          TST      R2    ; REG. 2 SHOULD STILL BE 0
          BEQ      2$    ; CONTINUE IF OK
          ERROR,   T31   ; *** SXT INSTRUCTION FAILED ***
2$:       SENV C
          SXT      R2    ; SIGN EXTEND
          CHECKCC+11 ; CHECK FOR CC = 11
          ERROR,   T31   ; *** INCORRECT CC BITS ***
          CMP      #-1,R2 ; REG. 2 SHOULD NOW HAVE -1
          BEQ      SWAB0  ; CONTINUE IF OK
          ERROR,   T31   ; *** SXT FAILED ***

SWAB0:    MOV      #125125,R3 ; LOAD BIT PATTERN INTO REGISTER
          SCC
          CLN
          SWAB     R3    ; SWAP BYTES OF REGISTER
          CHECKCC+10 ; CHECK FOR CC = 10
          ERROR,   SWAB0 ; *** INCORRECT CC BITS ***
          CMP      #52652,R3 ; CHECK IT
          BEQ      1$    ; CONTINUE IF OK
          ERROR,   SWAB0 ; *** SWAB INSTRUCTION FAILED ***
1$:       MOV      #377,R3
          SCC
          CLZ
          SWAB     R3    ; SWAP BYTES OF REGISTER
          CHECKCC+4 ; CHECK FOR CC = 4
          ERROR,   SWAB0 ; *** INCORRECT CC BITS ***
          CMP      #177400,R3 ; CHECK IT
          BEQ      XOR0   ; CONTINUE IF OK
          ERROR,   SWAB0 ; *** SWAB FAILED ***
  
```

PC 14546 = SWAB FAILED.

1119 014552 012704 177777
 1120 014556 012703 177777
 1121 014562 000277
 1122 014564 074403
 1123 014566 104405
 1124 014570 104423 014552
 1125 014574 012703 077777
 1126 014600 010400
 1127 014602 000263
 1128 014604 000244
 1129 014606 074003
 1130 014610 104411
 1131 014612 104423 014552
 1132 014616 012702 125252
 1133 014622 012704 052525
 1134 014626 000277
 1135 014630 074204
 1136 014632 104411
 1137 014634 104423 014552
 1138 014640 022704 177777
 1139 014644 001402
 1140 014646 104423 014552
 1141 014652

XOR0: MOV #-1,R4 ; LOAD REGISTERS
 MOV #-1,R3 ;
 SCC ;
 XOR R4,R3 ; SHOULD PRODUCE 0'S IN REG. 3
 CHECKCC+5 ; CHECK FOR CC = 5
 ERROR, XOR0 ; *** INCORRECT CC BITS ***
 MOV #77777,R3 ;
 MOV R4,R0 ; PLACE A -1 IN R0
 SEVC ; SET V & C BITS
 CLZ ;
 XOR 1,R3 ;
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, XOR0 ; *** INCORRECT CC BITS ***
 MOV #125252,R2 ; LOAD REGISTERS
 MOV #52525,R4 ;
 SCC ;
 XOR R2,R4 ; SHOULD PRODUCE ALL 1'S IN REG. 4
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, XOR0 ; *** INCORRECT CC BITS ***
 CMP #-1,R4 ; CHECK IT
 BEQ 1\$; CONTINUE IF OK
 ERROR, XOR0 ; *** XOR FAILED ***

1\$:

 .SBTTL T32 -- ADD AND SUB

1142 014652 104420 000032
 1143 014656 012701 021421
 1144 014662 060101
 1145 014664 104400
 1146 014666 104423 014652
 1147 014672 022701 043042
 1148 014676 001402
 1149 014700 104423 014652
 1150 014704 012700 156357
 1151 014710 060000
 1152 014712 104411
 1153 014714 104423 014652
 1154 014720 022700 134736
 1155 014724 001402
 1156 014726 104423 014652
 1157 014732 012702 100000
 1158 014736 060202
 1159 014740 104407
 1160 014742 104423 014652
 1161 014746 012704 021421
 1162 014752 012701 156357
 1163 014756 060401
 1164 014760 001402
 1165 014762 104423 014652
 1166 014766 005404
 1167 014770 012701 021421
 1168 014774 060104
 1169 014776 001402
 1170 015000 104423 014652

T32: BEGINTEST, 32 ; CHECK SWITCH REGISTER OPTIONS.
 ADD0: MOV #21421,R1 ; LOAD REGISTERS
 ADD R1,R1 ;
 CHECKCC+0 ; CHECK FOR CC = 0
 ERROR, T32 ; *** INCORRECT CC BITS ***
 CMP #43042,R1 ; CHECK IT
 BEQ 1\$; CONTINUE IF OK
 ERROR, T32 ; *** ADD INSTRUCTION FAILED ***
 1\$: MOV #-21421,R0 ; LOAD REGISTERS
 ADD R0,R0 ;
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, T32 ; *** INCORRECT CC BITS ***
 CMP #-43042,R0 ; CHECK IT
 BEQ 2\$; CONTINUE IF OK
 ERROR, T32 ; *** ADD INSTRUCTION FAILED ***
 2\$: MOV #100000,R2 ; LOAD REGISTERS
 ADD R2,R2 ; ADD SHOULD RESULT AS 0'S
 CHECKCC+7 ; CHECK FOR CC = 7
 ERROR, T32 ; *** INCORRECT CC BITS ***
 MOV #21421,R4 ; LOAD REGISTERS
 MOV #-21421,R1 ;
 ADD R4,R1 ; ADD SHOULD RESULT AS 0'S
 BEQ 3\$; CONTINUE IF OK
 ERROR, T32 ; *** ADD INSTRUCTION FAILED ***
 3\$: NEG R4 ; SWITCH SOURCE AND DESTINATION
 MOV #21421,R1 ;
 ADD R1,R4 ; SHOULD RESULT AS 0'S
 BEQ SUB0 ; CONTINUE IF OK
 ERROR, T32 ; *** ADD FAILED ***

```

1172
1173 015004 012702 021421      SUB0:  MOV    #21421,R2      ; LOAD REGISTERS
1174 015010 012703 156357      MOV    #-21421,R3      ;
1175 015014 160203              SUB    R2,R3      ; RESULT SHOULD=-43042
1176 015016 104410              CHECKCC+10      ; CHECK FOR CC = 10
1177 015020 104423 015004      ERROR, SUB0      ; *** INCORRECT CC BITS ***
1178 015024 022703 134736      CMP    #-43042,R3      ; CHECK IT
1179 015030 001402              BEQ    4$      ; CONTINUE IF OK
1180 015032 104423 015004      ERROR, SUB0      ; *** SUB INSTRUCTION FAILED ***
1181 015036 012703 021421      4$:  MOV    #21421,R3      ; LOAD REGISTER
1182 015042 010204              MOV    R2,R4      ; NOW R4 = #21421
1183 015044 160403              SUB    R4,R3      ; RESULT SHOULD=0
1184 015046 001402              BEQ    6$      ;
1185 015050 104423 015004      ERROR, SUB0      ; *** SUB INSTRUCTION FAILED ***
1186 015054 012703 177777      6$:  MOV    #-1,R3      ; LOAD REGISTERS
1187 015050 012702 077777      MOV    #77777,R2      ; LOAD REGISTERS
1188 015064 160302              SUB    R3,R2      ; RESULT SHOULD BE 100000 AND OVERFLOW
1189 015066 104413              CHECKCC+13      ; CHECK FOR CC = 13
1190 015070 104423 015004      ERROR, SUB0      ; *** INCORRECT CC BITS ***
1191 015074 022702 100000      CMP    #100000,R2      ; CHECK IT
1192 015100 001402              BEQ    8$      ; CONTINUE IF OK
1193 015102 104423 015004      ERROR, SUB0      ; *** SUB INSTRUCTION FAILED ***
1194 015106 012704 177777      8$:  MOV    #-1,R4      ;
1195 015112 160304              SUB    R3,R4      ;
1196 015114 104404              CHECKCC+4      ; CHECK FOR CC = 4
1197 015116 104423 015004      ERROR, SUB0      ; *** INCORRECT CC BITS ***
1198
1199
1200
1201
1202 015122 104420 000033      ;*****
1203 015126 012701 177777      ;SBTTL      T33 -- MTPS AND MFPS
1204 015132 005000              ;*****
1205 015134 106400              T33:  BEGINTST, 33      ; CHECK SWITCH REGISTER OPTIONS.
1206 015136 104400              TPSW: MOV    #177777,R1      ;
1207 015140 104423 015122      CLR    R0      ; SET PSW TO 0
1208 015144 106701              MTPS   R0      ; CHECK FOR CC = 0
1209 015146 001402              CHECKCC+0      ; *** INCORRECT CC BITS ***
1210 015150 104423 015122      ERROR, T33      ; MOVE PSW TO R1
1211 015154 104404              MFPS   R1      ; BR IF BIT 8 OF PSW WAS EXTENDED IN R1
1212 015156 012700 000337      BEQ    2$      ; *** MTPS OR MFPS INSTRUCTION FAILED ***
1213 015162 106400              ERROR, T33      ; CHECK FOR CC = 4
1214 015166 104417              CHECKCC+4      ; *** INCORRECT CC BITS ***
1215 015170 104423 015122      ERROR, T33      ; EXPECT 317 SINCE MTPS DOESN'T SET T BIT
1216 015172 106701              CHECKCC+17      ; CHECK FOR CC = 17
1217 015176 104411              ERROR, T33      ; *** INCORRECT CC BITS ***
1218 015200 104423 015122      MFPS   R1      ; MOVE PSW TO R1
1219 015202 104423 015122      CHECKCC+11      ; CHECK FOR CC = 11 (C BIT SHOULD NOT BE EFFECTED BY MFP
1220 015206 022701 177717      ERROR, T33      ; *** INCORRECT CC BITS ***
1221 015212 001402              CMP    #177717,R1      ; CHECK TO SEE IF BIT 8 OF PSW WAS EXTENDED THRU R1
1222 015214 104423 015122      BEQ    3$      ; *** MTPS OR MFPS INSTRUCTION FAILED ***
1223 015220
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
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
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```

```

1225 (2)
1226 (2)
1227 (2) 015220 104420 000074
1228 015224 112700 000252
1229 015230 110001
1230 015232 110102
1231 015234 122702 000252
1232 015240 001402
1233 015242 104423 015220
1234 015246 012700 125252
1235 015252 010001
1236 015254 010102
1237 015256 022702 125252
1238 015262 001402
1239 015264 104423 015220
1240 015270 012700 001006
1241 015274 012701 001010
1242 015300 012702 001012
1243 015304 005067 163502
1244 015310 112710 000125
1245 015314 111011
1246 015316 111112
1247 015320 122767 000125 163464
1248 015326 001402
1249 015330 104423 015220
1250 015334 012710 052525
1251 015340 011011
1252 015342 011112
1253 015344 022767 052525 163440
1254 015352 001402
1255 015354 104423 015220
1256 015360
1257 (2)
1258 (2)
1259 (2) 015360 104420 000035
1260 015364 012700 001006
1261 015370 012701 001010
1262 015374 012702 001012
1263 015400 105022
1264 015402 112710 000252
1265 015406 112021
1266 015410 105201
1267 015412 111167 163370
1268 015416 105200
1269 015420 112021
1270 015422 124227 000252
1271 015426 001003
1272 015430 105767 163352
1273 015434 001402
1274 (2) 015436 104423 015360
1275 (2)

;*****
;SBTTL T34 -- TEST MODES 0 AND 1 USING MOVB AND MOV
;*****
T34: BEGINTEST, 34 ;: CHECK SWITCH REGISTER OPTIONS.
MODE0: MOV #252,R0 ;: LOAD REGISTERS
      MOV R0,R1
      MOV R1,R2
      CMPB #252,R2 ;: CHECK IT
      BEQ 1$ ;: OK, CONTINUE
      ERROR, T34 ;: *** MOVB INSTRUCTION FAILED IN MODE 0 ***
1$: MOV #125252,R0 ;: LOAD REGISTERS
      MOV R0,R1
      MOV R1,R2
      CMP #125252,R2 ;: CHECK IT
      BEQ MODE1 ;: OK, CONTINUE
      ERROR, T34 ;: *** MOV INSTRUCTION FAILED IN MODE 0 ***

MODE1: MOV #TEMP,R0 ;: LOAD ADDRESSES INTO REGS.
      MOV #TEMP1,R1
      MOV #TEMP2,R2
      CLR TEMP2
      MOVB #125,(R0) ;: START CLEAN
      MOVB (R0),(R1) ;: LOAD THE LOCATIONS
      MOVB (R1),(R2) ;: TEMP => TEMP1
      CMPB #125,TEMP2 ;: TEMP1 => TEMP2
      BEQ 1$ ;: CHECK IT
      ERROR, T34 ;: OK, CONTINUE
1$: MOV #52525,(R0) ;: *** MOV INSTRUCTION FAILED IN MODE 1 ***
      MOV (R0),(R1) ;: LOAD THE LOCATIONS
      MOV (R1),(R2) ;: TEMP => TEMP1
      CMP #52525,TEMP2 ;: TEMP1 => TEMP2
      BEQ 2$ ;: CHECK IT
      ERROR, T34 ;: OK, CONTINUE
2$: ;: *** MOV INSTRUCTION FAILED IN MODE 1 ***

;*****
;SBTTL T35 -- TEST MODE 2 USING MOVB AND MOV
;*****
T35: BEGINTEST, 35 ;: CHECK SWITCH REGISTER OPTIONS.
MODE2: MOV #TEMP,R0 ;: LOAD ADDRESSES
      MOV #TEMP1,R1
      MOV #TEMP2,R2
      CLRB (R2)+ ;: START CLEAN
      MOVB #252,(R0) ;: LOAD THE LOCATIONS
      MOVB (R0)+,(R1)+ ;: TEMP => TEMP1
      INCB R1 ;: MAKE IT EVEN
      MOVB (R1),TEMP ;: MORE 0'S INTO TEMP
      INCB R0 ;: MAKE IT EVEN
      MOVB (R0)+,(R1)+ ;: TEMP1 => TEMP2
      CMPB -(R2),#252 ;: CHECK IT
      BNE 1$ ;: FAILED
      TSTB TEMP ;: CHECK IT
      BEQ 2$ ;: OK, CONTINUE
1$: ERROR, T35 ;: *** INSTRUCTIONS FAILED IN MODE 2 ***
2$:

```

CNMVA.P11 17-MAR-82 14:02

PC 15436 = INSTRUCTIONS FAILED IN MODE 2.

```

1274 015442 005741      2$:  TST      -(R1)
1275 015444 005022      CLR      (R2)+      ; START CLEAN
1276 015446 012740 125252  MOV      #125252, -(R0)      ; LOAD LOCATIONS
1277 015452 012020      MOV      (R0)+, (R0)+      ; TEMP => TEMP1
1278 015454 011067 163326  MOV      (R0), TEMP      ; 0 => TEMP
1279 015460 012121      MOV      (R1)+, (R1)+      ; 125252 => TEMP2
1280 015462 024227 125252  CMP      -(R2), #125252      ; CHECK IT
1281 015466 001003      BNE      3$      ; FAILED
1282 015470 005767 163312  TST      TEMP      ; CHECK IT
1283 015474 001402      BEQ      4$      ; OK, CONTINUE
1284 015476      3$:  ERROR, T35
1285 015476 104423 015360      4$:  *** INSTRUCTIONS FAILED IN MODE 2 ***
1286 015502

```

```

;*****
.SBTTL      T36 -- TEST MODE 3 USING MOV B AND MOV
;*****

```

```

T36:  BEGINTEST, 36      ; CHECK SWITCH REGISTER OPTIONS.
MODE3: MOV      #TEMP, ADR      ; LOAD ADDRESSES
      MOV      #TEMP1, ADR1
      MOV      #TEMP2, ADR2
      MOV      #ADR, R0      ; LOAD ADDRESSES OF ADDRESSES
      MOV      #ADR1, R1
      CLRB     TEMP2      ; START CLEAN
      MOV B    #125, TEMP      ; TEMP => TEMP1
      MOV B    @ (R0)+, @ (R1)+      ; TEMP2 => TEMP
      MOV B    @ (R1)+, TEMP      ; TEMP1 => TEMP2
      MOV B    @ (R0)+, @ (R0)+      ; CHECK IT
      CMP B    #125, TEMP2      ; FAILED
      BNE     1$      ; CHECK IT
      TST B    TEMP      ; OK, CONTINUE
      BEQ     2$

```

```

1$:  ERROR, T36      ; *** INSTRUCTIONS FAILED IN MODE 3 ***

```

```

2$:  CLR      TEMP2      ; START CLEAN
      MOV      #52525, TEMP      ; LOAD LOCATIONS
      MOV      #ADR, R0      ; LOAD ADDRESSES OF ADDRESSES
      MOV      #ADR1, R1
      MOV      @ (R0)+, @ (R0)+      ; TEMP => TEMP1
      MOV      @ (R0)+, TEMP      ; TEMP2 => TEMP
      MOV      @ (R1)+, @ (R1)+      ; TEMP1 => TEMP2
      CMP      #52525, TEMP2      ; CHECK IT
      BNE     3$      ; FAILED
      TST      TEMP      ; CHECK IT
      BEQ     4$      ; OK, CONTINUE

```

```

3$:  ERROR, T36      ; *** INSTRUCTIONS FAILED IN MODE 3 ***
4$:

```

```

;*****
.SBTTL      T37 -- TEST MODE 4 USING MOV B AND MOV
;*****

```

```

T37:  BEGINTEST, 37      ; CHECK SWITCH REGISTER OPTIONS.
MODE4: CLRB     TEMP      ; START CLEAN
      MOV      #TEMP, R0      ; LOAD ADDRESSES

```

```

1312 015660 104420 000037
1313 015664 105067 163116
1314 015670 012700 001006

```

CNKXAA.P11 17-MAR-82 14:02

T37 -- TEST MODE 4 USING MOV8 AND MOV

```

1521 015674 012701 001010      MOV      #TEMP1,R1      ;
1522 015700 012702 001012      MOV      #TEMP2,R2      ;
1523 015704 005202              INC      R2              ; ADJUST THE POINTER
1524 015706 021267 163101      CMP      (R2),TEMP2+1
1525 015712 001402              BEQ      1$
1526 015714 104423 015660      ERROR,   T37              ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1527 015720 112742 000252      1$:      MOV8   #252,-(R2)      ; LOAD TEMP2
1528 015724 005201              INC      R1              ; ADJUST THE POINTERS
1529 015726 005202              INC      R2
1530 015730 114241              MOV8   -(R2),-(R1)      ; TEMP2 => TEMP1
1531 015732 005200              INC      R0              ; ADJUST THE POINTERS
1532 015734 005202              INC      R2
1533 015736 114042              MOV8   -(R0),-(R2)      ; TEMP => TEMP2
1534 015740 105200              INC8    R0              ; ADJUST THE POINTERS
1535 015742 021067 163041      CMP      (R0),TEMP+1
1536 015746 001402              BEQ      2$
1537 015750 104423 015660      ERROR,   T37              ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1538 015754 105201              2$:      INC8    R1
1539 015756 114140              MOV8   -(R1),-(R0)      ; TEMP1 => TEMP
1540 015760 122767 000252 163020      CMP8   #252,TEMP      ; CHECK IT
1541 015766 001003              BNE      3$              ; FAILED
1542 015770 105767 163016      TSTB    TEMP2              ; CHECK IT
1543 015774 001402              BEQ      4$              ; OK, CONTINUE
1544 015776
1545 015778 104423 015660      3$:      ERROR,   T37              ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1546 016002 005067 163000      4$:      CLR      TEMP      ; START CLEAN
1547 016006 012700 001006      MOV      #TEMP,R0      ; LOAD ADDRESSES
1548 016012 012701 001010      MOV      #TEMP1,R1
1549 016016 012702 001012      MOV      #TEMP2,R2
1550 016022 005722              TST      (R2)+      ; ADJUST THE POINTER
1551 016024 021267 162764      CMP      (R2),TEMP2+2
1552 016030 001402              BEQ      5$
1553 016032 104423 015660      ERROR,   T37              ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1554 016036 012742 125252      5$:      MOV      #125252,-(R2)      ; LOAD TEMP2
1555 016042 005721              TST      (R1)+      ; ADJUST THE POINTERS
1556 016044 005722              TST      (R2)+
1557 016046 014241              MOV      -(R2),-(R1)      ; TEMP2 => TEMP1
1558 016050 005720              TST      (R0)+      ; ADJUST POINTERS
1559 016052 005722              TST      (R2)+
1560 016054 014042              MOV      -(R0),-(R2)      ; TEMP => TEMP2
1561 016056 005720              TST      (R0)+      ; ADJUST THE POINTERS
1562 016060 005721              TST      (R1)+
1563 016062 014140              MOV      -(R1),-(R0)      ; TEMP1 => TEMP
1564 016064 022767 125252 162714      CMP      #125252,TEMP      ; CHECK IT
1565 016072 001003              BNE      6$              ; FAILED
1566 016074 005767 162712      TST      TEMP2              ; CHECK IT
1567 016100 001402              BEQ      7$              ; OK, CONTINUE
1568 016102
1569 016106      6$:      ERROR,   T37              ; *** INSTRUCTIONS FAILED IN MODE 4 ***
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
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
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
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
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
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

```

```

;*****
;SBTTL      T40 -- TEST MODE 5 USING MOV8 AND MOV
;*****
T40:      BEGINTEST, 40      ; CHECK SWITCH REGISTER POINTERS.

```

```

1372 016112 105067 162670
1373 016116 012737 001006 162654
1374 016124 012767 001010 162650
1375 016132 012767 001012 162644
1376 016140 012700 001000
1377 016144 012701 001002
1378 016150 012702 001004
1379 016154 005722
1380 016156 112752 000125
1381 016162 022122
1382 016164 115251
1383 016166 022022
1384 016170 115052
1385 016172 022022
1386 016174 125052
1387 016176 001402
1388 016200 104423 016106
1389 016204 022120
1390 016206 115150
1391 016210 122767 000125 162570
1392 016216 001003
1393 016220 105767 162566
1394 016224 001402
1395 016226
1396 016226 104423 016106
1397 016232 005067 162550
1398 016236 012700 001000
1399 016242 012701 001002
1400 016246 012702 001004
1401 016252 005722
1402 016254 012752 052525
1403 016260 022122
1404 016262 015251
1405 016264 022022
1406 016266 015052
1407 016270 022021
1408 016272 015150
1409 016274 022767 052525 162504
1410 016302 001003
1411 016304 005767 162502
1412 016310 001402
1413 016312
1414 016312 104423 016106
1415 016316
1416
1417
1418
1419
1420
1421
1422

MODE5: CLRB TEMP ; START CLEAN
MOV #TEMP,ADR ; LOAD ADDRESSES
MOV #TEMP1,ADR1 ;
MOV #TEMP2,ADR2 ;
MOV #ADR,R0 ; LOAD ADDRESSES OF ADDRESSES
MOV #ADR1,R1 ;
MOV #ADR2,R2 ;
TST (R2)+ ; ADJUST THE POINTER
MOVB #125,a-(R2) ; LOAD TEMP2
CMP (R1)+,(R2)+ ; ADJUST THE POINTERS
MOVB a-(R2),a-(R1) ; TEMP2 => TEMP1
CMP (R0)+,(R2)+ ; ADJUST THE POINTERS
MOVB a-(R0),a-(R2) ; TEMP => TEMP2
CMP (R0)+,(R2)+ ; ADJUST THE POINTERS
CMPB a-(R0),a-(R2) ; CHECK IT
BEQ 1$
ERROR, T40 ; *** INSTRUCTION FAILED IN MODE 5 ***
1$: CMP (R1)+,(R0)+ ; ADJUST THE POINTERS
MOVB a-(R1),a-(R0) ; TEMP1 => TEMP
CMPB #125,TEMP ; CHECK IT
BNE 2$ ; FAILED
TSTB TEMP2 ; CHECK IT
BEQ 3$ ; OK, CONTINUE
2$: ERROR, T40 ; *** INSTRUCTIONS FAILED IN MODE 5 ***
3$: CLRB TEMP ; START CLEAN
MOV #ADR,R0 ; LOAD ADDRESSES OF ADDRESSES
MOV #ADR1,R1 ;
MOV #ADR2,R2 ;
TST (R2)+ ; ADJUST THE POINTER
MOVB #52525,a-(R2) ; LOAD TEMP2
CMP (R1)+,(R2)+ ; ADJUST THE POINTERS
MOVB a-(R2),a-(R1) ; TEMP2 => TEMP1
CMP (R0)+,(R2)+ ; ADJUST THE POINTERS
MOVB a-(R0),a-(R2) ; TEMP => TEMP2
CMP (R0)+,(R1)+ ; ADJUST THE POINTERS
MOVB a-(R1),a-(R0) ; TEMP1 => TEMP
CMP #52525,TEMP ; CHECK IT
BNE 4$ ; FAILED
TSTB TEMP2 ; CHECK IT
BEQ 5$ ; OK, CONTINUE
4$: ERROR, T40 ; *** INSTRUCTIONS FAILED IN MODE 5 ***
5$:

;*****
;SBTTL T41 -- TEST MODE 6 USING MOV8 AND MOV
;*****
T41: BEGINTEST, 41 ; CHECK SWITCH REGISTER OPTIONS.
MODE6: CLRB TEMP2 ; START CLEAN
MOV #TEMP,R0 ; LOAD ADDRESSES
MOV #TEMP1,R1 ;
MOV #TEMP2,R2 ;
MOVB #252,0(R0) ; LOAD TEMP (LOW BYTE)
MOVB #252,1(R0) ; LOAD TEMP (HIGH BYTE)

```

CNKXAA.P11 17-MAR-82 14:02

T41 -- TEST MODE 6 USING MOVB AND MOV

```

1423 016356 022757 125252 162422      CMP      #125252,TEMP      ; CHECK IT
1424 016354 001012                      BNE      1$              ; FAILED
1425 016366 116062 000001 000000      MOVB     1(R0),0(R2)      ; TEMP(H) => TEMP2(L)
1426 016374 116160 000002 000005      MOVB     2(R1),5(R0)      ; TEMP2(L) => TEMP2(H)
1427 016402 022767 125252 162402      CMP      #125252,TEMP2    ; CHECK IT
1428 016410 001402                      BEQ      2$              ; OK, CONTINUE
1429 016412                      1$:      ERROR,   T41              ;: *** INSTRUCTIONS FAILED IN MODE 6 ***
(2) 016412 104423 016316
1430
1431 016416 005067 162366      2$:      CLR      TEMP1          ; START CLEAN
1432 016422 012760 052525 000000      MOV      #52525,0(R0)    ; LOAD TEMP
1433 016430 016260 177774 000002      MOV      -4(R2),2(R0)    ; TEMP => TEMP1
1434 016436 022767 052525 162344      CMP      #52525,TEMP1    ; CHECK IT
1435 016444 001402                      BEQ      3$              ; OK, CONTINUE
1436 016446 104423 016316      3$:      ERROR,   T41              ;: *** INSTRUCTIONS FAILED IN MODE 6 ***
1437 016452
1438
1439
(2)
(2)
(2) 016452 104420 000042
1440 016456 005067 162326
1441 016462 012767 001006 162310
1442 016470 012767 001010 162304
1443 016476 012767 001012 162300
1444 016504 012700 001000
1445 016510 012701 001002
1446 016514 012702 001004
1447 016520 112770 000252 000000
1448 016526 112770 177774 000002
1449 016534 122767 000252 162246
1450 016542 001402
1451 016544 104423 016452
1452 016550 012770 125252 000000 1$:      MOV      #125252,00(R0) ; LOAD TEMP
1453 016556 012770 177774 000002      MOV      -4(R2),02(R0)    ; TEMP => TEMP1
1454 016564 022767 125252 162216      CMP      #125252,TEMP1    ; CHECK IT
1455 016572 001402                      BEQ      2$              ; OK, CONTINUE
1456 016574 104423 016452      2$:      ERROR,   T42              ;: *** INSTRUCTIONS FAILED IN MODE 7 ***
1457
1458 016600
1459
(2)
(2)
(2) 016600 104420 000043
1460 016604 012700 001006
1461 016610 012701 001010
1462 016614 000277
1463 016616 105010
1464 016620 104704
1465 016622 104704 016600
1466 016626 10510
1467 016630 104704
1468 016632 104423 016600
1469 016636 112711 000377
1470 016642 104410
1471 016644 104423 016600

```

 .SBTTL T42 -- TEST MODE 7 USING MOVB AND MOV

```

T42:      BEGINTEST, 42          ;: CHECK SWITCH REGISTER OPTIONS.
MODE7:    CLR      TEMP1        ;: START CLEAN
          MOV      #TEMP,ADR     ;: LOAD ADDRESSES
          MOV      #TEMP1,ADR1
          MOV      #TEMP2,ADR2
          MOV      #ADR,R0
          MOV      #ADR1,R1
          MOV      #ADR2,R2
          MOVB     #252,00(R0)   ;: LOAD TEMP
          MOVB     -4(R2),02(R0) ;: TEMP => TEMP1
          CMPB     #252,TEMP1    ;: CHECK IT
          BEQ      1$           ;: OK, CONTINUE
          ERROR,   T42          ;: *** MODE 7 IS FAILING ***
1$:      MOV      #125252,00(R0) ;: LOAD TEMP
          MOV      -4(R2),02(R0) ;: TEMP => TEMP1
          CMP      #125252,TEMP1 ;: CHECK IT
          BEQ      2$           ;: OK, CONTINUE
          ERROR,   T42          ;: *** INSTRUCTIONS FAILED IN MODE 7 ***
2$:

```

 .SBTTL T43 -- TSTB, CLRB, AND MOVB

```

T43:      BEGINTEST, 43          ;: CHECK SWITCH REGISTER OPTIONS.
TSTB1:    MOV      #TEMP,R0     ;: LOAD ADDRESSES
          MOV      #TEMP1,R1
          SCC
          CLRB     (R0)         ;: CLEAR THE LOCATION
          CHECKCC+4             ;: CHECK FOR CC = 4
          ERROR,   T43          ;: *** INCORRECT CC BITS ***
          TSTB     (R0)         ;: CHECK IT
          CHECKCC+4             ;: CHECK FOR CC = 4
          ERROR,   T43          ;: *** INCORRECT CC BITS ***
          MOVB     #377,(R1)    ;: LOAD THE LOCATION
          CHECKCC+10            ;: CHECK FOR CC = 10
          ERROR,   T43          ;: *** INCORRECT CC BITS ***

```

```
1472 016650 105711          TSTB      (R1)          ; CHECK IT
1473 016652 104410          CHECKCC+10        ; CHECK FOR CC = 10
1474 016654 104423 016600    ERROR,  T43        ; *** INCORRECT CC BITS ***
1475 016660 010002          MOV      R0,R2        ; R2 IS NOW POINTING TO LOCATION TEMP
1476 016662 112762 000200 000000    MOVB     #200,0(R2)      ; PLACE #200 IN LOCATION TEMP
1477 016670 112241          MOVB     (R2)+,-(R1)    ; MOVE #200 TO LOCATION TEMP+1
1478 016672 026127 177777 100200    CMP      -1(R1),#100200 ; CHECK THE DATA IN LOCATION TEMP
1479 016700 001402          BEQ      4$
1480 016702 104423 016600          ERROR,  T43        ; *** MOVB INSTRUCTION FAILED ***
1481 016706 020102          4$:  CMP      R1,R2        ; CHECK THE REGISTERS FOR PROPER VALUE
1482 016710 001402          BEQ      5$
1483 016712 104423 016600          ERROR,  T43        ; *** " INSTRUCTION FAILED ***
1484 016716
1485
1486
1487 (2)
1488 (2)
1489 (2) 016716 104420 000044          ;*****
1490 016722 012701 001012          .SBTTL      T44 -- CMPB AND BISB
1491 016726 012702 001006          ;*****
1492 016732 012711 000077          T44:  BEGINTEST, 44          ;: CHECK SWITCH REGISTER OPTIONS.
1493 016736 112704 000377          CMPB1:  MOV      #TEMP2,R1      ;: LOAD ADDRESS
1494 016742 150412          MOV      #TEMP,R2        ;: PLACE 77 IN LOCATION TEMP2
1495 016744 104410          MOV      #77,(R1)        ;: R4 SHOULD CONTAIN #177777
1496 016746 104423 016716          MOVB     #377,R4        ;: LOAD LOCATION
1497 016752 120412          BISB      R4,(R2)        ;: CHECK FOR CC = 10
1498 016754 001402          CHECKCC+10        ;: *** INCORRECT CC BITS ***
1499 016756 104423 016716          ERROR,  T44        ;: CHECK COMPARE
1500 016762 121112          CMPB      R4,(R2)        ;: CONTINUE IF OK
1501 016764 100002          BEQ      2$
1502 016766 104423 016716          ERROR,  T44        ;: *** BISB OR CMPB INSTRUCTION FAILED ***
1503 017002          2$:  CMPB      (R1),(R2)        ;: CHECK IT AGAIN
1504          BPL      3$
1505          ERROR,  T44        ;: CONTINUE IF OK
1506          3$:  CMPB      (R2),(R1)        ;: *** CMPB INSTRUCTION FAILED (WRONG CC) ***
1507          BMI      4$
1508          ERROR,  T44        ;: CONTINUE IF OK
1509          4$:  ;: *** INSTR FAILED ***
1510
1511
1512 (2)
1513 (2)
1514 (2) 017002 104420 000045          ;*****
1515 017006 012703 001006          .SBTTL      T45 -- BICB AND BITB
1516 017012 112713 000377          ;*****
1517 017016 012700 001010          T45:  BEGINTEST, 45          ;: CHECK SWITCH REGISTER OPTIONS.
1518 017022 010001          BICB1:  MOV      #TEMP,R3        ;: LOAD ADDRESS
1519 017024 112721 000252          MOVB     #377,(R3)      ;: LOAD LOCATION
1520 017030 000277          MOV      #TEMP1,R0        ;: PLACE THE ADDRESS OF TEMP1 IN R0
1521 017032 146013 000000          MOV      R0,R1        ;: AND R1
1522 017036 104401          MOVB     #252,(R1)+        ;: PLACE #252 IN TEMP1
1523 017040 104423 017002          SCC
1524 017044 136113 177777          BICB      0(R0),(R3)      ;: CLEAR EVERY OTHER BIT
1525 017050 001402          CHECKCC+1          ;: CHECK FOR CC = 1
1526 017052 104423 017002          ERROR,  T45        ;: *** INCORRECT CC BITS ***
1527 017056 132713 000125          BITB      -1(R1),(R3)    ;: CHECK IT
1528 017062 104401          BEQ      4$
1529 017064 104423 017002          ERROR,  T45        ;: CONTINUE IF OK
1530 017070 154113          4$:  BITB      #125,(R3)      ;: *** BICB OR BITB INSTRUCTION FAILED ***
1531          CHECKCC+1          ;: CHECK IT
1532          ERROR,  T45        ;: CHECK FOR CC = 1
1533          BISB      -(R1),(R3)    ;: *** INCORRECT CC BITS ***
1534          ;: SET THE BITS THAT WERE CLEARED
```

CNKXAA.P11 17-MAR-82 14:02

PC 17064 = INCORRECT CC BITS.

1522	017072	100402		BMI	6\$	:	CONTINUE IF OK
1523	017074	104423	017002	ERROR,	T45	:	*** BITB OR BISB INSTRUCTION FAILED ***
1524	017100	012746	000177	6\$: MOV	#177, -(SP)	:	STORE #177 ON THE STACK
1525	017104	142613		BICB	(SP)+, (R3)	:	CLEAR ALL THE BITS EXCEPT SIGN BIT
1526	017106	104411		CHECKCC+11		:	CHECK FOR CC = 11
1527	017110	104423	017002	ERROR,	T45	:	*** INCORRECT CC BITS ***
1528	017114	132713	000377	BITB	#377, (R3)	:	CHECK IT
1529	017120	104411		CHECKCC+11		:	CHECK FOR CC = 11
1530	017122	104423	017002	ERROR,	T45	:	*** INCORRECT CC BITS ***
1531	017126	010300		MOV	R3, R0	:	PLACE THE ADDRESS OF TEMP IN R0
1532	017130	012710	001010	MOV	#TEMP1, (R0)	:	PLACE THE ADDRESS OF TEMP1 IN TEMP
1533	017134	012730	000377	MOV	#377, @ (R0)+	:	WRITE A 377 IN LOCATION TEMP1
1534	017140	000263		SEVC		:	SET V & C BITS
1535	017142	145070	000000	BICB	@-(R0), @ (R0)	:	BIT CLEAR THE CONTENTS
1536						:	OF TEMP1 TO THE CONTENTS OF TEMP1
1537	017146	104405		CHECKCC+5		:	CHECK FOR CC = 5
1538	017150	104423	017002	ERROR,	T45	:	*** INCORRECT CC BITS ***
1539	017154	022027	001010	CMP	(R0)+, #TEMP1	:	MAKE SURE THAT (R0) IS POINTING TO LOCATION TEMP1
1540	017160	001402		BEQ	8\$	:	
1541	017162	104423	017002	ERROR,	T45	:	*** BICB OR CMP INSTRUCTION FAILED IN THE SPECIFIC MO
1542	017166	005750		8\$: TST	@-(R0)	:	TEST LOCATION TEMP1
1543	017170	001402		BEQ	10\$	:	
1544	017172	104423	017002	ERROR,	T45	:	*** BICB INSTRUCTION FAILED ***
1545	017176	000257		10\$: CCC		:	
1546	017200	141010		BICB	(R0), (R0)	:	CLEAR THE LOCATION TEMP
1547	017202	104404		CHECKCC+4		:	CHECK FOR CC = 4
1548	017204	104423	017002	ERROR,	T45	:	*** INCORRECT CC BITS ***
1549						:	
1550						:	
(2)				*****		:	*****
(2)				.SBTTL	T46 -- INCB AND DECB	:	
(2)				*****		:	*****
(2)	017210	104420	000046	T46: BEGINTEST,	46	:	CHECK SWITCH REGISTER OPTIONS.
1551	017214	012704	001006	INCB1: MOV	#TEMP, R4	:	LOAD ADDRESS
1552	017220	112714	000177	MOV	#177, (R4)	:	TEMP LOCATION=177
1553	017224	000261		SEC		:	
1554	017226	105214		INCB	(R4)	:	ADD ONES INTO LOCATION
1555	017230	104413		CHECKCC+13		:	CHECK FOR CC = 13
1556	017232	104423	017210	ERROR,	T46	:	*** INCORRECT CC BITS ***
1557	017236	012714	000376	MOV	#376, (R4)	:	
1558	017242	105224		INCB	(R4)+	:	
1559	017244	104411		CHECKCC+11		:	CHECK FOR CC = 11
1560	017246	104423	017210	ERROR,	T46	:	*** INCORRECT CC BITS ***
1561	017252	105744		TSTB	-(R4)	:	DECREMENT R4 BY 1
1562	017254	005746		TST	-(SP)	:	AND SP BY 2
1563	017256	010426		MOV	R4, (SP)+	:	PLACE THE ADDRESS OF TEMP ON THE STACK
1564	017260	000241		CLC		:	CLEAR C BIT
1565	017262	105256		INCB	@-(SP)	:	INCREMENT THE CONTENTS OF LOCATION TEMP
1566	017264	104404		CHECKCC+4		:	CHECK FOR CC = 4
1567	017266	104423	017210	ERROR,	T46	:	*** INCORRECT CC BITS ***
1568	017272	123634		CMPB	@(SP)+, @ (R4)+	:	RESTORE STACK POINTER
1569	017274	000261		SEC		:	SET C BIT
1570	017276	105264	177777	INCB	-1(R4)	:	
1571	017302	104401		CHECKCC+1		:	CHECK FOR CC = 1
1572	017304	104423	017210	ERROR,	T46	:	*** INCORRECT CC BITS ***
1573	017310	124427	000001	CMPB	-(R4), #1	:	CHECK IT
1574	017314	001402		BEQ	2\$	:	CONTINUE IF OK

PC 17316 = INCB INSTRUCTION FAILED.

```

1575 017316 104423 017210          2$: ERROR, T46          ;; *** INCB INSTRUCTION FAILED ***
1576 017322 000261                SEC
1577 017324 105314                DECB (R4)          ; SUBTRACT ONES FROM LOCATION
1578 017326 104405                CHECKCC+5          ; CHECK FOR CC = 5
1579 017330 104423 017210          ERROR, T46          ;; *** INCORRECT CC BITS ***
1580 017334 105324                DECB (R4)+
1581 017336 104411                CHECKCC+11          ; CHECK FOR CC = 11
1582 017340 104423 017210          ERROR, T46          ;; *** INCORRECT CC BITS ***
1583 017344 112764 000200 177777    MOVB #200,-1(R4)
1584 017352 105344                DECB -(R4)
1585 017354 104403                CHECKCC+3          ; CHECK CC = 3
1586 017356 104423 017210          ERROR, T46          ;; *** INCORRECT CC BITS ***
1587 017362 105364 000000          DECB 0(R4)
1588 017366 104401                CHECKCC+1          ; CHECK FOR CC = 1
1589 017370 104423 017210          ERROR, T46          ;; *** INCORRECT CC BITS ***
1590 017374 126427 000000 000176    CMPB 0(R4),#176
1591 017402 001402                BEQ 3$
1592 017404 104423 017210          ERROR, T46          ;; *** DECB INSTRUCTION FAILED ***
1593 017410
1594
1595
(2)
(2)
(2) 017410 104420 000047          ;*****
.SBTTL T47 -- COMB AND NEGB          ;*****
1596 017414 012703 001006          T47: BEGINTEST, 47          ;; CHECK SWITCH REGISTER OPTIONS.
1597 017420 012704 001010          COMB1: MOV #TEMP,R3          ; LOAD ADDRESS
1598 017424 012714 000252          MOV #TEMP1,R4
1599 017430 112413                MOV #252,(R4)
1600 017432 000277                MOV (R4)+,(R3)          ; LOAD EVERY OTHER BIT
1601 017434 105113                SCC
1602 017436 104401                COMB (R3)          ; 1'S COMPLEMENT
1603 017440 104423 017410          CHECKCC+1          ; CHECK FOR CC = 1
1604 017444 122713 000125          ERROR, T47          ;; *** INCORRECT CC BITS ***
1605 017450 001402                CMPB #125,(R3)          ; CHECK IT
1606 017452 104423 017410          BEQ 2$          ; CONTINUE IF OK
1607 017456 000277                ERROR, T47          ;; *** COMB INSTRUCTION FAILED ***
1608 017460 105113                2$: SCC
1609 017462 104411                COMB (R3)          ; COMPLEMENT BACK
1610 017464 104423 017410          CHECKCC+11          ; CHECK FOR CC = 11
1611 017470 010400                ERROR, T47          ;; *** INCORRECT CC BITS ***
1612 017472 126013 177777          MOV R4,R0
1613 017476 001402                CMPB -1(R0),(R3)          ; CHECK IT
1614 017500 104423 017410          BEQ 3$          ; CONTINUE IF OK
1615 017504 112724 000377          ERROR, T47          ;; *** COMB INSTRUCTION FAILED ***
1616 017510 114413                3$: MOV #377,(R4)+
1617 017512 000277                MOVB -(R4),(R3)          ; PLACE #377 IN (R3)
1618 017514 105113                SCC
1619 017516 104405                COMB (R3)
1620 017520 104423 017410          CHECKCC+5          ; CHECK FOR CC = 5
1621                ERROR, T47          ;; *** INCORRECT CC BITS ***
1622 017524 012700 001006          NEGB1: MOV #TEMP,R0          ; LOAD ADDRESS
1623 017530 112710 000001          MOVB #1,(R0)          ; LOAD THE LOCATION
1624 017534 105410                NEGB (R0)          ; 2'S COMPLEMENT
1625 017536 104411                CHECKCC+11          ; CHECK FOR CC = 11
1626 017540 104423 017524          ERROR, NEGB1          ;; *** INCORRECT CC BITS ***
1627 017544 122710 000377          CMPB #377,(R0)          ; CHECK IT

```

1628 017550 001402
1629 017552 104423 017524
1630 017556 012710 000200
1631 017562 105410
1632 017564 104413
1633 017566 104423 017524
1634 017572 122710 000200
1635 017576 001402
1636 017600 104423 017524
1637 017604

2\$: BEQ 2\$; CONTINUE IF OK
ERROR, NEGB1 ; *** NEGB INSTRUCTION FAILED ***
MOV #200,(R0)
NEGB (R0) ; 2'S COMPLEMENT
CHECKCC+13 ; CHECK FOR CC = 13
ERROR, NEGB1 ; *** INCORRECT CC BITS ***
CMPB #200,(R0) ; CHECK IT
BEQ 3\$; CONTINUE IF OK
ERROR, NEGB1 ; *** NEGB FAILED. ***
3\$:

1638
1639
(2)
(2)

; SBTTL T50 -- ROLB AND RORB

(2) 017604 104420 000050
1640 017610 012701 001010
1641 017614 112711 000040

T50: BEGINTEST, 50 ; CHECK SWITCH REGISTER OPTIONS.
ROLB1: MOV #TEMP1,R1 ; LOAD ADDRESS
MOV #40,(R1) ; LOAD LOCATION
CCC ; CLEAR FLAGS
ROLB (R1) ; SHIFT

1642 017620 000257
1643 017622 106111
1644 017624 106111

CHECKCC+12 ; CHECK FOR CC = 12
ERROR, T50 ; *** INCORRECT CC BITS ***
CMPB #200,(R1) ; CHECK IT

1645 017626 104412
1646 017630 104423 017604
1647 017634 122711 000200

BEQ 1\$; CONTINUE IF OK
ERROR, T50 ; *** ROLB INSTRUCTION FAILED ***
1\$: ROLB (R1) ; SHIFT

1648 017640 001402
1649 017642 104423 017604
1650 017646 106111

CHECKCC+7 ; CHECK FOR CC = 7
ERROR, T50 ; *** INCORRECT CC BITS ***
ROLB (R1) ; SHIFT

1651 017650 104407
1652 017652 104423 017604
1653 017656 106111

CMPB #1,(R1) ; CHECK IT
BEQ RORB1 ; CONTINUE IF OK
ERROR, T50 ; *** ROLB FAILED ***

1654 017660 122711 000001
1655 017664 001402
1656 017666 104423 017604

RORB1: MOV #TEMP1,R2 ; LOAD ADDRESS
MOV #4,(R2) ; LOAD LOCATION
CCC ; CLEAR FLAGS
RORB (R2) ; SHIFT

1657
1658 017672 012702 001010
1659 017676 112712 000004

CMPB #1,(R2) ; CHECK IT
BEQ 1\$; CONTINUE IF OK
ERROR, RORB1 ; *** RORB INSTRUCTION FAILED ***

1660 017702 000257
1661 017704 106012
1662 017706 106012

1\$: RORB (R2) ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, RORB1 ; *** INCORRECT CC BITS ***

1663 017710 122712 000001
1664 017714 001402
1665 017716 104423 017672

RORB (R2) ; SHIFT
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, RORB1 ; *** INCORRECT CC BITS ***

1666 017722 106012
1667 017724 104407
1668 017726 104423 017672

CMPB #200,(R2) ; CHECK IT
BEQ 2\$; CONTINUE IF OK
ERROR, RORB1 ; *** RORB FAILED ***

1669 017732 106012
1670 017734 104412
1671 017736 104413 017672

2\$: ERROR, RORB1

1672 017742 122712 000200
1673 017746 001402
1674 017750 104423 017672
1675 017754
1676
1677

; SBTTL T51 -- ASLB AND ASRB

(2)
(2)
(2) 017754 104420 000051

T51: BEGINTEST, 51 ; CHECK SWITCH REGISTER OPTIONS.

1678 017760 012703 001010
 1679 017764 112713 000040
 1680 017770 000257
 1681 017772 106313
 1682 017774 106313
 1683 017776 104412
 1684 020000 104423 017754
 1685 020004 122713 000200
 1686 020010 001402
 1687 020012 104423 017754
 1688 020016 106313
 1689 020020 104407
 1690 020022 104423 017754
 1691 020026 106313
 1692 020030 104404
 1693 020032 104423 017754
 1694
 1695 020036 012704 001010
 1696 020042 012703 001012
 1697 020046 112714 000004
 1698 020052 000257
 1699 020054 106214
 1700 020056 106214
 1701 020060 122714 000001
 1702 020064 001402
 1703 020066 104423 020036
 1704 020072 106214
 1705 020074 104407
 1706 020076 104423 020036
 1707 020102 106214
 1708 020104 104404
 1709 020106 104423 020036
 1710 020112 112713 000202
 1711 020116 106213
 1712 020120 106213
 1713 020122 104411
 1714 020124 104423 020036
 1715 020130 122713 000340
 1716 020134 001402
 1717 020136 104423 020036
 1718 020142
 1719
 1720
 (2)
 (2)
 (2) 020142 104420 000052
 1721 020146 012700 001012
 1722 020152 105010
 1723 020154 000257
 1724 020156 105510
 1725 020160 104404
 1726 020162 104423 020142
 1727 020166 000261
 1728 020170 105510
 1729 020172 000261
 1730 020174 105510

ASLB1: MOV #TEMP1,R3 ; LOAD ADDRESS
 MOV #40,(R3) ; LOAD LOCATION
 CCC ; CLEAR FLAGS
 ASLB (R3) ; SHIFT
 ASLB (R3)
 CHECKCC+12 ; CHECK FOR CC = 12
 ERROR, T51 ; *** INCORRECT CC BITS ***
 CMPB #200,(R3) ; CHECK IT
 BEQ 4\$; CONTINUE IF OK
 ERROR, T51 ; *** ASLB INSTRUCTION FAILED ***
 4\$: ASLB (R3) ; SHIFT
 CHECKCC+7 ; CHECK FOR CC = 7
 ERROR, T51 ; *** INCORRECT CC BITS ***
 ASLB (R3) ; SHIFT
 CHECKCC+4 ; CHECK FOR CC = 4
 ERROR, T51 ; *** INCORRECT CC BITS ***
 ASRB1: MOV #TEMP1,R4 ; LOAD ADDRESSES
 MOV #TEMP2,R3 ;
 MOV #4,(R4) ; LOAD LOCATION
 CCC ; CLEAR FLAGS
 ASRB (R4) ; SHIFT
 ASRB (R4)
 CMPB #1,(R4) ; CHECK IT
 BEQ 2\$; CONTINUE IF OK
 ERROR, ASRB1 ; *** ASRB INSTRUCTION FAILED ***
 2\$: ASRB (R4) ; SHIFT
 CHECKCC+7 ; CHECK FOR CC = 7
 ERROR, ASRB1 ; *** INCORRECT CC BITS ***
 ASRB (R4) ; SHIFT
 CHECKCC+4 ; CHECK FOR CC = 4
 ERROR, ASRB1 ; *** INCORRECT CC BITS ***
 MOV #202,(R3) ; LOAD LOCATION
 ASRB (R3) ; SHIFT
 ASRB (R3)
 CHECKCC+11 ; CHECK FOR CC = 11
 ERROR, ASRB1 ; *** INCORRECT CC BITS ***
 CMPB #340,(R3) ; CHECK IT
 BEQ 3\$; CONTINUE IF OK
 ERROR, ASRB1 ; *** ASRB FAILED ***
 3\$:

 .SBTTL 152 -- ADCB AND SBCB

 T52: BEGINTEST, 52 ; CHECK SWITCH REGISTER OPTIONS.
 ADCB1: MOV #TEMP2,R0 ; LOAD ADDRESS
 CLRB (R0) ; CLEAR THE LOCATION
 CCC ; CLEAR FLAGS
 ADCB (R0) ; ADD C BIT = 0
 CHECKCC+4 ; CHECK FOR CC = 4
 ERROR, T52 ; *** INCORRECT CC BITS ***
 SEC ; C=1
 ADCB (R0) ; ADD C BIT=1
 SEC ; C=1
 ADCB (R0) ; AGAIN

```
1731 020176 104400          CHECKCC+0          ; CHECK FOR CC = 0
1732 020200 104423 020142    ERROR, T52          ; *** INCORRECT CC BITS ***
1733 020204 122710 000002    CMPB #2,(R0)        ; CHECK IT
1734 020210 001402          BEQ 4$              ; CONTINUE IF OK
1735 020212 104423 020142    ERROR, T52          ; *** ADCB INSTRUCTION FAILED ***
1736 020216 112710 000177    4$: MOVB #77,(R0)    ; LOAD LARGEST POSITIVE BYTE
1737 020222 000261          SEC                  ; C=1
1738 020224 105510          ADCB (R0)            ; ADD C BIT=1
1739 020226 104412          CHECKCC+12          ; CHECK FOR CC = 12
1740 020230 104423 020142    ERROR, T52          ; *** INCORRECT CC BITS ***
1741 020234 122710 000200    CMPB #200,(R0)      ; CHECK IT
1742 020240 001402          BEQ 6$              ; CONTINUE IF OK
1743 020242 104423 020142    ERROR, T52          ; *** ADCB INSTRUCTION FAILED ***
1744 020246 112710 000377    6$: MOVB #377,(R0)   ; LOAD -1
1745 020252 000261          SEC                  ; C=1
1746 020254 105510          ADCB (R0)            ; ADD C BIT=1
1747 020256 104405          CHECKCC+5           ; CHECK FOR CC = 5
1748 020260 104423 020142    ERROR, T52          ; *** INCORRECT CC BITS ***
1749
1750 020264 012701 001012    SBCB1: MOV #TEMP2,R1 ; LOAD ADDRESS
1751 020270 112711 000003    MOVB #3,(R1)        ; LOAD LOCATION
1752 020274 000257          CCC                  ; CLEAR FLAGS
1753 020276 105611          SBCB (R1)            ; SUBTRACT C BIT=0
1754 020300 104400          CHECKCC+0           ; CHECK FOR CC = 0
1755 020302 104423 020264    ERROR, SBCB1        ; *** INCORRECT CC BITS ***
1756 020306 122711 000003    CMPB #3,(R1)        ; CHECK IT
1757 020312 001402          BEQ 2$              ; CONTINUE IF OK
1758 020314 104423 020264    ERROR, SBCB1        ; *** SBCB INSTRUCTION FAILED ***
1759 020320 000261          SEC                  ; C=1
1760 020322 105611          SBCB (R1)            ; SUBTRACT C BIT=1
1761 020324 000261          SEC                  ; C=1
1762 020326 105611          SBCB (R1)            ;
1763 020330 104400          CHECKCC+0           ; CHECK FOR CC = 0
1764 020332 104423 020264    ERROR, SBCB1        ; *** INCORRECT CC BITS ***
1765 020336 122711 000001    CMPB #1,(R1)        ; CHECK IT
1766 020342 001402          BEQ 3$              ; CONTINUE IF OK
1767 020344 104423 020264    ERROR, SBCB1        ; *** SBCB INSTRUCTION FAILED ***
1768 020350 000261          SEC                  ; C=1
1769 020352 105611          SBCB (R1)            ; SUBTRACT C BIT=1
1770 020354 104404          CHECKCC+4           ; CHECK FOR CC = 4
1771 020356 104423 020264    ERROR, SBCB1        ; *** INCORRECT CC BITS ***
1772 020362 000261          SEC                  ; C=1
1773 020364 105611          SBCB (R1)            ; SUBTRACT C BIT = 1
1774 020366 104411          CHECKCC+11          ; CHECK FOR CC = 11
1775 020370 104423 020264    ERROR, SBCB1        ; *** INCORRECT CC BITS ***
1776 020374 122711 000377    CMPB #377,(R1)      ; CHECK IT
1777 020400 001402          BEQ 4$              ; CONTINUE IF OK
1778 020402 104423 020264    ERROR, SBCB1        ; *** SBCB INSTRUCTION FAILED ***
1779 020406 112711 000200    4$: MOVB #200,(R1)   ; LOAD R1
1780 020412 000261          SEC                  ; C=1
1781 020414 105611          SBCB (R1)            ; SUBTRACT C BIT = 1
1782 020416 104402          CHECKCC+2           ; CHECK FOR CC = 2
1783 020420 104423 020264    ERROR, SBCB1        ; *** INCORRECT CC BITS ***
1784
1785
```

:SBTTI. T53 -- TST, CLR, AND MOV

(2)
(2) 020424 104420 000053
1786 020430 012701 001006
1787 020434 012700 001010
1788 020440 000277
1789 020442 005010
1790 020444 104404
1791 020446 104423 020424
1792 020452 005720
1793 020454 104404
1794 020456 104423 020424
1795 020462 010040
1796 020464 012730 177777
1797 020470 016011 177776
1798 020474 104410
1799 020476 104423 020424
1800 020502 005711
1801 020504 104410
1802 020506 104423 020424
1803
1804

```
*****
T53: BEGINTEST, 53          ;; CHECK SWITCH REGISTER OPTIONS.
MOV1: MOV #TEMP,R1          ;; LOAD ADDRESSES
      MOV #TEMP1,R0
      SCC
      CLR (R0)              ;; (CLEAR THE LOCATION
      CHECKCC+4              ;; CHECK FOR CC = 4
      ERROR, T53            ;; *** INCORRECT CC BITS ***
      TST (R0)+              ;; CHECK IT
      CHECKCC+4              ;; CHECK FOR CC = 4
      ERROR, T53            ;; *** INCORRECT CC BITS ***
      MOV R0, -(R0)
      MOV #177777, @ (R0)+
      MOV -2(R0), (R1)      ;; LOAD THE LOCATION.
      CHECKCC+10            ;; CHECK FOR CC = 10
      ERROR, T53            ;; *** INCORRECT CC BITS ***
      TST (R1)              ;; CHECK IT
      CHECKCC+10            ;; CHECK FOR CC = 10
      ERROR, T53            ;; *** INCORRECT CC BITS ***
*****
```

(2)
(2) 020512 104420 000054
1805 020516 012702 001010
1806 020522 012700 001006
1807 020526 000257
1808 020530 012720 177777
1809 020534 054012
1810 020536 104410
1811 020540 104423 020512
1812 020544 022227 177777
1813 020550 001402
1814 020552 104423 020512
1815 020556 020227 001012
1816 020562 001402
1817 020564 104423 020512
1818 020570 022742 000077
1819 020574 104401
1820 020576 104423 020512
1821 020602 022722 077777
1822 020606 104413
1823 020610 104423 020512
1824 020614 022227 077777
1825 020620 104410
1826 020622 104423 020512
1827 020626 012767 052525 160156
1828 020634 012767 001012 160146
1829 020642 012704 001000
1830 020646 012714 001002
1831 020652 012734 125252
1832 020656 057432 177776
1833
1834 020662 010200
1835 020664 025027 177777
1836 020670 001402

```
*****
SBTTL T54 -- CMP AND BIS
*****
T54: BEGINTEST, 54          ;; CHECK SWITCH REGISTER OPTIONS.
CMP1: MOV #TEMP1,R2          ;; LOAD ADDRESS
      MOV #TEMP,R0           ;; PLACE THE ADDRESS OF TEMP IN R0
      CCC                    ;; CLEAR CC BITS
      MOV #177777, (R0)+     ;; PLACE #177777 IN TEMP AND INC R0 BY 2
      BIS -(R0), (R2)        ;; LOAD LOCATION
      CHECKCC+10            ;; CHECK FOR CC = 10
      ERROR, T54            ;; *** INCORRECT CC BITS ***
      CMP (R2)+, #177777     ;; CHECK COMPARE
      BEQ 2$                 ;; CONTINUE IF OK
      ERROR, T54            ;; *** CMP OR BIS INSTRUCTION FAILED ***
2$:  CMP R2, #TEMP1+2        ;; CHECK R2 TO CONTAIN ADDRESS OF TEMP1+2
      BEQ 3$
      ERROR, T54            ;; *** NO AUTO INCREMENT ***
3$:  CMP #77, -(R2)          ;; CHECK IT AGAIN
      CHECKCC+1              ;; CHECK FOR CC = 1
      ERROR, T54            ;; *** INCORRECT CC BITS ***
      CMP #77777, (R2)+
      CHECKCC+13            ;; CHECK FOR CC = 13
      ERROR, T54            ;; *** INCORRECT CC BITS ***
      CMP -(R2), #77777     ;; ONCE MORE
      CHECKCC+10            ;; CHECK FOR CC = 10
      ERROR, T54            ;; *** INCORRECT CC BITS ***
      MOV #52525, TEMP2     ;; SET EVERY OTHER BIT IN TEMP2
      MOV #TEMP2, TEMP1     ;; PLACE THE ADDRESS OF TEMP2 IN TEMP1
      MOV #ADR, R4
      MOV #ADR1, (R4)        ;; PLACE THE ADDRESS OF ADR1 IN ADR POINTED BY R4
      MOV #125252, @ (R4)+   ;; PLACE THE #125252 IN LOCATION ADR1
      BIS @-2(R4), @ (R2)+   ;; SET EVERY OTHER BIT AT LOCATION TEMP2
      AND INCREMENT R2 BY 2
      MOV R2, R0            ;; PLACE ADDRESS OF TEMP2 IN R0
      CMP @-(R0), #177777   ;; TEMP2 SHOULD CONTAIN ALL 1'S
      BEQ 4$
*****
```

```
1837 020672 104423 020512      ERROR, T54      ;; *** CMP OR BIS INSTRUCTIONS FAILED IN MODES OTHER THAN
1838 020676 020227 001012      4$: CMP R2,#TEMP1+2      ; R2 SHOULD CONTAIN THE ADDRESS OF TEMP2
1839 020702 001402              BEQ 5$
1840 020704 104423 020512      ERROR, T54      ;; *** MODE 5 IS FAILING ***
1841 020710 005040              5$: CLR -(R0)      ; PLACE A 0 IN TEMP
1842 020712 010067 160074      MOV R0,TEMP2      ; PLACE ADDRESS OF TEMP IN TEMP2
1843 020716 022020              CMP (R0)+,(R0)+      ; BUMP R0 BY 4
1844 020720 055070 000002      BIS @-(R0),@2(R0)      ; PLACE THE CONTENTS OF TEMP2 AT TEMP
1845 020724 022767 001006      CMP @TEMP,TEMP      ; TEMP SHOULD CONTAIN ITS OWN ADDRESS
1846 020732 001402              BEQ 6$
1847 020734 104423 020512      6$: ERROR, T54      ;; *** CMP OR BIS INSTRUCTIONS FAILED ***
1848 020740
1849
1850      ;*****
1851      ;SBTTL T55 -- BIC AND BIT
1852      ;*****
1853      T55: BEGINTEST, 55      ;; CHECK SWITCH REGISTER OPTIONS.
1854      BIC1: MOV #TEMP,R3      ; LOAD ADDRESS
1855      MOV #177777,(R3)      ; LOAD LOCATION
1856      MOV #ADR,R4      ; PLACE THE ADDRESS OF ADR IN R4
1857      MOV #ADR1,(R4)      ; PLACE THE ADDRESS OF ADR1 IN ADR
1858      MOV (R3),@2(R4)+      ; LOAD LOCATION ADR1 WITH #177777
1859      MOV #TEMP1,R0      ; PLACE THE ADDRESS OF TEMP1 IN R0
1860      MOV #125252,(R0)      ; SET EVERY OTHER BIT AT LOCATION TEMP1
1861      SCC
1862      BIC (R0)+,(R3)      ; CLEAR EVERY OTHER BIT
1863      CHECKCC+1      ; CHECK FOR CC = 1
1864      ERROR, T55      ;; *** INCORRECT CC BITS ***
1865      BIT -(R0),(R3)      ; CHECK IT
1866      BEQ 1$      ; CONTINUE IF OK
1867      ERROR, T55      ;; *** BIC OR BIT INSTRUCTION FAILED ***
1868      1$: BIT #52525,(R3)      ; CHECK IT
1869      CHECKCC+1      ; CHECK FOR CC = 1
1870      ERROR, T55      ;; *** INCORRECT CC BITS ***
1871      BIS 0(R0),(R3)      ; SET THE BITS THAT WERE CLEARED
1872      BMI 2$      ; CONTINUE IF OK
1873      ERROR, T55      ;; *** BIT OR BIS INSTRUCTION FAILED ***
1874      2$: MOV #77777,(R0)+      ; SET ALL THE BITS AT TEMP1 EXCEPT SIGN
1875      MOV R0,R2
1876      BIC -2(R2),(R3)      ; TRY CLEARING THE OTHER BITS
1877      CHECKCC+1      ; CHECK FOR CC = 11
1878      ERROR, T55      ;; *** INCORRECT CC BITS ***
1879      CMP R0,#TEMP1+2      ; R0 SHOULD CONTAIN THE ADDRESS OF TEMP1+2
1880      BEQ 3$
1881      ERROR, T55      ;; *** R0 WRONG ***
1882      3$: MOV R0,(R0)      ; PLACE ADDRESS OF TEMP2 IN TEMP2.
1883      SEVC      ; SET V AND C BITS.
1884      BIC -(R0),R0      ; SHOULD CLEAR R0.
1885      CHECKCC+5      ; CHECK FOR CC = 5
1886      ERROR, T55      ;; *** INCORRECT CC BITS ***
1887      BIT @-2(R4),(R3)      ; CHECK IT
1888      CHECKCC+1      ; CHECK FOR CC = 11
1889      ERROR, T55      ;; *** INCORRECT CC BITS ***
1890      MOV #125252,-(SP)      ; SET EVERY OTHER BIT ON THE STACK
1891      MOV @-2(R4),(R3)+      ; SET ALL THE BITS AT LOCATION TEMP
1892      BIC 0(SP),-(R3)      ; CLEAR EVERY OTHER BIT AT LOCATION TEMP
```

```
1890 021140 022327 052525      CMP      (R3)+,#52525      ; TEMP SHOULD CONTAIN # 52525
1891 021144 001402      BEQ      4$
1892 021146 104423 020740      ERROR,   T55      ;: *** BIC FAILED IN MODE 6 ***
1893 021152 012700 001014      4$:      MOV      #TEMP2+2,R0      ; PLACE THE ADDRESS OF TEMP2+2 IN R0
1894 021156 010340      MOV      R3,-(R0)      ; PLACE THE ADDRESS OF TEMP1 IN TEMP2
1895 021160 014330      MOV      -(R3),a(R0)+      ; MOVE # 52525 IN LOCATION TEMP1
1896 021162 000263      SEVC      ; SET V & C BITS
1897 021164 035026      BIT      a-(R0),(SP)+      ; BIT TEST TEMP1 WITH STACK AND RESTORE STACK POINTER
1898 021166 104405      CHECKCC+5      ; CHECK FOR CC = 5
1899 021170 104423 020740      ERROR,   T55      ;: *** INCORRECT CC BITS ***
1900 021174 020627 003776      CMP      SP,#STACK      ; MAKE SURE THAT THE SP IS OK
1901 021200 001402      BEQ      5$
1902 021202 104423 020740      ERROR,   T55      ;: *** STACK POINTER FOULED UP ***
1903 021206
1904
1905
1906 021206 104420 000056      ;*****
1907 021212 012704 001010      ;SBTTL      T56 -- INC AND DEC
1908 021216 012714 077777      ;*****
1909 021222 000261      T56:      BEGINTEST, 56      ;: CHECK SWITCH REGISTER OPTIONS
1910 021224 005214      INC1:      MOV      #TEMP1,R4      ;: LOAD ADDRESS
1911 021226 104413      MOV      #77777,(R4)      ;: TEMP1 = 77777
1912 021230 104423 021206      SEC      ;:
1913 021234 012714 177776      INC      (R4)      ;: ADD ONES INTO LOCATION
1914 021240 012700 001006      CHECKCC+13      ;: CHECK FOR CC = 13
1915 021244 005214      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1916 021246 104411      MOV      #177776,(R4)
1917 021250 104423 021206      MOV      #TEMP,R0      ;: R0 IS POINTING TO LOCATION TEMP
1918 021254 005214      INC      (R4)
1919 021256 104405      CHECKCC+11      ;: CHECK CC = 11.
1920 021260 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1921 021264 005214      INC      (R4)
1922 021266 104401      CHECKCC+5      ;: CHECK FOR CC = 5
1923 021270 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1924 021274 026427 000000 000001      CMP      0(R4),#1      ;: CHECK IT
1925 021302 001402      BEQ      4$      ;: CONTINUE IF OK
1926 021304 104423 021206      ERROR,   T56      ;: *** INC INSTRUCTION FAILED ***
1927 021310 000261      4$:      SEC
1928 021312 005314      DEC      (R4)      ;: SUBTRACT ONES FROM LOCATION
1929 021314 104405      CHECKCC+5      ;: CHECK FOR CC = 5
1930 021316 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1931 021322 005314      DEC      (R4)
1932 021324 104411      CHECKCC+11      ;: CHECK CC = 11.
1933 021326 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1934 021332 012714 100000      MOV      #100000,(R4)
1935 021336 005314      DEC      (R4)
1936 021340 104403      CHECKCC+3      ;: CHECK FOR CC = 3
1937 021342 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1938 021344 005314      DEC      (R4)
1939 021350 104401      CHECKCC+1      ;: CHECK FOR CC = 1
1940 021352 104423 021206      ERROR,   T56      ;: *** INCORRECT CC BITS ***
1941
1942 021352 104423 021206      ;*****
1943 021352 104423 021206      ;SBTTL      T57 -- COM AND NEG
```

```

(2)
(2) 021356 104420 000057
1942 021362 012703 001010
1943 021366 012713 125252
1944 021372 000277
1945 021374 005163 000000
1946 021400 104401
1947 021402 104423 021356
1948 021406 022713 052525
1949 021412 001402
1950 021414 104423 021356
1951 021420 000277
1952 021422 005123
1953 021424 104411
1954 021426 104423 021356
1955 021432 022743 125252
1956 021436 001402
1957 021440 104423 021356
1958 021444 010300
1959 021446 012710 177777
1960 021452 000277
1961 021454 005110
1962 021456 104405
1963 021460 104423 021356
1964
1965 021464 012704 001010
1966 021470 012724 000001
1967 021474 010402
1968 021476 012762 100000 000000
1969 021504 005444
1970 021506 104411
1971 021510 104423 021464
1972 021514 022724 177777
1973 021520 001402
1974 021522 104423 021464
1975 021526 016444 000000
1976 021532 005411
1977 021534 104413
1978 021536 104423 021464
1979 021542 026214 000000
1980 021546 001402
1981 021550 104423 021464
1982 021554
1983
1984
(2)
(2)
(2) 021554 104420 000060
1985 021560 012701 001012
1986 021564 012711 020000
1987 021570 000257
1988 021572 006121
1989 021574 006141
1990 021576 104412
1991 021600 104423 021554
1992 021604 022711 100000

*****
T57: BEGINTEST, 57 ; CHECK SWITCH REGISTER OPTIONS.
COM1: MOV #TEMP1,R3 ; LOAD ADDRESS
MOV #125252,(R3) ; LOAD EVERY OTHER BIT
SCC
COM 0(R3) ; 1'S COMPLEMENT
CHECKCC+1 ; CHECK FOR CC = 1
ERROR, T57 ; *** INCORRECT CC BITS ***
CMP #52525,(R3) ; CHECK IT
BEQ 2$ ; CONTINUE IF OK
ERROR, T57 ; *** COM INSTRUCTION FAILED ***
2$: SCC
COM (R3)+ ; COMPLEMENT BACK
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T57 ; *** INCORRECT CC BITS ***
CMP #125252,-(R3) ; CHECK IT
BEQ 3$ ; CONTINUE IF OK
ERROR, T57 ; *** COM INSTRUCTION FAILED ***
3$: MOV R3,R0 ; R0 IS NOW POINTING TO LOCATION TEMP1
MOV #177777,(R0)
SCC
COM (R0)
CHECKCC+5 ; CHECK FOR CC = 5
ERROR, T57 ; *** INCORRECT CC BITS ***
NEG1: MOV #TEMP1,R4 ; LOAD ADDRESS
MOV #1,(R4)+ ; LOAD THE LOCATION
MOV R4,R2
MOV #100000,0(R2)
NEG -(R4) ; 2'S COMPLEMENT
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, NEG1 ; *** INCORRECT CC BITS ***
CMP #177777,(R4)+ ; CHECK IT
BEQ 2$ ; CONTINUE IF OK
ERROR, NEG1 ; *** NEG INSTRUCTION FAILED ***
2$: MOV 0(R4),-(R4) ; TEMP1 HOLDS LARGEST NEGATIVE NUMBER
NEG (R4) ; 2'S COMPLEMENT
CHECKCC+13 ; CHECK FOR CC = 13
ERROR, NEG1 ; *** INCORRECT CC BITS ***
CMP 0(R2),(R4) ; CHECK IT
BEQ 3$ ; CONTINUE IF OK
ERROR, NEG1 ; *** NEG FAILED. ***
3$:
*****
.SBTTL T60 -- ROL AND ROR
*****
T60: BEGINTEST, 60 ; CHECK SWITCH REGISTER OPTIONS.
ROL1: MOV #TEMP2,R1 ; LOAD ADDRESS
MOV #20000,(R1) ; LOAD LOCATION
CCC ; CLEAR FLAGS
ROL (R1)+ ; SHIFT
ROL -(R1)
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, T60 ; *** INCORRECT CC BITS ***
CMP #100000,(R1) ; CHECK IT

```

```

1993 021610 001402      BEQ      1$      ; CONTINUE IF OK
1994 021612 104423 021554 1$: ERROR, T60      ; *** ROL INSTRUCTION FAILED ***
1995 021616 006161 000000      ROL      0(R1)      ; SHIFT
1996 021622 104407      CHECKCC+7      ; CHECK FOR CC = 7
1997 021624 104423 021554      ERROR, T60      ; *** INCORRECT CC BITS ***
1998 021630 010102      MOV      R1,R2      ; R2 IS NOW POINTING TO LOCATION TEMP2
1999 021632 006112      ROL      (R2)      ; SHIFT
2000 021634 022711 000001      CMP      #1,(R1)      ; CHECK IT
2001 021640 001402      BEQ      ROR1      ; CONTINUE IF OK
2002 021642 104423 021554      ERROR, T60      ; *** ROL FAILED. ***
2003
2004 021646 012702 001012      ROR1: MOV      #TEMP2,R2      ; LOAD ADDRESS
2005 021652 012712 000004      MOV      #4,(R2)      ; LOAD LOCATION
2006 021656 000257      CCC      ; CLEAR FLAGS
2007 021660 006012      ROR      (R2)      ; SHIFT
2008 021662 006012      ROR      (R2)      ;
2009 021664 022712 000001      CMP      #1,(R2)      ; CHECK IT
2010 021670 001402      BEQ      1$      ; CONTINUE IF OK
2011 021672 104423 021646      ERROR, ROR1      ; *** ROR INSTRUCTION FAILED ***
2012 021676 006012      1$: ROR      (R2)      ; SHIFT
2013 021700 104407      CHECKCC+7      ; CHECK FOR CC = 7
2014 021702 104423 021646      ERROR, ROR1      ; *** INCORRECT CC BITS ***
2015 021706 006012      ROR      (R2)      ; SHIFT
2016 021710 104412      CHECKCC+12      ; CHECK FOR CC = 12
2017 021712 104423 021646      ERROR, ROR1      ; *** INCORRECT CC BITS ***
2018 021714 022712 100000      CMP      #100000,(R2)      ; CHECK IT
2019 021722 001402      BEQ      2$      ; CONTINUE IF OK
2020 021724 104423 021646      ERROR, ROR1      ; *** ROR FAILED. ***
2021
2022
2023 ;*****
(2) ;SBTTL      T61 -- ASL AND ASR
(2) ;*****
(2) 021730 104420 000061      T61: BEGINTEST, 61      ; CHECK SWITCH REGISTER OPTIONS.
2024 021734 012703 001012      ASL1: MOV      #TEMP2,R3      ; LOAD ADDRESS
2025 021740 012713 020000      MOV      #20000,(R3)      ; LOAD LOCATION
2026 021744 000257      CCC      ; CLEAR FLAGS
2027 021746 006313      ASL      (R3)      ; SHIFT
2028 021750 006313      ASL      (R3)      ;
2029 021752 104412      CHECKCC+12      ; CHECK FOR CC = 12
2030 021754 104423 021730      ERROR, T61      ; *** INCORRECT CC BITS ***
2031 021760 022713 100000      CMP      #100000,(R3)      ; CHECK IT
2032 021764 001402      BEQ      4$      ; CONTINUE IF OK
2033 021766 104423 021730      ERROR, T61      ; *** ASL INSTRUCTION FAILED ***
2034 021772 006313      4$: ASL      (R3)      ; SHIFT
2035 021774 104407      CHECKCC+7      ; CHECK FOR CC = 7
2036 021776 104423 021730      ERROR, T61      ; *** INCORRECT CC BITS ***
2037 022002 006313      ASL      (R3)      ; SHIFT
2038 022004 104404      CHECKCC+4      ; CHECK FOR CC = 4
2039 022006 104423 021730      ERROR, T61      ; *** INCORRECT CC BITS ***
2040
2041 022012 012704 001012      ASR1: MOV      #TEMP2,R4      ; LOAD ADDRESSES
2042 022016 012703 001006      MOV      #TEMP,R3      ;
2043 022022 012714 000004      MOV      #4,(R4)      ; LOAD LOCATION
2044 022026 000257      CCC      ; CLEAR FLAGS
2045 022030 006214      ASR      (R4)      ; SHIFT

```

PC 22006 = INCORRECT CC BITS.

2046 022032 006214
2047 022034 022714 000001
2048 022040 001402
2049 022042 104423 022012
2050 022046 006214
2051 022050 104407
2052 022052 104423 022012
2053 022056 006214
2054 022060 104404
2055 022062 104423 022012
2056 022066 012713 100002
2057 022072 006213
2058 022074 006213
2059 022076 104411
2060 022100 104423 022012
2061 022104 022713 160000
2062 022110 001402
2063 022112 104423 022012
2064 022116

2\$:

```

ASR      (R4)
CMP      #1,(R4)
BEQ      2$
ERROR,   ASR1
ASR      (R4)
CHECKCC+7
ERROR,   ASR1
ASR      (R4)
CHECKCC+4
ERROR,   ASR1
MOV      #100002,(R3)
ASR      (R3)
ASR      (R3)
CHECKCC+11
ERROR,   ASR1
CMP      #160000,(R3)
BEQ      3$
ERROR,   ASR1

```

```

: CHECK IT
: CONTINUE IF OK
: *** ASR INSTRUCTION FAILED ***
: SHIFT
: CHECK FOR CC = 7
: *** INCORRECT CC BITS ***
: SHIFT
: CHECK FOR CC = 4
: *** INCORRECT CC BITS ***
: LOAD LOCATION
: SHIFT
: CHECK FOR CC = 11
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** ASR FAILED ***

```

3\$:

2065
2066
(2)
(2)
(2) 022116 104420 000062
2067 022122 012700 001006
2068 022126 005010
2069 022130 000257
2070 022132 005510
2071 022134 104404
2072 022136 104423 022116
2073 022142 000261
2074 022144 005510
2075 022146 000261
2076 022150 005510
2077 022152 104400
2078 022154 104423 022116
2079 022160 022710 000002
2080 022164 001402
2081 022166 104423 022116
2082 022172 012700 077777
2083 022176 000261
2084 022200 005510
2085 022202 104412
2086 022204 104423 022116
2087 022210 022710 100000
2088 022214 001402
2089 022216 104423 022116
2090 022222 012710 177777
2091 022226 000261
2092 022230 005510
2093 022232 104405
2094 022234 104423 022116
2095
2096 022240 012701 001006
2097 022244 012711 000003
2098 022250 000257

```

:*****
.SBTTL      T62 -- ADC AND SBC
:*****
T62:  BEGINTEST, 62
ADC1:  MOV      #TEMP,P0
      CLR      (R0)
      CCC
      ADC      (R0)
      CHECKCC+4
      ERROR,   T62
      SEC
      ADC      (R0)
      SEC
      ADC      (R0)
      CHECK    +0
      ERROR,   T62
      CMP      #2,(R0)
      BEQ      4$
      ERROR,   T62
      MOV      #77777,(R0)
      SEC
      ADC      (R0)
      CHECKCC+12
      ERROR,   T62
      CMP      #100000,(R0)
      BEQ      6$
      ERROR,   T62
      MOV      #-1,(R0)
      SEC
      ADC      (R0)
      CHECKCC+5
      ERROR,   T62
SBC1:  MOV      #TEMP,R1
      MOV      #3,(R1)
      CCC

```

```

: CHECK SWITCH REGISTER OPTIONS.
: LOAD ADDRESS
: CLEAR THE LOCATION
: CLEAR FLAGS
: ADD C BIT = 0
: CHECK FOR CC = 4
: *** INCORRECT CC BITS ***
: C=1
: ADD C BIT=1
: C=1
: AGAIN
: CHECK FOR CC = 0
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** ADC INSTRUCTION FAILED ***
: LOAD LARGEST POSITIVE NUMBER
: C=1
: ADD C BIT=1
: CHECK FOR CC = 12
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** ADC INSTRUCTION FAILED ***
: LOAD -1
: C=1
: ADD C BIT=1
: CHECK FOR CC = 5
: *** INCORRECT CC BITS ***
: LOAD ADDRESS
: LOAD LOCATION
: CLEAR FLAGS

```

```
2099 022252 005611 SBC (R1) ; SUBTRACT C BIT=0
2100 022254 104400 CHECKCC+0 ; CHECK FOR CC = 0
2101 022256 104423 022240 ERROR, SBC1 ; *** INCORRECT CC BITS ***
2102 022262 022711 000003 CMP #3,(R1) ; CHECK IT
2103 022266 001402 BEQ 2$ ; CONTINUE IF OK
2104 022270 104423 022240 ERROR, SBC1 ; *** SBC INSTRUCTION FAILED ***
2105 022274 000261 2$: SEC ; C=1
2106 022276 005611 SBC (R1) ; SUBTRACT C BIT=1
2107 022300 000261 SEC ; C=1
2108 022302 005611 SBC (R1) ;
2109 022304 104400 CHECKCC+0 ; CHECK FOR CC = 0
2110 022306 104423 022240 ERROR, SBC1 ; *** INCORRECT CC BITS ***
2111 022312 022711 000001 CMP #1,(R1) ; CHECK IT
2112 022316 001402 BEQ 3$ ;
2113 022320 104423 022240 ERROR, SBC1 ; *** SBC INSTRUCTION FAILED ***
2114 022324 000261 3$: SEC ; C=1
2115 022326 005611 SBC (R1) ; SUBTRACT C BIT=1
2116 022330 104404 CHECKCC+4 ; CHECK FOR CC = 4
2117 022332 104423 022240 ERROR, SBC1 ; *** INCORRECT CC BITS ***
2118 022336 000261 SEC ; C=1
2119 022340 005611 SBC (R1) ; SUBTRACT C BIT = 1
2120 022342 104411 CHECKCC+11 ; CHECK FOR CC = 11
2121 022344 104423 022240 ERROR, SBC1 ; *** INCORRECT CC BITS ***
2122 022350 022711 177777 CMP #-1,(R1) ; CHECK IT
2123 022354 001402 BEQ 4$ ; CONTINUE IF OK
2124 022356 104423 022240 ERROR, SBC1 ; *** SBC INSTRUCTION FAILED ***
2125 022362 012711 100000 4$: MOV #100000,(R1) ; LOAD R1
2126 022366 000261 SEC ; C=1
2127 022370 005611 SBC (R1) ; SUBTRACT C BIT = 1
2128 022372 104402 CHECKCC+2 ; CHECK FOR CC = 2
2129 022374 104423 022240 ERROR, SBC1 ; *** INCORRECT CC BITS ***
2130
2131 ;*****
(2) .SBTTL T63 -- SXT, SWAB, AND XOR
(2) ;*****
(2) 022400 104420 000063 T63: BEGINTEST, 63 ; CHECK SWITCH REGISTER OPTIONS.
2132 022404 012702 001010 SXT1: MOV #TEMP1,R2 ; LOAD ADDRESS
2133 022410 005012 CLR (R2) ; CLEAR LOCATIONS
2134 022412 000277 ORC
2135 022414 000254 CLNZ
2136 022416 006712 SXT (R2) ; SIGN EXTEND
2137 022420 104405 CHECKCC+5 ; CHECK FOR CC = 5
2138 022422 104423 022400 ERROR, T63 ; *** INCORRECT CC BITS ***
2139 022426 005712 TST (R2) ; LOCATION SHOULD STILL BE 0
2140 022430 001402 BEQ 2$ ; CONTINUE IF OK
2141 022432 104423 022400 ERROR, T63 ; *** SXT INSTRUCTION FAILED ***
2142 022436 000273 2$: SENVC ; SET N, V & C BITS
2143 022440 006712 SXT (R2) ; SIGN EXTEND
2144 022442 104411 CHECKCC+11 ; CHECK FOR CC = 11
2145 022444 104423 022400 ERROR, T63 ; *** INCORRECT CC BITS ***
2146 022450 022712 177777 CMP #-1,(R2) ; LOCATION SHOULD NOW HAVE -1
2147 022454 001402 BEQ SWAB1 ; CONTINUE IF OK
2148 022456 104423 022400 ERROR, T63 ; *** SXT FAILED ***
2149
2150 022462 012703 001012 SWAB1: MOV #TEMP2,R3 ; LOAD ADDRESS
2151 022466 012713 125125 MOV #125125,(R3) ; LOAD BIT PATTERN INTO LOCATION
```

CNKXAA.P11 17-MAR-82 14:02

PC 22456 = SXT FAILED.

```

2152 022472 000277      SCC
2153 022474 000250      CLN
2154 022476 000313      SWAB      (R3)      ; SWAP BYTES OF LOCATIONS
2155 022500 104410      CHECKCC+10      ; CHECK FOR CC = 10
2156 022502 104423 022462  ERROR, SWAB1      ;: *** INCORRECT CC BITS ***
2157 022506 022713 052652  CMP      #52652,(R3)      ; CHECK IT
2158 022512 001402      BEQ      1$      ; CONTINUE IF OK
2159 022514 104423 022462  ERROR, SWAB1      ;: *** SWAB INSTRUCTION FAILED ***
2160 022520 012713 000377      1$: MOV      #377,(R3)
2161 022524 000277      SCC
2162 022526 000244      CLZ
2163 022530 000363 000000      SWAB      0(R3)
2164 022534 104404      CHECKCC+4      ; CHECK FOR CC = 4
2165 022536 104423 022462  ERROR, SWAB1      ;: *** INCORRECT CC BITS ***
2166 022542 022713 177400  CMP      #177400,(R3)
2167 022546 001402      BEQ      XOR1
2168 022550 104423 022462  ERROR, SWAB1      ;: *** SWAB FAILED. ***
2169
2170 022554 012704 177777      XOR1: MOV      #-1,R4      ; LOAD LOCATIONS
2171 022560 012767 177777 156222 MOV      #-1,TEMP1
2172 022566 000277      SCC
2173 022570 074467 156214      XOR      R4,TEMP1      ; SHOULD PRODUCE 0'S IN TEMP1
2174 022574 104405      CHECKCC+5      ; CHECK FOR CC = 5
2175 022576 104423 022554  ERROR, XOR1      ;: *** INCORRECT CC BITS ***
2176 022602 012767 077777 156200 MOV      #77777,TEMP1
2177 022610 012700 001010      MOV      #TEMP1,R0      ; PLACE THE ADDRESS OF TEMP1 IN R0
2178 022614 000263      SEVC      ; SET V & C BITS
2179 022616 000244      CLZ
2180 022620 074410      XOR      R4,(R0)
2181 022622 104411      CHECKCC+11      ; CHECK FOR CC = 11
2182 022624 104423 022554  ERROR, XOR1      ;: *** INCORRECT CC BITS ***
2183 022630 012701 125252      MOV      #125252,R1      ; LOAD LOCATIONS
2184 022634 012720 052525      MOV      #52525,(R0)+
2185 022640 000277      SCC
2186 022642 074140      XOR      R1,-(R0)      ; SHOULD PRODUCE ALL 1'S IN TEMP1
2187 022644 104411      CHECKCC+11      ; CHECK FOR CC = 11
2188 022646 104423 022554  ERROR, XOR1      ;: *** INCORRECT CC BITS ***
2189 022652 022767 177777 156130 CMP      #-1,TEMP1      ; CHECK IT
2190 022660 001402      BEQ      1$      ; CONTINUE IF OK
2191 022662 104423 022554  ERROR, XOR1      ;: *** XOR FAILED. ***
2192 022666
2193
2194
(2)
(2)
(2) 022666 104420 000064      ;*****
;SBTTL      T64 -- ADD, SUB, AND SOB
;*****
T64: BEGINTEST, 64      ;: CHECK SWITCH REGISTER OPTIONS.
ADD1: MOV      #TEMP2,R0      ; LOAD ADDRESSES
      MOV      #TEMP,R1
      MOV      #21421,TEMP2      ; LOAD LOCATIONS
      MOV      (R0),(R1)
      ADD      (R0),(R1)      ; ADD
      CHECKCC+0      ; CHECK FOR CC = 0
2201 022716 104423 022666  ERROR, T64      ;: *** INCORRECT CC BITS ***
2202 022722 022767 043042 156056 CMP      #43042,TEMP
2203 022730 001402      BEQ      1$      ; CHECK IT
2204 022732 104423 022666  ERROR, T64      ; CONTINUE IF OK
;: *** ADD INSTRUCTION FAILED ***

```

PC 22732 = ADD INSTRUCTION FAILED.

2205	022736	005010		1\$:	CLR	(R0)	; CLEAR LOCATION TEMP2
2206	022740	060020			ADD	R0,(R0)+	; PLACE THE ADDRESS OF TEMP2 IN TEMP2
2207	022742	024027	001014		CMP	-(R0),#TEMP2+2	; CHECK IT.
2208	022746	001402			BEG	2\$	
2209	022750	104423	022666		ERROR,	T64	:: *** ADD INSTRUCTION FAILED IN MODE 2 ***
2210	022754	012757	156357	156030	2\$:	MOV	#-21421,TEMP2
2211	022762	012011			MOV	(R0)+,(R1)	; LOAD LOCATIONS
2212	022764	064011			ADD	-(R0),(R1)	; ADD
2213	022766	104411			CHECKCC+11		; CHECK FOR CC = 11
2214	022770	104423	022666		ERROR,	T64	:: *** INCORRECT CC BITS ***
2215	022774	022767	134736	156004		CMP	#-43042,TEMP
2216	023002	001402			BEQ	3\$	; CHECK IT
2217	023004	104423	022666		ERROR,	T64	; CONTINUE IF OK
2218	023010	012767	100000	155774	3\$:	MOV	#100000,TEMP2
2219	023016	011061	000000		MOV	(R0),0(R1)	; *** ADD INSTRUCTION FAILED ***
2220	023022	066011	000000		ADD	0(R0),(R1)	; LOAD LOCATIONS
2221	023026	104407			CHECKCC+7		; ADD SHOULD RESULT AS 0'S
2222	023030	104423	022666		ERROR,	T64	; CHECK FOR CC=7
2223	023034	012767	021421	155746		MOV	#21421,TEMP1
2224	023042	012760	001010	000000		MOV	#TEMP1,0(R0)
2225	023050	012711	156357		MOV	#-21421,(R1)	; LOAD LOCATION TEMP1
2226	023054	010004			MOV	R0,R4	; PLACE THE ADDRESS OF TEMP1 IN TEMP2
2227	023056	067411	000000		ADD	@0(R4),(R1)	; LOAD LOCATION TEMP
2228	023062	104405			CHECKCC+5		; MAKE R4 POINT TO LOCATION TEMP2
2229	023064	104423	022666		ERROR,	T64	; ADD SHOULD RESULT AS 0'S
2230	023070	005430			NEG	@(R0)+	; CHECK FOR CC=5
2231	023072	012746	021421		MOV	#21421,-(SP)	; *** INCORRECT CC BITS ***
2232	023076	065066	000000		ADD	@-(R0),0(SP)	; NEGATE THE CONTENTS OF TEMP1
2233	023102	104405			CHECKCC+5		; PLACE # 21421 ON THE STACK
2234	023104	104423	022666		ERROR,	T64	; ADD, SHOULD=0'S
2235	023110	005726			TST	(SP)+	; CHECK FOR CC=5
2236							; *** INCORRECT CC BITS ***
2237	023112	001402			BEQ	4\$	; CHECK THE STACK TO CONTAIN 0, ALSO
2238	023114	104423	022666		ERROR,	T64	; RESTORE THE STACK PCINTER
2239	023120	012767	137777	155664	4\$:	MOV	#137777,TEMP2
2240	023126	062767	137777	155656		ADD	#137777,TEMP2
2241	023134	104403			CHECKCC+3		; *** ADD INSTRUCTION FAILED IN MODE 5 ***
2242	023136	104423	022666		ERROR,	T64	; CHECK CC=3
2243	023142	022767	077776	155642		CMP	#77776,TEMP2
2244	023150	001402			BEQ	SUB1	; *** INCORRECT CC BITS ***
2245	023152	104423	022666		ERROR,	T64	; *** ADD FAILED ***
2246							
2247	023156	012702	001006		SUB1:	MOV	#TEMP,R2
2248	023162	012703	001010			MOV	#TEMP1,R3
2249	023166	012767	021421	155612		MOV	#21421,TEMP
2250	023174	012767	156357	155606		MOV	#-21421,TEMP1
2251	023202	161213			SUB	(R2),(R3)	; LOAD ADDRESSES
2252	023204	104410			CHECKCC+10		; LOAD LOCATIONS
2253	023206	104423	023156		ERROR,	SUB1	; RESULT SHOULD=-43042
2254	023212	022767	134736	155570		CMP	#-43042,TEMP1
2255	023220	001402			BEQ	1\$	; CHECK FOR CC = 10
2256	023222	104423	023156		ERROR,	SUB1	; *** INCORRECT CC BITS ***
2257	023226	012767	021421	155554	1\$:	MOV	#21421,TEMP1
2258	023234	161213			SUB	(R2),(R3)	; CHECK IT
2259	023236	001402			BEQ	2\$	; CONTINUE IF OK
2260	023240	104423	023156		ERROR,	SUB1	; *** SUB INSTRUCTION FAILED ***

PC 23240 = SUB INSTRUCTION FAILED.

2261	023244	012767	177777	155536	2\$:	MOV	#-1,TEMP1	:	LOAD LOCATIONS
2262	023252	012767	077777	155526		MOV	#77777,TEMP	:	LOAD LOCATIONS
2263	023260	161312				SUB	(R3),(R2)	:	RESULT SHOULD GIVE 100000 AND OVERFLOW
2264	023262	104413				CHECKCC+13		:	CHECK FOR CC = 13
2265	023264	104423	023156			ERROR, SUB1		:	*** INCORRECT CC BITS ***
2266	023270	022767	100000	155510		CMP	#100000,TEMP	:	CHECK IT
2267	023276	001402				BEQ	3\$	:	CONTINUE IF OK
2268	023300	104423	023156			ERROR, SUB1		:	*** SUB INSTRUCTION FAILED ***
2269	023304	012712	177777		3\$:	MOV	#-1,(R2)		
2270	023310	161312				SUB	(R3),(R2)		
2271	023312	104404				CHECKCC+4		:	CHECK FOR CC = 4
2272	023314	104423	023156			ERROR, SUB1		:	*** INCORRECT CC BITS ***
2273	023320	012767	077777	155460		MOV	#77777,TEMP		
2274	023326	162767	077777	155452		SUB	#77777,TEMP		
2275	023334	104404				CHECKCC+4		:	CHECK FOR CC=4
2276	023336	104423	023156			ERROR, SUB1		:	*** INCORRECT CC BITS ***
2277	023342	005767	155440			TST	TEMP		
2278	023346	001402				BEQ	SOB1	:	TEMP SHOULD BE =0
2279	023350	104423	023156			ERROR, SUB1		:	*** SUB INSTRUCTION FAILED ***
2280									
2281	023354	012700	000012		SOB1:	MOV	#10.,R0	:	LOAD REGISTERS
2282	023360	005001				CLR	R1	:	
2283	023362	005201			1\$:	INC	R1	:	KEEP COUNT
2284	023364	020127	000012			CMP	R1,#10.		
2285	023370	003402				BLE	2\$		
2286	023372	104423	023354			ERROR, SOB1		:	*** SOB INSTRUCTION FAILED ***
2287	023376	000277			2\$:	SCC			
2288	023400	077010				SOB	R0,1\$	:	SUB. 1 FROM REG. 0, GO BACK TO 1\$
2289	023402	104417				CHECKCC+17		:	CHECK FOR CC = 17
2290	023404	104423	023354			ERROR, SOB1		:	*** INCORRECT CC BITS ***
2291	023410	005700				TST	R0	:	REG. 0 = 0 ?
2292	023412	001402				BEQ	3\$	:	NO, FAILED
2293	023414	104423	023354			ERROR, SOB1		:	*** SOB INSTRUCTION FAILED ***
2294	023420	022701	000012		3\$:	CMP	#10.,R1	:	DID IT GO THRU 10 TIMES ?
2295	023424	001402				BEQ	4\$	:	CONTINUE IF OK
2296	023426	104423	023354			ERROR, SOB1		:	*** SOB INSTRUCTION FAILED ***
2297	023432	012704	000010		4\$:	MOV	#10,R4	:	PLACE #10 IN R4
2298	023436	077401			5\$:	SOB	R4,5\$	:	STAY HERE UNTILL R4 = 0
2299	023440	005704				TST	R4		
2300	023442	001402				BEQ	6\$	:	CONTINUE IF OK
2301	023444	104423	023354			ERROR, SOB1		:	*** SOB FAILED ***
2302	023450				6\$:				

```

*****
.SBTTL      T65 -- MTPS AND MFPS
*****
T65:  BEGINTEST, 65      ;; CHECK SWITCH REGISTER OPTIONS.
TPSW1: MOV      #TEMP,R0  ;; PUT THE ADDRESS OF TEMP IN R0
      MOV      #TEMP1,R1  ;; PUT THE ADDRESS OF TEMP1 IN R1
      MOV      #177777,(R1)
      CLR      (R0)
      MTPS     (R0)
      CHECKCC+0
      ERROR,   T65
      MFPS     (R1)
      CHECKCC+4

```

(2)	023450	104420	000065
2305	023454	012700	001006
2306	023460	012701	001010
2307	023464	012711	177777
2308	023470	005010	
2309	023472	106410	
2310	023474	104400	
2311	023476	104423	023450
2312	023502	106711	
2313	023504	104404	

PC 23506 = INCORRECT CC BITS.

2314	023506	104423	023450	ERROR, T65	:: *** INCORRECT CC BITS ***
2315	023512	022711	177400	CMP #177400,(R1)	:: CHECK TEMP1 TO MAKE SURE THAT ONLY
2316					:: THE LOWER BYTE WAS AFFECTED BY MFPS
2317	023516	001402		BEQ 1\$	
2318	023520	104423	023450	ERROR, T65	:: *** MTPS OR MFPS INSTRUCTION FAILED ***
2319	023524	005011		CLR (R1)	
2320	023526	106427	000337	MTPS #337	:: SET PSW = 317 SINCE T BIT CAN NOT BE SET BY MTPS
2321	023532	104417		CHECKCC+17	:: CHECK FOR CC = 17
2322	023534	104423	023450	ERROR, T65	:: *** INCORRECT CC BITS ***
2323	023540	106767	155244	MFPS TEMP1	:: MOVE PSW TO TEMP1
2324	023544	104411		CHECKCC+11	:: CHECK FOR CC = 11 (C BIT SHOULD NOT BE EFFECTED BY MFP
2325	023546	104423	023450	ERROR, T65	:: *** INCORRECT CC BITS ***
2326	023552	022767	000317	CMP #317,TEMP1	
2327	023560	001402		BEQ 2\$	
2328	023562	104423	023450	ERROR, T65	:: *** MFPS INSTRUCTION FAILED IN MODE 6 ***
2329	023566				
2330					
2331					
(2)					
(2)					
(2)	023566	104420	000066		
2332	023572	005000			
2333	023574	000277			
2334	023576	112700	000200		
2335					
2336	023602	104411		CHECKCC+11	
2337	023604	104423	023566	ERROR, T66	
2338	023610	022700	177600	CMP #177600,R0	
2339	023614	001402		BEQ 1\$	
2340	023616	104423	023566	ERRCR, T66	:: *** SIGN WAS NOT EXTENDED IN R0 ***
2341	023622	000277		SCC	
2342	023624	012700	177777	MOV #177777,R0	
2343	023630	112700	000000	MOVB #0,R0	:: CLEAR THE LOWER BYTE OF R0.
2344	023634	104405		CHECKCC+5	:: CHECK FOR CC=5
2345	023636	104423	023566	ERROR, T66	:: *** INCORRECT CC BITS ***
2346	023642	005700		1ST R0	:: CHECK R0 FOR SIGN EXTENTION
2347	023644	001402		BEQ 2\$	
2348	023646	104423	023566	ERROR, T66	:: *** SIGN WAS NOT EXTENDED IN R0. ***
2349	023652	012704	001012	MOV #TEMP2,R4	:: R4 IS POINTING TO TEMP2
2350	023656	012714	000377	MOV #377,(R4)	:: PLACE #377 IN LOCATION TEMP2
2351	023662	012706	003774	MOV #STACK-2,R6	
2352	023666	116426	000000	MOVB 0(R4),(R6)+	:: PUSH # 377 ON STACK
2353	023672	022706	003776	CMP #STACK,R6	
2354	023676	001402		BEQ 3\$	
2355	023700	104423	023566	ERROR, T66	:: *** R6 DID NOT GET INCREMENTED ***
2356	023704	124627	000377	CMPE -(R6),#377	:: CHECK LOCATION STACK-2 TO
2357					:: CONTAIN PROPER DATA
2358	023710	001402		BEQ 4\$	
2359	023712	104423	023566	ERROR, T66	:: *** BYTE INSTRUCTION IS FAILING WITH R6 ***
2360	023716	022706	003774	CMP #STACK-2,R6	:: CHECK THAT R6 WAS DECREMENTED
2361					:: BY 2 BY A BYTE INSTRUCTION
2362	023722	001402		BEQ 5\$	
2363	023724	104423	023566	ERROR, T66	:: *** R6 WAS NOT DECREMENTED ***
2364	023730	016467	000000	MOV 0(R4),TEMP	:: SET THE LOWER BYTE OF LOCATION TEMP
2365	023736	005726		TST (R6)+	:: RESTORE STACK POINTER
2366	023740	000277		SCC	

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
CNKXAA.P11 17-MAR-82 14:02

B 6

MACY11 30G(1063) 17-MAR-82 14:03 PAGE 2-40
SEQ 0066

PC 23724 = R6 WAS NOT DECREMENTED.

```
2367 023742 114667 155041      MOVB    -(SP),TEMP+1      ; SET THE HIGHER BYTE OF LOCATION TEMP
2368 023746 104411              CHECKCC+11      ; CHECK FOR CC=11
2369 023750 104423 023566      ERROR,    T66      ;: *** INCORRECT CC BITS ***
2370 023754 022767 177777 155024  CMP      #177777,TEMP      ; CHECK TEMP FOR THE CORRECT VALUE
2371 023762 001402              BEQ      6$
2372 023764 104423 023566      ERROR,    T66      ;: *** TEMP FOULED UP ***
2373 023770 005067 155012      6$:    CLR      TEMP
2374 023774 000241              CLC
2375 023776 105167 155005      COMB      TEMP+1      ; WRITE 1'S IN THE HIGHER BYTE OF TEMP
2376 024002 104411              CHECKCC+11      ; CHECK FOR CC=11
2377 024004 104423 023566      ERROR,    T66      ;: *** INCORRECT CC BITS ***
2378 024010 022767 177400 154770  CMP      #177400,TEMP
2379 024016 001402              BEQ      7$
2380 024020 104423 023566      ERROR,    T66      ;: *** TEMP FOULED UP ***
2381
2382 024024 004767 010700      7$:    CALL    ENDSEG      ; CHECK FOR LOOP-IN-SEG...
2383                                     ;...RETURN AND FALL THRU IF NOT.
2384                                     .SBTTL
```

.SBTTL SECTION 2 -- T11 TRAPS AND INTERRUPTS.

```
2386
2387
2388 024030 012767 036222 012142 S2:  MOV    #SEG2,SEGN    ; SEGMENT 2 TEXT.
2389
2390 ;*****
(2) .SBTTL      T67 -- TEST AUTO INCREMENT/DECREMENT OF STACK POINTER (R6)
(2) ;*****
(2) 024036 104420 000067
2391 024042 012700 001026 R6TST: MOV    #K4,R0    ; CHECK SWITCH REGISTER OPTIONS.
2392 024046 010006      MOV    R0,R6    ; USE K4 AS PSUEDO STACK...
2393 024050 112667 154732      MOV    (R6)+,TEMP    ; INIT SP.
2394 024054 020627 001030      CMP    R6,#K4+2    ; SIX SHOULD INCREMENT BY TWO
2395 024060 001402      BEQ     1$
2396 024062 104423 024036      ERROR,  T67    ; *** R6 DID NOT AUTO INCREMENT BY TWO ***
2397 024066 010006 1$:  MOV    R0,R6    ; SHOULD DECREMENT BY TWO
2398 024070 114667 154712      MOV    -(R6),TEMP
2399 024074 020627 001024      CMP    R6,#K4-2
2400 024100 001402      BEQ     2$
2401 024102 104423 024036      ERROR,  T67    ; *** R6 DID NOT AUTO DECREMENT BY 2 ***
2402 024106 010006 2$:  MOV    R0,R6
2403 024110 112626      MOV    (R6)+,(R6)+    ; DOUBLE AUTO INCREMENT OF R6
2404 024112 020627 001032      CMP    R6,#K4+4
2405 024116 001402      BEQ     3$
2406 024120 104423 024036      ERROR,  T67    ; *** WRONG AUTO INCREMENT OF R6 ***
2407 024124 010006 3$:  MOV    R0,R6
2408 024126 005001      CLR     R1
2409 024130 122621      CMPB   (R6)+,(R1)+    ; TEST INCREMENT OF R6 AND R1.
2410 024132 020627 001030      CMP    R6,#K4+2
2411 024136 001402      BEQ     4$
2412 024140 104423 024036      ERROR,  T67    ; *** WRONG INCREMENT OF R6 ***
2413 024144 020127 000001 4$:  CMP    R1,#1
2414 024150 001402      BEQ     5$
2415 024152 104423 024036      ERROR,  T67    ; *** WRONG INCREMENT OF R1 ***
2416 024156 010006 5$:  MOV    R0,R6
2417 024160 005001      CLR     R1
2418 024162 122126      CMPB   (R1)+,(R6)+    ; TEST INCREMENT OF R1 AND R6.
2419 024164 020127 000001      CMP    R1,#1
2420 024170 001402      BEQ     6$
2421 024172 104423 024036      ERROR,  T67    ; *** WRONG INCREMENT OF R1 ***
2422 024176 020627 001030 6$:  CMP    R6,#K4+2
2423 024202 001402      BEQ     7$
2424 024204 104423 024036      ERROR,  T67    ; *** WRONG INCREMENT OF R6 ***
2425 024210 7$:
2426
2427 ;*****
(2) .SBTTL      T70 -- TEST BYTE TRANSFERS USING STACK POINTER (R6)
(2) ;*****
(2) 024210 104420 000070
2428 024214 012767 123456 154606 T70:  BEGINTEST, 70    ; CHECK SWITCH REGISTER OPTIONS.
2429 024222 012767 050505 154570      MOV    #123456,K5
2430 024230 012701 001020      MOV    #050505,K1
2431 024234 012706 001030      MOV    #K1,R1    ; R1=(050505)K1
2432 024240 112621      MOV    #K5,R6    ; R6=(123456)K5
2433 024242 022767 050456 154550      MOV    (R6)+,(R1)+    ; LOW .BYTE OF R6 0 R1
2434 024250 001402      CMP    #050456,K1
2435 024252 104423 024210      BEQ     1$
      ERROR,  T70    ; *** FALSE TRANSFER OF .BYTE ***
```

```
2436
2437 024256 012767 123456 154544 1$: MOV #123456,K5
2438 024264 012767 050505 154526 MOV #050505,K1
2439 024272 012701 001020 MOV #K1,R1 ;R1(050505)K1
2440 024276 012706 001032 MOV #K6,R6 ;R6(123456)K5
2441 024302 114621 MOVB -(R6),(R1)+ ;LOW .BYTE OF R6 TO R1 (DECREMENT)
2442 024304 026727 154510 050456 CMP K1,#050456
2443 024312 001402 BEQ 2$
2444 024314 104423 024210 ERROR, T70 ;; *** FALSE R6 .BYTE TRANSFER ***
2445
2446 024320 012767 123456 154472 2$: MOV #123456,K1
2447 024326 012767 050505 154474 MOV #050505,K5
2448 024334 012701 001020 MOV #K1,R1 ;(123456)
2449 024340 012706 001030 MOV #K5,R6 ;(050505)
2450 024344 112126 MOVB (R1)+(R6)+ ;LOW OF R1 TO LOW OF R6
2451 024346 022767 050456 154454 CMP #050456,K5
2452 024354 001402 BEQ 3$
2453 024356 104423 024210 ERROR, T70 ;; *** FALSE R6 .BYTE TRANSFER ***
2454
2455 024362 012767 123456 154430 3$: MOV #123456,K1
2456 024370 012767 050505 154432 MOV #050505,K5
2457 024376 012701 001021 MOV #K1+1,R1 ;123456
2458 024402 012706 001030 MOV #K5,R6 ;050505
2459 024406 112126 MOVB (R1)+(R6)+ ;HIGH OF R1 TO LOW OF R6
2460 024410 026727 154414 050647 CMP K5,#050647
2461 024416 001402 BEQ 4$
2462 024420 104423 024210 ERROR, T70 ;; *** FALSE R6 .BYTE TRANSFER ***
2463
2464 024424 012767 123456 154366 4$: MOV #123456,K1
2465 024432 012767 050505 154370 MOV #050505,K5
2466 024440 012701 001021 MOV #K1+1,R1 ;R1-123456-ODD ADDRESS
2467 024444 012706 001030 MOV #K5,R6 ;R6-050505--.EVEN ADDRESS
2468 024450 112521 MOVB (R6)+(R1)+ ;LOW OF R6 TO HIGH OF R1
2469 024452 022767 042456 154340 CMP #042456,K1
2470 024460 001402 BEQ 5$
2471 024462 104423 024210 ERROR, T70 ;; *** FAILED LOW OF 6 TO HIGH OF 1 ***
2472 024466
2473
2474
(2) ;*****
(2) .SBTTL T71 -- TEST BYTE COMPARES ON SEQUENTIAL ODD/EVEN ADDRESSES
(2) ;*****
2475 024466 104420 000071 T71: BEGINTEST, 71 ;; CHECK SWITCH REGISTER OPTIONS.
2476 024472 005067 154320 CLR K0 ; K0 <= 000000
2477 024476 012767 052525 154314 MOV #52525,K1 ; K1 <= 052525
2478 024504 012767 052400 154310 MOV #52400,K2 ; K2 <= 052400
2479 024512 126767 154302 154301 CMPB K1,K1+1
2480 024520 001402 BEQ 1$
2481 024522 104423 024466 ERROR, T71 ;; *** LOW TO HIGH IN SAME WORD FAILS ***
2482 024526 126767 154267 154264 1$: CMPB K1+1,K1
2483 024534 001412 BEQ 2$
2484 024536 104423 024466 ERROR, T71 ;; *** HIGH TO LOW IN SAME WORD FAILS ***
2485 024542 126767 154255 154250 2$: CMPB K2+1,K1
2486 024550 001402 BEQ 3$
2487 024552 104423 024466 ERROR, T71 ;; *** HIGH TO LOW IN DIFFERENT WORDS FAILS ***
2488 024556 126767 154240 154232 3$: CMPB K2,K0
2489 024564 001402 BEQ 4$
```

CNKXAA.P11 17-MAR-82 14:02

PC 24566 = EVEN TO EVEN FAILED.

```

2489 024566 104423 024466      ERROR, T71      ;; *** EVEN TO EVEN FAILED ***
2490 024572 126767 154223 154223 4$: CMPB      K1+1,K2+1
2491 024600 001402      BEQ          5$
2492 024602 104423 024466      ERROR, T71      ;; *** ODD TO ODD FAILED ***
2493 024606 126767 154210 154207 5$: CMPB      K2,K2+1
2494 024614 001002      BNE          6$
2495 024616 104423 024466      ERROR, T71      ;; *** LOW TO HIGH IN SAME WORD FAILED ***
2496 024622 126757 154175 154173 6$: CMPB      K2+1,K2+1
2497 024630 001402      BEQ          7$
2498 024632 104423 024466      ERROR, T71      ;; *** HIGH TO HIGH IN SAME WORD FAILED ***
2499 024636 126767 154160 154155 7$: CMPB      K2,K1+1
2500 024644 001002      BNE          8$
2501 024646 104423 024466      ERROR, T71      ;; *** EVEN TO ODD FAILED ***
2502 024652      8$:
2503
2504 ;*****
2505 ;SBTTL      T72 -- TEST THAT BUS TIME-OUT TRAPS TO THE 'RESTART' ADDRESS
2506 ;*****
2507 T72:      BEGINTEST, 72      ;; CHECK SWITCH REGISTER OPTIONS.
2508 ;
2509 ; T11 TRAPS TO ODT 'RESTART' ON TIME-OUT. THE CODE THERE WILL
2510 ; EMULATE A TRAP THRU THE TRADITIONAL BUS ERROR VECTOR (4).
2511 ;
2512 2509 024656 012706 003776      MOV      #STACK,SP      ; SET STACK POINTER.
2513 2510 024662 012767 024712 153114 MOV      #3$,ERRVEC      ; SET TRAP VECTOR.
2514 2511 024670 106427 000000      MTPS     #PRIO
2515 2512 024674 005737 160000      1$: TST      @#160000      ; NEXM -- TIME-OUT AND TRAP.
2516 2513 024700 000257      CCC      ; SHOULD EXECUTE THIS...
2517 2514 024702 000240      2$: NOP      ;...THEN TRAP AND STACK THIS PC.
2518 2515 024704 104423 024652      ERROR, T72      ;; *** BUS TIME-OUT DIDN'T TRAP AT ALL ***
2519 2516 024710 000414      BR          5$
2520 2517 024712 011600      3$: MOV      (SP),R0      ; GET STACKED PC...
2521 2518 024714 016601 000002      MOV      2(SP),R1      ;...AND PSW.
2522 2519 024720 020027 024702      CMP      R0,#2$
2523 2520 024724 001402      BEQ          4$      ; BR IF PC IS RIGHT.
2524 2521 024726 104423 024652      ERROR, T72      ;; *** STACKED PC INCORRECT ON TIME-OUT TRAP ***
2525 2522
2526 2523 024732 005701      4$: TST      R1
2527 2524 024734 001402      BEQ          5$      ; BR IF PSW IS RIGHT.
2528 2525 024736 104423 024652      ERROR, T72      ;; *** STACKED PSW INCORRECT ON TIME-OUT TRAP ***
2529 2526
2530 2527 024742 012767 000006 153034 5$: MOV      #ERRVEC+2,ERRVEC ; RESET TRAP CATCHER.
2531 2528
2532 ;*****
2533 ;SBTTL      T73 -- TEST THAT AN 'ILLEGAL' INSTRUCTION TRAPS TO 4
2534 ;*****
2535 (2) 024750 104420 000073      T73:      BEGINTEST, 73      ;; CHECK SWITCH REGISTER OPTIONS.
2536 ;
2537 ; DO BOTH 'JMP R' AND 'JSR R,R'
2538 ; ON ERROR, R0 POINTS TO THE OFFENDER.
2539 ;
2540 2534 024754 012700 025012      MOV      #1$,R0      ; 1ST PASS USES A 'JMP R'
2541 2535 024760 000402      SKP2
2542 2536 024762 012700 025016      8$: MOV      #2$,R0      ; 2ND PASS USES A 'JSR R,R'
2543 2537 024766 012706 003776      MOV      #STACK,SP      ; STACK POINTER SETUP
2544 2538 024772 012767 025030 153004      MOV      #3$,ERRVEC      ; SET TRAP PC...

```

```
2539 025000 005067 153002      CLR      ERRVEC+2      ;...AND PSW.
2540 025004 106427 000317      MTPS     #PRI6!17      ; SET CURRENT (OLD) PSW.
2541 025010 000110              JMP      (R0)          ; DISPATCH TO ONE OR THE OTHER.
2542 025012 000101      1$:     JMP      R1          ; ILLEGAL JMP -- SHOULD TRAP.
2543 025014 000401              SKP1
2544 025016 004101      2$:     JSR      R1,R1          ; ILLEGAL JSR -- SHOULD TRAP.
2545 025020 000240              NOP
2546 025022 104423 024750      ERROR,   T73          ;; *** ILLEGAL JMP/JSR DIDN'T TRAP ***
2547 025026 000427              BR       7$
2548
2549 025030 106701      3$:     MFPS     R1          ; OK, GET CURRENT (NEW) PSW.
2550 025032 001402              BEQ      4$          ; BR IF IT'S RIGHT.
2551 025034 104423 024750      ERROR,   T73          ;; *** PSW INCORRECT AFTER ILLEGAL INSTR TRAP TO 4 ***
2552 025040 020627 003772      4$:     CMP      SP,#STACK-4      ; BR IF STACK IS RIGHT.
2553 025044 001402              BEQ      5$          ; *** STACK POINTER INCORRECT ***
2554 025046 104423 024750      ERROR,   T73
2555 025052 021627 025014      5$:     CMP      (SP),#1$+2      ; BR IF STACKED PC IS RIGHT (JMP).
2556 025056 001405              BEQ      6$          ; BR IF STACKED PC IS RIGHT (JSR).
2557 025060 021627 025020      CMP      (SP),#2$+2
2558 025064 001402              BEQ      6$          ; BR IF STACKED PC IS RIGHT (JSR).
2559 025066 104423 024750      ERROR,   T73          ;; *** INCORRECT PC SAVED ON STACK ***
2560 025072 026627 000002 000317 6$:     CMP      2(SP),#PRI6!17
2561 025100 001402              BEQ      7$          ; BR IF STACKED (OLD) PSW IS RIGHT.
2562 025102 104423 024750      ERROR,   T73          ; *** INCORRECT PSW SAVED ON STACK ***
2563
2564 025106 020027 025012      7$:     CMP      R0,#1$          ; IS THIS THE 1ST PASS ??
2565 025112 001723              BEQ      8$          ; LOOP ONCE IF SO...
2566 025114 012767 000006 152662      MOV      #ERRVEC+2,ERRVEC ;...OTHERWISE, RESET VECTOR...
2567
2568 ;*****
(2) ;SBTTL      T74 -- TEST THAT A RESERVED (UNDEFINED) INSTRUCTION TRAPS TO 10
(2) ;*****
(2) 2569 025122 104420 000074      T74:     BEGINTEST, 74      ; CHECK SWITCH REGISTER OPTIONS.
2570 025126 012706 003776      MOV      #STACK,SP      ; STACK POINTER SETUP
2571 025132 012767 025162 152650      MOV      #2$,RESVEC      ; SET TRAP PC...
2572 025140 005067 152646      CLR      RESVEC+2      ;...AND PSW.
2573 025144 106427 000317      MTPS     #PRI6!17      ; SET CURRENT (OLD) PSW.
2574 025150 000010      RESRVD      ; RESERVED INSTRUCTION -- SHOULD TRAP
2575 025152 000240      1$:     NOP
2576 025154 104423 025122      ERROR,   T74          ; *** RESERVED OPCODE DIDN'T TRAP ***
2577 025156 000424              BR       6$
2578
2578 025160 000700      2$:     MFPS     R0          ; BR IF NEW PSW IS RIGHT.
2579 025164 001402              BEQ      3$          ; *** NEW PSW INCORRECT AFTER TRAP TO 10 ***
2580 025166 104423 025122      ERROR,   T74
2581 025172 020627 003772      3$:     CMP      SP,#STACK-4      ; BR IF SP PUSHED DOWN RIGHT.
2582 025176 001402              BEQ      4$          ; *** STACK POINTER INCORRECT ***
2583 025200 104423 025122      ERROR,   T74
2584 025204 021627 025132      4$:     CMP      (SP),#1$          ; BR IF STACKED PC IS RIGHT.
2585 025210 001402              BEQ      5$          ; *** INCORRECT PC SAVED ON STACK ***
2586 025212 104423 025122      ERROR,   T74
2587 025216 026627 000002 000317 5$:     CMP      2(SP),#PRI6!17
2588 025224 001402              BEQ      6$          ; BR IF STACKED (OLD) PSW IS RIGHT.
2589 025226 104423 025122      ERROR,   T74          ; *** INCORRECT PSW SAVED ON STACK ***
2590
2591 025232 012767 000012 152550 6$:     MOV      #RESVEC+2,RESVEC ; RESET VECTOR.
```

```
2592
2593
(2)
(2)
(2) 025240 104420 000016
2594 025244 012706 000016 152536
2595 025250 012767 025300
2596 025256 005067 152534
2597 025262 106427 000317
2598 025266 000003
2599 025270 000240
2600 025272 104423 025240
2601 025276 000424
2602
2603 025300 106700
2604 025302 001402
2605 025304 104423 025240
2606 025310 020627 003772
2607 025314 001402
2608 025316 104423 025240
2609 025322 021627 025270
2610 025326 001402
2611 025330 104423 025240
2612 025334 026627 000002 000317
2613 025342 001402
2614 025344 104423 025240
2615
2616 025350 012767 000016 152436
2617
2618
(2)
(2)
(2) 025356 104420 000076
2619 025362 012706 003776 152424
2620 025366 012767 025416
2621 025374 005067 152422
2622 025400 106427 000317
2623 025404 000004
2624 025406 000240
2625 025410 104423 025356
2626 025414 000424
2627
2628 025416 106700
2629 025420 001402
2630 025422 104423 025356
2631 025426 020627 003772
2632 025432 001402
2633 025434 104423 025356
2634 025440 021627 025406
2635 025444 001402
2636 025446 104423 025356
2637 025452 026627 000002 000317
2638 025460 001402
2639 025462 104423 025356
2640
2641 025466 012767 000022 152324
```

```
*****
.SBTTL T75 -- TEST THAT A 'BPT' INSTRUCTION TRAPS TO 14
*****
T75: BEGINTEST, 75 ; CHECK SWITCH REGISTER OPTIONS.
      MOV #STACK.SP ; STACK POINTER SETUP
      MOV #2$,BPTVEC ; SET TRAP PC...
      CLR BPTVEC+2 ; ...AND PSW.
      MTPS #PRI6!17 ; SET CURRENT (OLD) PSW.
      BPT ; BPT -- SHOULD TRAP.
1$: NOP ;
      ERROR, T75 ; *** BPT DIDN'T TRAP ***
      BR 6$

2$: MFPS R0 ; BR IF NEW PSW IS RIGHT.
      BEQ 3$ ; *** PSW INCORRECT AFTER BPT TRAP TO 14 ***
      ERROR, T75
3$: CMP SP,#STACK-4 ; BR IF STACK IS RIGHT.
      BEQ 4$ ; *** STACK POINTER INCORRECT ***
      ERROR, T75
4$: CMP (SP),#1$ ; BR IF STACKED PC IS RIGHT.
      BEQ 5$ ; *** INCORRECT PC SAVED ON STACK ***
      ERROR, T75 ; BR IF STACKED (OLD) PSW IS RIGHT.
      BEQ 6$ ; *** INCORRECT PSW SAVED ON STACK ***
      ERROR, T75
6$: MOV #BPTVEC+2,BPTVEC ; RESET VECTOR.

*****
.SBTTL T76 -- TEST THAT AN 'IOT' INSTRUCTION TRAPS TO 20
*****
T76: BEGINTEST, 76 ; CHECK SWITCH REGISTER OPTIONS.
      MOV #STACK.SP ; STACK POINTER SETUP
      MOV #2$,IOTVEC ; SET TRAP PC...
      CLR IOTVEC+2 ; ...AND PSW.
      MTPS #PRI6!17 ; SET CURRENT (OLD) PSW.
      IOT ; IOT -- SHOULD TRAP.
1$: NOP ;
      ERROR, T76 ; *** IOT DID NOT TRAP ***
      BR 6$

2$: MFPS R0 ; BR IF NEW PSW IS RIGHT.
      BEQ 3$ ; *** PSW INCORRECT AFTER IOT TRAP TO 20 ***
      ERROR, T76
3$: CMP SP,#STACK-4 ; BR IF STACK IS RIGHT.
      BEQ 4$ ; *** STACK POINTER INCORRECT ***
      ERROR, T76
4$: CMP (SP),#1$ ; BR IF STACKED PC IS RIGHT.
      BEQ 5$ ; *** INCORRECT PC SAVED ON STACK ***
      ERROR, T76 ; BR IF STACKED (OLD) PSW IS RIGHT.
      BEQ 6$ ; *** INCORRECT PSW SAVED ON STACK ***
      ERROR, T76
6$: MOV #IOTVEC+2,IOTVEC ; RESET VECTOR.
```

```

2642
2643
(2)
(2)
(2) 025474 104420 000077
2644 025500 012706 003776
2645 025504 004767 007552
2646 025510 001456
2647 025512 010004
2648
2649
2650
2651
2652
2653
2654 025514 012767 025564 152302
2655 025522 005067 152300
2656 025526 106427 000300
2657 025532 012700 140000
2658 025536 077001
2659
2660
2661 025540 012700 000005
2662 025544 052714 020000
2663 025550 000257
2664 025552 077001
2665 025554 005014
2666 025556 104423 025474
2667 025562 000426
2668
2669 025564 106700
2670 025566 005014
2671 025570 005700
2672 025572 001402
2673 025574 104423 025474
2674 025600 020627 003772
2675 025604 001402
2676 025606 104423 025474
2677 025612 021627 02555
2678 025616 001402
2679 025620 104423 025474
2680 025624 026627 000002 000300
2681 025632 001402
2682 025634 104423 025474
2683
2684 025640 012767 000026 152156
2685 025646
2686
2687
(2)
(2)
(2) 025646 104420 000100
2688
2689
2690
2691

```

```

*****
.SBTTL      T77 -- TEST THAT 'POWER-FAIL' TRAPS TO 24 (REQUIRES Q-BUS EXERCISER)
*****
T77:  BEGINTEST, 77          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      #STACK,SP      ;STACK POINTER SETUP
      CALL     QBXIN          ; Q-BUS EXERCISER AVAILABLE ??
      BEQ      7$            ; BYPASS TEST IF NOT.
      MOV      R0,R4          ; YES, MOVE POINTER TO R4 AND FALL THRU.

      ; QBX CSR1<13> IS USED TO EMULATE A POWER DOWN/UP SEQUENCE.
      ; CAUTION:
      ; WE ONLY WANT TO RECOGNIZE THE 'DOWN' HALF OF THE POWER SEQUENCE.
      ; ABORT THE PWR-CYCLE AT THAT POINT SO WE DON'T CRASH TO ODT.

      MOV      #2$,PWRVEC      ; SET POWER TRAP PC...
      CLR      PWRVEC+2        ; ...AND PSW
      MTPS     #PRI6          ; SET CURRENT (OLD) PSW.
      MOV      #140000,R0      ; IF LOOPING, WE HAVE TO WAIT LONG...
      SOB      R0,,           ; ...ENOUGH (==250MSEC) TO INSURE THE...
                                ; ...PREVIOUS BPOK CYCLE WAS FINISHED...
                                ; ... (IN QBX) BEFORE TRYING ANOTHER.
      MOV      #5,R0           ; OK, SET A 25 USEC TIMER.
      BIS      #20000,(R4)     ; SET PWR-CYCLE REQUEST...
      CCC      R0,,           ; ...SHOULD SEE IT AFTER THIS...
      SOB      R0,,           ; ...AND STACK THIS PC.
      CLR      (R4)           ; TRAP NOT RECEIVED, CLEAR THE REQUEST.
      ERROR,   T77            ;: *** POWER-FAIL DID NOT TRAP ***
      BR       6$

2$:  MFPS     R0              ; OK, GET NEW PSW...
      CLR      (R4)           ; ...AND CLEAR PWR-CYCLE REQUEST.
      TST      R0
      BEQ      3$            ; BR IF NEW PSW IS RIGHT.
      ERROR,   T77            ;: *** PSW INCORRECT AFTER PWR-FAIL TRAP TO 24 ***
      CMP      SP,#STACK-4
      BEQ      4$            ; BR IS STACK IS RIGHT.
      ERROR,   T77            ;: *** STACK POINTER INCORRECT ***
      CMP      (SP),#1$
      BEQ      5$            ; BR IF STACKED PC IS RIGHT.
      ERROR,   T77            ;: *** INCORRECT PC SAVED ON STACK ***
      CMP      2(SP),#PRI6
      BEQ      6$            ; BR IF STACKED (OLD) PSW IS RIGHT.
      ERROR,   T77            ;: *** INCORRECT PSW SAVED ON STACK ***

6$:  MOV      #PWRVEC+2,PWRVEC ; RESTORE VECTOR.
7$:

*****
.SBTTL      T100 -- TEST THAT AN 'EMT' INSTRUCTION TRAPS TO 30
*****
T100: BEGINTEST, 100        ;; CHECK SWITCH REGISTER OPTIONS.

      ; NOW EMT'S HAVE TO BE OK OR WE'D NEVER HAVE GOTTEN STARTED IN THE FIRST
      ; PLACE. WE'VE ALREADY VERIFIED ALL VARIATIONS OF THE EMT OPCODE,
      ; SO WE'LL JUST TEST FOR CORRECT STACK MANIPULATION HERE.

```

```
2692 ;
2693 025652 012706 003776 ; MOV #STACK,SP ; STACK POINTER SETUP
2694 025656 016701 152146 ; MOV EMTVEC,R1 ; SAVE EMT VECTOR IN R1.
2695 025662 012767 025706 152140 ; MOV #2$,EMTVEC ; SET TRAP PC...
2696 025670 005067 152136 ; CLR EMTVEC+2 ; ...AND PSW.
2697 025674 106427 000317 ; MTPS #PRI6!17 ; SET CURRENT (OLD) PSW.
2698 025700 104000 ; EMT ; EMT -- SHOULD TRAP...
2699 025702 000240 1$: NOP ; ...AND STACK THIS PC.
2700 025704 000240 ; NOP ; IF IT DOESN'T, JUST FALL THRU ANYWAY.
2701
2702 025706 106700 2$: MFPS R0 ; GET NEW PSW...
2703 025710 010167 152114 ; MOV R1,EMTVEC ; ...AND RESTORE VECTOR.
2704 025714 005700 ; TST R0
2705 025716 001402 ; BEQ 3$ ; BR IF NEW PSW IS RIGHT.
2706 025720 104423 025646 ; ERROR, T100 ;: *** PSW INCORRECT AFTER EMT TRAP TO 30 ***
2707 025724 020627 003772 3$: CMP SP,#STACK-4
2708 025730 001402 ; BEQ 4$ ; BR IF SP IS RIGHT.
2709 025732 104423 025646 ; ERROR, T100 ;: *** STACK POINTER INCORRECT ***
2710 025736 021627 025702 4$: CMP (SP),#1$
2711 025742 001402 ; BEQ 5$ ; BR IF STACKED PC IS RIGHT.
2712 025744 104423 025646 ; ERROR, T100 ;: *** INCORRECT PC SAVED ON STACK ***
2713 025750 026627 000002 000317 5$: CMP 2(SP),#PRI6!17
2714 025756 001402 ; BEQ 6$ ; BR IF STACKED (OLD) PSW IS RIGHT.
2715 025760 104423 025646 ; ERROR, T100 ;: *** INCORRECT PSW SAVED ON STACK ***
2716 025764 6$:
2717
2718 ;*****
(2) ;SBTTL T101 -- TEST THAT A "TRAP" INSTRUCTION TRAPS TO 34
(2) ;*****
(2) 025764 104420 000101 T101: BEGINTEST, 101 ;: CHECK SWITCH REGISTER OPTIONS.
2719 ;
2720 ; WE ALREADY KNOW THAT ALL VARIATIONS OF THE TRAP OPCODE WORK.
2721 ; SO WE'LL JUST CHECK THAT THE STACK IS HANDLED PROPERLY.
2722 ;
2723 025770 012706 003776 ; MOV #STACK,SP ; STACK POINTER SETUP
2724 025774 016701 152034 ; MOV TRPVEC,R1 ; SAVE TRAP VECTOR IN R1.
2725 026000 012767 026024 152026 ; MOV #2$,TRPVEC ; SET TRAP PC...
2726 026006 005067 152024 ; CLR TRPVEC+2 ; ...AND PSW.
2727 026012 106427 000317 ; MTPS #PRI6!17 ; SET CURRENT (OLD) PSW.
2728 026016 104400 ; TRAP+0 ; TRAP -- SHOULD TRAP...
2729 026020 000240 1$: NOP ; ...AND STACK THIS PC.
2730 026022 000240 ; NOP ; IF IT DOESN'T, JUST FALL THRU ANYWAY.
2731
2732 026024 106700 2$: MFPS R0 ; GET NEW PSW...
2733 026026 010167 152002 ; MOV R1,TRPVEC ; ...AND RESTORE VECTOR.
2734 026032 005700 ; TST R0
2735 026034 001402 ; BEQ 3$ ; BR IF NEW PSW IS RIGHT.
2736 026036 104423 025764 ; ERROR, T101 ;: *** PSW INCORRECT AFTER TRAP TO 34 ***
2737 026042 020627 003772 3$: CMP SP,#STACK-4
2738 026046 001402 ; BEQ 4$ ; BR IF STACK IS RIGHT.
2739 026050 104423 025764 ; ERROR, T101 ;: *** STACK POINTER INCORRECT ***
2740 026054 021627 026020 4$: CMP (SP),#1$
2741 026060 001402 ; BEQ 5$ ; BR IF STACKED PC IS RIGHT.
2742 026062 104423 025764 ; ERROR, T101 ;: *** INCORRECT PC SAVED ON STACK ***
2743 026066 026627 000002 000317 5$: CMP 2(SP),#PRI6!17
2744 026074 001402 ; BEQ 6$ ; BR IF STACKED (OLD) PSW IS RIGHT.
```

```

2745 026076 104423 025764          ERROR, T101          ;; *** INCORRECT PSW SAVED ON STACK ***
2746 026102
2747
2748
(2)
(2)
(2) 026102 104420 000102          ;*****
;SBTTL      T102 -- TEST THAT THE 'T' BIT CAUSES A TRACE TRAP
;*****
T102: BEGINTEST, 102          ;; CHECK SWITCH REGISTER OPTIONS.
; FIRST, SET 'T' VIA RTI -- SHOULD TRAP TO 14 BEFORE NEXT INSTRUCTION.
;
2752 026106 012706 003776          MOV      #STACK,SP
2753 026112 012767 026150 151674  MOV      #3$,TRTVEC          ; SET TRAP PC...
2754 026120 005067 151672          CLR      TRTVEC+2          ;...AND NEW PSW.
2755 026124 012746 000020          MOV      #20,-(SP)          ; PUSH OLD PSW WITH T BIT...
2756 026130 012746 026136          MOV      #.+6,-(SP)          ;...AND PC.
2757 026134 000002          RTI          ; SHOULD SET T BIT...
2758 026136 000240          1$: NOP          ;...AND TRAP BEFORE THIS EXECUTES.
2759 026140 000240          2$: NOP
2760 026142 104423 026102          ERROR,   T102          ;; *** TRACE BIT DID NOT TRAP ***
2761 026146 000417          BR      6$
2762
2763 026150 106700          3$: MFPS      R0          ; TRAPPED OK, T SHOULD BE CLEAR NOW.
2764 026152 001402          BEQ      4$          ; BR IF SO.
2765 026154 104423 026102          ERROR,   T102          ; *** NEW PSW INCORRECT AFTER TRACE TRAP TO 14 ***
2766 026160 021627 026136          4$: CMP      (SP),#1$          ; IT SHOULD HAVE TRAPPED BEFORE...
2767 026164 001402          BEQ      5$          ;...THE INSTRUCTION AT 1$.
2768 026166 104423 026102          ERROR,   T102          ; *** TRACE TRAP HAPPENED AT WRONG TIME ***
2769 026172 026627 000002 000020  5$: CMP      2(SP),#20          ; AND T BIT SHOULD HAVE BEEN STACKED...
2770 026200 001402          BEQ      6$          ;...ALONG WITH THE OLD PSW.
2771 026202 104423 026102          ERROR,   T102          ; *** T BIT WASN'T SAVED IN STACKED PSW ***
2772
2773          ; NOW SET 'T' VIA RTT -- SHOULD ALLOW NEXT INSTRUCTION, THEN TRAP.
2774
2775 026206 012767 026240 151600  6$: MOV      #8$,TRTVEC          ; CHANGE VECTOR.
2776 026214 012746 000020          MOV      #20,-(SP)          ; PUSH OLD PSW WITH T BIT...
2777 026220 012746 026226          MOV      #.+6,-(SP)          ;...AND PC.
2778 026224 000006          RTT          ; SHOULD SET T BIT...
2779 026226 000240          NOP          ;...EXECUTE THIS...
2780 026230 000240          7$: NOP          ;...AND TRAP BEFORE THIS.
2781 026232 104423 026102          ERROR,   T102          ; *** TRACE TRAP DIDN'T HAPPEN ***
2782 026236 000417          BR      11$
2783
2784 026240 106700          8$: MFPS      R0          ; OK, T SHOULD BE CLEAR NOW.
2785 026242 001402          BEQ      9$
2786 026244 104423 026102          ERROR,   T102          ; *** NEW PSW INCORRECT AFTER TRACE TRAP TO 14 ***
2787 026250 021627 026230          9$: CMP      (SP),#7$          ; AND IT SHOULD HAVE ALLOWED...
2788 026254 001402          BEQ      10$          ;...THE 1ST NOP TO EXECUTE.
2789 026256 101123 026102          ERROR,   T102          ; *** RTT DIDN'T ALLOW NEXT INSTRUCTION TO EXECUTE ***
2790 026262 026627 000002 000020 10$: CMP      2(SP),#20          ; AND THE T BIT SHOULD BE ON STACK.
2791 026270 001402          BEQ      11$
2792 026272 104423 026102          ERROR,   T102          ; *** T BIT WASN'T SAVED IN STACKED PSW ***
2793 026276
2794
2795
(2)
(2)
;*****
;SBTTL      T103 -- TEST THAT TRACE TRAP ON A TRAP IS INHIBITED
;*****

```

```
(2) 026276 104420 000103 T103: BEGINTEST, 103 ;: CHECK SWITCH REGISTER OPTIONS.
2796 026302 012706 003776 MOV #STACK,SP ;: SET STACK
2797 026306 012767 026360 151500 MOV #1$,TRTVEC ;: SET TRACE VECTOR...
2798 026314 005067 151476 CLR TRTVEC+2 ;: ...AND PSW.
2799 025320 012767 026370 151472 MOV #2$,IOTVEC ;: SET IOT VECTOR...
2800 026326 005067 151470 CLR IOTVEC+2 ;: ...AND PSW.
2801 026332 012746 000020 MOV #20,-(SP) ;: PUSH PSW WITH T BIT...
2802 025336 012746 026344 MOV #.+6,-(SP) ;: ...AND PC.
2803 026342 000006 RTT ;: SET T BIT...
2804 026344 000004 IOT ;: ...AND XCT ANOTHER TRAP...
2805 026346 000240 NOP ;: THE IOT SHOULD WIN...
2806 026350 000240 NOP ;: ...AND INHIBIT THE TRACE.
2807 026352 104423 026276 ERROR, T103 ;: *** NEITHER TRAP OCCURRED ??? ***
2808 026356 000404 BR 2$
2809
2810 026360 000240 1$: NOP ;: IOT SHOULD HAVE CLEARED T...
2811 026362 000240 NOP ;: ...AND INHIBITED THE TRACE.
2812 026364 104423 026276 ERROR, T103 ;: *** TRACE TRAP ON TRAP WASN'T INHIBITED ***
2813
2814 026370 012767 000022 151422 2$: MOV #IOTVEC+2,IOTVEC ;: RESET IOT VECTOR.
2815
2816 ;*****
(2) ;SBTTL T104 -- TEST THAT RESET HAS NO EFFECT ON A TRACE TRAP
(2) ;*****
(2) 026376 104420 000104 T104: BEGINTEST, 104 ;: CHECK SWITCH REGISTER OPTIONS.
2817 026402 012706 003776 MOV #STACK,SP ;: SET STACK
2818 026406 012767 026446 151400 MOV #1$,TRTVEC ;: SET TRACE VECTOR.
2819 026414 005067 151376 CLR TRTVEC+2
2820 026420 012746 000020 MOV #20,-(SP) ;: PUSH PSW WITH T BIT...
2821 026424 012746 026432 MOV #.+6,-(SP) ;: ...AND PC.
2822 026430 000006 RTT ;: SET 'T' BIT...
2823 026432 000005 RESET ;: ...AND XCT BUS RESET.
2824 026434 000240 NOP ;: SHOULD HAVE NO EFFECT...
2825 026436 000240 NOP ;: ... I.E. TRAP SHOULD STILL OCCUR...
2826 026440 000240 NOP ;: ...AFTER THE RESET INSTRUCTION.
2827 026442 104423 026376 ERROR, T104 ;: *** BUS-RESET DISABLED OR INHIBITED TRACE TRAP ***
2828
2829 026446 106427 000000 1$: MTPS #PRIO ;: WE'RE ALL DONE.
2830 026452 012767 000016 151334 MOV #TRTVEC+2,TRTVEC ;: RESET TRACE VECTOR.
2831
2832 ;*****
(2) ;SBTTL T105 -- TEST THAT ODD ADDRESS TRAPS ARE NOT IMPLEMENTED
(2) ;*****
(2) 026460 104420 000105 T105: BEGINTEST, 105 ;: CHECK SWITCH REGISTER OPTIONS.
2833 026464 012706 003776 MOV #STACK,SP ;: SET STACK.
2834 026470 012767 026514 151306 MOV #2$,ERRVEC ;: IN CASE ODD ADDRESS TRAPS.
2835 026476 000167 000007 JMP 1$+1
2836 026502 000240 NOP ;: IT DIDN'T DO ANYTHING AT ALL !!
2837 026504 104423 026460 ERROR, T105 ;: *** JUMP TO AN ODD ADDRESS DID NOTHING ***
2838 026510 012707 026520 1$: MOV #3$,PC ;: ADJUST ODD PC AND EXIT.
2839 026514 2$:
(2) 026514 104423 026460 3$: ERROR, T105 ;: *** JUMP TO AN ODD ADDRESS TIMED OUT ***
2840 026520 012767 000006 151256 MOV #ERRVEC+2,ERRVEC ;: RESET THE TRAP CATCHER.
2841
2842 ;*****
(2) ;SBTTL T106 -- TEST THE CPU TYPE INSTRUCTION (MFPT)
```

```

(2)
(2) 026526 104420 000106
2843 026532 012706 003776
2844 026536 012767 026602 151244
2845 026544 005000
2846 026546 000257
2847 026550 000007
2848 026552 106701
2849 026554 032701 000017
2850 026560 001402
2851 026562 104423 026526
2852 026566 020027 000004
2853 026572 001405
2854 026574 104423 026526
2855 026600 000402
2856 026602
(2) 026602 104423 026526
2857 026606 012767 000012 151174
2858
2859
(2)
(2)
(2) 026614 104420 000107
2860 026620 012706 003776
2861 026624 004767 006432
2862 026630 001444
2863 026632 010004
2864
2865
2866
2867 026634 012703 000110
2868 026640 005014
2869 026642 012713 026706
2870 026646 012763 000200 000002
2871 026654 106427 000200
2872 026660 012764 000001 000002
2873 026666 052764 000020 000002
2874 026674 042764 000001 000002
2875 026702 000241
2876 026704 000401
2877 026706
(2) 026706 104423 026614
2878 026712 000407
2879 026714 012713 026732
2880 026720 106427 000000
2881 026724 000240
2882 026726 104423 026614
2883 026732 005014
2884 026734 012723 000112
2885 026740 005013
2886 026742
2887
2888
(2)
(2)
(2) 026742 104420 000110
  
```

```

*****
T106: BEGINTEST, 106      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      #STACK,SP  ;; SET STACK.
      MOV      #2$,RESVEC ;; SET TRAP CATCHER.
      CLR      R0
      CCC      ;; CLEAR CC BITS.
      MFPT     ;; MOVE FROM PROC TYPE (TO R0).
      MFPS     R1
      BIT      #17,R1
      BEQ      1$
      ERROR,   T106      ;; *** CC BITS CHANGED ON MFPT INSTRUCTION ***
1$:    CMP      R0,#KXT11
      BEQ      3$
      ERROR,   T106      ;; *** RETURNED VALUE INCORRECT ON MFPT ***
2$:    BR       3$
3$:    ERROR,   T106      ;; *** MFPT TRAPPED TO 10 ***
      MOV      #RESVEC+2,RESVEC
*****
.SBTTL      T107 -- TEST EXTERNAL Q-BUS INTERRUPT (REQUIRES Q-BUS EXERCISER)
*****
T107: BEGINTEST, 107      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      #STACK,SP
      CALL     QBXIN      ;; Q-BUS EXERCISER AVAILABLE ??
      BEQ      4$         ;; IF NOT, BYPASS TEST.
      MOV      R0,R4      ;; YES, CSR1 POINTER => R4 AND FALL THRU.
      ;;
      ;; USE QBX TO FAKE A Q-BUS INTERRUPT (NO DMA).
      ;;
      MOV      #QBXVEC,R3 ;; GET VECTOR POINTER.
      CLR      (R4)       ;; CLEAR CSR1 (NO DMA REQUIRED).
      MOV      #1$(R3)    ;; SET Q-BUS VECTOR...
      MOV      #PRI4,2(R3) ;; ...AND PRIORITY.
      MTPS     #PRI4      ;; RAISE CPU TO Q-BUS LEVEL.
      MOV      #1,2(R4)    ;; SET QBX GO (CLEAR DONE)...
      BIS      #20,2(R4)   ;; ...ENABLE INT 4 ON DONE...
      BIC      #1,2(R4)    ;; ...AND CLEAR GO (SET DONE).
      240        ;; INTERRUPT SHOULD BE HELD OFF.
      BR       2$
1$:    ERROR,   T107      ;; *** Q-BUS INTERRUPTS AT WRONG LEVEL ***
      BR       3$
2$:    MOV      #3$(R3)    ;; CHANGE THE VECTOR.
      MTPS     #PRI0      ;; LOWER CPU...
      240        ;; ...INTERRUPT SHOULD COME IN.
      ERROR,   T107      ;; *** Q-BUS INTERRUPT NOT RECEIVED ***
3$:    CLR      (R4)      ;; THAT'S ALL THERE IS TO IT.
      MOV      #QBXVEC+2,(R3)+ ;; RESET Q-BUS VECTOR.
      CLR      (R3)
4$:
*****
.SBTTL      T110 -- TEST THE 'BEVNT' INTERRUPT (REQUIRES Q-BUS EXERCISER)
*****
T110: BEGINTEST, 110      ;; CHECK SWITCH REGISTER OPTIONS.
  
```

```

2889 026746 012706 003776      MOV    #STACK,SP      ; SET STACK POINTER.
2890 026752 004767 006304      CALL   QBXIN        ; Q-BUS EXERCISER AVAILABLE ??
2891 026756 001433              BEQ     4$             ; IF NOT, BYPASS TEST.
2892 026760 010004              MOV     R0,R4           ; YES, CSR1 POINTER => R4 AND FALL THRU.
2893
2894      ; QBX CSR1<11> EMULATES A 'BEVNT' INTERRUPT.
2895      ;
2896 026762 106427 000300      MTPS    #PRI6        ; SET PRI 06
2897 026766 012767 027006 151104  MOV     #1$,BEVNT      ; SET BEVNT VECTOR.
2898 026774 005000              CLR      R0
2899 026776 012714 004000      MOV     #4000,(R4)    ; SET 'BEVNT' REQUEST.
2900 027002 077001              SOB     R0,,          ; DELAY ABOUT 250 MSEC, INTERRUPT...
2901 027004 000403              BR      2$             ;...SHOULD BE MASKED AT THIS LEVEL.
2902 027006              1$:
2903 027012 000410              ERROR,  T110          ;; *** BEVNT INTERRUPT LEVEL INCORRECT ***
2904 027014 012767 027034 151056 2$:  MOV     #3$,BEVNT      ; OK, CHANGE VECTOR...
2905 027022 106427 000240      MTPS    #PRI5        ;...AND LOWER CPU TO PRI 5.
2906 027026 077001              SOB     R0,,          ; DELAY AGAIN, INT SHOULD COME IN.
2907 027030 104423 026742      ERROR,  T110          ;; *** BEVNT INTERRUPT NOT RECEIVED ***
2908 027034 012767 000102 151036 3$:  MOV     #BEVNT+2,BEVNT  ; RESET VECTOR.
2909 027042 106427 000000      MTPS    #PRI0        ; LOWER CPU TO 0.
2910
2911 027046 004767 005656      4$:  CALL    ENDSEG      ; CHECK FOR LOOP-IN-SEGMENT...
2912                                ;...RETURN AND FALL THRU IF NOT.
2913      .SBTTL

```

CNKXAA.P11 17-MAR-82 14:02

SECTION 3 -- LOCAL 2K RAM TESTS.

.SBTTL SECTION 3 -- LOCAL 2K RAM TESTS.

```
2915
2916
2917 027052 012767 036245 007120 S3:  MOV    #SEG3,SEGN    ; SEGMENT 3 TEXT.
2918 027060 012767 160010 151720      MOV    #RAMADR,TEMP    ; SAVE RAM ADDRESS...
2919 027066 012767 003764 151714      MOV    #RAMSIZ,TEMP1    ; ...AND SIZE.
2920
2921 ;*****
2922 (2) .SBTTL      T111 -- LOCAL 2K RAM ADDRESS TEST
2923 (2) ;*****
2924 (2) 027074 104420 000111 T111:  BEGINTEST, 111    ;: CHECK SWITCH REGISTER OPTIONS.
2925 ;
2926 ; WRITE EACH RAM LOCATI'N WITH ITS OWN ADDRESS, READ BACK AND VERIFY.
2927 ; REPEAT A SECOND TIME USING BYTE-SWAPPED ADDRESS AS DATA.
2928
2929      MOV    #STACK,SP
2930      MOV    #7$,ERRVEC    ; SET TIME-OUT VECTOR.
2931      MOV    TEMP,R0        ; GET RAM ADDRESS...
2932      MOV    TEMP1,R1       ; ...AND SIZE.
2933      MOV    R1, -(SP)      ; SAVE A COPY.
2934
2935 1$:   MOV    R0,(R0)        ; LOAD ADDRESSES (FROM BOTTOM UP).
2936      TST    (R0)+          ; BUMP
2937      SOB    R1,1$         ; LOOP 'TIL DONE.
2938
2939      MOV    (SP),R2        ; RECOVER SIZE K.
2940      MOV    -(R0),R1       ; READ 'EM BACK (FROM TOP DOWN).
2941      CMP    R1,R0          ; DATA = ADDRESS ??
2942      BEQ    3$             ; BR IF SO.
2943      ERROR, T111           ;: *** RAM DATA INCORRECT AT ADDRESS IN R0 ***
2944      SOB    R2,2$         ; LOOP 'TIL DONE.
2945
2946 4$:   MOV    (SP),R2        ; RECOVER SIZE K.
2947      MOV    R0,R1          ; ADDRESS => R1...
2948      SWAB   R1             ; ...SWAP BYTES...
2949      MOV    R1,(R0)+       ; ...AND LOAD SWAPPED ADDRESSES.
2950      SOB    R2,4$
2951
2952 5$:   MOV    (SP),R3        ; RECOVER SIZE K.
2953      MOV    -(R0),R1       ; GET DATA.
2954      MOV    R0,R2          ;
2955      SWAB   R2             ;
2956      CMP    R1,R2          ; DATA = SWAPPED ADDRESS ??
2957      BEQ    6$             ; BR IF SO.
2958      ERROR, T111           ;: *** RAM DATA (SWAPPED) INCORRECT AT ADDRESS IN R0 ***
2959      SOB    R3,5$         ;
2960      BR     8$             ; DONE, EXIT.
2961
2962 7$:   ERROR, T111           ;: *** BUS TIME-OUT -- RAM ADDRESS IN R0 ***
2963 8$:
2964 ;*****
2965 (2) .SBTTL      T112 -- LOCAL 2K RAM DATA TEST
2966 (2) ;*****
2967 (2) 027212 104420 000112 T112:  BEGINTEST, 112    ;: CHECK SWITCH REGISTER OPTIONS.
2968 ;
```

```

2964      ; FILL RAM WITH A BACKGROUND PATTERN OF 0'S.
2965      ; FLOAT AND VERIFY A 1 BIT THRU EACH WORD, AND WHEN DONE
2966      ; RESTORE THAT WORD TO THE BACKGROUND.
2967      ; WHEN ALL WORDS DONE, VERIFY THAT THE ENTIRE RAM STILL CONTAINS
2968      ; THE CORRECT BACKGROUND DATA.
2969
2970      ; REPEAT A SECOND TIME WITH BACKGROUND 1, AND A FLOATING 0.
2971
2972 027216 012706 003776      MOV      #STACK,SP
2973 027222 012767 027376 150554  MOV      #10$,ERRVEC      ; SET TIME-OUT VECTOR.
2974 027230 005046      CLR      -(SP)      ; BACKGROUND = 0'S FIRST...
2975 027232 000402      SKP2
2976 027234 012746 177777 1$:      MOV      #-1,-(SP)      ;...THEN 1'S.
2977 027240 016700 151542      MOV      TEMP,R0      ; SET RAM ADDRESS...
2978 027244 016701 151540      MOV      TEMP1,R1      ;...AND SIZE.
2979 027250 011620 2$:      MOV      (SP),(R0)+      ; FILL RAM WITH BACKGROUND DATA.
2980 027252 077102      SOB      R1,2$
2981
2982 027254 016700 151526      MOV      TEMP,R0      ; RESET ADDRESS...
2983 027260 016701 151524      MOV      TEMP1,R1      ;...AND WORD COUNT...
2984 027264 011004 3$:      MOV      (R0),R4      ;...AND EXERCISE 1ST/NEXT LOCATION.
2985 027266 020416      CMP      R4,(SP)
2986 027270 001402      BEQ      4$
2987 027272 104423 027264      ERROR, 3$      ; BR IF BACKGROUND IS RIGHT.
2988 027276 012702 000001 4$:      MOV      #1,R2      ;: *** INITIAL BACKGROUND INCORRECT AT ADDRESS IN R0 ***
2989 027302 011603      MOV      (SP),R3
2990 027304 074203      XOR      R2,R3      ; COPY CURRENT BACKGROUND TO R3.
2991 027306 012702 000020      MOV      #16,R2      ; SET OR CLEAR BIT 0.
2992 027312 000402      SKP2      ; SET BIT COUNTER.
2993 027314 006303 5$:      ASL      R3
2994 027316 005503      ADC      R3      ; ROTATE DATA...
2995 027320 010310 6$:      MOV      R3,(R0)      ;...BIT 15 => BIT 0.
2996 027322 011004      MOV      (R0),R4      ; WRITE FLOATING DATA...
2997 027324 020304      CMP      R3,R4      ;...AND READ IT BACK.
2998 027326 001402      BEQ      7$
2999 027330 104423 027320      ERROR, 6$      ; BR IF ITS RIGHT.
3000 027334 077211 7$:      SOB      R2,5$      ;: *** FLOATING DATA INCORRECT AT ADDRESS IN R0 ***
3001 027336 011620      MOV      (SP),(R0)+      ; LOOP FOR ALL BITS.
3002 027340 077127      SOB      R1,3$      ; THEN RESTORE, BUMP ADDRESS...
3003      ;...AND LOOP FOR ALL WORDS.
3004 027342 016700 151440      MOV      TEMP,R0      ; NOW, RESET POINTERS...
3005 027346 016701 151436      MOV      TEMP1,R1      ;...AND VERIFY THAT RESTORED..
3006 027352 011004 8$:      MOV      (R0),R4      ;...BACKGROUND IS STILL CORRECT.
3007 027354 020416      CMP      R4,(SP)
3008 027356 001402      BEQ      9$
3009 027360 104423 027352      ERROR, 8$      ; BR IF SO.
3010 027364 005720 9$:      TST      (R0)+      ;: *** BACKGROUND CHANGED AT ADDRESS IN R0 ***
3011 027366 077107      SOB      R1,8$      ; BUMP...
3012      ;...AND LOOP 'TIL DONE.
3013 027370 005726      TST      (SP)+      ; FINALLY, IF BACKGROUND IS 0...
3014 027372 001720      BEQ      1$      ;...LOOP ONCE MORE...
3015 027374 000402      BR      11$      ;...OTHERWISE, WE'RE ALL DONE.
3016
3017 027376 104423 027212 10$:      ERROR, T112
3018 027402 11$:

```

```

3019
3020 ;*****
3021 ; SBTTL T113 -- OPTIONAL DMA TEST (REQUIRES Q-BUS EXERCISER)
3022 ;*****
3023 T113: BEGINTST, 113 ; CHECK SWITCH REGISTER OPTIONS.
3024 MOV #STACK, SP ; Q-BUS EXERCISOR AVAILABLE ??
3025 CALL QBXIN ; IF NOT, BYPASS TEST.
3026 BEQ 10$ ; YES, CSR1 POINTER => R4 AND FALL THRU.
3027 MOV R0, R4
3028 ;
3029 ; TRANSFER EACH RAM WORD TO THE QBX AND BACK.
3030 ; VERIFY THAT RETURNED DATA MATCHES THAT WHICH WAS SENT.
3031 ;
3032 MOV #9$, ERRVEC ; SET TRAP CATCHER.
3033 MOV TEMP, R1 ; RAM ADDRESS POINTER...
3034 MOV TEMP1, R2 ; ...AND WORD COUNT.
3035 2$: MOV R1, (R1) ; SET 1ST/NEXT RAM WORD.
3036 MOV #140407, (R4) ; DAT1 (MEM => QBX) IN 'HOG' MODE.
3037 MOV R1, 4(R4) ; FROM RAM (R1) TO DATA REGISTER...
3038 MOV #-4, 6(R4) ; ...4 TIMES.
3039 MOV #401, 2(R4) ; GO, INHIBIT BA+.
3040 CLR R0
3041 3$: TSTB 2(R4)
3042 BMI 33$ ; PROCEED WHEN DONE...
3043 SOB R0, 3$
3044 33$: BIT #40000, 2(R4) ; ...AND CHECK FOR NXM ERROR.
3045 BEQ 4$ ; BR IF OK.
3046 4$: CMP 10(R4), R1 ; *** NXM ERROR IN QBX ON DAT1 (MEM TO QBX) ***
3047 BEQ 5$ ; DATA REGISTER CORRECT ??
3048 ERROR, 2$ ; *** DATA INCORRECT IN QBX ON DAT1 (MEM TO QBX) ***
3049 5$: MOV #000601, (R4) ; DAT0 (QBX => MEM) IN 'HOG' MODE.
3050 MOV #TEMP2, 4(R4) ; FROM DATA REGISTER TO TEMP2...
3051 MOV #-4, 6(R4) ; ...4 TIMES.
3052 MOV #401, 2(R4) ; GO, INHIBIT BA+.
3053 CLR R0
3054 6$: TSTB 2(R4)
3055 BMI 66$ ; PROCEED WHEN DONE...
3056 SOB R0, 6$
3057 66$: BIT #40000, 2(R4) ; ...AND CHECK FOR NXM.
3058 BEQ 7$ ; BR IF OK.
3059 7$: CMP TEMP2, R1 ; *** NXM ERROR IN QBX ON DAT0 (QBX TO MEM) ***
3060 BEQ 8$ ; RETURNED DATA CORRECT ??
3061 ERROR, 5$ ; *** DATA INCORRECT IN TEMP2 ON DAT0 (QBX TO MEM) ***
3062 8$: TST (R1)+ ; SET NEXT ADDRESS...
3063 SOB R2, 2$ ; ...AND LOOP 'TIL DONE.
3064 CLR (R4) ; CLEAR QBX...
3065 BR 10$ ; ...AND EXIT.
3066 9$: ERROR, T113 ; *** BUS TIME-OUT -- QBX (R4) OR RAM (R1) ***
3067
3068
3069
3070

```

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).

D 7

MACY11 306(1063) 17-MAR-82 14:03 PAGE 4-3

CNKXAA.P11 17-MAR-82 14:02

PC 27624 = BUS TIME-OUT -- QBX (R4) OR RAM (R1).

SEQ 0081

```
3071 027630 012767 000006 150146 10$: MOV #ERRVEC+2,ERRVEC ; RESET ERROR VECTOR.
3072 027636 004767 005066 CALL ENDSEG ; CHECK FOR LOOP-IN-SEGMENT...
3073 ;...RETURN AND FALL THRU IF NOT.
3074 .SBTTL
```

SECTION 4 -- LOCAL PROM/RAM TESTS (EMPTY SOCKETS).

.SBTTL SECTION 4 -- LOCAL PROM/RAM TESTS (EMPTY SOCKETS).

3076						
3077						
3078	027642	012767	036263	006330	S4:	MOV #SEG4,SEGN ; SEGMENT 4 TEXT.
3079	027650	012767	170000	151130		MOV #ROMADR,TEMP ; SAVE ROM ADDRESS...
3080	027656	012767	002000	151124		MOV #ROMSIZ,TEMP1 ;...AND SIZE.
3081						
3082						*****
(2)						.SBTTL T114 -- LOCAL PROM/RAM ADDRESS TEST
(2)						*****
(2)	027664	104420	000114		T114:	BEGINTEST, 114 ;: CHECK SWITCH REGISTER OPTIONS.
3083						
3084						: PROM/RAM SOCKETS CONTAIN THE 1KW MACRO ODT PROM SET.
3085						: ATTEMPT TO WRITE (123456) INTO EACH LOCATION.
3086						: IF AN ADDRESS HAPPENS TO CONTAIN THAT VALUE, THEN WRITE AGAIN
3087						: WITH THE COMPLIMENT OF THAT VALUE.
3088						
3089	027670	012706	003776			MOV #STACK,SP
3090	027674	012767	027746	150102		MOV #3\$,ERRVEC ; SET TIME-OUT VECTOR.
3091	027702	016700	151100			MOV TEMP,R0 ; SET ADDRESS...
3092	027706	016701	151076			MOV TEMP1,R1 ;...AND SIZE.
3093						
3094	027712	012702	123456		1\$:	MOV #123456,R2
3095	027716	010210				MOV R2,(R0) ; 1 TO WRITE DATA (SHOULDN'T TRAP).
3096	027720	021002				CMP (R0),R2 ; D, IT WRITE ??
3097	027722	001006				BNE 2\$; BR IF NOT.
3098	027724	005102				COM R2 ; MAYBE...
3099	027726	010210				MOV R2,(R0) ;...TRY COMPLIMENT.
3100	027730	021002				CMP (R0),R2 ; HOW NOW ??
3101	027732	001002				BNE 2\$; BR IF OK.
3102	027734	104423	027712			ERROR, 1\$;: *** WRITE TO ROM ALTERED DATA ????? ***
3103						
3104	027740	005720			2\$:	TST (R0)+ ; BUMP POINTER.
3105	027742	077115				SQB R1,1\$; LOOP 'TIL DONE...
3106	027744	000402				BR 4\$;...AND PROCEED.
3107						
3108	027746				3\$:	
(2)	027746	104423	027664			ERROR, T114 ;: *** BUS TIME-OUT -- ROM ADDRESS IN R0 ***
3109						
3110	027752	012767	000006	150024	4\$:	MOV #ERRVEC+2,ERRVEC
3111	027760	004767	004744			CALL ENDSEG ; CHECK FOR LOOP-IN-SEGMENT...
3112						;...RETURN AND FALL THRU IF NOT.
3113						.SBTTL

.SBTTL SECTIONS 5 AND 6 -- SERIAL LINES 1 AND 2 TESTS.

: DLART BIT ASSIGNMENTS.

XMTBRK= 001
 XMTIE= 100
 XMTRDY= 200

RCVIE= 000100
 RCDUN= 000200
 RCVACT= 004000
 RCVBRK= 004000
 FRERR= 020000
 ORERR= 040000
 RCVERR= 100000

: THE FOLLOWING DLART TESTS ARE RUN TWICE PER PASS.
 : FIRST, ON SLU1 (THE CONSOLE SLOT).

S5: MOV #SEG5,SEGN ; SEGMENT 5 TEXT.
 MOV #SLU1,R0 ; 1ST UNIT CSR...
 MOV #SLU1V,R1 ; ...AND VECTOR(S).
 MOV #SLU1P,R2 ; SLU1 INTERRUPTS AT LEVEL 4.
 MOV #BADU1,R3 ; TIME-OUT VECTOR.
 BR DLSET ; SET-UP POINTERS AND XCT.

: AND THEN ON SLU2.

S6: MOV #SEG6,SEGN ; SEGMENT 6 TEXT.
 MOV #SLU2,R0 ; 2ND UNIT CSR...
 MOV #SLU2V,R1 ; ...AND VECTOR.
 MOV #SLU2P,R2 ; SLU2 INTERRUPTS AT LEVEL 5.
 MOV #BADU2,R3 ; TIME-OUT VECTOR.

DLSET:

:*****
 :.SBTTL T115 -- TEST SERIAL LINE REGISTER ADDRESSES
 :*****

T115: BEGINTEST, 115 ;: CHECK SWITCH REGISTER OPTIONS.
 MOV #STACK,SP ;: INIT THE STACK.
 MOV R3,ERRVEC ;: SET BUS TIME-OUT TRAP CATCHER.
 MOV R0,RCSR ;: LOAD UP THE REGISTER POINTERS.
 TST (R0)+ ;: IF ANY OF THESE TRAP, WE'RE DEAD !!
 MOV R0,RBUF
 TST (R0)+
 MOV R0,XCSR
 TST (R0)+
 MOV R0,XBUF
 MOV R1,RVEC ;: SAVE RCVR VECTOR POINTER...
 MOV #BADRI,(R1)+ ;: ...AND INIT VECTOR.
 MOV R2,(R1)+
 MOV R1,XVEC ;: SAVE XMTR VECTOR POINTER...
 MOV #BADTI,(R1)+ ;: ...AND INT VECTOR.
 MOV R2,(R1)+
 MOV @XCSR,R0 ;: NOW GET THE DEFAULT XMIT STATUS...
 BIC #^C72,R0 ;: ...AND STRIP THE BAUD RATE BITS.

3115
 3116
 3117
 3118
 3119 000001
 3120 000100
 3121 000200
 3122
 3123 000100
 3124 000200
 3125 004000
 3126 004000
 3127 020000
 3128 040000
 3129 100000
 3130
 3131
 3132
 3133
 3134 027764 012767 036301 006206
 3135 027772 012700 177560
 3136 027776 012701 000060
 3137 030002 012702 010200
 3138 030006 012703 030152
 3139 030012 000413
 3140
 3141
 3142
 3143 030014 012767 036317 006156
 3144 030022 012700 176540
 3145 030026 012701 000120
 3146 030032 012702 000240
 3147 030036 012703 030162
 3148
 3149 030042
 3150
 (2)
 (2)
 (2) 030042 104420 000115
 3151 030046 012706 003776
 3152 030052 010367 147726
 3153 030056 010037 150762
 3154 030062 005720
 3155 030064 010067 150716
 3156 030070 005720
 3157 030072 010067 150752
 3158 030076 005720
 3159 030100 010067 150746
 3160 030104 010167 150744
 3161 030110 012721 030172
 3162 030114 010221
 3163 030116 010167 150734
 3164 030122 012721 030202
 3165 030126 010221
 3166 030130 017700 150714
 3167 030134 042700 177705

```
3168 030140 052700 000002      BIS      #2,R0      ; INSURE 'PBRE' IS SET...
3169 030144 010067 150670      MOV      R0,DFPBR    ; ...AND SAVE AS THE DEFAULT BAUD RATE.
3170 030150 000432      BR       DLTSTS    ; START 'EM UP.
3171      ;
3172      ; COME HERE IF BUS TIME-OUT OR UNEXPECTED INTERRUPT.
3173      ;
3174 030152      BADU1:      ERROR, S5      ;; *** BUS TIME-OUT ON SLU1 ADDRESS ***
(1) 030152 104423 027764      JMP      DLDUN
3175 030156 000167 003270      BADU2:      ERROR, S6      ;; *** BUS TIME-OUT ON SLU2 ADDRESS ***
(1) 030162 104423 030014      JMP      DLDUN
3176 030162 000167 003260      BADRI:      ERROR, DLTSTS    ;; *** UNEXPECTED RCVR INTERRUPT. ***
(1) 030172 104423 030236      JMP      DLDUN
3177 030166 000167 003260      BADTI:      ERROR, DLTSTS    ;; *** UNEXPECTED XMTR INTERRUPT. ***
(1) 030172 104423 030236      JMP      DLDUN
3178 030172 000167 003250      BADTI:      ERROR, DLTSTS    ;; *** UNEXPECTED XMTR INTERRUPT. ***
(1) 030202 104423 030236      JMP      DLDUN
3179 030176 000167 003250      BADTI:      ERROR, DLTSTS    ;; *** UNEXPECTED XMTR INTERRUPT. ***
3180 030202 000167 003240      BADTI:      ERROR, DLTSTS    ;; *** UNEXPECTED XMTR INTERRUPT. ***
3181 030206 000167 003240      BADTI:      ERROR, DLTSTS    ;; *** UNEXPECTED XMTR INTERRUPT. ***
3182      ;
3183      ; FLOATING 1 AND 0 DATA TABLE, USED IN VARIOUS TESTS.
3184      ;
3185 030212      015      012      SYNC: .BYTE 015, 012      ; SYNC PAIR.
3186 030214      001      002      FLT1: .BYTE 001, 002, 004, 010, 020, 040, 100, 200
3187 030224      376      375      373  FLT0: .BYTE 376, 375, 373, 367, 357, 337, 277, 177
3188 030234 000000      .WORD 0      ; TERMINATOR.
3189      ;
3190 030236      DLTSTS:
3191      ;*****
(2) .SBTTL      T116 -- TEST THAT BUS-RESET CLEARS THE RIGHT BITS
(2)      ;*****
(2) 030236 104420 000116      T116: BEGINTEST, 116      ;; CHECK SWITCH REGISTER OPTIONS.
3192      ;
3193      ; SET ALL WRITABLE BITS AND RESET. CHECK THAT ONLY
3194      ; RCV-IE, XMIT-IE, MAINT, AND XMIT-BRK GET CLEARED.
3195      ;
3196 030242 106427 000300      MTPS      #PRI6      ; RAISE CPU
3197 030246 012777 000177 150574  MOV      #177,@XCSR    ; SET WRITABLE BITS IN XCSR...
3198 030254 012777 000377 150570  MOV      #377,@XBUF    ; ...AND XBUF...
3199 030262 012777 000100 150554  MOV      #100,@RCSR    ; ...AND RCSR.
3200 030270 000240 000240      240,240      ; IN CASE WE NEED A DELAY.
3201 030274 000005      RESET      240,240
3202 030276 000240 000240      MOV      @RCSR,R0      ; GET RCSR...
3203 030302 017700 150536      MOV      @RBUF,R1      ; ...RBUF (DUMMY READ)...
3204 030306 017701 150534      MOV      @XCSR,R2      ; ...XCSR...
3205 030312 017702 150532      MOV      @XBUF,R3      ; ...AND XBUF.
3206 030316 017703 150530      MOV      DFPBR,@XCSR    ; RESTORE DEFAULT XMIT...
3207 030322 016777 150512 150520  CLR      @RCSR      ; ...AND RCVR STATUS.
3208 030330 005077 150510      BIT      #100,R0      ; NOW, DID RCV-IE CLEAR ??
3209 030334 032700 000100      BEQ      1$
3210 030340 001402      ERROR, T116      ;; *** BUS-RESET DIDN'T CLEAR RCV-IE BIT IN RCSR ***
3211 030342 104423 030236      BIT      #105,R2      ; DID XIE, MAINT AND XBRK CLEAR ??
3212 030346 032702 000105      BEQ      2$
3213 030352 001402      ERROR, T116      ;; *** BUS-RESET DIDN'T CLEAR XIE, MAINT, AND/OR XBRK **
3214 030354 104423 030236      CMPB     R2,#272    ; ALL OTHERS SET (INCLUDING XMTRDY) ??
3215 030360 120227 000272      BEQ      3$
3216 030364 001402
```

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
CNKXAA.P11 17-MAR-82 14:02

H 7

MACY11 30G(1063) 17-MAR-82 14:03 PAGE 6-2
PC 30366 = BUS-RESET CLEARED WRONG BITS IN XCSR. SEQ 0085

```
3217 030366 104423 030236      ERROR, T116      ;; *** BUS-RESET CLEARED WRONG BITS IN XCSR ***
3218 030372 120327 177777      3$: CMPB R3,#-1      ; ALL BITS STILL IN XBUF ??
3219 030376 001402              BEQ 4$
3220 030400 104423 030236      ERROR, T116      ;; *** BUS-RESET CLEARED BITS IN XBUF ***
3221 030404 106427 000000      4$: MTPS #PRIO      ; LOWER THE CPU.
3222
3223 ;*****
(2) .SBTTL T117 -- TEST THAT ALL UNDEFINED BITS ARE ZERO
(2) ;*****
(2) T117: BEGINTEST, 117      ;; CHECK SWITCH REGISTER OPTIONS.
3224 030410 104420 000117      MOV @RCSR,R0      ; GET REGISTERS AGAIN.
3225 030414 017700 150424      MOV @RBUF,R1
3226 030420 017701 150422      MOV @XCSR,R2
3227 030424 017702 150420      MOV @XBUF,R3
3228 030430 017703 150416      BIT #173477,R0      ; RCSR, UNDEFINED BITS SET ??
3229 030434 032700 173477      BEQ 1$
3230 030440 001402              ERROR, T117      ;; *** UNDEFINED BITS SET IN RCVR STATUS ***
3231 030442 104423 030410      1$: BIT #13400,R1      ; RBUF, UNDEFINED BITS SET ??
3232 030446 032701 013400      BEQ 2$
3233 030452 001402              ERROR, T117      ;; *** UNDEFINED BITS SET IN RCVR DATA BUFFER ***
3234 030454 104423 030410      2$: BIT #177400,R2      ; XCSR, UNDEFINED BITS SET ??
3235 030460 032702 177400      BEQ 3$
3236 030464 001402              ERROR, T117      ;; *** UNDEFINED BITS SET IN XMTR STATUS ***
3237 030466 104423 030410      3$: BIT #177400,R3      ; XBUF, UNDEFINED BITS SET ??
3238 030472 032703 177400      BEQ 4$
3239 030476 001402              ERROR, T117      ;; *** UNDEFINED BITS SET IN XMTR DATA BUFFER ***
3240 030500 104423 030410      4$: TSTB @XCSR      ; XMTR READY ??
3241 030504 105777 150340      BMI 5$
3242 030510 100402              ERROR, T117      ;; *** XMITTER NOT READY ***
3243 030512 104423 030410      .PRINT      ; *** DEBUG, NOP ***
3244 030516 030516              5$: SKP2
3245 030516 000402              JMP DLDUN
3246 030520 000167 002726      .SBTTL
```

```

3248 .SBTTL TRANSMITTERS
3249 030524 XMTRS: ; TRANSMITTER TEST SECTION.
3250 ;*****
3251 (2) .SBTTL T120 -- FLOAT 1 AND 0 THRU THE XMIT DATA BUFFER
3252 (2) ;*****
3253 (2) 030524 104420 000120 T120: BEGINTEST, 120 ; CHECK SWITCH REGISTER OPTIONS.
3254 030530 012700 030214 MOV #FLT1,R0 ; TABLE POINTER.
3255 030534 112001 MOVB (R0)+,R1 ; GET 1ST CHARACTER.
3256 030536 010177 150310 1$: MOV R1,@XBUF ; LOAD DATA INTO 'XMIT DATA BUF'...
3257 030542 017702 150304 MOV @XBUF,R2 ; ...AND READ IT BACK.
3258 030546 120102 CMPB R1,R2
3259 030550 001402 BEQ 2$
3260 030552 104423 030536 ERROR, 1$ ; *** FLOATING 1 OR 0 INCORRECT IN XBUF ***
3261 030556 112001 2$: MOVB (R0)+,R1 ; GET NEXT CHAR...
3262 030560 001366 BNE 1$ ; ...AND LOOP 'TIL DONE.
3263 ;*****
3264 (2) .SBTTL T121 -- TEST THAT 'XMIT-BRK' CAN BE SET AND CLEARED
3265 (2) ;*****
3266 (2) 030562 104420 000121 T121: BEGINTEST, 121 ; CHECK SWITCH REGISTER OPTIONS.
3267 030566 106427 000340 MTPS #PRI7 ; RAISE CPU.
3268 030572 052777 000001 150250 BIS #1,@XCSR ; SET 'XMIT-BRK' BIT...
3269 030600 017700 150244 MOV @XCSR,R0 ; ...AND SAVE IN R0.
3270 030604 042777 000001 150236 BIC #1,@XCSR ; CLEAR IT...
3271 030612 017701 150232 MOV @XCSR,R1 ; ...AND SAVE IN R1.
3272 030616 016777 150216 150224 MOV DFPBR,@XCSR ; RESTORE DEFAULT XMTR.
3273 030624 106427 000000 MTPS #PRI0 ; AND PRIORITY.
3274 030630 032700 000001 BI #1,R0
3275 030634 001002 BNE 1$ ; BR IF IT SET
3276 030636 104423 030562 1$: ERROR, T121 ; *** XMIT-BRK BIT FAILED TO SET ***
3277 030642 032701 000001 BEQ 2$ ; BR IF IT CLEARED
3278 030646 001402 2$: ERROR, T121 ; *** XMIT-BRK BIT FAILED TO CLEAR ***
3279 030650 104423 030562
3280 030654
3281 ;*****
3282 (2) .SBTTL T122 -- TEST THAT 'XMIT-MAINT' CAN BE SET AND CLEARED
3283 (2) ;*****
3284 (2) 030654 104420 000122 T122: BEGINTEST, 122 ; CHECK SWITCH REGISTER OPTIONS.
3285 030660 052777 000004 150162 BIS #4,@XCSR ; SET 'XMIT-MAINT' BIT...
3286 030666 017700 150156 MOV @XCSR,R0 ; ...AND SAVE IN R0.
3287 030672 042777 000004 150150 BIC #4,@XCSR ; CLEAR IT...
3288 030700 017701 150144 MOV @XCSR,R1 ; ...AND SAVE IN R1.
3289 030704 016777 150130 150156 MOV DFPBR,@XCSR
3290 030712 032700 000004 BIT #4,R0
3291 030716 001002 BNE 1$ ; BR IF IT SET
3292 030720 104423 030654 1$: ERROR, T122 ; *** XMIT-MAINT BIT FAILED TO SET ***
3293 030724 032701 000004 BEQ 2$ ; BR IF IT CLEARED
3294 030730 001402 2$: ERROR, T122 ; *** XMIT-MAINT BIT FAILED TO CLEAR ***
3295 030732 104423 030654
3296 030736
3297 ;*****
3298 (2) .SBTTL T123 -- TEST THAT 'XMIT-PBRE' CAN BE SET AND CLEARED
3299 (2) ;*****
3300 (2) 030736 104420 000123 T123: BEGINTEST, 123 ; CHECK SWITCH REGISTER OPTIONS.

```

CNKXAA.P11 17-MAR-82 14:02

T123 -- TEST THAT 'XMIT-PBRE' CAN BE SET AND CLEARED

3292	030742	052777	000002	150100	RIS	#2,@XCSR	;SET 'XMIT-PBRE' BIT...
3293	030750	017700	150074		MOV	@XCSR,R0	...AND SAVE IN R0.
3294	030754	042777	000002	150066	BIC	#2,@XCSR	;CLEAR IT...
3295	030762	017701	150062		MOV	@XCSR,R1	...AND SAVE IN R1.
3296	030766	016777	150046	150054	MOV	DFPBR,@XCSR	
3297	030774	032700	000002		BIT	#2,R0	
3298	031000	001002			BNE	1\$	;BR IF IT SET
3299	031002	104423	030736		ERROR,	T123	;: *** XMIT-PBRE BIT FAILED TO SET ***
3300	031006	032701	000002		1\$: BIT	#2,R1	
3301	031012	001402			BEQ	2\$	;BR IF IT CLEARED
3302	031014	104423	030736		2\$: ERROR,	T123	;: *** XMIT-PBRE BIT FAILED TO CLEAR ***

3303 031020
3304
3305
3306 (2)
3307 (2)
3308 (2)
3309 (2)
3310 031020 104420 000124 150016
3311 031024 052777 000010 150016
3312 031032 017700 150012
3313 031036 042777 000010 150004
3314 031044 017701 150000
3315 031050 016777 147764 147772
3316 031056 032700 000010
3317 031062 001002
3318 031064 104423 031020
3319 031070 032701 000010
3320 031074 001402
3321 031076 104423 031020
3322 031102

:SBTTL T124 -- TEST THAT 'XMIT-PBR0' CAN BE SET AND CLEARED

T124: BEGINTEST, 124 ;: CHECK SWITCH REGISTER OPTIONS.
BIS #10,@XCSR ;SET 'XMIT-PBR0' BIT...
MOV @XCSR,R0 ;...AND SAVE IN R0.
BIC #10,@XCSR ;CLEAR IT...
MOV @XCSR,R1 ;...AND SAVE IN R1.
MOV DFPBR,@XCSR
BIT #10,R0
BNE 1\$;BR IF IT SET
ERROR, T124 ;: *** XMIT-PBR0 BIT FAILED TO SET ***
1\$: BIT #10,R1
BEQ 2\$;BR IF IT CLEARED
2\$: ERROR, T124 ;: *** XMIT-PBR0 BIT FAILED TO CLEAR ***

:SBTTL T125 -- TEST THAT 'XMIT-PBR1' CAN BE SET AND CLEARED

T125: BEGINTEST, 125 ;: CHECK SWITCH REGISTER OPTIONS.
BIS #20,@XCSR ;SET 'XMIT-PBR1' BIT...
MOV @XCSR,R0 ;...AND SAVE IN R0.
BIC #20,@XCSR ;CLEAR IT...
MOV @XCSR,R1 ;...AND SAVE IN R1.
MOV DFPBR,@XCSR
BIT #20,R0
BNE 1\$;BR IF IT SET
ERROR, T125 ;: *** XMIT-PBR1 BIT FAILED TO SET ***
1\$: BIT #20,R1
BEQ 2\$;BR IF IT CLEARED
2\$: ERROR, T125 ;: *** XMIT-PBR1 BIT FAILED TO CLEAR ***

:SBTTL T126 -- TEST THAT 'XMIT-PBR2' CAN BE SET AND CLEARED

T126: BEGINTEST, 126 ;: CHECK SWITCH REGISTER OPTIONS.
BIS #40,@XCSR ;SET 'XMIT-PBR2' BIT...
MOV @XCSR,R0 ;...AND SAVE IN R0.
BIC #40,@XCSR ;CLEAR IT...
MOV @XCSR,R1 ;...AND SAVE IN R1.
MOV DFPBR,@XCSR

3323 031102 104420 000125 147734
3324 031106 052777 000020 147730
3325 031114 017700 147730
3326 031120 042777 000020 147722
3327 031126 017701 147716
3328 031132 016777 147702 147710
3329 031140 032700 000020
3330 031144 001002
3331 031146 104423 031102
3332 031152 032701 000020
3333 031156 001402
3334 031160 104423 031102
3335 031164
3336 (2)
3337 (2)
3338 (2) 031164 104420 000126 147652
3339 031170 052777 000040 147652
3340 031176 017700 147646
3341 031202 042777 000040 147640
3342 031210 017701 147634
3343 031214 016777 147620 147626

```
3339 031222 032700 000040          BIT    #40,R0
3340 031226 001002          BNE     1$          ;BR IF IT SET
3341 031230 104423 031164          ERROR, T126          ;; *** XMIT-PBR2 BIT FAILED TO SET ***
3342 031234 032701 000040          1$: BIT    #40,R1
3343 031240 001402          BEQ     2$          ;BR IF IT CLEARED
3344 031242 104423 031164          ERROR, T126          ;; *** XMIT-PBR2 BIT FAILED TO CLEAR ***
3345 031246
3346
3347
(2)
(2)
(2) 031246 104420 000127          ;*****
3348 031252 106427 000300          .SBTTL    T127 -- TEST THAT 'XMIT-IE' CAN BE SET AND CLEARED
3349 031256 052777 000100 147564          ;*****
3350 031264 017700 147560          T127: BEGINTEST, 127          ;; CHECK SWITCH REGISTER OPTIONS.
3351 031270 042777 000100 147552          MTPS    #PRI6          ;; RAISE CPU PRIORITY
3352 031276 017701 147546          BIS     #100, @XCSR          ;SET 'XMIT-IE' BIT...
3353 031302 016777 147532 147540          MOV     @XCSR, R0          ;...AND SAVE IN R0.
3354 031310 106427 000000          BIC     #100, @XCSR          ; CLEAR IT...
3355 031314 032700 000100          MOV     @XCSR, R1          ;...AND SAVE IN R1.
3356 031320 001002          MOV     DFPBR, @XCSR          ; RESTORE XMTR...
3357 031322 104423 031246          MTPS    #PRIO          ;...AND CPU TOO.
3358 031326 032701 000100          BIT     #100, R0
3359 031332 001402          BNE     1$          ;BR IF IT SET
3360 031334 104423 031246          ERROR, T127          ;; *** XMIT-IE BIT FAILED TO SET ***
3361 031340          1$: BIT     #100, R1
3362          BEQ     2$          ; BR IF IT CLEARED.
3363          ERROR, T127          ;; *** XMIT-IE BIT FAILED TO CLEAR ***
3364
3365
(2)
(2)
(2) 031340 104420 000130          ;*****
3366 031344 016777 147470 147476          .SBTTL    T130 -- TEST THAT 'XMIT-RDY' CAN BE SET AND CLEARED
3367 031352 005001          ;*****
3368 031354 005002          T130: BEGINTEST, 130          ;; CHECK SWITCH REGISTER OPTIONS.
3369 031356 105777 147466          MOV     DFPBR, @XCSR          ; DELAY COUNT.
3370 031366 104423 031340          CLR     R1          ; DITTO
3371 031372 012777 000000 147452          CLR     R2          ;WAIT FOR 'XMIT-RDY' TO SET
3372 031400 017700 147444          1$: TSTB    @XCSR          ;KEEP TRYING.
3373 031404 032700 000200          BMI     2$          ;; *** XMIT-RDY FAILED TO SET ***
3374 031410 001402          SOB     R1, 1$          ;TRANSMIT A NULL CHARACTER
3375 031412 104423 031340          ERROR, T130          ;READ XMIT STATUS REG
3376 031416 105777 147426          MOV     @XCSR, R0          ; READY SHOULD BE GONE.
3377 031422 100403          BIT     #XMITRDY, R0          ; BR IF SO.
3378 031424 077204          BEQ     3$          ;; *** XMIT-RDY BIT FAILED TO CLEAR ***
3379 031426 104423 031340          TSTB    @XCSR          ;WAIT FOR 'XMIT-RDY' TO SET
3380 031432          BMI     4$          ; KEEP TRYING
3381          SOB     R2, 3$          ;; *** XMIT-RDY FAILED TO SET AFTER XMITTING CHARACTER *
3382          ERROR, T130
3383
3384
(2)
(2)
(2) 031432 104420 000131          ;*****
3385 031436 016777 147376 147404          .SBTTL    T131 -- TEST THAT WE CAN INTERRUPT ON 'XMT-RDY'
3386 031444 016701 147406          ;*****
3387 031450 106461 000002          T131: BEGINTEST, 131          ;; CHECK SWITCH REGISTER OPTIONS.
3388          MOV     DFPBR, @XCSR          ; GET VECTOR POINTER.
3389          MOV     XVEC, R1          ; RAISE CPU TO XMTR LEVEL.
3390          MTPS    2(R1)
```

```

3386 031454 012711 031500      MOV    #2$, (R1)      ; SET XMITTER VECTOR.
3387 031460 105777 147364      TSTB   @XCSR        ; WAIT FOR 'XMIT-RDY'
3388 031464 100375              BPL     1$              ;
3389 031466 052777 000100 147354 BIS    #100, @XCSR      ; SET ENABLE 'XMIT-IE' BIT
3390 031474 000240              240                    ; INTERRUPT SHOULD BE HELD OFF...
3391 031476 000406              BR      3$              ; ...BR IF SO.
3392 031500 022626              CMP    (SP)+, (SP)+      ; IT WASN'T.
3393 031502 042777 000100 147340 BIC    #100, @XCSR      ; REMOVE 'XMIT-IE'
3394 031510 104423 031432      ERROR, T131             ; *** XMT-RDY INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
3395
3396 031514 012711 031542      MOV    #4$, (R1)      ; CHANGE THE VECTOR.
3397 031520 106427 000000      MTPS   #PR10           ; LOWER CPU PRIORITY
3398 031524 000240              240                    ; INTERRUPT SHOULD HAVE COME IN.
3399 031526 042777 000100 147314 BIC    #100, @XCSR      ; IT DIDN'T, REMOVE 'XMIT-IE' BIT
3400 031534 104423 031432      ERROR, T131             ; *** XMT-RDY INTERRUPT NOT RECEIVED AT CPU LEVEL 0 ***
3401 031540 000415              BR      6$              ;
3402
3403      ; NOW, ON INTERRUPT, WE'RE HERE AT CPU LEVEL 4 (5 IF SLU2).
3404      ; LOWER CPU AGAIN AND CHECK THAT WE DON'T GET ANOTHER INTERRUPT.
3405
3406 031542 022626              4$: CMP    (SP)+, (SP)+
3407 031544 012711 031560      MOV    #5$, (R1)      ; CHANGE THE VECTOR.
3408 031550 106427 000000      MTPS   #PR10           ; LOWER THE CPU PRIORITY AGAIN.
3409 031554 000240              240                    ; SHOULDN'T GET ANOTHER...
3410 031556 000406              BR      6$              ; ...OK, EXIT.
3411 031560 022626              5$: CMP    (SP)+, (SP)+
3412 031562 042777 000100 147260 BIC    #100, @XCSR      ; 2ND INTERRUPT -- BAD NEWS.
3413 031570 104423 031432      ERROR, T131             ; REMOVE 'XMIT-IE'
3414      ; *** XMT INTERRUPT ACK FAILED TO REMOVE THE REQUEST **
3415 031574 042777 000100 147246 6$: BIC    #100, @XCSR      ; BE SURE THE ENABLE IS REMOVED.
3416 031602 012711 030202      MOV    #BADTI, (R1)      ; RESET FALSE INTR VECTOR
3417
3418      ;*****
3419      ;SBTTL      T132 -- TEST THAT EACH XMIT BAUD RATE SETS A DIFFERENT SPEED
3420      ;*****
3421      ;T132:  BEGINTEST, 132              ;: CHECK SWITCH REGISTER OPTIONS.
3422      ;
3423      ; FIRST DETERMINE THE RELATIVE SPEED OF EACH BAUD RATE.
3424      ; TRANSMIT A CHARACTER AND COUNT CPU BUS TRANSACTION CYCLES UNTIL
3425      ; TRANSMISSION IS COMPLETE (XMT-RDY ASSERTS).
3426      ; SAVE THE OBSERVED CYCLE COUNT IN THE SPEED TABLE.
3427      ; IF NO FLAG BY THE TIME THE COUNT OVERFLOWS (65536 CYCLES), SAVE
3428      ; THE ZERO, WHICH IMPLIES XMT RDY NEVER SET AT THAT SPEED.
3429      ;
3430      ; NOTE: ALL FOLLOWING RECEIVER TESTS WILL USE THE SPEED VALUES
3431      ; OBSERVED HERE AS LOOP CONTROL COUNTERS.
3432      ; NOTE:NOTE: IF THERE'S A TTY ATTACHED, YOU'LL SEE SOME GARBAGE
3433      ; CHARACTERS ON THE SCREEN.
3434
3435 031612 012700 000002      MOV    #2, R0              ; START WITH SPEED 0 (300 BAUD).
3436 031616 012701 001016      MOV    #SPDO, R1         ; TABLE POINTER.
3437 031622 005002              1$: CLR     R2              ; CLEAR THE ACCUMULATOR.
3438 031624 010077 147220      MOV    R0, @XCSR          ; SET PBR BITS.
3439 031630 012777 000000 147214 MOV    #0, @XBUF        ; XMIT A CHAR.
3440 031636 105777 147206      2$: TSTB   @XCSR          ; ON READY, CHAR HAS BEEN COPIED...
3441      ; ...INTO THE SERIAL OUTPUT REGISTER.

```

```

3439 031644 012777 000000 147200      MOV    #0,@XBUF      ; XMIT AGAIN AND MEASURE THE TIME...
3440                                     ; ...UNTIL READY RETURNS WHICH IS THE...
3441                                     ; ...SERIAL OUTPUT TIME.
3442 031652 105777 147172      3$:    TSTB    @XCSR      ; [ 4 ]
3443 031656 100403              BMI     4$              ; [ 1 ] WE'RE DONE ON XMT-RDY.
3444 031660 062702 000010      ADD     #8,R2          ; [ 2 ] COUNT 8 CYCLES/LOOP...
3445 031664 001372              BNE     3$              ; [ 1 ] ...UNTIL DONE (OR OVERFLOW).
3446 031666 010221      4$:    MOV     R2,(R1)+        ; OK, R2 = TX TIME IN BUS CYCLES...
3447                                     ; ...AT APPROX 3 USEC/COUN...
3448                                     ; ...SAVE IT IN THE SPEED TABLE.
3449 031670 020067 147144      CMP     R0,DFPBR        ; CURRENT RATE = DEFAULT ???
3450 031674 001002              BNE     5$              ; SKIP IF NOT.
3451 031676 010267 147140      MOV     R2,DFSPD        ; YES, SAVE AS DEFAULT SPEED.
3452 031702 062700 000010      5$:    ADD     #10,R0      ; INCREMENT PBR BITS.
3453 031706 020027 000072      CMP     R0,#72
3454 031712 101743              BLOS    1$
3455 031714 016777 147120 147126      MOV     DFPBR,@XCSR    ; LOOP FOR ALL 8 SETTINGS.
3456                                     ; DONE, RESTORE DEFAULT XMTTR...
3457                                     ; ...AND FALL THRU.
3458                                     ;
3459                                     ; NOW THE RELATIVE SPEED AT EACH SETTING HAS BEEN DETERMINED.
3460                                     ; CHECK THAT THE DIFFERENTIAL OF EACH ONE TO THE NEXT LOWER
3461                                     ; IS 2:1 (MORE OR LESS). THIS IS A GROSS SPEED CHECK SINCE
3462                                     ; WE CAN'T RELY ON A REAL CLOCK FOR TIMING.
3463                                     ;
3464                                     ; DIFFERENTIAL IS CALCULATED BY:
3465                                     ; (LOWER/2)-HIGHER = 0 +/- 25.0% OF THE LOWER.
3466                                     ;
3467                                     ; ON ERROR, R0 DISPLAYS THE HIGHER OF THE TWO RATES UNDER TEST.
3468 031722 012700 000072      MOV     #72,R0          ; WORK FROM THE HIGHEST DOWN.
3469 031726 012701 001034      MOV     #SPD7,R1         ; TABLE POINTER.
3470 031732 011102      10$:    MOV     (R1),R2        ; GET HIGHER (SMALLER) VALUE.
3471 031734 001002              BNE     11$
3472 031736 104423 031606      ERROR,  T132          ; *** XMT-RDY NEVER SET AT PBR IN R0 ***
3473 031742 020127 001016      11$:    CMP     R1,#SPD0    ; IF PBR 0, THERE'S NOTHING LOWER...
3474 031746 001421              BEQ     14$          ; ...SO JUST QUIT.
3475                                     ;
3476 031750 014103              MOV     -(R1),R3        ; GET NEXT LOWER (LARGER) VALUE.
3477 031752 010304              MOV     R3,R4          ; SAVE A COPY FOR THE TOLERANCE.
3478 031754 006203              ASR     R3              ; NOW, 1/2 THE LARGER...
3479 031756 160203              SUB     R2,R3          ; ...MINUS THE SMALLER = DELTA.
3480 031760 100001              BPL     12$
3481 031762 005403              NEG     R3              ; MAKE ABSOLUTE IF NECESSARY.
3482                                     ;
3483 031764 006204      12$:    ASR     R4              ; SET ALLOWABLE TOLERANCE...
3484 031766 006204              ASR     R4              ; ... = 25.0% OF THE LARGER.
3485 031770 000240              NOP
3486 031772 000240              NOP
3487 031774 020304              CMP     R3,R4          ; DELTA <= 25.0% ???
3488 031776 003402              BLE     13$          ; BR IF SO.
3489 032000 104423 031606      ERROR,  T132          ; *** SPEED DIFFERENTIAL ERROR ***
3490 032004 162700 000010      13$:    SUB     #10,R0      ; DEC RATE COUNT...
3491 032010 100350              BPL     10$          ; ...AND LOOP 'TIL DONE.
3492 032012
3493      .SBTTI

```

```

3495 .SBTTL RECEIVERS
3496 RCVR: ; RECEIVER TEST SECTION.
3497 ;*****
(2) .SBTTL T133 -- TEST THAT 'RCV-IE' CAN BE SET AND CLEARED
(2) ;*****
(2) T133: BEGINTEST, 133 ; CHECK SWITCH REGISTER OPTIONS.
3498 MTPS #PRI6 ; RAISE CPU PRIORITY
3499 MOV #100, @RCR ; SET 'RCV-IE' BIT...
3500 MOV @RCR, R0 ; ...AND SAVE IN R0.
3501 CLR @RCR ; CLEAR IT...
3502 MOV @RCR, R1 ; ...AND SAVE IN R1.
3503 BIT #100, R0
3504 BNE 1$ ; BR IF IT SET.
3505 ERROR, T133 ; *** RCV-IE BIT FAILED TO SET ***
3506 1$: BIT #100, R1
3507 BEQ 2$ ; BR IF IT CLEARED.
3508 ERROR, T133 ; *** RCV-IE BIT FAILED TO CLEAR ***
3509 2$: MTPS #PRI0
3510 ;*****
3511 .SBTTL T134 -- TEST THAT 'RCV-ACT' AND 'RCV-DONE' CAN SET AND CLEAR
3512 ;*****
(2) T134: BEGINTEST, 134 ; CHECK SWITCH REGISTER OPTIONS.
(2) ; THE TEST ENSURES THAT:
3513 1. THE INTERNAL 'MAINT' PATH IS FUNCTIONAL
3514 2. RCV-ACT CLEARS BEFORE RCV-DONE SETS.
3515 3. RCV-ACT CAN SET AND CLEAR
3516 4. READING RCV DATA REGISTER CLEARS RCV DONE
3517 ;
3518 ;
3519 MOV DFPBR, @XCSR
3520 BIS #4, @XCSR ; SET MAINT BIT.
3521 TSTB @XCSR ; INSURE WE'RE 'XMIT-RDY'
3522 BPL 1$
3523 MOV DFSPD, R0
3524 ASL R0 ; SET A 2 CHAR (20 BITS) DELAY COUNT...
3525 MOV R0, R1
3526 MOV R0, R2 ; ...AND SAVE 3 COPIES FOR LATER.
3527 MOV R0, R3
3528 SOB R0 ; WAIT 20 BIT TIMES...
3529 @RBUF, R0 ; ...AND DUMMY READ.
3530 MOV @RCR, R0 ; GET RCVR STATUS.
3531 BIT #RCVACT!RCVDUN, R0 ; BOTH ACTIVE AND DONE SHOULD BE CLEAR
3532 BEQ 2$
3533 ERROR, T134 ; *** RCV-ACT AND/OR RCV-DUN ARE SET IN ERROR ***
3534 2$: MOV #0, @RBUF ; NOW, XMIT A NULL CHARACTER
3535 240
3536 3$: MOV @RCR, R0 ; GET RCVR STATUS
3537 BIT #RCVACT, R0
3538 BNE 4$ ; BR IF AND WHEN ACTIVE SETS.
3539 SOB R1, 3$
3540 ERROR, T134 ; *** RCV-ACT FAILED TO SET ***
3541 4$: BIT #RCVDUN, R0 ; TEST THAT 'RCV-DUN' IS NOT SET YET.
3542 BEQ 5$
3543 ERROR, T134 ; *** RCV-DUN SET WHEN RCV-ACT WAS SET ***
3544 5$:

```

```
3545 032224 017700 146614      5$:  MOV  @RCSR,R0      ;READ RCVR STATUS AGAIN.
3546 032230 032700 004000      BIT  #RCVACT,R0
3547 032234 001403              BEQ   6$              ; BR IF AND WHEN ACTIVE CLEARS.
3548 032236 077206              SOB   R2,5$
3549 032240 104423 032074      ERROR, T134      ;: *** RCV-ACT SET BUT FAILED TO CLEAR ***
3550 032244 017700 146574      6$:  MOV  @RCSR,R0      ;READ RCVR STATUS
3551 032250 032700 000200      BIT  #RCVDUN,R0
3552 032254 001003              BNE   7$              ; BR IF AND WHEN DONE SETS.
3553 032256 077306              SOB   R3,6$
3554 032260 104423 032074      ERROR, T134      ;: *** RCV-DUN FAILED TO SET WHEN RCV-ACT CLEARED ***
3555 032264 017700 146556      7$:  MOV  @RBUF,R0      ; OK, READ SHOULD LOWER DONE FLAG.
3556 032270 017700 146550      MOV  @RCSR,R0      ; GET STATUS.
3557 032274 032700 000200      BIT  #RCVDUN,R0
3558 032300 001402              BEQ   8$              ; BR IF DONE IS CLEAR.
3559 032302 104423 032074      ERROR, T134      ;: *** RCV-DUN FAILED TO CLEAR AFTER READING DATA ***
3560 032306
3561
3562 ;*****
3563 (2) ;SBTTL      T135 -- TEST THE 'RCV-BRK', 'RCV-ERR', AND 'FR-FRR' BITS
3564 (2) ;*****
3565 (2) 032306 104420 000135 ;T135.  BEGINTEST, 135      ;: CHECK SWITCH REGISTER OPTIONS.
3566 ;
3567 ; GENERATE A FRAMING ERROR BY SETTING THE RCV-BRK BIT.
3568 ; ON SLU1 -- EXPECT THAT BRK INVOKES A MASKABLE LEVEL 7 HALT REQUEST.
3569 ;
3570 032312 013746 000140      1$:  MOV  @#BHALT,--(SP) ; SAVE BREAK VECTOR.
3571 032316 106427 000370      MTPS  #PRI7      ; RAISE CPU.
3572 032322 016777 146512      MOV  DFPBR,@XCSR
3573 032330 052777 000004 146512  BIS  #4,@XCSR      ; SET MAINT BIT.
3574 032336 017700 146504      MOV  @RBUF,R0      ; DUMMY READ.
3575 032342 012757 032476 145570  MOV  #6$,@BHALT      ; SET THE HALT VECTOR (FOR SLU1).
3576 032350 012737 000340 000142  MOV  #PRI7,@#BHALT+2
3577 032356 016701 146460      MOV  DFPD,R1
3578 032362 006301              ASL   R1              ; SET A 2 CHAR (20 BITS) DELAY.
3579 032364 052777 000001 146456  BIS  #XMTBRK,@XCSR      ; SET 'XBRK'...
3580 032372 012777 000025 146452  MCV  #25,@XBUF      ;...START XMITTER...
3581 032400 077101              SOB   R1,,          ;...AND DELAY. IF SLU1, INTERRUPT...
3582 032402 105777 146436      TSTB  @RCSR      ;...SHOULD BE HELD OFF (CPU IS AT 7).
3583 032406 100402              BMI   2$              ; DONE SHOULD BE THERE BY NOW.
3584 032410 104423 032316      ERROR, 1$              ; BR IF SO.
3585 032414 026727 146424 176540  2$:  CMP  RCSR,#SLU2      ;: *** NO RCV-DUN AFTER XMIT WITH XMT-BRK SET ***
3586 032422 001430              BEQ   7$              ; IF SLU2...
3587 ;: JUST VERIFY STATUS/DATA.
3588 ;
3589 ; SLU1, VERIFY THAT 'BHALT' INTERRUPT DOES THE RIGHT THINGS.
3590 032424 012767 032446 145506  3$:  MOV  #4$,@BHALT      ; SO FAR, SO GOOD, CHANGE THE VECTOR...
3591 032432 106427 000300      MTPS  #PRI6      ;...AND LOWER CPU TO 6...
3592 032436 000240 240              ;...INTERRUPT SHOULD COME IN.
3593 032440 104423 032316      ERROR, 1$              ;: *** HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6 ***
3594 032444 000417 7$              BR      7$
3595 032446 022626 4$:  CMP  (SP)+,(SP)+      ; OK, INT RECEIVED, FIX STACK.
3596 032450 012767 032466 145462  MOV  #5$,@BHALT      ; NOW, CHANGE THE VECTOR...
3597 032456 106427 000300      MTPS  #PRI6      ;...LOWER CPU AGAIN...
3598 032462 000240 240              ;...AND SEE THAT WE DON'T GET ANOTHER.
3599 032464 000407 7$              BR      7$              ; TERRIFIC -- PROCEED.
```

```

3598 032466 022626 032316 5$: CMP (SP)+,(SP)+ ; 2ND HALT TRAP -- TOUGH LUCK !!
3599 032470 104423 032316 ERROR, 1$ ;: *** DOUBLE HALT TRAP RECEIVED ***
3600 032474 000403 BR 7$
3601 032476 022626 6$: CMP (SP)+,(SP)+ ; WE'RE HERE IF HALT WASN'T MASKED !!
3602 032500 104423 032316 ERROR, 1$ ;: *** HALT TRAP INTERRUPT NOT MASKED BY CPU LEVEL 7 ***
3603
3604 ; BOTH LINES VERIFY CORRECT STATUS AND DATA BITS.
3605
3606 032504 017700 146336 7$: MOV @RBUF,R0 ; READ (IF SLU1, LOWERS BRK REQ)..
3607 032510 012637 000140 MOV (SP)+,@#BHALT ;...AND RESTORE BREAK VECTOR.
3608 032514 032700 004000 BIT #RCVBRK,R0 ; RCV-BRK SHOULD BE SET.
3609 032520 001002 BNE 8$
3610 032522 104423 032316 ERROR, 1$ ;: *** RCV-BRK BIT DIDN'T SET ***
3611 032526 032700 100000 8$: BIT #RCVERR,R0 ; SO SHOULD RCV-ERR...
3612 032532 001403 BEQ 9$
3613 032534 032700 020000 BIT #FRERR,R0 ;...AND FR-ERR.
3614 032540 001002 BNE 10$
3615 032542 9$: ERROR, 1$ ;: *** RCV-ERR AND/OR FR-ERR BITS DIDN'T SET ***
(1) 032542 104423 032316 10$: TSTB R0 ; RCV DATA SHOULD BE ZERO.
3616 032546 105700 BEQ 11$
3617 032550 001402 ERROR, 1$ ;: *** RECEIVED DATA NON-ZERO ON 'BRK' ***
3618 032552 104423 032316
3619 ; AND FINALLY, WE SHOULD BE ABLE TO RESTORE A NORMAL RECEIVER.
3620
3621 11$: BIC #XMTBRK,@XCSR ; NOW, CLEAR THE BREAK BIT...
3622 032556 042777 000001 146264 MOV DFSPD,R0
3623 032564 016700 146252 SOB R0 ;...DELAY...
3624 032570 077001 MOV @RBUF,R0 ;...DUMMY READ.
3625 032572 017700 146250 MOV #25,@XBUF ; XMIT A CHAR...
3626 032576 012777 000025 146246 12$: TSTB @RCSR
3627 032604 105777 146234 BPL 13$ ;...WAIT FOR RCVR DONE...
3628 032610 100375 MOV @RBUF,R0 ;...AND READ.
3629 032612 017700 146230 BIT #RCVERR!FRERR!RCVBRK,R0 ; ERRORS SHOULD BE GONE.
3630 032616 032700 124000 BEQ 13$ ; BR IF SO.
3631 032622 001402 ERROR, 1$ ;: *** RCV-ERR, FR-ERR, AND/OR BRK BITS DIDN'T CLEAR ***
3632 032624 104423 032556 13$: CMPB R0,#25
3633 032630 120027 000025 BEQ 14$ ; AND CORRECT DATA RECEIVED.
3634 032634 001402 ERROR, 1$ ;: *** DATA INCORRECT AFTER 'BRK' SEQUENCE ***
3635 032636 104423 032556 14$: MTPS #PRIO ; DONE, LOWER THE CPU.
3636 032642 106427 000000
3637
3638 ;*****
(2) ;BTTL 1136 -- TEST THAT 'RCV-ERR' AND 'FR-ERR' BITS SET AND CLEAR
(2) ;*****
(2) 032646 104420 000136 1136: BEGINTST, 136 ;: CHECK SWITCH REGISTER OPTIONS.
3639 032652 016777 146162 146170 MOV DFSPD,@XCSR
3640 032660 052777 000004 146162 BIS #4,@XCSR ; SET MAINT BIT.
3641 032666 017700 146154 MOV @RBUF,R0 ; DUMMY READ.
3642 032672 012701 000004 MOV #4,R1
3643 032676 012777 000000 146146 1$: MOV #0,@XBUF ;XMIT A CHARACTER
3644 032704 105777 146140 2$: TSTB @XCSR ;WAIT FOR XMIT-RDY
3645 032710 100375 BPL 3$
3646 032712 077107 SOB R1,1$ ; LOOP 4 TIMES...
3647 ;...SHOULD CAUSE AN OVER-RUN ERROR.
3648 032714 016700 146122 MOV DFSPD,R0
3649 032720 077001 SOB R0 ; WAIT 1 ADDITIONAL CHAR TIME...

```

```

3650 032722 017700 146120      MOV    @RBUF,R0      ;...AND READ RCV DATA BUFFER.
3651 032726 100003      BPL     3$          ; BR IF NO 'RCV-ERR'.
3652 032730 052700 000000      BIT     #ORERR,R0      ; 'OR-ERR' SHOULD BE SET AS WELL.
3653 032734 001002      BNE     4$          ; BOTH SET -- TERRIFIC.
3654 032736      3$:      ERROR, T136      ;: *** RCV-ERR AND/OR OR-ERR NOT SET ON FORCED OVER-RUN
(2) 032738 104423 000000
3655      ;
3656 032742 012777 000000 146102 4$:      MOV    #0,@XBUF      ; NOW XMIT AGAIN...
3657 032750 105777 146070      5$:      TSTB   @RCSR      ;...WAIT FOR RCVR DUN...
3658 032754 100375      BPL     5$
3659 032756 017700 146066      MOV    @RBUF,R0      ;...AND READ, ERROR SHOULD BE GONE.
3660 032762 032700 140000      BIT     #RCVERR!ORERR,R0
3661 032766 001402      BEQ     6$          ; BR IF SO.
3662 032770 104423 032646      6$:      ERROR, T136      ;: *** RCV-ERR AND/OR OR-ERR BITS FAILED TO CLEAR ***
3663 032774
3664
3665 ;*****
(2) ;SBTTL      T137 -- TEST THAT WE CAN INTERRUPT ON 'RCV-DUN'
(2) ;*****
(2) 032774 104420 000137      T137:    BEGINTEST, 137      ;: CHECK SWITCH REGISTER OPTIONS.
3666 033000 016777 146034 146042      MOV    DFPBR,@XCSR
3667 033006 052777 000004 146034      BIS     #4,@XCSR      ; SET MAINT BIT.
3668 033014 016701 146034      MOV    RVEC,R1      ; GET RCVR VECTOR POINTER.
3669 033020 106461 000002      MTPS   2(R1)      ; RAISE CPU TO RCVR LEVEL.
3670 033024 012711 033056      MOV    #2$,(R1)      ; SET RCVR VECTOR.
3671 033030 012777 000000 146014      MOV    #0,@XBUF      ; TRANSMIT A CHARACTER
3672 033036 105777 146002      1$:      TSTB   @RCSR      ; WAIT FOR 'RCV-DUN'
3673 033042 100375      BPL     1$
3674 033044 052777 000100 145772      BIS     #100,@RCSR      ; SET ENABLE 'RCV-IE' BIT
3675 033052 000240      240      ; INTERRUPT SHOULD BE HELD OFF...
3676 033054 000406      BR      3$          ;...BR IF SO.
3677 033056 022626      2$:      CMP     (SP)+,(SP)+      ; BUT IT WASN'T.
3678 033060 042777 000100 145756      BIC     #100,@RCSR      ; REMOVE 'RCV-IE'
3679 033066 104423 032774      ERROR, T137      ;: *** RCV-DUN INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
3680
3681 033072 012711 033120      3$:      MOV    #4$,(R1)      ; CHANGE VECTOR.
3682 033076 106427 000000      MTPS   #PR10      ; LOWER CPU PRIORITY
3683 033102 000240      240      ; INTERRUPT SHOULD COME IN.
3684 033104 042777 000100 145732      BIC     #100,@RCSR      ; BUT IT DIDN'T, REMOVE 'RCV-IE' BIT
3685 033112 104423 032774      ERROR, T137      ;: *** RCV-DUN INTERRUPT NOT RECEIVED AT ***
3686 033116 000415      BR      6$
3687
3688 ;
3689 ; NOW, ON INTERRUPT, WE'RE HERE AT LEVEL 4 (5 IF SLU2).
3690 ; LOWER THE CPU AND VERIFY THAT WE DON'T GET ANOTHER ONE.
3691 033120 022626      4$:      CMP     (SP)+,(SP)+
3692 033122 012711 033136      MOV    #5$,(R1)      ; CHANGE THE VECTOR.
3693 033126 106427 000000      MTPS   #PR10      ; LOWER THE CPU PRIORITY AGAIN.
3694 033132 000240      240      ; SHOULDN'T GET ANOTHER...
3695 033134 000406      BR      6$          ;...OK, EXIT.
3696 033136 022626      5$:      CMP     (SP)+,(SP)+      ; 2ND INTERRUPT, -- TOO BAD !!
3697 033140 042777 000100 145676      BIC     #100,@RCSR      ; REMOVE 'RCV-IE'
3698 033146 104423 032774      ERROR, T137      ;: *** INTERRUPT ACK FAILED TO REMOVE THE REQUEST ***
3699
3700 033152 042777 000100 145664      6$:      BIC     #100,@RCSR      ; INSURE THE ENABLE IS OFF.
3701 033160 017700 145662      MOV    @RBUF,R0      ; READ

```

```

3702 033164 012711 030172          MOV      #BADRI,(R1)          ;RESET FALSE INTR VECTOR
3703
3704
(2)
(2)
(2) 033170 104420 000140          ;*****
;SBTTL      T140 -- FLOAT 1 AND 0 THRU THE 'MAINT' DATA LOOP (INTERNAL)
;*****
T140:  BEGINTEST, 140              ;; CHECK SWITCH REGISTER OPTIONS.
;
; NOTE: IF THERE'S A TERMINAL ATTACHED, YOU'LL SEE GARBAGE
; CHARACTERS ON THE SCREEN -- DON'T WORRY ABOUT IT !!
;
3709 033174 016777 145640 145646    MOV      DFPBR,@XCSR
3710 033202 052777 000004 145640    BIS      #4,@XCSR          ; SET MAINT BIT.
3711 033210 012700 030212          MOV      #SYNC,R0          ; DATA TABLE POINTER.
3712 033214 112077 145632          1$:  MOVVB   (R0)+,@XBUF      ; XMIT THE SYNC PAIR <CR>...
3713 033220 105777 145624          2$:  TSTB    @XCSR
3714 033224 100375                BPL      2$
3715 033226 112077 145620          MOVVB   (R0)+,@XBUF      ;...AND <LF>.
3716 033232 105777 145606          3$:  TSTB    @RCR          ; WAIT FOR RCVR DONE...
3717 033236 100375                BPL      3$
3718 033240 127727 145602 000012    CMPB    @RBUF,#12        ; AND PROCEED ON THE RCVD <LF>.
3719 033246 001371                BNE      3$
3720
3721 033250 112001                4$:  MOVVB   (R0)+,R1          ; NOW FLOAT DATA THRU THE LOOP.
3722 033252 010177 145574          5$:  MOV      R1,@XBUF          ; XMIT A BYTE...
3723 033256 105777 145562          6$:  TSTB    @RCR
3724 033262 100375                BPL      6$
3725 033264 017702 145556          MOV      @RBUF,R2          ;...AND READ IT BACK.
3726 033270 120102                CMPB    R1,R2
3727 033272 001402                BEQ      7$
3728 033274 104423 033252          ERROR,  5$
3729 033300 105701                7$:  TSTB    R1
3730 033302 001362                BNE      4$          ;: *** RCVD DATA INCORRECT, MAINT DATA PATH FAILURE ***
;: TERMINATOR DONE ??
;: NOT YET, LOOP.
;*****
;SBTTL      T141 -- FLOAT 1 AND 0 THRU THE LOOP-BACK (EXTERNAL, SLU2 ONLY)
;*****
(2)
(2)
(2) 033304 104420 000141          T141:  BEGINTEST, 141      ;; CHECK SWITCH REGISTER OPTIONS.
3733 033310 026727 145530 177560    CMP      RCSR,#SLU1      ; CONSOLE LINE ??
3734 033316 001455                BEQ      DLDUN          ; DON'T IF SO.
3735
3736 033320 016777 145514 145522    MOV      DFPBR,@XCSR      ; DEFAULT XMIT STATUS (NO MAINT).
3737 033326 017700 145514          MOV      @RBUF,R0          ; DUMMY READ.
3738 033332 016700 145504          MOV      DFSPD,R0
3739 033336 006300                ASL      R0
3740 033340 006300                ASL      R0          ; 4 CHAR LOOP TIMER.
3741 033342 010001                MOV      R0,R1          ; SAVE A COPY.
3742 033344 112777 000033 145500    MOVVB   #33,@XBUF      ; XMIT AN 'ESC'.
3743 033352 105777 145466          1$:  TSTB    @RCR          ; IF RCVR-DUN SETS NOW, WE CAN ASSUME...
3744 033356 100411                BMI      3$          ;...THAT THE LOOP-BACK IS INSTALLED.
3745 033360 077001                SOB      R0,1$
3746 033362 112777 000132 145462    MOVVB   #12,@XBUF      ; IF NOT, XMIT 'Z'.
3747 033370 105777 145450          2$:  TSTB    @RCR          ; IF IT SETS NOW, WE HAVE A TTY...
3748 033374 100416                BMI      DLDUN          ;...RESPONDING TO THE 'ESC Z' SEQUENCE.
3749 033376 077104                SOB      R1,2$          ; IF NEITHER, THE LOOP MUST BE OPEN.
3750 033400 104423 033304          ERROR,  T141
3751 033404 000422                BR       DLDUN          ;: *** NO RCV-DUN -- IS LOOP CONNECTOR INSTALLED ??? ***
;: QUIT (WHILE WE'RE AHEAD) !!

```

```

3752
3753 033406 017700 145334      3$:  MOV  @RBUF,R0      ; OK, DUMMY READ...
3754 033412 012700 030214      MOV  #FLT1,R0      ;...SET TABLE POINTER...
3755 033416 112001              4$:  MOVB  (R0)+,R1      ;...AND FLOAT DATA THRU THE LOOP.
3756 033420 010177 145426      5$:  MOV  R1,@XBUF      ; XMIT A BYTE...
3757 033424 105777 145414      6$:  TSTB  @RCSR
3758 033430 100375              BPL  6$
3759 033432 017702 145410      MOV  @RBUF,R2      ;...AND READ IT BACK.
3760 033436 120102              CMPB  R1,R2
3761 033440 001402              BEJ  7$
3762 033442 104423 033420      ERROR, 5$
3763 033446 105701              7$:  TSTB  R1
3764 033450 001362              BNE  4$
3765
3766 033452 016777 145362 145370 DLDUN: MOV  DFPBR,@XCSR      ; INSURE A DEFAULT XMITTER...
3767 033460 017700 145362      MOV  @RBUF,R0      ;...AND RECEIVER.
3768 033464 012767 000006 144312      MOV  #ERRVEC+2,ERRVEC
3769 033472 004767 001232      CALL  ENDSEG
3770
3771 033476 026727 145342 176540      CMP  RCSR,#SLU2      ; CHECK FOR LOOP-IN-SEGMENT...
3772 033504 001402              BEQ  S7      ;...RETURN AND FALL THRU IF NOT.
3773 033506 000167 174302              JMP  S6      ; LINE 2 ??
3774                                ; BR IF SO...
                                ;...ELSE, GO 'ROUND ONCE.
                                .SBTTL

```

```
3776 .SBTTL SECTION 7 -- PARALLEL I/O PORT (8255) TESTS.
3777
3778 : PORTS A, B, AND C MUST BE INTERCONNECTED THRU A SPECIAL
3779 : LOOP-BACK CONNECTOR. IF NOT THESE TESTS BREAK.
3780
3781 PPM0= 233 ; CONTROL WORD TO SET MODE 0 (ALL INPUT).
3782 PPM1= 264 ; ANOTHER FOR MODE 1, A IN, B OUT, C<7:6> OUT.
3783
3784 LEDLIT= 200 ; LED CONTROL BIT (0 = ON, 1 = OFF).
3785 INRDY= 040 ; INPUT DATA READY (IBF).
3786 INENA= 020 ; ENABLE INTERRUPT ON 'INRDY'
3787 ININT= 010 ; INPUT INTERRUPT BIT.
3788
3789 OUTENA= 004 ; ENABLE INTERRUPT ON 'OUTRDY'.
3790 OUTRDY= 002 ; OUTPUT PORT READY (OBF).
3791 OUTINT= 001 ; OUTPUT INTERRUPT BIT.
3792
3793 033512 012767 036335 002460 S7: MOV #SEG7,SEGN ; SEGMENT 7 TEXT.
3794
3795 *****
3796 (2) .SBTTL T142 -- TEST PIO REGISTER ADDRESSES
3797 (2) *****
3798 (2) T142: BEGINTEST, 142 ;: CHECK SWITCH REGISTER OPTIONS.
3799 033520 104420 000142 ;: *** TRY A RESET HERE ***
3800 033524 000240 000240 ;: A IS 1ST OF 4 REGISTERS.
3801 033530 012700 176200 ;: B IS 1ST OF 2 VECTOR PAIRS.
3802 033534 012701 000130 ;: PRIORITY (5) FOR BOTH.
3803 033540 012702 000240 ;: SET TIME-OUT VECTOR.
3804 033544 012767 033634 144232 MOV #BADPIO,ERRVEC
3805 033552 012706 003776 MOV #STACK,SP
3806 033556 010067 145276 MOV R0,PPA ;: SET REGISTER POINTERS.
3807 033562 005720 TST (R0)+ ;: IF ANY TRAP -- TROUBLE !a!
3808 033564 010067 145272 MOV R0,PPB
3809 033570 005720 TST (R0)+
3810 033572 010067 145266 MOV R0,PPC
3811 033576 005720 TST (R0)+
3812 033600 010067 145262 MOV R0,PPCW
3813 033604 005710 TST (R0)
3814 033606 010167 145260 MOV R1,PPBV ;: SAVE B VECTOR POINTER.
3815 033612 012721 033644 MOV #BADBI,(R1)+ ;: SET TO CATCH UNEXPECTED INTERRUPTS.
3816 033616 010221 MOV R2,(R1)+
3817 033620 010167 145244 MOV R1,PPAV ;: DITTO FOR PORT A VECTOR.
3818 033624 012721 033654 MOV #BADA1,(R1)+
3819 033630 010221 MOV R2,(R1)+
3820 033632 000414 BR PIOTST
3821
3822 : COME HERE IF TIME-OUT OR UNEXPECTED PIO INTERRUPTS.
3823
3824 BADPIO:
3825 ERROR, T142 ;: ** BUS TIME-OUT ON PIO ADDRESS ***
3826 JMP PIODUN
3827
3828 BADBI:
3829 ERROR, PIOTST ;: *** UNEXPECTED PORT B INTERRUPT ***
3830 JMP PIODIN
3831
3832 BADA1:
3833 ERROR, PIOTST ;: *** UNEXPECTED PORT A INTERRUPT ***
3834 JMP PIODUN
```

3826 033664
3827 (2)
3828 (2)
3829 (2) 033664 104420 000143
3830
3831
3832
3833 033670 000240 000240
3834 033674 000005
3835 033676 005000
3836 033700 077001
3837 033702 117700 145152
3838 033706 120027 177777
3839 033712 001402
3840 033714 104423 033664
3841 033720
3842
3843 (2)
3844 (2)
3845 (2) 033720 104420 000144
3846 033724 112777 000264 145134
3847 033732 112777 000016 145126
3848 033740 112777 000014 145120
3849 033746 012700 000012
3850 033752 077001
3851 033754 117703 145100
3852 033760 117703 145100
3853 033764 120327 000002
3854 033770 001402
3855 033772 104423 033720
3856 033776 005001
3857 034000 012702 177777
3858 034004 005000
3859 034006 110177 145050
3860 034012 117703 145046
3861 034016 032703 000002
3862 034022 001402
3863 034024 104423 034004
3864 034030 117703 145030
3865 034034 120327 000042
3866 034040 001403
3867 034042 077006
3868 034044 104423 034004
3869 034050 117702 145004
3870 034054 117703 145004
3871 034060 120327 000002
3872 034064 001402
3873 034066 104423 034004
3874 034072 120201
3875 034074 001402
3876 034076 104423 034004

PIOTST:

:SBTTL T143 -- TEST THAT RESET INVOKES MODE 0 AND CLEARS THE CHIP

T143: BEGINTEST, 143 ;: CHECK SWITCH REGISTER OPTIONS.
:
: VERIFY THAT PORT A (AND IT'S BUFFER) IS IN THE 'RESET' STATE.
: PORTS B AND C WILL BE INDETERMINATE DUE TO THE LOOP-BACK INTERCONNECTS
: AND THE CONTROL WORD PORT IS 'WO', SO DON'T BOTHER WITH THOSE.
:
: 240,240
: RESET ;: LED (ON C<7>) SHOULD BE OFF.
: CLR R0 ;: DELAY, CHIP SEEMS TO TAKE QUITE...
: SOB R0 ;: ...A WHILE (200USEC) TO GET 'RESET'.
: MOVB @PPA,R0 ;: READ PORT A (INPUT).
: CMPB R0,#-1 ;: LO BYTE ONLY -- SHOULD BE HIGH.
: BEQ 1\$
: ERROR, T143 ;: *** PORT A RESET STATE INCORRECT ***
1\$:

:SBTTL T144 -- TEST MODE 1 I/O THRU THE LOOP CONNECTOR (FLAG MODE)

T144: BEGINTEST, 144 ;: CHECK SWITCH REGISTER OPTIONS.
: MOVB #PPM1,@PPCW ;: SET MODE 1, A IN, B OUT, C<7:6> OUT.
: MOVB #<7-1>!0,@PPCW ;: CLEAR C(7) TO TURN ON THE TINY LED.
: MOVB #<6-1>!0,@PPCW ;: CLEAR C(6) JUST FOR THE HELL OF IT.
: MOV #10,R0
: SOB R0 ;: DELAY ABOUT 50 USEC...
: MOVB @PPA,R3 ;: ...AND DUMMY READ TO DROP 'INRDY'.
: MOVB @PPC,R3 ;: NOW, GET STATUS.
: CMPB R3,#OUTRDY ;: SHOULD HAVE THIS STATUS SET.
: BEQ 1\$
: ERROR, T144 ;: *** PORT C MODE 1 STATUS INCORRECT ***
1\$: CLR R1 ;: SEND A BINARY COUNT FROM R1...
: MOV #-1,R2 ;: ...TO R2 THRU THE 8255.
2\$: CLR R0 ;: LOOP DELAY.
: MOVB R1,@PPB ;: XMIT DATA OUTPUT (THRU B).
3\$: MCVB @PPC,R3 ;: GET STATUS.
: BIT #OUTRDY,R3 ;: OUTRDY SHOULD BE GONE.
: BEQ 4\$
: ERROR, 2\$;: *** OUTRDY STILL SET AFTER OUTPUT XFER ***
4\$: MOVB @PPC,R3 ;: GET STATUS AGAIN...
: CMPB R3,#INRDY!OUTRDY ;: ...AND WAIT FOR THIS.
: BEQ 5\$;: WONDERFUL, MOVE ON.
: SOB R0,4\$
: ERROR, 2\$;: *** STATUS INCORRECT AFTER OUTPUT XFER ***
5\$: MOVB @PPA,R2 ;: DATA INPUT (THRU A).
: MOVB @PPC,R3 ;: GET STATUS
: CMPB R3,#OUTRDY ;: EXPECT THIS STATUS.
: BEQ 6\$;: BR IF SO.
: ERROR, 2\$;: *** STATUS INCORRECT AFTER INPUT XFER ***
6\$: CMPB R2,R1 ;: DATA LOOP INTACT ??
: BEQ 7\$;: BR IF SO.
: ERROR, 2\$;: *** DATA INCORRECT AFTER OUT/IN EXCHANGE ***

CNKXAA.P11 17-MAR-82 14:02

PC 34076 = DATA INCORRECT AFTER OUT/IN EXCHANGE.

```

3876 034102 105201          7$:   INCB   R1
3877 034104 001337          BNE    2$           ; LOOP 'TIL DONE.
3878
3879
(2)
(2)
(2) 034106 104420 000145
3880 034112 016704 144754
3881 034116 106464 000002
3882 034122 012714 034140
3883 034126 112777 000005 144732
3884 034134 000240
3885 034136 000403
3886
3887 034140 022626          1$:   CMP     (SP)+,(SP)+
3888 034142 104423 034106    ERROR,  T145
3889 034146 117703 144712    2$:   MOVB   @PPC,R3
3890 034152 120327 000007    CMPB   R3,#OUTENA!OUTRDY!OUTINT
3891 034156 001402
3892 034160 104423 034106    BEQ     3$
3893
3894 034164 012714 034204    3$:   MOV     #4$,(R4)
3895 034170 106427 000200    MTPS   #PRI4
3896 034174 000240
3897 034176 104423 034106    240
3898 034201 000401          ERROR,  T145
3899
3900 034204 022626          4$:   CMP     (SP)+,(SP)+
3901 034206 012714 034222    MOV     #5$,(R4)
3902 034212 106427 000200    MTPS   #PRI4
3903 034216 000240
3904 034220 000403
3905 034222 022626          240
3906 034224 104423 034106    BR      6$
3907
3908 034230 112777 000004 144630 6$:   MOVB   #<2,1>!0,@PPCW
3909 034236 112777 000001 144616    MOVB   #1,@PPB
3910 034244 005000
3911 034246 117703 144612    CLR     R0
3912 034252 032703 000001    7$:   MOVB   @PPC,R3
3913 034256 001403
3914 034260 077006
3915 034262 104423 034106    BIT     #OUTINT,R3
3916
3917 034266 132777 000040 144570 8$:   BEQ     8$
3918 034274 001001          SOB     R0,7$
3919 034276 077005
3920 034300 117700 144554    9$:   MOVB   @PPA,R0
3921 034304 012714 033644    MOV     #BADBI,(R4)
3922
3923
(2)
(2)
(2) 034310 104420 000146
3924 034314 016704 144550
3925 034320 106464 000002

```

```

;*****
;SBTTL      T145 -- TEST THE PORT B (OUTPUT) INTERRUPT
;*****
T145:  BEGINTEST, 145          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV     PPBV,R4          ;; GET B VECTOR POINTER.
      MTPS    2(R4)           ;; RAISE CPU TO PIO LEVEL...
      MOV     #1$,(R4)        ;; ...AND SET VECTOR.
      MOVB    #<2,1>!1,@PPCW  ;; SET C<2> B INTR ENABLE.
      240
      BR      2$              ;; OUTRDY IS ALREADY THERE, SO...
                               ;; ...INTERRUPT REQUEST SHOULD BE THERE...
                               ;; ...BUT HELD OFF BY THE CPU LEVEL.
      1$:   CMP     (SP)+,(SP)+  ;; BUT IT WASN'T !!
      ERROR,  T145            ;; *** PORT B INTERRUPT LEVEL INCORRECT ***
      2$:   MOVB    @PPC,R3
      CMPB    R3,#OUTENA!OUTRDY!OUTINT
      BEQ     3$
      ERROR,  T145            ;; *** STATUS INCORRECT WITH PORT B INTERRUPT PENDING **
      3$:   MOV     #4$,(R4)
      MTPS    #PRI4
      240
      ERROR,  T145            ;; OK, CHANGE VECTOR.
                               ;; LOWER CPU TO LEVEL 4...
                               ;; ...INTERRUPT SHOULD COME IN.
                               ;; *** PORT B INTERRUPT NOT RECEIVED ***
      4$:   CMP     (SP)+,(SP)+  ;; WE'RE HERE ON INTERRUPT, FIX STACK.
      MOV     #5$,(R4)
      MTPS    #PRI4
      240
      BR      6$              ;; CHANGE THE VECTOR...
                               ;; ...AND LOWER CPU AGAIN...
                               ;; ...SHOULDN'T GET ANOTHER INTERRUPT.
      5$:   CMP     (SP)+,(SP)+  ;; BUT WE DID -- TROUBLE.
      ERROR,  T145            ;; *** DOUBLE PORT B INTERRUPT ***
      6$:   MOVB    #<2,1>!0,@PPCW  ;; OK, CLEAR THE ENABLE...
      MOVB    #1,@PPB         ;; ...WRITE -- SHOULD LOWER INT REQUEST.
      CLR     R0
      7$:   MOVB    @PPC,R3
      BIT     #OUTINT,R3
      BEQ     8$
      SOB     R0,7$
      ERROR,  T145            ;; BR WHEN REQUEST CLEARS.
                               ;; *** PORT B INT REQUEST DIDN'T CLEAR AFTER WRITE ***
      8$:   BITB    #INRDY,@PPC
      BNE     9$
      SOB     R0,8$
      9$:   MOVB    @PPA,R0
      MOV     #BADBI,(R4)
      ;; BR WHEN 'INRDY' COMES UP...
      ;; ...BUT DON'T WAIT FOREVER.
      ;; DUMMY READ.
      ;; RESET VECTOR.

```

```

;*****
;SBTTL      T146 -- TEST THE PORT A (INPUT) INTERRUPT
;*****
T146:  BEGINTEST, 146          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV     PPAV,R4          ;; GET PORT A VECTOR POINTER.
      MTPS    2(R4)           ;; RAISE CPU TO PIO LEVEL.

```

```
3926 034324 112777 000011 144534      MOVB    #<4 1.>!1,@PPCW ; SET C<4> A INTR ENABLE.
3927 034332 117703 144526      MOVB    @PPC,R3
3928 034336 120327 000022      CMPB    R3,#INENA!OUTRDY ; EXPECT TO SEE THIS.
3929 034342 001402              BEQ      1$
3930 034344 104423 034310      ERROR,   T146          ;; *** STATUS INCORRECT ***
3931
3932 034350 012714 034372      1$:      MOV     #2$(R4)      ; SET THE VECTOR.
3933 034354 005000              CLR      R0
3934 034356 012701 000123      MOV     #123,R1
3935 034362 110177 144474      MOVB    R1,@PPB      ; SEND DATA (B => A) THRU THE LOOP.
3936 034366 077001              SOB     R0,,          ; INTERRUPT SHOULD BE HELD OFF.
3937 034370 000403              BR      3$
3938 034372 022626              2$:      CMP     (SP)+,(SP)+      ; BUT IT WASN'T.
3939 034374 104423 034310      ERROR,   T146          ;; *** PORT A INTERRUPT LEVEL INCORRECT ***
3940
3941 034400 117703 144460      3$:      MOVB    @PPC,R3      ; GET STATUS.
3942 034404 120327 000072      CMPB    R3,#INRDY!INENA!ININT!OUTRDY
3943 034410 001402              BEQ      4$      ; BR IF STATUS IS OK.
3944 034412 104423 034310      ERROR,   T146          ;; *** STATUS INCORRECT WITH PORT A INTERRUPT PENDING **
3945
3946 034416 012714 034436      4$:      MOV     #5$(R4)      ; CHANGE VECTOR.
3947 034422 106427 000200      MTPS    #PRI4      ; LOWER CPU TO LEVEL 4...
3948 034426 000240              240      ;...INTERRUPT SHOULD COME IN.
3949 034430 104423 034310      ERROR,   T146          ;; *** PORT A INTERRUPT NOT RECEIVED ***
3950 034434 000401              SKP1
3951
3952 034436 022626              5$:      CMP     (SP)+,(SP)+      ; WE'RE HERE ON INTERRUPT, FIX STACK.
3953 034440 012714 034454      MOV     #6$(R4)      ; CHANGE VECTOR...
3954 034444 106427 000200      MTPS    #PRI4      ;...AND LOWER AGAIN...
3955 034450 000240              240      ;...SHOULDN'T GET ANOTHER.
3956 034452 000403              BR      7$
3957 034454 022626              6$:      CMP     (SP)+,(SP)+      ; 2ND INTERRUPT, FIX STACK.
3958 034456 104423 034310      ERROR,   T146          ;; *** DOUBLE PORT A INTERRUPT ***
3959
3960 034462 112777 000010 144376      7$:      MOVB    #<4 1.>!0,@PPCW ; OK, CLEAR THE ENABLE...
3961 034470 117702 144364      MOVB    @PPA,R2      ;...READ DATA (SHOULD LOWER FLAGS).
3962 034474 005000              CLR      R0
3963 034476 117703 144362      8$:      MOVB    @PPC,R3      ; READ STATUS.
3964 034502 032703 000010      BIT     #ININT,R3
3965 034506 001403              BEQ      9$      ; BR WHEN INT REQUEST CLEARS.
3966 034510 077006              SOB     R0,8$
3967 034512 104423 034310      ERROR,   T146          ;; *** PORT A INT REQUEST DIDN'T CLEAR AFTER READ ***
3968
3969 034516 012714 033654      9$:      MOV     #BADA1,(R4)      ; RESET PORT A VECTOR.
3970
3971
(2)
(2)
(2)
3972 034522 104420 000147      ;*****
3973 034526 016704 144340      .SBTTL    T147 -- TEST THAT B PRIORITY IS HIGHER THAN A
3974 034532 106464 000002      ;*****
3975 034536 012714 034632      T147:    BEGINTEST, 147      ; CHECK SWITCH REGISTER OPTIONS.
3976 034542 012764 034622 000004      MOV     PPBV,R4      ; GET B VECTOR POINTER.
3977 034550 112777 000005 144310      MTPS    2(R4)      ; RAISE CPU TO DEVICE LEVEL.
3978 034556 112777 000011 144302      MOV     #3$(R4)      ; SET B VECTOR.
3979 034564 112777 000001 144270      MOV     #2$,4(R4)      ; SET A VECTOR.
3980 034564 112777 000001 144270      MOVB    #<2 1.>!1,@PPCW ; ENABLE B...
3981 034564 112777 000001 144270      MOVB    #<4 1.>!1,@PPCW ;...AND A INTERRUPTS.
3982 034564 112777 000001 144270      MOVB    #1,@PPB      ; OUTPUT DATA...
```

```

3979 034572 005000          CLR      R0
3980 034574 132777 000040 144262 1$:  BITB   #INRDY,@PPC      ;...AND WAIT 'TIL DONE.
3981 034602 001001          BNE     .+4
3982 034604 077005          SOB     R0,1$      ; KEEP-ALIVE.
3983
3984 034606 106427 000200          MTPS   #PRI4      ; LOWER CPU, B SHOULD COME IN FIRST.
3985 034612 000240          240
3986 034614 104423 034522          ERROR, T147      ;; *** NEITHER ONE INTERRUPTED -- WHAT THE HELL !! ***
3987 034620 000415          BR       4$
3988 034622 022626          2$:  CMP     (SP)+,(SP)+      ; WE'RE HERE IF A CAME IN FIRST.
3989 034624 104423 034522          ERROR, T147      ;; *** A INTERRUPTED BEFORE B -- BAD NEWS !! ***
3990 034630 000411          BR       4$
3991
3992 034632 022626          3$:  CMP     (SP)+,(SP)+      ; WE'RE HERE IF B INTERRUPTED FIRST.
3993 034634 012764 034654 000004      MOV     #4$,4(R4)      ; CHANGE THE A VECTOR...
3994 034642 106427 000200          MTPS   #PRI4      ;...AND LET IT IN NOW.
3995 034646 000240          240
3996 034650 104423 034522          ERROR, T147      ;; *** A INT NOT RECEIVED AFTER B INT ***
3997
3998 034654 112777 000004 144204 4$:  MOVB   #<2_1.>!0,@PPCW ; CLEAR THE ENABLES.
3999 034662 112777 000010 144176      MOVB   #<4_1.>!0,@PPCW
4000 034670 112777 000001 144164      MOVB   #1,@PPB      ; LOWER REQUESTS.
4001 034676 117700 144156      MOVB   @PPA,R0
4002 034702 012714 033644          MOV     #BADBI,(R4)      ; RESET VECTORS.
4003 034706 012764 033654 000004      MOV     #BADAI,4(R4)
4004
4005 034714 012767 000006 143062 P10DUN: MOV    #ERRVEC+2,ERRVEC
4006 034722 004767 000002          CALL   ENDSEG      ; CHECK FOR LOOP-IN-SEGMENT...
4007 034726 000410          BR       ENDPAS      ;...RETURN AND DO ENDPASS IF NOT.
4008          .SBTTL

```

```

4010 .SBTTL END OF PASS ROUTINES.
4011 *****
4012 : AT LAST -- WE'VE HIT THE END OF THE TRAIL, PODNAH !!!
4013 : REPORT END-PASS, TAKE A LITTLE NAP, AND THEN DO IT ALL AGAIN.
4014
4015 : SOB RC,, EXECUTES IN 3.66 USEC IN KXT11.
4016 : IF Q-BUS TRANSACTION REQUIRED, ADD 609 NSEC (CYCLE SLIP).
4017 : THEREFORE, SOB FOR TIMING TAKES ABOUT 4.27 USEC/COUNT.
4018
4019 034730 005767 000730 ENDSLG: TST INSEG : LOOP-IN-SEGMENT ??
4020 034734 001001 BNE 1$ : YES.
4021 034736 000207 RETURN : JUST RETURN IF NOT.
4022
4023 1$: EMT+7 : <CRLF>
4024 034742 016700 001232 MOV SEGN,RO : CURRENT SEGMENT DONE.
4025 034746 000402 SKP2
4026 034750 012700 035130 ENDPAS: MOV #EOP,RO : ALL SEGMENTS DONE.
4027 034754 104003 EMT+3
4028 034756 012700 035141 MOV #EOP1,RO
4029 034762 104003 EMT+3 : END PASS.
4030 034764 005267 147012 INC TPASS : BUMP TOTAL PASS COUNT.
4031 034770 026727 147004 CMP ERRCNT,(PC)+ : ANY ERRORS THIS PASS ??
4032 034774 000000 1$: 0
4033 034776 001002 BNE 2$ : SKIP IF SO.
4034 035000 005267 147000 INC GPASS : NO, BUMP CONSECUTIVE-GOOD PASS COUNT.
4035 035004 016767 146770 177762 2$: MOV ERRCNT,1$ : UPDATE SAVED COUNT.
4036 035012 016700 146764 MOV TPASS,RO
4037 035016 104424 TRAP+24 : PASS COUNT...
4038 035020 012700 035155 MOV #EOPE,RO
4039 035024 104003 EMT+3
4040 035026 016700 146746 MOV ERRCNT,RO
4041 035032 104424 TRAP+24 : ...AND TOTAL ERROR COUNT.
4042 035034 104007 EMT+7 : <CRLF>
4043 035036 052767 010000 143132 BIT #10000,SWR : ERROR SUMMARY ??
4044 035044 001402 BEQ 3$
4045 035046 004767 000124 CALL ERKSUM : PRINT IT IF SO.
4046 035052 106427 000000 3$: MTPS #PRIO : LOWER THE CPU.
4047 035056 005000 CLR RO
4048 035060 012701 000004 MOV #4,R1
4049 035064 077001 4$: SOB RO,, : ENDPASS DELAY...
4050 035066 077102 SOB R1,4$ : ...ABOUT 1 SECOND.
4051
4052 035070 013700 000042 MOV #42,RO
4053 035074 001405 BEQ 5$ : BR IF NOT CHAINING.
4054 035076 000240 NOP : OR RESET.
4055 035100 004710 CALL (RO) : SYSMAC STYLE CHAIN RETURN.
4056 035102 000240 000240 240,240,240
4057
4058 035110 016700 000550 5$: MOV INSEG,RO : GET ITERATION ADDRESS.
4059 035114 001002 BNE 6$
4060 035116 012700 010336 MOV #DOALL,RO : IF NONE, BACK TO SQUARE 1.
4061 035122 012706 003776 6$: MOV #STACK,SP : RESET STACK...
4062 035126 000110 JMP (RO) : ...AND LOOP FOREVER.
4063
4064 035130 005015 045516 040530 EOP: .ASCIZ <15><12>'NKXAA0'
4065 035141 054 042440 042116 EOP1: .ASCIZ ', END PASS '

```

```

4066 035155 054 052040 052117 EOPE: .ASCIZ ', TOTAL ERRORS '
4067 035176 .EVEN
4068
4069 ;*****
4070 ; ERROR SUMMARY REPORT.
4071 ; CALLED IN 'END-PASS' IF SWR<12> IS SET.
4072 ;
4073 ERRSUM: MOV #EPCTBL,R2
4074 TST (R2)
4075 BEQ 2$ ; BR IF NO ERROR DATA LOGGED.
4076 MOV #ERHEAD,R0
4077 EMT+3 ; HEADER.
4078 1$:
4082 EMT+10 ;: <TAB>
4083 (1) MOV (R2)+,R0
4084 (1) TRAP+24 ;: PRINT (R2)+
4085 (1) EMT+10 ;: <TAB>
4086 (1) MOV (R2)+,R0
4087 (1) TRAP+24 ;: PRINT (R2)+
4088 035230 104007 EMT+7 ;: <CRLF>
4089 035232 005712 TST (R2)
4090 035234 00367 BNE 1$ ; LOOP 'TIL DONE.
4091 035236 000207 2$: RETURN
4092
4093 ERHEAD: .ASCIZ <15><12><11>'ERRPC'<11>' TIMES'<15><12>
4094 .EVEN
4095 ;*****
4096 ; SUBROUTINE TO CHECK FOR Q-BUS EXERCISOR.
4097 ; RETURN QBX CSR1 ADDRESS IN R0 IF IT'S THERE.
4098 ;
4099 QBXIN: TST QBXA ; QBX INSTALLED ??
4100 BNE 3$ ; BR IF SO.
4101 MOV #1$,ERRVEC ; DON'T KNOW YET, SET TRAP CATCHER...
4102 MOV #QBXCsr,R0 ; ...AND A POINTER.
4103 CLR (R0) ; CLEAR QBX CSR1 (IF POSSIBLE)...
4104 240 ; ...TRAP TO 1$ IF NOT.
4105 MOV R0,QBXA ; IT'S THERE, SET FLAG = CSR ADDRESS...
4106 BR 2$ ; ...AND RETURN.
4107 1$: CLR QBXA ; NOT THERE, CLEAR FLAG.
4108 CMP (SP)+,(SP)+ ; FIX STACK.
4109 MOV #ERRVEC+2,ERRVEC ; RESET ERROR VECTOR.
4110 MOV QBXA,R0 ; RETURN CSR POINTER OR ZERO.
4111 RETURN

```

```

4110 .SBTTL TRAP HANDLER
4111 *****
4112 : TRAP HANDLER
4113 :
4114 TRAPHAN: MOV R0,-(SP) ; SAVE CALLERS R0...
4115 MOV R1,-(SP) ; ...AND R1.
4116 MOV 4(SP),R0 ; GET TRAP PC.
4117 MOVB -2(R0),R0 ; GET TRAP CODE.
4118 MOV R0,R1 ; WORKING COPY.
4119 SUB #17,R1
4120 BPL 1$ ; BR IF TRAP+20 OR HIGHER.
4121 CLR R1 ; TRAP+0 THRU TRAP+17 ARE COMMON.
4122 1$: ASL R1 ; SHIFT UP TO WORD OFFSET...
4123 MOV 2$(R1),PC ; ...AND DISPATCH THERE.
4124
4125 2$: CCBIT ; TRAP0 - TRAP17 = CC BIT CHECK.
4126 BGNTST ; TRAP20 = BEGIN NEW TEST.
4127 CHKSWR ; TRAP21 = CHECK SWR.
4128 NEWSWR ; TRAP22 = GET NEW SWR.
4129 LOGERR ; TRAP23 = LOG AND REPORT ERROR.
4130 ITOA ; TRAP24 = CONVERT INTEGER TO ASCII AND PRINT.
4131 :
4132 : ADD MORE AS NEEDED.
4133 :
4134 :
4135 *****
4136 : CONDITION CODE BIT CHECKER.
4137 : CALL: TRAP+N, WHERE N IS THE CONDITION CODE TO EXPECT (0 - 17).
4138 :
4139 : PSW TO EXAMINE IS AT 6(SP), N IS IN R0.
4140 : RETURN TO CALL+2 IF CC N.E. N
4141 : RETURN TO CALL+6 OTHERWISE.
4142 :
4143 CCBIT: MOV 6(SP),R1 ; GET PSW TO CHECK.
4144 BIC #^C17,R1 ; KEEP CC BITS ONLY.
4145 CMP R0,R1
4146 BNE 1$ ; BR IF CC BITS INCORRECT.
4147 ADD #4,4(SP) ; OK, ADJUST RETURN PC.
4148 1$: MOV (SP)+,R1
4149 MOV (SP)+,R0
4150 RTI
4151 :
4152 *****
4153 : BEGIN TEST ROUTINE.
4154 : CALL VIA TRAP+20
4155 :
4156 BGNTST: CHKSWR
4157 MOV (SP)+,R1 ; RESTORE CALLERS REGISTERS.
4158 MOV (SP)+,R0
4159 MOV 8(SP),TSTNUM ; SAVE TEST NUMBER...
4160 ; ...FOR THE ERROR HANDLER...
4161 ADD #2,(SP) ; ...BUMP RETURN PC...
4162 RTI ; ...AND RETURN.

```

```

4164 ;*****
4165 ; SWITCH REGISTER HANDLER.
4166 ; CALLED VIA TRAP+21 (CHECKSWR) AND TRAP+22 (GETSWR).
4167 ;
4168 ;
4169 035460 104005 CHKSWR: .ENABL LSB ; CHECK KEYBOARD ( => R0 ).
4170 035462 120027 000007 EMT+5 ; <^G> ??
4171 035466 001024 CMPB RO,#7 ; JUST RETURN IF NOT.
4172 BNE 1$
4173 035470 005737 000042 NEWSWR: TST @#42 ; CHAINING ??
4174 035474 001021 BNE 1$ ; DON'T IF SO.
4175 035476 012700 035546 MOV #DSWR,R0
4176 035502 104003 EMT+3 ; OLD SWR...
4177 035504 016700 142466 MOV SWR,R0
4178 035510 104424 TRAP+24 ;...= THIS.
4179 035512 012700 035557 MOV #NSWR,R0
4180 035516 104003 EMT+3 ; NEW SWR...
4181 035520 104000 EMT+0 ;...AND WAIT FOR INPUT.
4182 035522 104011 EMT+11 ; PARSE OCTAL NUMBER.
4183 035524 000405 BR 1$ ; JUST EXIT IF NO CHANGE.
4184 035526 120067 142444 CMPB RO,SWR ; NEW SEGMENT SELECTED ??
4185 035532 001016 BNE NEWSW ; BR IF SO.
4186 035534 010067 142436 MOV RO,SWR ; OTHERWISE, SAVE NEW SWR...
4187 035540 012601 1$: MOV (SP)+,R1 ;...RESTORE...
4188 035542 012600 MOV (SP)+,R0
4189 035544 000002 RTI ;...AND RETURN TO CALLER.
4190 ;
4191 035546 005015 053523 020122 DSWR: .DSABL LSB
4192 035557 040 047040 053505 NSWR: .ASCIZ <15><12>'SWR = '
4193 .ASCIZ ' NEW = '
4194 .EVEN
4195 ;
4196 ; NEW SEGMENT REQUIRED. GET ADDRESS AND DO A RESTART-IN-SEGMENT.
4197 035570 010067 142402 NEWSWEG: MOV RO,SWR ; SAVE NEW SWR.
4198 035574 042700 177770 BIC #^C7,R0 ; STRIP OFF SEGMENT NUMBER.
4199 035600 006300 ASL RO ; SHIFT NUMBER UP TO WORD OFFSET...
4200 035602 016067 035644 000054 MOV 3$(R0),INSEG ;...AND SET/CLEAR 'INSEG' FLAG.
4201 035610 012700 004000 MOV #ERRCNT,R0
4202 035614 012701 002000 MOV #1024,,R1
4203 035620 005020 1$: CLR (R0)+ ; CLEAR THE ERROR TABLE AND COUNTERS.
4204 035622 077102 SOB R1,1$
4205 035624 016700 000034 MOV INSEG,R0 ; GET LOOP-SEGMENT ADDRESS.
4206 035630 001002 BNE 2$
4207 035632 012700 010336 MOV #DOALL,R0 ; IF ZERO, RESTORE 'RUN-ALL' MODE.
4208 035636 012706 003776 2$: MOV #STACK,SP ; NOW, RESET STACK...
4209 035642 000110 JMP (R0) ;...AND RESTART-IN-SEGMENT.
4210 035644 000000 3$: 0 ; SEGMENT = 0 = RUN 'EM ALL...
4211 035646 010336 024030 027052 S1,S2,S3 ;...OTHERWISE, RUN JUST ONE OF THESE.
4212 035654 027642 027764 030014 S4,S5,S6
4213 035662 033512 S7
4214 035664 000000 INSEG: 0 ; ZERO = RUN ALL...
4215 ;...NZ = ADDRESS FOR LOOP-IN-SEGMENT.

```

```

4217 *****
4218 ERROR HANDLER.
4219 CALLED VIA TRAP+23.
4220
4221 ACCUMULATE ERROR DATA (PC AND NUMBER OF OCCURRENCES) IN THE ERROR LOG.
4222 PRINT THE STATE OF THE MACHINE AT TIME OF ERROR (UNLESS INHIBITED).
4223 TEST THE LOOP (SWR<9>) AND HALT (SWR<15>) SWITCHES
4224 AND EXIT ACCORDINGLY.
4225
4226 LOGERR: MOV #ERRCNT,R0 ; GET TABLE POINTER...
4227 MOV 4(SP),R1 ; ...AND CURRENT (ERROR) PC.
4228 INC (R0)+ ; BUMP TOTAL ERROR COUNT.
4229 CMPB (R0)+,(R0)+ ; SKIP OVER TOTAL PASS COUNT...
4230 CLR (R0)+ ; ...AND CLEAR "CONSECUTIVE-GOOD" COUNT.
4231 1$: CMP R1,(R0) ; THIS PC ALREADY LOGGED ??
4232 BEQ 3$ ; JUST BUMP ITS COUNT IF SO.
4233 TST (R0) ; NO, END OF TABLE ??
4234 BEQ 2$ ; STORE PC HERE IF SO.
4235 CMP R0,#LOGEND-4 ; TABLE FULL ??
4236 BHIS 4$ ; YES, FORGET IT !!!
4237 CMP (R0)+,(R0)+ ; NONE OF THE ABOVE, BUMP POINTER...
4238 BR 1$ ; ...AND LOOP 'TIL WE FIND ONE...
4239 ; ...OR THE OTHER.
4240 2$: MOV R1,(R0) ; STORE ERROR PC...
4241 3$: INC 2(R0) ; ...AND BUMP ITS COUNTER...
4242 4$: ; ...AND FALL THRU.
4243
4244 PNTERR: BIT #20000,SWR ; ERROR MESSAGES INHIBITED ??
4245 BNF 1$ ; BR IF SO.
4246 MOV SP,ESP ; GET STACK POINTER...
4247 ADD #10,ESP ; ...ADJUST TO SP AT TIME OF ERROR.
4248 EMT+7 ; <CRIF>
4249 MOV SEGN,R0 ; CURRENT SEGMENT...
4250 EMT+3 ; ...AND TEST...
4251 MOV #FINTST,R0 ; ...NUMBER.
4252 EMT+3 ; ...HEADER.
4253 MOV TSTNUM,R0 ; ...PRINT 4(SP)
4254 TRAP+24 ; ; <TAB>
4255 MOV #HDR,R0 ; ...PRINT 6(SP)
4256 EMT+3 ; ; <TAB>
4261 MOV 4(SP),R0 ; ...PRINT 2(SP)
(1) TRAP+24 ; ; <TAB>
(1) EMT+10 ; ...PRINT (SP)
(1) MOV 6(SP),R0 ; ; <TAB>
(1) TRAP+24 ; ...PRINT R2
(1) EMT+10 ; ; <TAB>
(1) MOV 2(SP),R0 ; ...PRINT R3
(1) TRAP+24 ;
(1) EMT+10 ;
(1) MOV (SP),R0 ;
(1) TRAP+24 ;
(1) EMT+10 ;
(1) MOV R2,R0 ;
(1) TRAP+24 ;
(1) EMT+10 ;
(1) MOV R3,R0 ;

```

(1)	036056	104424			TRAP+24	:: PRINT R3
(1)	036060	104010			EMT+10	:: <TAB>
(1)	036062	010400			MOV R4,R0	
(1)	036064	104424			TRAP+24	:: PRINT R4
(1)	036066	104010			EMT+10	:: <TAB>
(1)	036070	010500			MOV R5,R0	
(1)	036072	104424			TRAP+24	:: PRINT R5
(1)	036074	104010			EMT+10	:: <TAB>
(1)	036076	016700	000072		MOV ESP,R0	
(1)	036102	104424			TRAP+24	:: PRINT ESP
(1)	036104	104010			EMT+10	:: <TAB>
4262	036106	104007			EMT+7	:: <CRLF>
4263	036110	012601		1\$:	MOV (SP)+,R1	:: RESTORE REGISTERS.
4264	036112	012600			MOV (SP)+,R0	
4265	036114	005767	142056		TST SWR	:: HALT-ON-ERROR (SWR<15>) ??
4266	036120	100012			BPL 2\$	:: BR IF NOT.
4267	036122	013767	177564	000042	MOV @#177564,SAVTY	:: YES, SAVE CONSOLE TTY STATUS...
4268	036130	042737	000004	177564	BIC #4,@#177564	:: ...AND INSURE 'MAINT' IS NOT SET.
4269	036136	000000			EHALT	:: ERROR HALT.
4270	036140	016737	000026	177564	MOV SAVTY,@#177564	:: ON PROCEED, RESTORE TTY STATUS.
4271	036146	032767	001000	142022	2\$: BIT #1000,SWR	:: LOOP-ON-ERROR (SWR<9>) ??
4272	036154	001003			BNE 3\$	:: BR IF SO.
4273	036156	062716	000002		ADD #2,(SP)	:: NO, ADJUST RETURN PC...
4274	036162	000002			RTI	:: ...AND RETURN TO CALLER.
4275	036164	017616	000000	3\$:	MOV @ (SP), (SP)	:: REPLACE RETURN PC WITH LOOP ADDRESS...
4276	036170	000002			RTI	:: ...AND DO IT.
4277						
4278	036172	000000			SAVTY: 0	
4279	036174	000000			ESP: 0	
4280	036176	000000			TSTNUM: 0	
4281	036200	036202			SEGN: SEG1	: POINTS TO ONE OF THE FOLLOWING:
4282	036202	030524	020061	047111	SEG1: .ASCIZ	'T11 INSTRUCTION'
4283	036222	030524	020061	051124	SEG2: .ASCIZ	'T11 TRAP/INTERRUPT'
4284	036245	114	041517	046101	SEG3: .ASCIZ	'LOCAL 2KW RAM'
4285	036263	114	041517	046101	SEG4: .ASCIZ	'LOCAL 1KW ROM'
4286	036301	123	051105	040511	SEG5: .ASCIZ	'SERIAL LINE 1'
4287	036317	123	051105	040511	SEG6: .ASCIZ	'SERIAL LINE 2'
4288	036335	120	051101	046101	SEG7: .ASCIZ	'PARALLEL I/O PORT'
4289	036357	040	040506	046111	FINTST: .ASCIZ	' FAILS IN TEST '
4290	036377	015	020012	041520	HDR: .ASCIZ	<15><12>' PC+2'<11>
4291	036407	040	050040	053523	.ASCIZ	' PSW'<11>
4292	036415	040	051040	004460	.ASCIZ	' R0'<11>
4293	036422	020040	030522	011	.ASCIZ	' R1'<11>
4294	036427	040	051040	004462	.ASCIZ	' R2'<11>
4295	036434	020040	031522	011	.ASCIZ	' R3'<11>
4296	036441	040	051040	004464	.ASCIZ	' R4'<11>
4297	036446	020040	032522	011	.ASCIZ	' R5'<11>
4298	036453	122	024066	050123	.ASCIZ	'R6(SP)'<15><12>
4299					.EVEN	

```

4301 ;*****
4302 ; CONVERT INTEGER TO ASCII AND PRINT.
4303 ; CALL VIA TRAP+24, NUMBER TO CONVERT IS 2ND WORD ON THE STACK.
4304 ;
4305 ; CODE ADDED BECAUSE EMT+30 (INTEGER TO ASCII) FUNCTIONALITY IS
4306 ; DIFFERENT BETWEEN XXDP+ V1.0 AND V1.1. THIS ROUTINE MAKES THAT
4307 ; DIFFERENCE TRANSPARENT TO THE CALLER.
4308 ;
4309 036464 016600 000002 ITOA: MOV 2(SP),R0 ; NUMBER => R0.
4310 036470 012701 036522 MOV #2$,R1 ; IF V1.1, PACK THE ASCII STRING AT 2$.
4311 036474 005011 CLR (R1)
4312 036476 104030 EMT+30 ; CONVERT I TO A.
4313 036500 005767 000016 TST 2$
4314 036504 001403 BEQ 1$ ; IF V1.0, IT'S ALREADY BEEN PRINTED.
4315 036506 012700 036522 MOV #2$,R0 ; IF V1.1, PRINT IT NOW.
4316 036512 104003 EMT+3
4317 036514 012601 1$: MOV (SP)+,R1 ; RESTORE...
4318 036516 012600 MOV (SP)+,R0
4319 036520 000002 RTI ;...AND RETURN.
4320
4321 036522 031061 032063 033065 2$: .ASCIZ /123456/
4322 036532 .EVEN
4323
4324 036532 036532 $LSTAD: .
4325
4326 010000 .END BEGIN
  
```

	1094	1104	1113	1123	1130	1136	1146	1153	1160	1176	1189	1196	1203
	1208	1212	1215	1464	1467	1470	1473	1492	1513	1519	1526	1529	1537
	1547	1555	1559	1566	1571	1578	1581	1585	1588	1602	1609	1619	1625
	1632	1645	1651	1667	1670	1683	1689	1692	1705	1708	1713	1725	1731
	1739	1747	1754	1763	1770	1774	1782	1790	1793	1798	1801	1810	1819
	1822	1825	1860	1866	1874	1882	1885	1898	1910	1915	1918	1921	1928
	1931	1935	1938	1946	1953	1962	1970	1977	1990	1996	2013	2016	2029
	2035	2038	2051	2054	2059	2071	2077	2085	2093	2100	2109	2116	2120
	2128	2137	2144	2155	2164	2174	2181	2187	2200	2213	2221	2228	2233
	2241	2252	2264	2271	2275	2289	2310	2313	2321	2324	2336	2344	2368
	2376												
CHECKS=	104421	93#	4156										
CHKSWR	035460	4127	4169#										
CLNZ =	000254	84#	1085	2135									
CMPBO	011476	540#											
CMPB1	016722	1487#											
CMP0	013104	822#											
CMP1	020516	1805#											
COMBO	012052	622#											
COMB1	017414	1596#											
COMO	013456	906#											
COM1	021362	1942#											
DFPBR =	001040	167#	3169*	3207	3267	3282	3296	3310	3324	3338	3353	3364	3383
		3455	3519	3569	3639	3666	3709	3736	3766				3449
DFSPD =	001042	168#	3451*	3523	3574	3623	3648	3738					
DLDUN	033452	3175	3177	3179	3181	3245	3734	3748	3751	3766#			
DLSET	030042	3139	3149#										
DLTSTS	030236	3170	3178	3180	3190#								
CJALL	010336	299	308#	4060	4207								
EHALI =	000000	101#	228	4269									
EMTTRP	010140	261#											
EMTVEC=	000030	70#	261	262*	289*	293*	2694	2695*	2696*	2703*			
ENDPAS	034750	4007	4026#										
ENDSEG	034730	2382*	2911*	3072*	3111*	3769*	4006*	4019#					
EOP	035130	4026	4064#										
EOPE	035155	4038	4066#										
EOP1	035141	4028	4065#										
EPCTBL=	004006	193#	4073										
ERHEAD	035240	4076	4088#										
ERRCNT=	004000	190#	4031	4035	4040	4201	4226						
ERROR =	104423	95#	314	318	334	345	358	370	384	390	394	411	418
		423	430	432	435	442	445	452	454</				

CROSS REFERENCE TABLE -- USER SYMBOLS

	1315	1326	1337	1344	1353	1368	1388	1395	1413	1429	1436	1451	1456
	1465	1468	1471	1474	1480	1483	1493	1496	1499	1502	1514	1517	1520
	1523	1527	1530	1538	1541	1544	1548	1556	1560	1567	1572	1575	1579
	1582	1586	1589	1592	1603	1606	1610	1614	1620	1626	1629	1633	1636
	1646	1649	1652	1656	1665	1668	1671	1674	1684	1687	1690	1693	1703
	1706	1709	1714	1717	1726	1732	1735	1740	1743	1748	1755	1758	1764
	1767	1771	1775	1778	1783	1791	1794	1799	1802	1811	1814	1817	1820
	1823	1826	1837	1840	1847	1861	1864	1867	1870	1875	1878	1883	1886
	1892	1899	1902	1911	1916	1919	1922	1925	1929	1932	1936	1939	1947
	1950	1954	1957	1963	1971	1974	1978	1981	1991	1994	1997	2002	201
	2014	2017	2020	2030	2033	2036	2039	2049	2052	2055	2060	2063	207
	2078	2081	2086	2089	2094	2101	2104	2110	2113	2117	2121	2124	2129
	2138	2141	2145	2148	2156	2159	2165	2168	2175	2182	2188	2191	2201
	2204	2209	2214	2217	2222	2229	2234	2238	2242	2245	2253	2256	2260
	2265	2268	2272	2276	2279	2286	2290	2293	2296	2301	2311	2314	2318
	2322	2325	2328	2337	2340	2345	2348	2355	2359	2363	2369	2372	2377
	2380	2396	2401	2406	2412	2415	2421	2424	2435	2444	2453	2462	2471
	2480	2483	2486	2489	2492	2495	2498	2501	2515	2521	2525	2546	2551
	2554	2559	2562	2575	2580	2583	2586	2589	2600	2605	2608	2611	2614
	2625	2630	2633	2636	2639	2666	2673	2676	2679	2682	2706	2709	2712
	2715	2736	2739	2742	2745	2760	2765	2768	2771	2781	2786	2789	2792
	2807	2812	2827	2837	2839	2851	2854	2856	2877	2882	2902	2907	2940
	2955	2959	2987	2999	3009	3017	3044	3047	3059	3062	3069	3102	3108
	3174	3176	3178	3180	3211	3214	3217	3220	3230	3233	3236	3239	3242
	3257	3271	3274	3285	3288	3299	3302	3313	3316	3327	3330	3341	3344
	3357	3360	3370	3375	3379	3394	3400	3413	3472	3489	3505	3508	3533
	3541	3544	3549	3554	3559	3582	3591	3599	3602	3610	3615	3618	3632
	3635	3654	3662	3679	3685	3698	3728	3750	3762	3820	3822	3824	3840
	3853	3862	3867	3872	3875	3888	3892	3897	3906	3915	3930	3939	3944
	3949	3958	3967	3986	3989	3996							
ERRSUM	035176	4045*	4073#										
ERRVEC=	000004	64#	2510*	2527*	2538*	2539*	2566*	2834*	2840*	2927*	2973*	3029*	3071*
		3110*	3152*	3768*	3800*	4005*	4097*	4106*					3090
ESP	036174	4246*	424										

[illegible]

CROSS REFERENCE TABLE -- USER SYMBOLS

PIOP =	000240	132#	3799															
PIOTST	033664	3816	3822	3824	3826#													
PNTERR	035734	4244#																
PPA =	001060	177#	3802*	3837	3849	3868	3920	3961	4001									
PPAV =	001070	181#	3813*	3924														
PPB =	001062	178#	3804*	3858*	3909*	3935*	3978*	4000*										
PPBV =	001072	182#	3810*	3880	3972													
PPC =	001064	179#	3806*	3850	3859	3863	3869	3889	3911	3917	3927	3941	3963	3980				
PPCW =	001066	180#	3808*	3844*	3845*	3846*	3883*	3908*	3926*	3960*	3976*	3977*	3998*	3999*				
PPMO =	000253	3781#																
PPM1 =	000264	3782#	3844															
PRGSIZ=	000076	19#																
PRI0 =	000000	79#	250*	2511*	2829*	2880*	2909*	3221*	3268*	3354*	3397*	3408*	3509*	3636*				
		3682*	3693*	4046*														
PRI1 =	000040	78#																
PRI2 =	000100	77#																
PRI3 =	000140	76#																
PRI4 =	000200	75#	123	2870	2871*	3895*	3902*	3947*	3954*	3984*	3994*							
PRI5 =	000240	74#	126	132	2905*													
PRI6 =	000300	73#	2540*	2560	2572*	2587	2597*	2612	2622*	2637	2656*	2680	2697*	2713				
		2727*	2743	2896*	3196*	3348*	3498*	3589*	3595*									
		72#	247	3262*	3568*	3573												
PRI7 =	000340	69#	2654*	2655*	2684*													
PWRVEC=	000024	157#	4095	4101*	4104*	4107												
QBXA =	001036	136#	4098															
QBXCSR=	174020	2645*	2861*	2890*	3022*	4095#												
QBXIN =	035262	137#	2867	2884														
QBXVEC=	000110	111#	2918															
RAMADR=	160010	112#	2919															
RAMSIZ=	003764	171#	3155*	3204	3225	3529	3555	3571	3606	3625	3629	3641	3650	3659				
RBUF =	001046	3701	3718	3725	3737	3753	3759	3767										
		170#	3153*	3199*	3203	3208*	3224	3499*	3500	3501*	3502	3530	3537	3545				
		3550	3556	3580	3583	3627	3657	3672	3674*	3678*	3684*	3697*	3700*	3716				
		3723	3733	3743	3747	3757	3771											
		3125#	3531	3538	3546													

CROSS REFERENCE TABLE

[illegible]

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
 CNKXAA.P11 17-MAR-82 14:02

M 9

CROSS REFERENCE TABLE -- USER SYMBOLS

MACY11 306(1063) 17-MAR-82 14:03 PAGE 9-7
 SEQ 0116

TPSW	015126	1200#									
TPSW1	023454	2305#									
TRAPHA	035336	294	4114#								
TRAP4 =	167776	116#	248*								
TRPVEC=	000034	71#	263*	290*	294*	2724	2725*	2726*	2733*		
TRTVEC=	000014	67#	2753*	2754*	2775*	2797*	2798*	2818*	2819*	2830*	
TSTB0	011422	524#									
TSTB1	016604	1460#									
TSTNUM	036176	4159*	4253	4280#							
T1	010344	311#	314	318	334	345	358	370	384		
T10	011316	498#	502	504	507	510	514	517			
T100	025646	2687#	2706	2709	2712	2715					
T101	025764	2718#	2736	2739	2742	2745					
T102	026102	2748#	2760	2765	2768	2771	2781	2786	2789	2792	
T103	026276	2795#	2807	2812							
T104	026376	2816#	2827								
T105	026460	2832#	2837	2839							
T106	026526	2842#	2851	2854	2856						
T107	026614	2859#	2877	2882							
T11	011416	523#	528	531	534	537					
T110	026742	2888#	2902	2907							
T111	027074	2921#	2940	2955	2959						
T112	027212	2962#	3017								
T113	027402	3020#	3069								
T114	027664	3082#	3108								
T115	030042	3150#									
T116	030236	3191#	3211	3214	3217	3220					
T117	030410	3223#	3230	3233	3236	3239	3242				
T12	011472	539#	543	546	550	553	557	561			
T120	030524	3250#									
T121	030562	3261#	3271	3274							
T122	030654	3277#	3285	3288							
T123	030736	3291#	3299	3302							
T124	031020	3305#	3313	3316							
T125	031102	3319#	3327	3330							
T126	031164	3333#	3341	3344							
T127	031246	3347#	3357	3360							
T13	011604	564#	570	573	576	579	582	585			
T130	031340	3363#	3370	3375	3379						
T131	031432	3382#	3394	3400	3413						
T132	031606	3418#	3472	3489							
T133	032012	3497#	3505	3508							
T134	032074	3511#	3533	3541	3544	3549	3554	3559			
T135	032306	3562#									
T136	032646	3638#	3654	3662							
T137	032774	3665#	3679	3685	3698						
T14	011710	587#	592	596	599	602	605	609	612	616	619
T140	033170	3704#									
T141	033304	3732#	3750								
T142	033520	3795#	3820								
T143	033664	3827#	3840								
T144	033720	3843#	3853								
T145	034106	3879#	3888	3892	3897	3906	3915				
T146	034310	3923#	3930	3939	3944	3949	3958	3967			
T147	034522	3971#	3986	3989	3996						
T15	012046	621#	626	629	633	636	641				

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
CNKXAA.P11 17-MAR-82 14:02 CROSS REFERENCE TABLE

N 9

-- USER SYMBOLS

MACY11 30G(1063) 17-MAR-82 14:03 PAGE 9-8
SEQ 0117

T16	012220	659#	665	668	671	675								
T17	012360	695#	701	704	707	710								
T2	010600	387#	390	394										
T20	012532	735#	740	746	749	754	757	762						
T21	013004	798#	802	805	809	812	815	819						
T22	013100	821#	826	829	832	835								
T23	013156	838#	845	848	851	854	857	861	865	868				
T24	013314	871#	876	880	883	886	889	893	896	900	903			
T25	013452	905#	910	913	917	920	925							
T26	013624	943#	949	952	955	959								
T27	013764	979#	985	988	991	994								
T3	010624	396#	411											
T30	014136	1019#	1024	1030	1033	1038	1041	1046						
T31	014410	1082#	1088	1091	1095	1098								
T32	014652	1143#	1147	1150	1154	1157	1161	1166	1171					
T33	015122	1199#	1204	1207	1209	1213	1216	1219						
T34	015220	1225#	1231	1237	1248	1254								
T35	015360	1257#	1272	1284										
T36	015502	1287#	1302	1315										
T37	015660	1318#	1326	1337	1344	1353	1368							
T4	010706	414#	418	420	423									
T40	016106	1371#	1388	1395	1413									
T41	016316	1416#	1429	1436										
T42	016452	1439#	1451	1456										
T43	016600	1459#	1465	1468	1471	1474	1480	1483						
T44	016716	1486#	1493	1496	1499	1502								
T45	017002	1505#	1514	1517	1520	1523	1527	1530	1538	1541	1544	1548		
T46	017210	1550#	1556	1560	1567	1572	1575	1579	1582	1586	1589	1592		
T47	017410	1595#	1603	1606	1610	1614	1620							
T5	010746	426#	430	432	435	442	445							
T50	017604	1639#	1646	1649	1652	1656								
T51	017754	1677#	1684	1687	1690	1693								
T52	020142	1720#	1726	1732	1735	1740	1743	1748						
T53	020424	1785#	1791	1794	1799	1802								
T54	020512	1804#	1811	1814	1817	1820	1823	1826	1837	1840	1847			
T55	020740	1850#	1861	1864	1867	1870	1875	1878	1883	1886	1892	1899	1902	
T56	021206	1905#	1911	1916	1919	1922	1925	1929	1932	1936	1939			
T57	021356	1941#	1947	1950	1954	1957	1963							
T6	011052	448#	452	454	457	463	464	467						
T60	021554	1984#	1991	1994	1997	2002								
T61	021730	2023#	2030	2033	2036	2039								
T62	022116	2066#	2072	2078	2081	2086	2089	2094						
T63	022400	2131#	2138	2141	2145	2148								
T64	022666	2194#	2201	2204	2209	2214	2217	2222	2229	2234	2238	2242	2245	
T65	023450	2304#	2311	2314	2318	2322	2325	2328						
T66	023566	2331#	2337	2340	2345	2348	2355	2359	2363	2369	2372	2377	2380	
T67	024036	2390#	2396	2401	2406	2412	2415	2421	2424					
T7	011156	470#	473	476	478	482	487	491	495					
T70	024210	2427#	2435	2444	2453	2462	2471							
T71	024466	2474#	2480	2483	2486	2489	2492	2495	2498	2501				
T72	024652	2504#	2515	2521	2525									
T73	024750	2529#	2546	2551	2554	2559	2562							
T74	025122	2568#	2575	2580	2583	2586	2589							
T75	025240	2593#	2600	2605	2608	2611	2614							
T76	025356	2618#	2625	2630	2633	2636	2639							
T77	025474	2643#	2666	2673	2676	2679	2682							

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
CNKXAA.P11 17-MAR-82 14:02

B 10

MACY11 306(1063) 17-MAR-82 14:03 PAGE 9-9
SEQ 0118

CROSS REFERENCE TABLE -- USER SYMBOLS

VBIT = 010462	347#												
XBUF = 001052	173#	3159*	3198*	3206	3227	3253*	3254	3371*	3436*	3439*	3535*	3577*	3626*
	3643*	3656*	3671*	3712*	3715*	3722*	3742*	3746*	3756*				
XCSR = 001050	172#	3157*	3166	3197*	3205	3207*	3226	3240	3263*	3264	3265*	3266	3267*
	3278*	3279	3280*	3281	3282*	3292*	3293	3294*	3295	3296*	3306*	3307	3308*
	3309	3310*	3320*	3321	3322*	3323	3324*	3334*	3335	3336*	3337	3338*	3349*
	3350	3351*	3352	3353*	3364*	3367	3372	3376	3383*	3387	3389*	3393*	3399*
	3412*	3415*	3435*	3437	3442	3455*	3519*	3520*	3521	3569*	3570*	3576*	3622*
	3639*	3640*	3644	3666*	3667*	3709*	3710*	3713	3736*	3766*			
	3119#	3576	3622										
XMTBRK= 000001	3120#												
XMTIE = 000100	3121#	3373											
XMTIRDY= 000200	3249#												
XMTIRS 030524	1116	1119#	1124	1131	1137	1140							
XORO 014552	2167	2170#	2175	2182	2188	2191							
XOR1 022554	175#	3163*	3384										
XVFC = 001056	372#												
ZBIT 010546	19	4324#											
SLSTAD 055531	218#	249	271	280	303#	314	318	334	345	358	370	384	390
. = 056534	394	411	418	420	423	430	432	435	442	445	452	454	457
	463	464	467	473	476	477	478	482	487	491	495	502	504
	507	510	514	517	528	531	534	537	543	546	550	553	557
	561	570	573	576	579	582	585	592	596	599	602	605	609
	612	616	619	626	629	633	636	641	646	649	653	656	665
	668	671	675	683	686	689	692	701	704	707	710	718	721
	724	729	732	740	746	749	754	757	762	768	771	777	780
	784	788	791	796	802	805	809	812	815	819	826	829	832
	835	845	848	851	854	857	861	865	868	876	880	883	886
	889	893	896	900	903	910	913	917	920	925	930	933	937
	940	949	952	955	959	967	970	973	976	985	988	991	994
	1002	1005	1008	1013	1016	1024	1030	1033	1038	1041	1046	1052	1055
	1061	1064	1068	1072	1075	1080	1088	1091	1095	1098	1105	1108	1114
	1117	1124	1131	1137	1140	1147	1150	1154	1157	1161	1166	1171	1177
	1180	1185	1190	1193	1197	1204	1207	1209	1213	1216	1219	1231	1237
	1248	1254	1272	1284	1302	1315	1326	1337	1344	1353	1368	1388	1395
	1413	1429	1436	1451	1456	1465	1468	1471	1474	1480	1483	1493	1496
	1499	1502	1514	1517	1520	1523	1527	1530	1538	1541	1544	1548	1556
	1560	1567	1572	1575	1579	1582	1586	1589	1592	1603	1606	1610	1614
	1620	1626	1629	1633	1636	1646	1649	1652	1656	1665	1668	1671	1674
	1684	1687	1690	1693	1703	1706	1709	1714	1717	1726	1732	1735	1740
	1743	1748	1755	1758	1764	1767	1771	1775	1778	1783	1791	1794	1799
	1802	1811	1814	1817	1820	1823	1826	1837	1840	1847	1861	1864	1867
	1870	1875	1878	1883	1886	1892	1899	1902	1911	1916	1919	1922	1925
	1929	1932	1936	1939	1947	1950	1954	1957	1963	1971	1974	1978	1981
	1991	1994	1997	2002	2011	2014	2017	2020	2030	2033	2036	2039	2049
	2052	2055	2060	2063	2072	2078	2081	2086	2089	2094	2101	2104	2110
	2113	2117	2121	2124	2129	2138	2141	2145	2148	2156	2159	2165	2168
	2175	2182	2188	2191	2201	2204	2209	2214	2217	2222	2229	2234	2238
	2242	2245	2253	2256	2260	2265	2268	2272	2276	2279	2286	2290	2293
	2296	2301	2311	2314	2318	2322	2325	2328	2337	2340	2345	2348	2355
	2359	2363	2369	2372	2377	2380	2396	2401	2406	2412	2415	2421	2424
	2435	2444	2453	2462	2471	2480	2483	2486	2489	2492	2495	2498	2501
	2515	2521	2525	2546	2551	2554	2559	2562	2575	2580	2583	2586	2589
	2600	2605	2608	2611	2614	2625	2630	2633	2636	2639	2658	2664	2666
	2673	2676	2679	2682	2706	2709	2712	2715	2736	2739	2742	2745	2756
	2760	2765	2768	2771	2777	2781	2786	2789	2792	2802	2807	2812	2821

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).

CNKXAA.P11 17-MAR-82 14:02

CROSS REFERENCE TABLE -- USER SYMBOLS

C 10

MACY11 306(1063) 17-MAR-82 14:03 PAGE 9-10
SEQ 0119

2827	2837	2839	2851	2854	2856	2877	2882	2900	2902	2906	2907	2940
2955	2959	2987	2999	3009	3017	3044	3047	3059	3062	3069	3102	3108
3174	3176	3178	3180	3211	3214	3217	3220	3230	3233	3236	3239	3242
3243	3257	3271	3274	3285	3288	3299	3302	3313	3316	3327	3330	3341
3344	3357	3360	3370	3375	3379	3394	3400	3413	3472	3489	3505	3508
3528	3533	3541	3544	3549	3554	3559	3578	3582	3591	3599	3602	3610
3615	3618	3624	3632	3635	3649	3654	3662	3679	3685	3698	3728	3750
3762	3820	3822	3824	3836	3840	3848	3853	3862	3867	3872	3875	3888
3852	3897	3906	3915	3930	3936	3939	3944	3949	3958	3967	3981	3986
3989	3996	4049	4067#	4322#	4324							

CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).
CNKXAA.P11 17-MAR-82 14:02

D 10

MACY11 306(1063) 17-MAR-82 14:03 PAGE 10

CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0120

ERCALL	45#	314	318	334	345	358	370	384	390	394	411	418	420	423	430
	432	435	442	445	452	454	457	463	464	467	473	476	478	482	487
	491	495	502	504	507	510	514	517	528	531	534	537	543	546	550
	553	557	561	570	573	576	579	582	585	592	596	599	607	605	609
	612	616	619	626	629	633	636	641	646	649	653	656	665	668	671
	675	683	686	689	692	701	704	707	710	718	721	724	729	732	740
	746	749	754	757	762	768	771	777	780	784	788	791	796	802	805
	809	812	815	819	826	829	832	835	845	848	851	854	857	861	865
	868	876	880	883	886	889	893	896	900	903	910	913	917	920	925
	930	933	937	940	949	952	955	959	967	970	973	976	985	988	991
	994	1002	1005	1008	1013	1016	1024	1030	1033	1038	1041	1046	1052	1055	1061
	1064	1068	1072	1075	1080	1089	1091	1095	1098	1105	1108	1114	1117	1124	1131
	1137	1140	1147	1150	1154	1157	1161	1166	1171	1177	1180	1185	1190	1193	1197
	1204	1207	1209	1213	1216	1219	1231	1237	1248	1254	1272	1284	1302	1315	1326
	1337	1344	1353	1368	1388	1395	1413	1429	1436	1451	1456	1465	1468	1471	1474
	1480	1483	1493	1496	1499	1502	1514	1517	1520	1523	1527	1530	1538	1541	1544
	1548	1556	1560	1567	1572	1575	1579	1582	1586	1589	1592	1603	1606	1610	1614
	1620	1626	1629	1633	1636	1646	1649	1652	1656	1665	1668	1671	1674	1684	1687
	1690	1693	1703	1706	1709	1714	1717	1726	1732	1735	1740	1743	1748	1755	1758
	1764	1767	1771	1775	1778	1783	1791	1794	1799	1802	1811	1814	1817	1820	1823
	1826	1837	1840	1847	1861	1864	1867	1870	1875	1878	1883	1886	1892	1899	1902
	1911	1916	1919	1922	1925	1929	1932	1936	1939	1947	1950	1954	1957	1963	1971
	1974	1978	1981	1991	1994	1997	2002	2011	2014	2017	2020	2030	2033	2036	2039
	2049	2052	2055	2060	2063	2072	2078	2081	2086	2089	2094	2101	2104	2110	2113
	2117	2121	2124	2129	2138	2141	2145	2148	2156	2159	2165	2168	2175	2182	2188
	2191	2201	2204	2209	2214	2217	2222	2229	2234	2238	2242	2245	2253	2256	2260
	2265	2268	2272	2276	2279	2286	2290	2293	2296	2301	2311	2314	2318	2322	2325
	2328	2337	2340	2345	2348	2355	2359	2363	2369	2372	2377	2380	2396	2401	2406
	2412	2415	2421	2424	2435	2444	2453	2462	2471	2480	2483	2486	2489	2492	2495
	2498	2501	2515	2521	2525	2546	2551	2554	2559	2562	2575	2580	2583	2586	2589
	2600	2605	2608	2611	2614	2625	2630	2633	2636	2639	2666	2673	2676	2679	2682
	2706	2709	2712	2715	2736	2739	2742	2745	2760	2765	2768	2771	2781	2786	2789
	2792	2807	2812	2827	2837	2839	2851	2854	2856	2877	2882	2902	2907	2940	2955
	2959	2987	2999	3009	3017	3044	3047	3059	3062	3069	3102	3108	3174	3176	3178
	3180	3211	3214	3217	3220	3230	3233	3236	3239	3242	3257	3271	3274	3285	3288
	3299	3302	3313	3316	3327	3330	3341	3344	3357	3360	3370	3375	3379	3394	3400
	3413	3472	3489	3505	3508	3533	3541	3544	3549	3554	3559	3582	3591	3599	3602
	3610	3615	3618	3632	3635	3654	3662	3679	3685	3698	3728	3750	3762	3820	3822
	3824	3840	3853	3862	3867	3872	3875	3888	3892	3897	3906	3915	3950	3939	3944
	3949	3958	3967	3986	3989	3996									
NEWTES	33#	311	387	396	414	426	448	470	498	523	539	564	587	621	659
	695	735	798	821	838	871	905	943	979	1019	1082	1143	1199	1225	1257
	1287	1318	1371	1416	1439	1459	1486	1505	1550	1595	1639	1677	1720	1785	1804
	1850	1905	1941	1984	2023	2066	2131	2194	2304	2331	2390	2427	2474	2504	2529
	2568	2593	2618	2643	2687	2718	2748	2795	2816	2832	2842	2859	2888	2921	2962
	3020	3082	3150	3191	3223	3250	3261	3277	3291	3305	3319	3333	3347	3363	3382
	3418	3497	3511	3562	3638	3665	3704	3732	3795	3827	3843	3879	3923	3971	

. ABS. 036534 000 OVR RO ABS GBL I

ERRORS DETECTED: 0

CNKXAA, CNKXAA/L1: TOC/CRF=CNKXAA
RUN-TIME: 16 21 1 SECONDS

