

M8063
SBC11/21

KXT-11 PROC DIAG
CNKXAAO

AH-S943A-MC
FICHE 1 OF 1

JUL 1982
COPYRIGHT © 1982
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows. Each cell in the grid contains a small, dense diagram or data table. The diagrams appear to be circuit board layouts or component placement guides, with labels and lines indicating connections. The text within these diagrams is too small to be legible, but the overall structure suggests a detailed technical specification for a hardware component.



IDENTIFICATION

PRODUCT CODE: AC-S942A-MC
PRODUCT NAME: CNKXAA0 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 INVOKE 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 AND 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.

TABLE OF CONTENTS

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	

	DESTINATION MODE 0
521	
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.

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
1225 NON-ZERO ADDRESS MODES
1231 T34 -- TEST MODES 0 AND 1 USING MOV B AND MOV
1237 PC 15242 = MOV B INSTRUCTION FAILED IN MODE 0.
1248 PC 15264 = MOV INSTRUCTION FAILED IN MODE 0.
1254 PC 15330 = MOV INSTRUCTION FAILED IN MODE 1.
1257 PC 15354 = MOV INSTRUCTION FAILED IN MODE 1.
1272 T35 -- TEST MODE 2 USING MOV B AND MOV
1284 PC 15436 = INSTRUCTIONS FAILED IN MODE 2.
1287 PC 15476 = INSTRUCTIONS FAILED IN MODE 2.
1302 T36 -- TEST MODE 3 USING MOV B AND MOV
1315 PC 15600 = INSTRUCTIONS FAILED IN MODE 3.
1318 PC 15654 = INSTRUCTIONS FAILED IN MODE 3.
1326 T37 -- TEST MODE 4 USING MOV B AND MOV
1337 PC 15714 = INSTRUCTIONS FAILED IN MODE 4.
1344 PC 15750 = INSTRUCTIONS FAILED IN MODE 4.
1353 PC 15776 = INSTRUCTIONS FAILED IN MODE 4.
1368 PC 16032 = INSTRUCTIONS FAILED IN MODE 4.
1371 PC 16102 = INSTRUCTIONS FAILED IN MODE 4.
1388 T40 -- TEST MODE 5 USING MOV B AND MOV
1395 PC 16200 = INSTRUCTION FAILED IN MODE 5.
1413 PC 16226 = INSTRUCTIONS FAILED IN MODE 5.
1416 PC 16312 = INSTRUCTIONS FAILED IN MODE 5.
1429 T41 -- TEST MODE 6 USING MOV B AND MOV
1436 PC 16412 = INSTRUCTIONS FAILED IN MODE 6.
1439 PC 16446 = INSTRUCTIONS FAILED IN MODE 6.
1451 T42 -- TEST MODE 7 USING MOV B AND MOV
1456 PC 16544 = MODE 7 IS FAILING.
1459 PC 16574 = INSTRUCTIONS FAILED IN MODE 7.
1465 T43 -- TSTB, CLRB, AND MOV B
1468 PC 16622 = INCORRECT CC BITS.
1471 PC 16632 = INCORRECT CC BITS.
1474 PC 16644 = INCORRECT CC BITS.
1480 PC 16654 = INCORRECT CC BITS.
1483 PC 16702 = MOV B INSTRUCTION FAILED.
1486 PC 16712 = MOV B INSTRUCTION FAILED.
1493 T44 -- CMP B AND BIS B
1496 PC 16746 = INCORRECT CC BITS.
1499 PC 16756 = BIS B OR CMP B INSTRUCTION FAILED.
1502 PC 16766 = CMP B INSTRUCTION FAILED (WRONG CC).
PC 16776 = INSTR FAILED.

1505 T45 -- BICB 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 = RO 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.
1936	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.

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 27360 = 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 DAT1 (MEM TO QBX).
3047	PC 27522 = DATA INCORRECT IN QBX ON DAT1 (MEM TO QBX).
3059	PC 27576 = NXM ERROR IN QBX ON DAT0 (QBX TO MEM).
3062	PC 27610 = DATA INCORRECT IN TEMP2 ON DAT0 (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
3236
3239
3242
3246
3248
3250
3257
3261
3271
3274
3277
3285
3288
3291
3299
3302
3305
3313
3316
3319
3327
3330
3333
3341
3344
3347
3357
3360
3363
3370
3375
3379
3382
3394
3400
3413
3418
3472
3489
3493
3495
3497
3505
3508
3511
3533
3541
3544
3549
3554
3559
3562
3582
3591
3599

PC 30454 = UNDEFINED BITS SET IN RCVR DATA BUFFER.
PC 30466 = UNDEFINED BITS SET IN XMTR STATUS.
PC 30500 = UNDEFINED BITS SET IN XMTR DATA BUFFER.
PC 30512 = XMITTER NOT READY.

TRANSMITTERS

T120 -- FLOAT 1 AND 0 THRU THE XMIT DATA BUFFER
PC 30552 = FLOATING 1 OR 0 INCORRECT IN XBUF.
T121 -- TEST THAT 'XMIT-BRK' CAN BE SET AND CLEARED
PC 30636 = XMIT-BRK BIT FAILED TO SET.
PC 30650 = XMIT-BRK BIT FAILED TO CLEAR.
T122 -- TEST THAT 'XMIT-MAINT' CAN BE SET AND CLEARED
PC 30720 = XMIT-MAINT BIT FAILED TO SET.
PC 30732 = XMIT-MAINT BIT FAILED TO CLEAR.
T123 -- TEST THAT 'XMIT-PBRE' CAN BE SET AND CLEARED
PC 31002 = XMIT-PBRE BIT FAILED TO SET.
PC 31014 = XMIT-PBRE BIT FAILED TO CLEAR.
T124 -- TEST THAT 'XMIT-PBRO' CAN BE SET AND CLEARED
PC 31064 = XMIT-PBRO BIT FAILED TO SET.
PC 31076 = XMIT-PBRO BIT FAILED TO CLEAR.
T125 -- TEST THAT 'XMIT-PBR1' CAN BE SET AND CLEARED
PC 31146 = XMIT-PBR1 BIT FAILED TO SET.
PC 31160 = XMIT-PBR1 BIT FAILED TO CLEAR.
T126 -- TEST THAT 'XMIT-PBR2' CAN BE SET AND CLEARED
PC 31230 = XMIT-PBR2 BIT FAILED TO SET.
PC 31242 = XMIT-PBR2 BIT FAILED TO CLEAR.
T127 -- TEST THAT 'XMIT-IE' CAN BE SET AND CLEARED
PC 31322 = XMIT-IE BIT FAILED TO SET.
PC 31334 = XMIT-IE BIT FAILED TO CLEAR.
T130 -- TEST THAT 'XMIT-RDY' CAN BE SET AND CLEARED
PC 31366 = XMIT-RDY FAILED TO SET.
PC 31412 = XMIT-RDY BIT FAILED TO CLEAR.
PC 31426 = XMIT-RDY FAILED TO SET AFTER XMITTING CHARACTER.
T131 -- TEST THAT WE CAN INTERRUPT ON 'XMT-RDY'
PC 31510 = XMT-RDY INTERRUPT AT WRONG LEVEL (TOO HIGH).
PC 31534 = XMT-RDY INTERRUPT NOT RECEIVED AT CPU LEVEL 0.
PC 31570 = XMT INTERRUPT ACK FAILED TO REMOVE THE REQUEST.
T132 -- TEST THAT EACH XMIT BAUD RATE SETS A DIFFERENT SPEED
PC 31736 = XMT-RDY NEVER SET AT PBR IN RO.
PC 32000 = SPEED DIFFERENTIAL ERROR.

RECEIVERS

T133 -- TEST THAT 'RCV-IE' CAN BE SET AND CLEARED
PC 32052 = RCV-IE BIT FAILED TO SET.
PC 32064 = RCV-IE BIT FAILED TO CLEAR.
T134 -- TEST THAT 'RCV-ACT' AND 'RCV-DONE' CAN SET AND CLEAR
PC 32156 = RCV-ACT AND/OR RCV-DUN ARE SET IN ERROR.
PC 32206 = RCV-ACT FAILED TO SET.
PC 32220 = RCV-DUN SET WHEN RCV-ACT WAS SET.
PC 32240 = RCV-ACT SET BUT FAILED TO CLEAR.
PC 32260 = RCV-DUN FAILED TO SET WHEN RCV-ACT CLEARED.
PC 32302 = RCV-DUN FAILED TO CLEAR AFTER READING DATA.
T135 -- TEST THE 'RCV-BRK', 'RCV-ERR', AND 'FR-ERR' BITS
PC 32410 = NO RCV-DUN AFTER XMIT WITH XMT-BRK SET.
PC 32440 = HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6.
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 CHIP
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 B 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
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81

```
.TITLE CNKXAA -- KXT-11 (FALCON) VERIFICATION TESTS (FIELD VERSION).  
: :  
: : REV B -- FIX RAM ADDRESS TEST.  
: : FIX SLU SPEED DIFFERENTIAL TEST.  
: : FIX PIO RESET TEST.  
: :  
: : PART 1 -- TEST THE T11 INSTRUCTION SET.  
: : PART 2 -- TEST THE T11 TRAPS AND INTERRUPTS.  
: : PART 3 -- TEST THE LOCAL 2K RAM MEMORY.  
: : PART 4 -- TEST THE LOCAL ROM/RAM MEMORY (EMPTY SOCKETS).  
: : PART 5 -- TEST SERIAL LINE UNIT 1 (DEC DC319 DLART).  
: : PART 6 -- TEST SERIAL LINE UNIT 2 ( DITTO ).  
: : PART 7 -- TEST THE PARALLEL I/O INTERFACE (INTEL 8255).  
: :  
: : .ABS  
: : .NLIST MD,MC,CND,TOC  
: : .LIST ME  
: : *****  
: : PRGSIZ= <^H<SLSTAD>>+1 ; PROGRAM SIZE IN 1/8 K UNITS.  
: : .SBTTL SWITCH REGISTER OPTIONS  
: : SWR= 176 ; THE STANDARD SOFT SWITCH REGISTER ADDRESS.  
: : SWR<15> 100000 HALT ON ERROR.  
: : SWR<14>  
: : SWR<13> 020000 INHIBIT ERROR MESSAGES.  
: : SWR<12> 010000 PRINT ERROR SUMMARY AT END-PASS.  
: : SWR<11>  
: : SWR<10>  
: : SWR<09> 001000 LOOP ON ERROR.  
: : SWR<08>  
: : SWR<02:00> N RUN SEGMENT N (1 THRU 7) ONLY.  
: : *****  
: : TNS= 0 ; INIT TEST NUMBER SEQUENCE.  
: :  
: : T11 CPU EQUATES.  
: :  
: : R6= SP  
: : R7= PC  
: : ERRVEC= 4 ; ILLEGAL INSTRUCTION VECTOR.  
: : RESVEC= 10 ; RESERVED OPCODE VECTOR.  
: : BPTVEC= 14 ; BREAKPOINT VECTOR...  
: : TRTVEC= BPTVEC ; ...TRACE TRAP TOO.  
: : IOTVEC= 20 ; IOT VECTOR.  
: : PWRVEC= 24 ; POWER FAIL VECTOR.  
: : EMTVEC= 30 ; EMT VECTOR.  
: : TRPVEC= 34 ; TRAP VECTOR.  
: :  
: : PRI7= 340  
: : PRI6= 300  
: : PRI5= 240  
: : PRI4= 200  
: : PRI3= 140  
: : PRI2= 100  
: : PRI1= 040  
: : PRI0= 000  
: :  
: : RESRVD= 000010 ; RESERVED (UNDEFINED) OPCODE.
```

000076
000176
000000
000006
000007
000004
000010
000014
000014
000020
000024
000030
000034
000340
000300
000240
000200
000140
000100
000040
000000
000010

82	000007	MFPT= 000007	: MFPT OPCODE (NEW INSTRUCTION)...
83	000004	KXT11= 4	: ...SHOULD RETURN #4 IN RO.
84	000254	CLNZ= CLN!CLZ	
85	000263	SEVC= SEV!SEC	
86	000273	SEVVC= 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	PIOMC= 176206	: ...AND MODE CONTROL.
135			
136	174020	QBXCSR= 174020	: Q-BUS EXERCISER CSR1 ADDRESS (OPTIONAL)...
137	000110	QBXVEC= 110	: ...AND VECTOR (AT PRI4).

```

138
139
140
141      001000
142      001002
143      001004
144      001006
145      001010
146      001012
147      001014
148      001016
149      001020
150      001022
151      001024
152      001026
153      001030
154      001032
155      001034
156
157      001036
158
159      001016
160      001020
161      001022
162      001024
163      001026
164      001030
165      001032
166      001034
167      001040
168      001042
169
170      001044
171      001046
172      001050
173      001052
174      001054
175      001056
176
177      001060
178      001062
179      001064
180      001066
181      001070
182      001072
183
184
185      003776
186
187
188
189
190      004000
191      004002
192      004004
193      004006

:
: NOW, EQUATE SOME GENERAL REGISTERS.
:
: INSTRUCTION TEST SCRATCH LOCATIONS.
ADR=      1000
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).
:
: DLART, VARIABLE SPEED COUNTERS.
SPD0=     K0
SPD1=     K1
SPD2=     K2
SPD3=     K3
SPD4=     K4
SPD5=     K5
SPD6=     K6
SPD7=     K7
DFPBR=    ADR+40      ; DEFAULT PBR BITS...
DFSPD=    ADR+42      ; ...AND THE CORRESPONDING SPEED COUNTER.
:
: DLART REGISTER POINTERS.
RCSR=     ADR+44
RBUF=     ADR+46
XCSR=     ADR+50
XBUF=     ADR+52
RVEC=     ADR+54
XVEC=     ADR+56
:
: PARALLEL I/O REGISTER POINTERS.
PPA=      ADR+60
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 2ND 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.
EPCTBL=   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

K 2

MACY11 30G(1063) 17-MAR-82 14:03 PAGE 1-3
SEQ 0023

KXT-11 (FALCON) DEFAULT CONFIGURATION

194
195
196

007776

LOGEND= 7776

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

INITIAL START AND CLEAR VECTORS.

```

198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218 010000
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 010021
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
238
239
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
  
```

.SBTTL INITIAL START AND CLEAR VECTORS.

ON INITIAL START, WE'LL ATTEMPT TO INITIALIZE THE Q-BUS MEMORY VECTOR SPACE WITH A TRAP CATCHER IN THE TRADITIONAL STYLE. LOCATIONS 0 THRU 776 WILL BE COATED WITH:

```

      .WORD    +2
      .WORD    EHALT
  
```

WE HAVE TO PRESUME THAT THE T11 IS CAPABLE OF EXECUTING THE FEW BASIC FUNCTIONS REQUIRED TO SET THE TRAP CATCHERS. IF NOT -- THEN HOW DID WE EVER GET STARTED IN THE FIRST PLACE ???

IN THE ENGINEERING AND FIELD VERSIONS, WE HAVE A MACRO-ODT IN ROM WHICH PROVIDES 'BUS TIME-OUT' AND 'HALT' EMULATION. ON A RESTART TRAP, WE'LL REMAIN IN ODT IF THE STACKED PC POINTS TO A 'HALT' OPCODE. OTHERWISE, WE'LL VECTOR THRU LOC 4.

'BHALT' AND 'BREAK' VECTOR THRU 140 TO THE ODT BREAK HANDLER.

```

      . = 10000          ; PROGRAM STARTS AT 2K.
BEGIN:
INIT:  NOP          ; RESET ???
      MOV    @#30,-(SP) ; SAVE XXDP EMT VECTOR...
      MOV    @#40,-(SP) ; ...LOAD DEV ID AND UNIT...
      MOV    @#42,-(SP) ; ...AND CHAIN RETURN.
      MOV    #2,R0     ; COAT THE ENTIRE VECTOR AREA...
      CLR   R1        ; ...WITH A TRADITIONAL TRAP CATCHER.
      MOV    #200,R2
1$:    MOV    R0,(R1)+ ;
      MOV    #EHALT,(R1)+ ;
      CMP   (R0)+,(R0)+ ;
      SOB   R2,1$
      MOV    (SP)+,@#42 ; RESTORE CHAIN RETURN...
      MOV    (SP)+,@#40 ; ...LOAD DEV...
      MOV    (SP)+,@#30 ; ...AND EMT VECTOR.
      MOV    #137,@#200 ; SET A STANDARD 'RESTART' ADDRESS.
      MOV    #INIT1,@#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,@#BHALT ; ENABLE 'BREAKS' TO ODT...
      MOV   #PRI7,@#BHALT+2 ; ...THRU VECTOR 140.
      MOV   #1,@#TRAP4 ; ENABLE ODT'S 'TRAP 4' EMULATOR.
.PRINT .-2          ; *** DEBUG, 100000 ***
      MTPS  #PRIO   ; AND LOWER THE CPU LEVEL.
  
```

.SBTTL QUICK-VERIFY EMT AND TRAP.

... NOW TRAPS (104400) AND EMTS (104000) ARE USED EXTENSIVELY FOR
 ... ERROR HANDLING, AND TTY SERVICE. THE EMT HAS TO BE FUNCTIONAL
 ... OTHERWISE, THE XXDP+ MONITOR WOULD HAVE CRASHED. TRAPS ARE
 ... UNKNOWN AT THIS POINT. SO TO BE SAFE, WE'LL DO A QUICK VERIFY
 ... OF BOTH THE TRAP AND THE EMT OPCODES HERE.
 ... ADDITIONAL TESTING WILL BE ATTEMPTED IN SEGMENT 2.

252									
253									
254									
255									
256									
257									
258									
259									
260									
261	010140	016746	167664			EMTTRP:	MOV	EMTVEC, -(SP)	: SAVE MONITORS EMT VECTOR.
262	010144	012767	010204	167656			MOV	#4\$, EMTVEC	: 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					GETSWR		: 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 010336 DOALL:
309 010336 012767 036202 025634 S1: MO' #SEG1,SEGN ; SEGMENT 1 TEXT.
310
311 ;*****
(2) .SBTTL T1 -- BRANCHES, FORWARD, BACKWARD, AND CONDITIONAL
(2) ;*****
(2) 010344 104420 000001 T1: BEGINTEST, 1 ;; CHECK SWITCH REGISTER OPTIONS.
312 010350 000405 BRANCH: BR 2$
313 010352 000405 1$: BR 3$
314 010354 104423 010344 4$: ERROR, T1 ;; *** FORWARD BRANCH FAILED ***
315 010360 012707 010374 4$: MOV #5$,PC
316 010364 000772 2$: BR 1$
317 010366 000774 3$: BR 4$
318 010370 104423 010344 5$: ERROR, T1 ;; *** BACKWARD BRANCH FAILED ***
319 010374 NOBIT: CCC ; ZERO CONDITION CODES, NZVC=0000
320 010374 000257 BCS 1$
321 010376 103414 BVS 1$
322 010400 102413 BEQ 1$
323 010402 001412 BMI 1$
324 010404 100411 NOP1 ; CHECK NOP1 INSTRUCTION (OPCODE 260).
325 010406 000260 BCS 1$
326 010410 103407 BVS 1$
327 010412 102406 BEQ 1$
328 010414 001405 BMI 1$
329 010416 100404 BLT 1$
330 010420 002403 BLE 1$
331 010422 003402 BLOS 1$
332 010424 101401 BHI 2$
333 010426 101002
334 010430 1$: ERROR, T1 ;; *** BRANCH FAILURE OR CC NOT 0000 ***
(2) 010430 104423 010344 2$:
335 010434 NBIT: SEN ; NBIT IS SET, NZVC=1000
336 010434 000270 BPL 1$
337 010436 100007 BEQ 1$
338 010440 001406 BGE 1$
339 010442 002005 BGT 1$
340 010444 003004 BCS 1$
341 010446 103403 BLOS 1$
342 010450 101402 BLO 1$
343 010452 103401 BLE 2$
344 010454 003402
345 010456 1$:
(2) 010456 104423 010344 2$: ERROR, T1 ;; *** BRANCH FAILURE OR CC NOT 1000 ***
346 010462 VBIT: SEN ; SET N...
347 010462 000270 SEV ;...AND V, NZVC = 1010
348 010464 000262 BVC 1$
349 010466 102010 BEQ 1$
350 010470 001407 BPL 1$
351 010472 100006 BCS 1$
352 010474 103405 BLT 1$
353 010476 002404 BLE 1$
354 010500 003403 BLOS 1$
355 010502 101402 BLO 1$
356 010504 103401

```

PC 10456 = BRANCH FAILURE OR CC NOT 1000.

```

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

```

T3 -- TEST REGISTER SELECTION

404 010662 060500
 405 010664 060400
 406 010666 060300
 407 010670 060200
 408 010672 060100
 409 010674 020027 005523
 410 010700 001402
 411 010702 104423 010624
 412 010706

```

ADD R5,R0      :...+ 1000...
ADD R4,R0      :...+ 400...
ADD R3,R0      :...+ 100...
ADD R2,R0      :...+ 20...
ADD R1,R0      :...+ 4...
CMP RO,#STACK+1525 :...SHOULD = THIS.
BEQ 1$         : BR IF OK.
ERROR, T3      :; *** REGISTER SELECTION FAILURE ***
1$:

```

413
 414
 (2)
 (2)
 (2) 010706 104420 000004
 415 010712 012700 010726
 416 010716 000277
 417 010720 000110
 418 010722 104423 010706
 419 010726 104417
 420 010730 104423 010706
 421 010734 020027 010726
 422 010740 001402
 423 010742 104423 010706
 424 010746

```

:*****
.SBTTL      T4 -- JMP INSTRUCTION -- MODE 1
:*****
T4:  BEGINTEST, 4      :; CHECK SWITCH REGISTER OPTIONS.
JMP1: MOV #1$,R0       :; TEST JUMP INSTRUCTION MODE 1
      SCC
      JMP (R0)
      ERROR, T4        :; *** MODE 1 JUMP FAILED ***
1$:  CHECKCC+17
      ERROR, T4        :; *** CC BITS ALTERED ON JMP ***
      CMP RO,#1$
      BEQ 2$           :; CONTINUE IF R0 IS OK
      ERROR, T4        :; *** R0 INCORRECT AFTER JUMP ***
2$:

```

425
 426
 (2)
 (2)
 (2) 010746 104420 000005
 427 010752 012700 010766
 428 010756 000277
 429 010760 000120
 430 010762 104423 010746
 431 010766 104417
 432 010770 104423 010746
 433 010774 020027 010770
 434 011000 001402
 435 011002 104423 010746
 436
 437 011006 012767 011036 167772
 438 011014 012767 011052 167766
 439 011022 012700 001006
 440 011026 000277
 441 011030 000130
 442 011032 104423 010746
 443 011036 027067 000000 000006
 444 011044 001402
 445 011046 104423 010746
 446 011052
 447
 448

```

:*****
.SBTTL      T5 -- JMP INSTRUCTION -- MODES 2 AND 3
:*****
T5:  BEGINTEST, 5      :; CHECK SWITCH REGISTER OPTIONS.
JMP2: MOV #1$,R0       :; TEST JUMP INSTRUCTION MODE 2
      SCC
      JMP (R0)+
      ERROR, T5        :; *** MODE 2 JUMP FAILED ***
1$:  CHECKCC+17
      ERROR, T5        :; *** CC BITS ALTERED AFTER JMP ***
      CMP RO,#1$+2
      BEQ JMP3
      ERROR, T5        :; *** MODE 2 FAILED ON JUMP ***
JMP3: MOV #3$,TEMP     :; TEST JUMP INSTRUCTION MODE 3
      MOV #4$,TEMP+2
      MOV #TEMP,R0
      SCC
      JMP @ (R0)+
      ERROR, T5        :; *** MODE 3 JUMP FAILED ***
3$:  CMP @ (R0),4$
      BEQ 4$
      ERROR, T5        :; *** MODE 3 JUMP FAILED ***
4$:

```

449 011052 104420 000006
 449 011056 012700 011074
 450 011062 000277

```

:*****
.SBTTL      T6 -- JMP INSTRUCTION -- MODES 4 AND 5
:*****
T6:  BEGINTEST, 6      :; CHECK SWITCH REGISTER OPTIONS.
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$,2,R0  ; CHECK R0
456 011104 001402          BEQ     JMP5
457 011106 104423 011052  ERROR,   T6          ;; *** R0 INCORRECT AFTER JUMP ***
458
459 011112 012767 011140 167670 JMP5:    MOV      #3$,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 ***
465 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
;*****

```

```

(2) 011156 104420 000007  T7:      BEGINTEST, 7      ;; CHECK SWITCH REGISTER OPTIONS.
471 011162 012703 011204  JMP6:    MOV      #1$,+6,R3
472 011166 000163 177772          JMP      -6(R3)
473 011172 104423 011156  ERROR,   T7          ;; *** MODE 6 JUMP FAILED ***
474 011176 020327 011204  1$:      CMP      R3,#1$,+6  ; CHECK R3
475 011202 001402          BEQ     2$
476 011204 104423 011156  ERROR,   T7          ;; *** R3 WRONG AFTER JMP6 ***
477 011210 000167 000004  2$:      JMP      3$,-4(PC)    ; TEST JUMP INSTRUCTION MODE 6
478 011214 104423 011156  ERROR,   T7          ;; *** JUMP FAILED ***
479
480 011220 012703 011234  3$:      MOV      #JMP7,R3    ; JUMP SHOULD LAND HERE
481 011224 000163 000000          JMP      0(R3)
482 011230 104423 011156  ERROR,   T7          ;; *** JUMP FAILED ***
483
484 011234 012703 001006  JMP7:    MOV      #TEMP,R3
485 011240 012713 011254          MOV      #1$, (R3)
486 011244 000173 000000          JMP      @ (R3)
487 011250 104423 011156  ERROR,   T7          ;; *** MODE 7 JUMP FAILED ***
488 011254 012713 011274  1$:      MOV      #3$, (R3)    ; TEST JUMP INSTRUCTION MODE 7
489 011260 012700 001002          MOV      #TEMP-4,R0
490 011264 000170 000004          JMP      @4(R0)
491 011270 104423 011156  ERROR,   T7          ;; *** JUMP FAILED ***
492 011274 012767 011316 167504  3$:      MOV      #4$,TEMP
493 011302 012700 001006          MOV      #TEMP,R0
494 011306 000170 000000          JMP      @0(R0)
495 011312 104423 011156  ERROR,   T7          ;; *** MODE 7 JUMP FAILED. ***
496 011316          4$:

```

```

;*****
.SBTTL      T10 -- JSR AND RTS INSTRUCTIONS
;*****

```

```

(2) 011316 104420 000010  T10:    BEGINTEST, 10    ;; CHECK SWITCH REGISTER OPTIONS.

```

T10 -- JSR AND RTS INSTRUCTIONS

```

499 011322 012706 003776 JSRST: MOV #STACK,SP ; SET UP STACK POINTER.
500 011326 000277 SCC
501 011330 004767 000004 JSR PC,2$
502 011334 1$:
(2) 011334 104423 011316 ERROR, T10 ;: *** JSR INSTRUCTION FAILED ***
503 011340 104417 2$: CHECKCC+17 ;: *** CC BITS CHANGED ON JSR ***
504 011342 104423 011316 ERROR, T10 ;: *** CC BITS CHANGED ON JSR ***
505 011346 022706 003774 4$: CMP #STACK-2,SP
506 011352 001402 BEQ 5$ ; BR IF STACK IS RIGHT.
507 011354 104423 011316 ERROR, T10 ;: *** STACK WRONG AFTER JSR ***
508 011360 022716 011334 5$: CMP #1$, (SP)
509 011364 001402 BEQ 6$ ; BR IF STACKED PC IS RIGHT.
510 011366 104423 011316 ERROR, T10 ;: *** SP DID NOT HAVE CORRECT RETURN ADDRESS ***
511
512 011372 012716 011404 6$: MOV #7$, (SP) ; CHANGE RETURN PC.
513 011376 000207 RTS PC
514 011400 104423 011316 ERROR, T10 ;: *** RTS INSTRUCTION FAILED ***
515 011404 022706 003776 7$: CMP #STACK,SP
516 011410 001402 BEQ 8$ ; BR IF STACK PROPERLY RESTORED.
517 011412 104423 011316 ERROR, T10 ;: *** SP WAS NOT RESTORED BY RTS INSTRUCTION ***
518
519
520
521 .SBTTL
522 .SBTTL DESTINATION MODE 0
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

```

```

.SBTTL
.SBTTL DESTINATION MODE 0

```

```

*****
.SBTTL T11 -- TSTB, CLRB, AND MOVB
*****

```

```

(2) 011416 104420 000011 T11: BEGINTEST, 11 ;: CHECK SWITCH REGISTER OPTIONS.
524 011422 012706 003776 TSTB0: MOV #STACK,SP ;: INIT THE STACK.
525 011426 000277 SCC
526 011430 105000 CLRB R0 ;: CLEAR THE REGISTER
527 011432 104404 CHECKCC+4 ;: CHECK FOR CC = 4
528 011434 104423 011416 ERROR, T11 ;: *** CLRB FAILED ***
529 011440 105700 TSTB R0 ;: CHECK IT
530 011442 104404 CHECKCC+4 ;: CHECK FOR CC = 4
531 011444 104423 011416 ERROR, T11 ;: *** TSTB FAILED. ***
532 011450 112701 000377 MOVB #377,R1 ;: LOAD THE REGISTER
533 011454 104410 CHECKCC+10 ;: CHECK FOR CC = 10
534 011456 104423 011416 ERROR, T11 ;: *** INCORRECT CC BITS ***
535 011462 105701 TSTB R1 ;: CHECK IT
536 011464 104410 CHECKCC+10 ;: CHECK FOR CC = 10
537 011466 104423 011416 ERROR, T11 ;: *** INCORRECT CC BITS ***
538
539

```

```

*****
.SBTTL T12 -- CMPB AND BISB
*****

```

```

(2) 011472 104420 000012 T12: BEGINTEST, 12 ;: CHECK SWITCH REGISTER OPTIONS.
540 011476 000277 CMPB0: SCC
541 011500 152702 000377 BISB #377,R2 ;: LOAD REGISTER
542 011504 104411 CHECKCC+11 ;: CHECK FOR CC = 11
543 011506 104423 011472 ERROR, T12 ;: *** INCORRECT CC BITS ***
544 011512 122702 000377 CMPB #377,R2 ;: CHECK COMPARE
545 011516 001402 BEQ 2$ ;: CONTINUE IF OK
546 011520 104423 011472 ERROR, T12 ;: *** BISB OR CMPB INSTRUCTION FAILED ***
547 011524 112700 000077 2$: MOVB #77,R0

```

PC 11520 = BISB OR CMPB INSTRUCTION FAILED.

548 011530 120002
 549 011532 100002
 550 011534 104423 011472
 551 011540 120200
 552 011542 100402
 553 011544 104423 011472
 554 011550 112702 000377
 555 011554 122702 000377
 556 011560 001402
 557 011562 104423 011472
 558 011566 112702 000377
 559 011572 120227 000377
 560 011576 001402
 561 011600 104423 011472
 562 011604

CMPB R0,R2 : CHECK IT AGAIN
 BPL 3\$: CONTINUE IF OK
 ERROR, T12 :
 3\$: CMPB R2,R0 :
 BMI 4\$: ONCE MORE
 ERROR, T12 : CONTINUE IF OK
 4\$: MOVB #377,R2 :
 CMPB #377,R2 : *** CMPB INSTRUCTION FAILED (WRONG CC) ***
 BEQ 5\$: LOAD REGISTER, SIGN EXTEND
 ERROR, T12 : CHECK IF BYTE INSTRUCTION
 5\$: MOVB #377,R2 : CONTINUE IF OK
 CMPB R2,#377 : *** CMPB BECAME CMP INSTRUCTION ***
 BEQ 6\$: LOAD REGISTER, SIGN EXTEND
 ERROR, T12 : CHECK IF BYTE INSTRUCTION
 6\$: : CONTINUE IF OK
 : *** MOVB OR CMPB FAILED ***

563
 564
 (2)
 (2)
 (2) 011604 104420 000013
 565 011610 112703 000377
 566 011614 112700 000252
 567 011620 000277
 568 011622 140003
 569 011624 104401
 570 011626 104423 011604
 571 011632 130003
 572 011634 001402
 573 011636 104423 011604
 574 011642 132703 000125
 575 011646 104401
 576 011650 104423 011604
 577 011654 150003
 578 011656 100402
 579 011660 104423 011604
 580 011664 142703 000177
 581 011670 104411
 582 011672 104423 011604
 583 011676 132703 000377
 584 011702 104411
 585 011704 104423 011604
 586
 587

:*****
 .SBTTL T13 -- BICB AND BITB
 :*****
 T13: BEGINTEST, 13 : CHECK SWITCH REGISTER OPTIONS.
 BICB0: MOVB #377,R3 : LOAD REGISTER
 MOVB #252,R0 : PLACE #252 IN R0
 SCC :
 BICB R0,R3 : CLEAR EVERY OTHER BIT
 CHECKCC+1 : CHECK FOR CC = 1
 ERROR, T13 : *** INCORRECT CC BITS ***
 BITB R0,R3 : CHECK IT
 BEQ 4\$: CONTINUE IF OK
 ERROR, T13 : *** BICB OR BITB INSTRUCTION FAILED ***
 4\$: BITB #125,R3 : CHECK IT
 CHECKCC+1 : CHECK FOR CC = 1
 ERROR, T13 : *** INCORRECT CC BITS ***
 BISB R0,R3 : SET THE BITS THAT WERE CLEARED
 BMI 6\$:
 ERROR, T13 : *** BISB INSTRUCTION FAILED ***
 6\$: BICB #177,R3 : CLEAR ALL THE BITS EXCEPT FOR SIGN
 CHECKCC+11 : CHECK FOR CC = 11
 ERROR, T13 : *** INCORRECT CC BITS ***
 BITB #377,R3 : CHECK IT
 CHECKCC+11 : CHECK FOR CC = 11
 ERROR, T13 : *** INCORRECT CC BITS ***

586
 587
 (2)
 (2)
 (2) 011710 104420 000014
 588 011714 112704 000177
 589 011720 000261
 590 011722 105204
 591 011724 104413
 592 011726 104423 011710
 593 011732 112704 000376
 594 011736 105204
 595 011740 104411
 596 011742 104423 011710
 597 011746 105204

:*****
 .SBTTL T14 -- INCB AND DECB
 :*****
 T14: BEGINTEST, 14 : CHECK SWITCH REGISTER OPTIONS.
 INCB0: MOVB #177,R4 : R4 = 177
 SEC :
 INCB R4 : ADD ONES INTO REG. 4
 CHECKCC+13 : CHECK FOR CC = 13
 ERROR, T14 : *** INCORRECT CC BITS ***
 MOVB #376,R4 :
 INCB R4 :
 CHECKCC+11 : CHECK FOR CC = 11
 ERROR, T14 : *** INCORRECT CC BITS ***
 INCB R4 :

PC 11742 = INCORRECT CC BITS.

598 011750 104405
599 011752 104423 011710
600 011756 105204
601 011760 104401
602 011762 104423 011710
603 011766 122704 000001
604 011772 001402
605 011774 104423 011710
606 012000 000261
607 012002 105304
608 012004 104405
609 012006 104423 011710
610 012012 105304
611 012014 104411
612 012016 104423 011710
613 012022 012704 000200
614 012026 105304
615 012030 104403
616 012032 104423 011710
617 012036 105304
618 012040 104401
619 012042 104423 011710
620
621
(2)
(2)
(2) 012046 104420 000015
622 012052 112703 000252
623 012056 000277
624 012060 105103
625 012062 104401
626 012064 104423 012046
627 012070 122703 000125
628 012074 001402
629 012076 104423 012046
630 012102 000277
631 012104 105103
632 012106 104411
633 012110 104423 012046
634 012114 122703 000252
635 012120 001402
636 012122 104423 012046
637 012126 012703 000377
638 012132 000277
639 012134 105103
640 012136 104405
641 012140 104423 012046
642
643 012144 112700 000001
644 012150 105400
645 012152 104411
646 012154 104423 012144
647 012160 122700 000377
648 012164 001402
649 012166 104423 012144
650 012172 012700 000200

```

CHECKCC+5          ; CHECK FOR CC = 5
ERROR, T14         ;; *** INCORRECT CC BITS ***
INCB R4
CHECKCC+1          ; CHECK FOR CC = 1
ERROR, T14         ;; *** INCORRECT CC BITS ***
CMPB #1,R4         ; CHECK IT
BEQ 2$             ; CONTINUE IF OK
ERROR, T14         ;; *** INCB INSTRUCTION FAILED ***

2$: SEC
DECB R4            ; SUBTRACT ONES FROM REG. 4
CHECKCC+5          ; CHECK FOR CC = 5
ERROR, T14         ;; *** INCORRECT CC BITS ***
DECB R4
CHECKCC+11         ; CHECK FOR CC = 11
ERROR, T14         ;; *** INCORRECT CC BITS ***
MOV #200,R4
DECB R4
CHECKCC+3          ; CHECK FOR CC = 3
ERROR, T14         ;; *** INCORRECT CC BITS ***
DECB R4
CHECKCC+1          ; CHECK FOR CC = 1
ERROR, T14         ;; *** INCORRECT CC BITS ***

*****
.SBTTL T15 -- COMB AND NEGB
*****
T15: BEGINTEST, 15 ; CHECK SWITCH REGISTER OPTIONS.
COMBO: MOVB #252,R3 ; LOAD EVERY OTHER BIT
      SCC
      COMB R3        ; 1'S COMPLEMENT
      CHECKCC+1     ; CHECK FOR CC = 1
      ERROR, T15    ; *** INCORRECT CC BITS ***
      CMPB #125,R3 ; CHECK IT
      BEQ 2$        ; CONTINUE IF OK
      ERROR, T15    ; *** COMB INSTRUCTION FAILED ***

2$: SCC
      LOMB R3       ; COMPLEMENT BACK
      CHECKCC+11    ; CHECK FOR CC = 11
      ERROR, T15    ; *** INCORRECT CC BITS ***
      CMPB #252,R3 ; CHECK IT
      BEQ 3$        ; CONTINUE IF OK
      ERROR, T15    ; *** COMB INSTRUCTION FAILED ***

3$: MOV #377,R3
      SCC
      COMB R3
      CHECKCC+5     ; CHECK FOR CC = 5
      ERROR, T15    ; *** INCORRECT CC BITS ***

NEGB0: MOVB #1,R0   ; LOAD THE REGISTER
      NEGB R0       ; 2'S COMPLEMENT
      CHECKCC+11    ; CHECK FOR CC = 11
      ERROR, NEGB0 ; *** INCORRECT CC BITS ***
      CMPB #377,R0 ; CHECK IT
      BEQ 2$        ; CONTINUE IF OK
      ERROR, NEGB0 ; *** NEGB INSTRUCTION FAILED ***

2$: MOV #200,R0

```

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
658
659

```

```

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$:

```

```

(2)
(2)
(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
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
694
695

```

```

:*****
.SBTTL T16 -- ROLB AND RORB
:*****
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$: ROLB R1 ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, T16 ; *** INCORRECT CC BITS ***
ROLB R1 ; SHIFT
CMPB #1,R1 ; CHECK IT
BEQ 2$ ; CONTINUE IF OK
ERROR, T16 ; *** ROLB FAILED. ***
2$: RORB0: MOVB #4,R2 ; LOAD REGISTER
CCC ; CLEAR FLAGS
RORB R2 ; SHIFT
RORB R2
CMPB #1,R2 ; CHECK IT
BEQ 1$ ; CONTINUE IF OK
ERROR, RORB0 ; *** RORB INSTRUCTION FAILED ***
1$: RORB R2 ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, RORB0 ; *** INCORRECT CC BITS ***
RORB R2 ; SHIFT
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, RORB0 ; *** INCORRECT CC BITS ***
CMPB #200,R2 ; CHECK IT
BEQ 2$ ; CONTINUE IF OK
ERROR, RORB0 ; *** RORB FAILED ***
2$:
:*****
.SBTTL T17 -- ASLB AND ASRB
:*****
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

```

```

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

```

PC 12400 = INCORRECT CC BITS.

701 012400 104423 012360
702 012404 122703 000200
703 012410 001402
704 012412 104423 012360
705 012416 106303
706 012420 104407
707 012422 104423 012360
708 012426 106303
709 012430 104404
710 012432 104423 012360
711
712 012436 112704 000004
713 012442 000257
714 012444 106204
715 012446 106204
716 012450 122704 000001
717 012454 001402
718 012456 104423 012436
719 012462 106204
720 012464 104407
721 012466 104423 012436
722 012472 106204
723 012474 104404
724 012476 104423 012436
725 012502 112703 000202
726 012506 106203
727 012510 106203
728 012512 104411
729 012514 104423 012436
730 012520 122703 000340
731 012524 001402
732 012526 104423 012436
733 012532
734
735
(2)
(2)
(2) 012532 104420 000020
736 012536 105000
737 012540 000257
738 012542 105500
739 012544 104404
740 012546 104423 012532
741 012552 000261
742 012554 105500
743 012556 000261
744 012560 105500
745 012562 104400
746 012564 104423 012532
747 012570 122700 000002
748 012574 001402
749 012576 104423 012532
750 012602 112700 000177
751 012606 000261
752 012610 105500
753 012612 104412

```

ERROR, T17
CMPB #200,R3
BEQ 4$
ERROR, T17
4$: ASLB R3
CHECKCC+7
ERROR, T17
ASLB R3
CHECKCC+4
ERROR, T17

ASRB0: MOVB #4,R4
CCC
ASRB R4
ASRB R4
CMPB #1,R4
BEQ 2$
ERROR, ASRB0
2$: ASRB R4
CHECKCC+7
ERROR, ASRB0
ASRB R4
CHECKCC+4
ERROR, ASRB0
MOVB #202,R3
ASRB R3
ASRB R3
CHECKCC+11
ERROR, ASRB0
CMPB #340,R3
BEQ 3$
ERROR, ASRB0
3$:

```

```

*** INCORRECT CC BITS ***
CHECK IT
CONTINUE IF OK
*** ASLB INSTRUCTION FAILED ***
SHIFT
CHECK FOR CC = 7
*** INCORRECT CC BITS ***
SHIFT
CHECK FOR CC = 4
*** INCORRECT CC BITS ***

LOAD REGISTER
CLEAR FLAGS
SHIFT

CHECK IT
CONTINUE IF OK
*** ASRB INSTRUCTION FAILED ***
SHIFT
CHECK FOR CC = 7
*** INCORRECT CC BITS ***
SHIFT
CHECK FOR CC = 4
*** INCORRECT CC BITS ***
LOAD REGISTER
SHIFT

CHECK FOR CC = 11
*** INCORRECT CC BITS ***
CHECK IT
CONTINUE IF OK
*** ASRB FAILED ***

```

```

*****
.SBTTL T20 -- ADCB AND SBCB
*****
T20: BEGINTEST, 20
ADCB0: CLRB R0
CCC
ADCB R0
CHECKCC+4
ERROR, T20
SEC
ADCB R0
SEC
ADCB R0
CHECKCC+0
ERROR, T20
CMPB #2,R0
BEQ 4$
ERROR, T20
4$: MOVB #177,R0
SEC
ADCB R0
CHECKCC+12

```

```

CHECK SWITCH REGISTER OPTIONS.
CLEAR THE REGISTER
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
*** ADCB INSTRUCTION FAILED ***
LOAD LARGEST POSITIVE NUMBER
C=1
ADD C BIT=1
CHECK FOR CC = 12

```

PC 12614 = INCORRECT CC BITS.

754 012614 104423 012532
755 012620 122700 000200
756 012624 001402
757 012626 104423 012532
758 012632 112700 000377
759 012636 000261
760 012640 105500
761 012642 104405
762 012644 104423 012532
763
764 012650 112701 000003
765 012654 000257
766 012656 105601
767 012660 104400
768 012662 104423 012650
769 012666 122701 000003
770 012672 001402
771 012674 104423 012650
772 012700 000261
773 012702 105601
774 012704 000261
775 012706 105601
776 012710 104400
777 012712 104423 012650
778 012716 122701 000001
779 012722 001402
780 012724 104423 012650
781 012730 000261
782 012732 105601
783 012734 104404
784 012736 104423 012650
785 012742 000261
786 012744 105601
787 012746 104411
788 012750 104423 012650
789 012754 122701 000377
790 012760 001402
791 012762 104423 012650
792 012766 112701 000200
793 012772 000261
794 012774 105601
795 012776 104402
796 013000 104423 012650
797
798
(2)
(2)
(2) 013004 104420 000021
799 013010 000277
800 013012 005000
801 013014 104404
802 013016 104423 013004
803 013022 005700
804 013024 104404
805 013026 104423 013004
806 013032 012704 177777

6S: ERROR, T20
CMPB #200,R0
BEQ 6S
6S: ERROR, T20
MOVB #377,R0
SEC
ADCB R0
CHECKCC+5
ERROR, T20
SBCB0: MOVB #3,R1
CCC
SBCB R1
CHECKCC+0
ERROR, SBCB0
CMPB #3,R1
BEQ 2S
2S: ERROR, SBCB0
SEC
SBCB R1
SEC
SBCB R1
CHECKCC+0
ERROR, SBCB0
CMPB #1,R1
BEQ 3S
3S: ERROR, SBCB0
SEC
SBCB R1
CHECKCC+4
ERROR, SBCB0
SEC
SBCB R1
CHECKCC+11
ERROR, SBCB0
CMPB #377,R1
BEQ 4S
4S: ERROR, SBCB0
MOVB #200,R1
SEC
SBCB R1
CHECKCC+2
ERROR, SBCB0
T21: BEGINTEST, 21
MOV0: SCC
CLR R0
CHECKCC+4
ERROR, T21
TST R0
CHECKCC+4
ERROR, T21
MOV #177777,R4

:: *** INCORRECT CC BITS ***
:: CHECK IT
:: CONTINUE IF OK
:: *** ADCB INSTRUCTION FAILED ***
:: LOAD -1
:: C=1
:: ADD C BIT=1
:: CHECK FOR CC = 5
:: *** INCORRECT CC BITS ***
:: LOAD REGISTER
:: CLEAR FLAGS
:: SUBTRACT C BIT=0
:: CHECK FOR CC = 0
:: *** INCORRECT CC BITS ***
:: CHECK IT
:: CONTINUE IF OK
:: *** SBCB INSTRUCTION FAILED ***
:: C=1
:: SUBTRACT C BIT=1
:: C=1
:: CHECK FOR CC = 0
:: *** INCORRECT CC BITS ***
:: CHECK IT
:: CONTINUE IF OK
:: *** SBCB INSTRUCTION FAILED ***
:: C=1
:: SUBTRACT C BIT=1
:: CHECK FOR CC = 4
:: *** INCORRECT CC BITS ***
:: C=1
:: SUBTRACT C BIT = 1
:: CHECK FOR CC = 11
:: *** INCORRECT CC BITS ***
:: CHECK IT
:: CONTINUE IF OK
:: *** SBCB INSTRUCTION FAILED ***
:: LOAD R1
:: C=1
:: SUBTRACT C BIT = 1
:: CHECK FOR CC = 2
:: *** INCORRECT CC BITS ***
:*****
.SBTTL T21 -- TST, CLR, AND MOV
:*****
T21: BEGINTEST, 21 :: CHECK SWITCH REGISTER OPTIONS.
MOV0: SCC
CLR R0
CHECKCC+4
ERROR, T21
TST R0
CHECKCC+4
ERROR, T21
MOV #177777,R4
:: CLEAR THE REGISTER
:: CHECK FOR CC = 4
:: *** INCORRECT CC BITS ***
:: CHECK IT
:: CHECK FOR CC = 4
:: *** INCORRECT CC BITS ***
:: LOAD THE REGISTER

PC 13026 = INCORRECT CC BITS.

807 013036 010401
808 013040 104410
809 013042 104423 013004
810 013046 005701
811 013050 104410
812 013052 104423 013004
813 013056 020401
814 013060 001402
815 013062 104423 013004
816 013066 000263
817 013070 010000
818 013072 104405
819 013074 104423 013004
820
821
(2)
(2)
(2) 013100 104420 000022
822 013104 000277
823 013106 012700 177777
824 013112 050002
825 013114 104411
826 013116 104423 013100
827 013122 020002
828 013124 001402
829 013126 104423 013100
830 013132 022702 000077
831 013136 100002
832 013140 104423 013100
833 013144 020227 000077
834 013150 100402
835 013152 104423 013100
836 013156
837
838
(2)
(2)
(2) 013156 104420 000023
839 013162 012703 177777
840 013166 012700 001006
841 013172 012710 125252
842 013176 000277
843 013200 041003
844 013202 104401
845 013204 104423 013156
846 013210 031003
847 013212 001402
848 013214 104423 013156
849 013220 032703 052525
850 013224 104401
851 013226 104423 013156
852 013232 052703 125252
853 013236 100402
854 013240 104423 013156
855 013244 042703 077777
856 013250 104411

```

MOV R4,R1
CHECKCC+10 ; CHECK FOR CC = 10
ERROR, T21 ; *** INCORRECT CC BITS ***
TST R1 ; CHECK IT
CHECKCC+10 ; CHECK FOR CC = 10
ERROR, T21 ; *** INCORRECT CC BITS ***
CMP R4,R1 ; CHECK R1 TO CONTAIN PROPER DATA
BEQ 2$
ERROR, T21 ; *** R1 WRONG AFTER MOV ***
2$: SEVC ; SET V & C BITS
MOV R0,R0
CHECKCC+5
ERROR, T21 ; *** INCORRECT CC BITS ***

;*****
.SBTTL T22 -- CMP AND BIS
;*****
T22: BEGINTEST, 22 ; CHECK SWITCH REGISTER OPTIONS.
CMP0: SCC
MOV #177777,R0 ; LOAD REGISTER
BIS R0,R2 ; CHECK THE BIS INSTRUCTION
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T22 ; *** INCORRECT CC BITS ***
CMP R0,R2 ; CHECK COMPARE
BEQ 2$ ; CONTINUE IF OK
ERROR, T22 ; *** BIS OR CMP INSTRUCTION FAILED ***
2$: CMP #77,R2 ; CHECK IT AGAIN
BPL 3$ ; CONTINUE IF OK
ERROR, T22 ; *** CMP INSTRUCTION FAILED (WRONG CC) ***
3$: CMP R2,#77 ; ONCE MORE
BMI 4$ ; CONTINUE IF OK
ERROR, T22 ; *** BIC OR CMP FAILED ***

;*****
.SBTTL T23 -- BIC AND BIT
;*****
T23: BEGINTEST, 23 ; CHECK SWITCH REGISTER OPTIONS.
BICO: MOV #177777,R3 ; LOAD REGISTER
MOV #TEMP,R0 ; PLACE THE ADDRESS OF TEMP IN R0
MOV #125252,(R0) ; SET (R0)
SCC
BIC (R0),R3 ; CLEAR EVERY OTHER BIT
CHECKCC+1 ; CHECK FOR CC = 1
ERROR, T23 ; *** INCORRECT CC BITS ***
BIT (R0),R3 ; CHECK IT
BEQ 1$ ; CONTINUE IF OK
ERROR, T23 ; *** BIC OR BIT INSTRUCTION FAILED ***
1$: BIT #52525,R3 ; CHECK IT
CHECKCC+1 ; CHECK FOR CC = 1
ERROR, T23 ; *** INCORRECT CC BITS ***
BIS #125252,R3 ; SET THE BITS THAT WERE CLEARED
BMI 2$ ; CONTINUE IF OK
ERROR, T23 ; *** BIT OR BIS INSTRUCTION FAILED ***
2$: BIC #77777,R3 ; CLEAR ALL THE BITS EXCEPT FOR SIGN
CHECKCC+11 ; CHECK FOR CC = 11

```

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
870
871
(2)
(2)
(2) 013314 104420 000024
872 013320 012704 077777
873 013324 000261
874 013326 005204
875 013330 104413
876 013332 104423 013314
877 013336 012704 177776
878 013342 005204
879 013344 104411
880 013346 104423 013314
881 013352 005204
882 013354 104405
883 013356 104423 013314
884 013362 005204
885 013364 104401
886 013366 104423 013314
887 013372 022704 000001
888 013376 001402
889 013400 104423 013314
890 013404 000261
891 013406 005304
892 013410 104405
893 013412 104423 013314
894 013416 005304
895 013420 104411
896 013422 104423 013314
897 013426 012704 100000
898 013432 005304
899 013434 104403
900 013436 104423 013314
901 013442 005304
902 013444 104401
903 013446 104423 013314
904
905
(2)
(2)
(2) 013452 104420 000025
906 013456 012703 125252

```

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 ***
TST R0              ; CHECK R0 TO CONTAIN 0
BEQ 3$
ERROR, T23          ;; *** BIC FAILED ***

3$:
;*****
.SBTTL T24 -- INC AND DEC
;*****
T24:  BEGINTEST, 24      ;; CHECK SWITCH REGISTER OPTIONS.
INCO:  MOV #77777,R4    ; R4=77777
SEC
INC R4              ; ADD ONES INTO REG. 4
CHECKCC+13         ; CHECK FOR CC = 13
ERROR, T24          ;; *** INCORRECT CC BITS ***
MOV #177776,R4
INC R4
CHECKCC+11         ; CHECK FOR CC = 11
ERROR, T24          ;; *** INCORRECT CC BITS ***
INC R4
CHECKCC+5           ; CHECK FOR CC = 5
ERROR, T24          ;; *** INCORRECT CC BITS ***
INC R4
CHECKCC+1           ; CHECK FOR CC = 1
ERROR, T24          ;; *** INCORRECT CC BITS ***
CMP #1,R4           ; CHECK IT
BEQ 4$              ; FAILED
ERROR, T24          ;; *** INC INSTRUCTION FAILED ***

4$:
SEC
DEC R4              ; SUBTRACT ONES FROM REG. 4
CHECKCC+5           ; CHECK FOR CC = 5
ERROR, T24          ;; *** INCORRECT CC BITS ***
DEC R4
CHECKCC+11         ; CHECK FOR CC = 11
ERROR, T24          ;; *** INCORRECT CC BITS ***
MOV #100000,R4
DEC R4
CHECKCC+3           ; CHECK FOR CC = 3
ERROR, T24          ;; *** INCORRECT CC BITS ***
DEC R4
CHECKCC+1           ; CHECK FOR CC = 1
ERROR, T24          ;; *** INCORRECT CC BITS ***

;*****
.SBTTL T25 -- COM AND NEG
;*****
T25:  BEGINTEST, 25      ;; CHECK SWITCH REGISTER OPTIONS.
COMO:  MOV #125252,R3   ; LOAD EVERY OTHER BIT

```

907 013462 000277
 908 013464 005103
 909 013466 104401
 910 013470 104423 013452
 911 013474 022703 052525
 912 013500 001402
 913 013502 104423 013452
 914 013506 000277
 915 013510 005103
 916 013512 104411
 917 013514 104423 013452
 918 013520 022703 125252
 919 013524 001402
 920 013526 104423 013452
 921 013532 012703 177777
 922 013536 000277
 923 013540 005103
 924 013542 104405
 925 013544 104423 013452
 926
 927 013550 012700 000001
 928 013554 005400
 929 013556 104411
 930 013560 104423 013550
 931 013564 022700 177777
 932 013570 001402
 933 013572 104423 013550
 934 013576 012700 100000
 935 013602 005400
 936 013604 104413
 937 013606 104423 013550
 938 013612 022700 100000
 939 013616 001402
 940 013620 104423 013550
 941 013624
 942
 943
 (2)
 (2)
 (2) 013624 104420 000026
 944 013630 012701 020000
 945 013634 000257
 946 013636 006101
 947 013640 006101
 948 013642 104412
 949 013644 104423 013624
 950 013650 022701 100000
 951 013654 001402
 952 013656 104423 013624
 953 013662 006101
 954 013664 104407
 955 013666 104423 013624
 956 013672 006101
 957 013674 022701 000001
 958 013700 001402
 959 013702 104423 013624

```

: SCC
COM R3
CHECKCC+1
ERROR, T25
CMP #52525,R3
BEQ 2$
ERROR, T25
2$: SCC
COM R3
CHECKCC+11
ERROR, T25
CMP #125252,R3
BEQ 3$
ERROR, T25
3$: MOV #177777,R3
SCC
COM R3
CHECKCC+5
ERROR, T25
NEGO: MOV #1,R0
NEG R0
CHECKCC+11
ERROR, NEGO
CMP #177777,R0
BEQ 2$
ERROR, NEGO
2$: MOV #100000,R0
NEG R0
CHECKCC+13
ERROR, NEGO
CMP #100000,R0
BEQ 3$
ERROR, NEGO
3$:

```

```

: 1'S COMPLEMENT
: CHECK FOR CC = 1
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** COM INSTRUCTION FAILED ***

: COMPLEMENT BACK
: CHECK FOR CC = 11
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** COM INSTRUCTION FAILED ***

: CHECK FOR CC = 5
: *** INCORRECT CC BITS ***

: LOAD THE REGISTER
: 2'S COMPLEMENT
: CHECK FOR CC = 11
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** NEG INSTRUCTION FAILED ***

: 2'S COMPLEMENT
: CHECK FOR CC = 13
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** NEG FAILED ***

```

```

:*****
:SBTTL T26 -- ROL AND ROR
:*****
T26: BEGINTEST, 26
ROLO: MOV #20000,R1
CCC
ROL R1
ROL R1
CHECKCC+12
ERROR, T26
CMP #100000,R1
BEQ 1$
ERROR, T26
1$: ROL R1
CHECKCC+7
ERROR, T26
ROL R1
CMP #1,R1
BEQ 2$
ERROR, T26

```

```

: CHECK SWITCH REGISTER OPTIONS.
: LOAD REGISTER
: CLEAR FLAGS
: SHIFT

: CHECK FOR CC = 12
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** ROL INSTRUCTION FAILED ***

: SHIFT
: CHECK FOR CC = 7
: *** INCORRECT CC BITS ***

: SHIFT
: CHECK IT
: CONTINUE IF OK
: *** ROL FAILED ***

```

PC 13702 = ROL FAILED.

```

960 013706
961 013706 012702 000004
962 013712 000257
963 013714 006002
964 013716 006002
965 013720 022702 000001
966 013724 001402
967 013726 104423 013706
968 013732 006002
969 013734 104407
970 013736 104423 013706
971 013742 006002
972 013744 104412
973 013746 104423 013706
974 013752 022702 100000
975 013756 001402
976 013760 104423 013706
977 013764
978
979
(2)
(2)
(2) 013764 104420 000027
980 013770 012703 020000
981 013774 000257
982 013776 006303
983 014000 006303
984 014002 104412
985 014004 104423 013764
986 014010 022703 100000
987 014014 001402
988 014016 104423 013764
989 014022 006303
990 014024 104407
991 014026 104423 013764
992 014032 006303
993 014034 104404
994 014036 104423 013764
995
996 014042 012704 000004
997 014046 000257
998 014050 006204
999 014052 006204
1000 014054 022704 000001
1001 014060 001402
1002 014062 104423 014042
1003 014066 006204
1004 014070 104407
1005 014072 104423 014042
1006 014076 006204
1007 014100 104404
1008 014102 104423 014042
1009 014106 012703 100002
1010 014112 006203
1011 014114 006203
1012 014116 104411
  
```

```

2$:
ROR0: MOV #4,R2          : LOAD REGISTER
      CCC              : CLEAR FLAGS
      ROR R2          : SHIFT
      ROR R2          :
      CMP #1,R2       : CHECK IT
      BEQ 1$          : CONTINUE IF OK
      ERROR, ROR0     : *** ROR INSTRUCTION FAILED ***
1$:   ROR R2          : SHIFT
      CHECKCC+7       : CHECK FOR CC = 7
      ERROR, ROR0     : *** INCORRECT CC BITS ***
      ROR R2          : SHIFT
      CHECKCC+12      : CHECK FOR CC = 12
      ERROR, ROR0     : *** INCORRECT CC BITS ***
      CMP #100000,R2  : CHECK IT
      BEQ 2$          : CONTINUE IF OK
      ERROR, ROR0     : *** ROR FAILED ***
2$:
:*****
.SBTTL T27 -- ASL AND ASR
:*****
T27: BEGINTEST, 27      : CHECK SWITCH REGISTER OPTIONS.
ASL0: MOV #20000,R3     : LOAD REGISTER
      CCC              : CLEAR FLAGS
      ASL R3           : SHIFT
      ASL R3           :
      CHECKCC+12       : CHECK FOR CC = 12
      ERROR, T27       : *** INCORRECT CC BITS ***
      CMP #100000,R3   : CHECK IT
      BEQ 4$          : CONTINUE IF OK
      ERROR, T27       : *** ASL INSTRUCTION FAILED ***
4$:   ASL R3           : SHIFT
      CHECKCC+7       : CHECK FOR CC = 7
      ERROR, T27       : *** INCORRECT CC BITS ***
      ASL R3           : SHIFT
      CHECKCC+4        : CHECK FOR CC = 4
      ERROR, T27       : *** INCORRECT CC BITS ***
ASR0: MOV #4,R4         : LOAD REGISTER
      CCC              : CLEAR FLAGS
      ASR R4          : SHIFT
      ASR R4          :
      CMP #1,R4       : CHECK IT
      BEQ 2$          : CONTINUE IF OK
      ERROR, ASR0     : *** ASR INSTRUCTION FAILED ***
2$:   ASR R4          : SHIFT
      CHECKCC+7       : CHECK FOR CC = 7
      ERROR, ASR0     : *** INCORRECT CC BITS ***
      ASR R4          : SHIFT
      CHECKCC+4        : CHECK FOR CC = 4
      ERROR, ASR0     : *** INCORRECT CC BITS ***
      MOV #100002,R3  : LOAD REGISTER
      ASR R3          : SHIFT
      ASR R3          :
      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		
1018			
1019			
(2)			
(2)			
(2)	014136	104420	000030
1020	014142	005000	
1021	014144	000257	
1022	014146	005500	
1023	014150	104404	
1024	014152	104423	014136
1025	014156	000261	
1026	014160	005500	
1027	014162	000261	
1028	014164	005500	
1029	014166	104400	
1030	014170	104423	014136
1031	014174	022700	000002
1032	014200	001402	
1033	014202	104423	014136
1034	014206	012700	077777
1035	014212	000261	
1036	014214	005500	
1037	014216	104412	
1038	014220	104423	014136
1039	014224	022700	100000
1040	014230	001402	
1041	014232	104423	014136
1042	014236	012700	177777
1043	014242	000261	
1044	014244	005500	
1045	014246	104405	
1046	014250	104423	014136
1047			
1048	014254	012701	000003
1049	014260	000257	
1050	014262	005601	
1051	014264	104400	
1052	014266	104423	014254
1053	014272	022701	000003
1054	014276	001402	
1055	014300	104423	014254
1056	014304	000261	
1057	014306	005601	
1058	014310	000261	
1059	014312	005601	
1060	014314	104400	
1061	014316	104423	014254
1062	014322	022701	000001
1063	014326	001402	
1064	014330	104423	014254
1065	014334	000261	

```

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

3$:

;*****
.SBTTL T30 -- ADC AND SBC
;*****
T30: BEGINTEST, 30    ;; CHECK SWITCH REGISTER OPTIONS.
ADC0: CLR R0          ; CLEAR THE REGISTER
      CCC            ; CLEAR FLAGS
      ADC R0         ; ADD C BIT = 0
      CHECKCC+4      ; CHECK FOR CC = 4
      ERROR, T30     ;; *** INCORRECT CC BITS ***
      SEC           ; C=1
      ADC R0         ; ADD C BIT=1
      SEC           ; C=1
      ADC R0         ; AGAIN
      CHECKCC+0      ; CHECK FOR CC = 0
      ERROR, T30     ;; *** INCORRECT CC BITS ***
      CMP #2,R0      ; CHECK IT
      BEQ 4$         ; CONTINUE IF OK
      ERROR, T30     ;; *** ADC INSTRUCTION FAILED ***
4$:  MOV #77777,R0    ; LOAD LARGEST POSITIVE NUMBER
      SEC           ; C=1
      ADC R0         ; ADD C BIT=1
      CHECKCC+12     ; CHECK FOR CC = 12
      ERROR, T30     ;; *** INCORRECT CC BITS ***
      CMP #100000,R0 ; CHECK IT
      BEQ 6$         ; FAILED
      ERROR, T30     ;; *** ADC INSTRUCTION FAILED ***
6$:  MOV #-1,R0      ; LOAD -1
      SEC           ; C=1
      ADC R0         ; ADD C BIT=1
      CHECKCC+5      ; CHECK FOR CC = 5
      ERROR, T30     ;; *** INCORRECT CC BITS ***

SBC0: MOV #3,R1      ; LOAD REGISTER
      CCC            ; CLEAR FLAGS
      SBC R1         ; SUBTRACT C BIT=0
      CHECKCC+0      ; CHECK FOR CC = 0
      ERROR, SBC0    ;; *** INCORRECT CC BITS ***
      CMP #3,R1      ; CHECK IT
      BEQ 2$         ; CONTINUE IF OK
      ERROR, SBC0    ;; *** SBC INSTRUCTION FAILED ***
2$:  SEC           ; C=1
      SBC R1         ; SUBTRACT C BIT=1
      SEC           ; C=1
      SBC R1         ;
      CHECKCC+0      ; CHECK FOR CC = 0
      ERROR, SBC0    ;; *** INCORRECT CC BITS ***
      CMP #1,R1      ; CHECK IT
      BEQ 3$         ; CONTINUE IF OK
      ERROR, SBC0    ;; *** SBC INSTRUCTION FAILED ***
3$:  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$: SENVC ; SET N, V & C BITS
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
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, SWAB0 ; *** INCORRECT CC BITS ***
CMP #177400,R3
BEQ XOR0
ERROR, SWAB0 ; *** 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
1142
1143
(2)
(2)
(2) 014652 104420 000032
1144 014656 012701 021421
1145 014662 060101
1146 014664 104400
1147 014666 104423 014652
1148 014672 022701 043042
1149 014676 001402
1150 014700 104423 014652
1151 014704 012700 156357
1152 014710 060000
1153 014712 104411
1154 014714 104423 014652
1155 014720 022700 134736
1156 014724 001402
1157 014726 104423 014652
1158 014732 012702 100000
1159 014736 060202
1160 014740 104407
1161 014742 104423 014652
1162 014746 012704 021421
1163 014752 012701 156357
1164 014756 060401
1165 014760 001402
1166 014762 104423 014652
1167 014766 005404
1168 014770 012701 021421
1169 014774 060104
1170 014776 001402
1171 015000 104423 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 R0,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$:
;*****
.SBTTL T32 -- ADD AND SUB
;*****
T32: BEGINTEST, 32 ;: CHECK SWITCH REGISTER OPTIONS.
ADD0: MOV #21421,R1 ;: LOAD REGISTERS
ADD R1,R1 ;: ADD
CHECKCC+0 ;: CHECK FOR CC = 0
ERROR, T32 ;: *** INCORRECT CC BITS ***
CMP #43042,R1 ;: CHECK IT
BEQ 1$ ;: CONTINUE IF OK
ERROR, T32 ;: *** ADD INSTRUCTION FAILED ***

1$: MOV # -21421,R0 ;: LOAD REGISTERS
ADD R0,R0 ;: ADD
CHECKCC+11 ;: CHECK FOR CC = 11
ERROR, T32 ;: *** INCORRECT CC BITS ***
CMP # -43042,R0 ;: CHECK IT
BEQ 2$ ;: 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      SUBO:  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, SUBO           : *** 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, SUBO           : *** 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, SUBO           : *** SUB NSTRUCTION FAILED ***
1186 015054 012703 177777      6$:  MOV    #-1,R3        : LOAD REGISTERS
1187 015060 012702 077777      MOV    #77777,R2      : LOAD REGISTERS
1188 015064 160302              SUB    R3,R2           : RESULT SHOULD BE 100000 AND OVERFLOW
1189 015066 104413              CHECKCC+13            : CHECK FOR CC = 13
1190 015070 104423 015004      ERROR, SUBO           : *** INCORRECT CC BITS ***
1191 015074 022702 100000      CMP    #100000,R2     : CHECK IT
1192 015100 001402              BEQ    8$              : CONTINUE IF OK
1193 015102 104423 015004      ERROR, SUBO           : *** SUB INSTRUCTION FAILED ***
1194 015106 012704 177777      8$:  MOV    #-1,R4        :
1195 015112 160304              SUB    R3,R4           :
1196 015114 104404              CHECKCC+4             : CHECK FOR CC = 4
1197 015116 104423 015004      ERROR, SUBO           : *** INCORRECT CC BITS ***
1198
1199
(2)
(2)
(2) 015122 104420 000033      :*****
1200 015126 012701 177777      .SBTTL   T33 -- MTPS AND MFPS
1201 015132 005000              :*****
1202 015134 106400              T33:  BEGINTEST, 33    : CHECK SWITCH REGISTER OPTIONS.
1203 015136 104400              TPSW:  MOV    #177777,R1
1204 015140 104423 015122      CLR    R0              : SET PSW TO 0
1205 015144 106701              MTPS   R0              : CHECK FOR CC = 0
1206 015146 001402              CHECKCC+0             : *** INCORRECT CC BITS ***
1207 015150 104423 015122      ERROR, T33            : MOVE PSW TO R1
1208 015154 104404              MFPS   R1              : BR IF BIT 8 OF PSW WAS EXTENDED IN R1
1209 015156 104423 015122      BEQ    2$              : *** MTPS OR MFPS INSTRUCTION FAILED ***
1210 015162 012700 000337      2$:  CHECKCC+4          : CHECK FOR CC = 4
1211 015166 106400              ERROR, T33            : *** INCORRECT CC BITS ***
1212 015170 104417              MOV    #337,R0        : EXPECT 317 SINCE MTPS DOESN'T SET T BIT
1213 015172 104423 015122      MTPS   R0              : CHECK FOR CC = 17
1214 015176 106701              CHECKCC+17           : *** INCORRECT CC BITS ***
1215 015200 104411              MFPS   R1              : MOVE PSW TO R1
1216 015202 104423 015122      CHECKCC+11           : CHECK FOR CC = 11 (C BIT SHOULD NOT BE EFFECTED BY MFP
1217 015206 022701 177717      ERROR, T33            : *** INCORRECT CC BITS ***
1218 015212 001402              CMP    #177717,R1     : CHECK TO SEE IF BIT 8 OF PSW WAS EXTENDED THRU R1
1219 015214 104423 015122      BEQ    3$              :
1220 015220              ERROR, T33            : *** MTPS OR MFPS INSTRUCTION FAILED ***
1221
1222
1223
1224
:
.SBTTL
.SBTTL  NON-ZERO ADDRESS MODES
:

```

```

1225 ;*****
(2) .SBTTL T34 -- TEST MODES 0 AND 1 USING MOVB AND MOV
(2) ;*****
(2) 015220 104420 000034 T34: BEGINTEST, 34 ;: CHECK SWITCH REGISTER OPTIONS.
1226 015224 112700 000252 MODE0: MOVB #252,R0 ;: LOAD REGISTERS
1227 015230 110001 MOVB R0,R1 ;:
1228 015232 110102 MOVB R1,R2 ;:
1229 015234 122702 000252 CMPB #252,R2 ;: CHECK IT
1230 015240 001402 BEQ 1$ ;: OK, CONTINUE
1231 015242 104423 015220 ERROR, T34 ;: *** MOVB INSTRUCTION FAILED IN MODE 0 ***
1232 015246 012700 125252 1$: MOV #125252,R0 ;: LOAD REGISTERS
1233 015252 010001 MOV R0,R1 ;:
1234 015254 010102 MOV R1,R2 ;:
1235 015256 022702 125252 CMP #125252,R2 ;: CHECK IT
1236 015262 001402 BEQ MODE1 ;: OK, CONTINUE
1237 015264 104423 015220 ERROR, T34 ;: *** MOV INSTRUCTION FAILED IN MODE 0 ***
1238
1239 015270 012700 001006 MODE1: MOV #TEMP,R0 ;: LOAD ADDRESSES INTO REGS.
1240 015274 012701 001010 MOV #TEMP1,R1 ;:
1241 015300 012702 001012 MOV #TEMP2,R2 ;:
1242 015304 005067 163502 CLR TEMP2 ;: START CLEAN
1243 015310 112710 000125 MOVB #125,(R0) ;: LOAD THE LOCATIONS
1244 015314 111011 MOVB (R0),(R1) ;: TEMP => TEMP1
1245 015316 111112 MOVB (R1),(R2) ;: TEMP1 => TEMP2
1246 015320 122767 000125 163464 CMPB #125,TEMP2 ;: CHECK IT
1247 015326 001402 BEQ 1$ ;: OK, CONTINUE
1248 015330 104423 015220 ERROR, T34 ;: *** MOV INSTRUCTION FAILED IN MODE 1 ***
1249 015334 012710 052525 1$: MOV #52525,(R0) ;: LOAD THE LOCATIONS
1250 015340 011011 MOV (R0),(R1) ;: TEMP => TEMP1
1251 015342 011112 MOV (R1),(R2) ;: TEMP1 => TEMP2
1252 015344 022767 052525 163440 CMP #52525,TEMP2 ;: CHECK IT
1253 015352 001402 BEQ 2$ ;: OK, CONTINUE
1254 015354 104423 015220 ERROR, T34 ;: *** MOV INSTRUCTION FAILED IN MODE 1 ***
1255 015360 2$:
1256
1257 ;*****
(2) .SBTTL T35 -- TEST MODE 2 USING MOVB AND MOV
(2) ;*****
(2) 015360 104420 000035 T35: BEGINTEST, 35 ;: CHECK SWITCH REGISTER OPTIONS.
1258 015364 012700 001006 MODE2: MOV #TEMP,R0 ;: LOAD ADDRESSES
1259 015370 012701 001010 MOV #TEMP1,R1 ;:
1260 015374 012702 001012 MOV #TEMP2,R2 ;:
1261 015400 105022 CLRB (R2)+ ;: START CLEAN
1262 015402 112710 000252 MOVB #252,(R0) ;: LOAD THE LOCATIONS
1263 015406 112021 000252 MOVB (R0)+,(R1)+ ;: TEMP => TEMP1
1264 015410 105201 INCB R1 ;: MAKE IT EVEN
1265 015412 111167 163370 MOVB (R1),TEMP ;: MORE 0'S INTO TEMP
1266 015416 105200 INCB R0 ;: MAKE IT EVEN
1267 015420 112021 000252 MOVB (R0)+,(R1)+ ;: TEMP1 => TEMP2
1268 015422 124227 000252 CMPB -(R2),#252 ;: CHECK IT
1269 015426 001003 BNE 1$ ;: FAILED
1270 015430 105767 163352 TSTB TEMP ;: CHECK IT
1271 015434 001402 BEQ 2$ ;: OK, CONTINUE
1272 015436 104423 015360 1$: ERROR, T35 ;: *** INSTRUCTIONS FAILED IN MODE 2 ***
1273

```

```

1274 015442 005741
1275 015444 005022
1276 015446 012740 125252
1277 015452 012020
1278 015454 011067 163326
1279 015460 012121
1280 015462 024227 125252
1281 015466 001003
1282 015470 005767 163312
1283 015474 001402
1284 015476
(2) 015476 104423 015360
1285 015502
1286
1287
(2)
(2)
(2) 015502 104420 000036
1288 015506 012767 001006 163264
1289 015514 012767 001010 163260
1290 015522 012767 001012 163254
1291 015530 012700 001000
1292 015534 012701 001002
1293 015540 105067 163246
1294 015544 112767 000125 163234
1295 015552 113031
1296 015554 113167 163226
1297 015560 113030
1298 015562 122767 000125 163222
1299 015570 001003
1300 015572 105767 163210
1301 015576 001402
1302 015600
(2) 015600 104423 015502
1303
1304 015604 005067 163202
1305 015610 012767 052525 163170
1306 015616 012700 001000
1307 015622 012701 001002
1308 015626 013030
1309 015630 013067 163152
1310 015634 013131
1311 015636 022767 052525 163146
1312 015644 001003
1313 015646 005767 163134
1314 015652 001402
1315 015654
(2) 015654 104423 015502
1316 015660
1317
1318
(2)
(2)
(2) 015660 104420 000037
1319 015664 105067 163116
1320 015670 012700 001006

```

```

2$: TST -(R1)
CLR (R2)+ ; START CLEAN
MOV #125252, -(R0) ; LOAD LOCATIONS
MOV (R0)+, (R0)+ ; TEMP => TEMP1
MOV (R0), TEMP ; 0 => TEMP
MOV (R1)+, (R1)+ ; 125252 => TEMP2
CMP -(R2), #125252 ; CHECK IT
BNE 3$ ; FAILED
TST TEMP ; CHECK IT
BEQ 4$ ; OK, CONTINUE

3$: ERROR, T35 ;: *** INSTRUCTIONS FAILED IN MODE 2 ***

4$:

:*****
.SBTTL T36 -- TEST MODE 3 USING MOV 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
MOVB #125, TEMP
MOVB @ (R0)+, @ (R1)+ ;: TEMP => TEMP1
MOVB @ (R1)+, TEMP ;: TEMP2 => TEMP
MOVB @ (R0)+, @ (R0)+ ;: TEMP1 => TEMP2
CMPB #125, TEMP2 ;: CHECK IT
BNE 1$ ;: FAILED
TSTB TEMP ;: CHECK IT
BEQ 2$ ;: OK, CONTINUE

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

```

T37 -- TEST MODE 4 USING MOVB AND MOV

```

1321 015674 012701 001010      MOV      #TEMP1,R1      :
1322 015700 012702 001012      MOV      #TEMP2,R2      :
1323 015704 005202              INC      R2              : ADJUST THE POINTER
1324 015706 021267 163101      CMP      (R2),TEMP2+1   :
1325 015712 001402              BEQ      1$              :
1326 015714 104423 015660      ERROR,  T37             ;; *** INSTRUCTIONS FAILED IN MODE 4 ***
1327 015720 112742 000252      1$:  MOVB   #252,-(R2)     : LOAD TEMP2
1328 015724 005201              INC      R1              : ADJUST THE POINTERS
1329 015726 005202              INC      R2              :
1330 015730 114241              MOVB   -(R2),-(R1)     : TEMP2 => TEMP1
1331 015732 005200              INC      R0              : ADJUST THE POINTERS
1332 015734 005202              INC      R2              :
1333 015736 114042              MOVB   -(R0),-(R2)     : TEMP => TEMP2
1334 015740 105200              INCB   R0              : ADJUST THE POINTERS
1335 015742 021067 163041      CMP      (R0),TEMP+1   :
1336 015746 001402              BEQ      2$              :
1337 015750 104423 015660      ERROR,  T37             ;; *** INSTRUCTIONS FAILED IN MODE 4 ***
1338 015754 105201              2$:  INCB   R1              :
1339 015756 114140              MOVB   -(R1),-(R0)     : TEMP1 => TEMP
1340 015760 122767 000252 163020  CMPB   #252,TEMP       : CHECK IT
1341 015766 001003              BNE     3$              : FAILED
1342 015770 105767 163016      TSTB   TEMP2           : CHECK IT
1343 015774 001402              BEQ     4$              : OK, CONTINUE
1344 015776              3$:  ERROR, T37             ;; *** INSTRUCTIONS FAILED IN MODE 4 ***
(2) 015776 104423 015660
1345              4$:  CLR     TEMP           : START CLEAN
1346 016002 005067 163000      MOV     #TEMP,R0       : LOAD ADDRESSES
1347 016006 012700 001006      MOV     #TEMP1,R1      :
1348 016012 012701 001010      MOV     #TEMP2,R2      :
1349 016016 012702 001012      TST     (R2)+          : ADJUST THE POINTER
1350 016022 005722              CMP     (R2),TEMP2+2   :
1351 016024 021267 162764      BEQ     5$              :
1352 016030 001402              ERROR,  T37             ;; *** INSTRUCTIONS FAILED IN MODE 4 ***
1353 016032 104423 015660      5$:  MOV     #125252,-(R2) : LOAD TEMP2
1354 016036 012742 125252      TST     (R1)+          : ADJUST THE POINTERS
1355 016042 005721              TST     (R2)+          :
1356 016044 005722              MOV     -(R2),-(R1)     : TEMP2 => TEMP1
1357 016046 014241              TST     (R0)+          : ADJUST POINTERS
1358 016050 005720              TST     (R2)+          :
1359 016052 005722              MOV     -(R0),-(R2)     : TEMP => TEMP2
1360 016054 014042              TST     (R0)+          : ADJUST THE POINTERS
1361 016056 005720              TST     (R1)+          :
1362 016060 005721              MOV     -(R1),-(R0)     : TEMP1 => TEMP
1363 016062 014140              CMP     #125252,TEMP    : CHECK IT
1364 016064 022767 125252 162714  BNE     6$              : FAILED
1365 016072 001003              TST     TEMP2           : CHECK IT
1366 016074 005767 162712      BEQ     7$              : OK, CONTINUE
1367 016100 001402              6$:  ERROR, T37             ;; *** INSTRUCTIONS FAILED IN MODE 4 ***
1368 016102              7$:
(2) 016102 104423 015660
1369 016106
1370
1371
(2)
(2)
(2) 016106 104420 000040
;*****
.SBTTL      T40 -- TEST MODE 5 USING MOVB AND MOV
;*****
T40:  BEGINTEST, 40      ;; CHECK SWITCH REGISTER OPTIONS.

```

T40 -- TEST MODE 5 USING MOVB AND MOV

```

1372 016112 105067 162670      MODE5: CLR      TEMP      : START CLEAN
1373 016116 012767 001006 162654  MOV     #TEMP,ADR     : LOAD ADDRESSES
1374 016124 012767 001010 162650  MOV     #TEMP1,ADR1   :
1375 016132 012767 001012 162644  MOV     #TEMP2,ADR2   :
1376 016140 012700 001000      MOV     #ADR,R0       : LOAD ADDRESSES OF ADDRESSES
1377 016144 012701 001002      MOV     #ADR1,R1      :
1378 016150 012702 001004      MOV     #ADR2,R2      :
1379 016154 005722      TST     (R2)+         : ADJUST THE POINTER
1380 016156 112752 000125  MOVB   #125,a-(R2)    : LOAD TEMP2
1381 016162 022122      CMP     (R1)+(R2)+    : ADJUST THE POINTERS
1382 016164 115251      MOVB   a-(R2),a-(R1) : TEMP2 => TEMP1
1383 016166 022022      CMP     (R0)+(R2)+    : ADJUST THE POINTERS
1384 016170 115052      MOVB   a-(R0),a-(R2) : TEMP => TEMP2
1385 016172 022022      CMP     (R0)+(R2)+    : ADJUST THE POINTERS
1386 016174 125052      CMPB   a-(R0),a-(R2) : CHECK IT
1387 016176 001402      BEQ     1$            :
1388 016200 104423 016106      ERROR, T40           : : *** INSTRUCTION FAILED IN MODE 5 ***
1389 016204 022120      1$:  CMP     (R1)+(R0)+  : ADJUST THE POINTERS
1390 016206 115150      MOVB   a-(R1),a-(R0) : TEMP1 => TEMP
1391 016210 122767 000125 162570  CMPB   #125,TEMP     : CHECK IT
1392 016216 001003      BNE     2$            : FAILED
1393 016220 105767 162566      TSTB   TEMP2         : CHECK IT
1394 016224 001402      BEQ     3$            : OK, CONTINUE
1395 016226      2$:
(2) 016226 104423 016106      ERROR, T40           : : *** INSTRUCTIONS FAILED IN MODE 5 ***
1396
1397 016232 005067 162550      3$:  CLR      TEMP      : START CLEAN
1398 016236 012700 001000      MOV     #ADR,R0       : LOAD ADDRESSES OF ADDRESSES
1399 016242 012701 001002      MOV     #ADR1,R1      :
1400 016246 012702 001004      MOV     #ADR2,R2      :
1401 016252 005722      TST     (R2)+         : ADJUST THE POINTER
1402 016254 012752 052525  MOVB   #52525,a-(R2) : LOAD TEMP2
1403 016260 022122      CMP     (R1)+(R2)+    : ADJUST THE POINTERS
1404 016262 015251      MOVB   a-(R2),a-(R1) : TEMP2 => TEMP1
1405 016264 022022      CMP     (R0)+(R2)+    : ADJUST THE POINTERS
1406 016266 015052      MOVB   a-(R0),a-(R2) : TEMP => TEMP2
1407 016270 022021      CMP     (R0)+(R1)+    : ADJUST THE POINTERS
1408 016272 015150      MOV     a-(R1),a-(R0) : TEMP1 => TEMP
1409 016274 022767 052525 162504  CMP     #52525,TEMP   : CHECK IT
1410 016302 001003      BNE     4$            : FAILED
1411 016304 005767 162502      TST     TEMP2         : CHECK IT
1412 016310 001402      BEQ     5$            : OK, CONTINUE
1413 016312      4$:
(2) 016312 104423 016106      ERROR, T40           : : *** INSTRUCTIONS FAILED IN MODE 5 ***
1414 016316      5$:
1415
1416
(2)
(2)
(2) 016316 104420 000041 :*****
:SBTTL      T41 -- TEST MODE 6 USING MOVB AND MOV
:*****
T41:  BEGINTEST, 41      : : CHECK SWITCH REGISTER OPTIONS.
MODE6: CLR      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)

```


T41 -- TEST MODE 6 USING MOVB AND MOV

```

1423 016356 022767 125252 162422      CMP      #125252,TEMP      : CHECK IT
1424 016364 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$:
(2) 016412 104423 016316      ERROR,   T41                :: *** INSTRUCTIONS FAILED IN MODE 6 ***
1430
1431 016416 005067 162366                    2$: 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      ERROR,   T41                :: *** INSTRUCTIONS FAILED IN MODE 6 ***
1437 016452                    3$:
1438
1439
(2)
(2)
(2) 016452 104420 000042
1440 016456 005067 162326
1441 016462 012767 001006 162310
1442 016470 012767 001010 162304
1443 016476 012767 001012 162300
1444 016504 012700 001000
1445 016510 012701 001002
1446 016514 012702 001004
1447 016520 112770 000252 000000    MOVB     #252,00(R0)      : LOAD TEMP
1448 016526 112770 177774 000002    MOVB     @-4(R2),@2(R0)   : TEMP => TEMP1
1449 016534 122767 000252 162246    CMPB     #252,TEMP1       : CHECK IT
1450 016542 001402                    BEQ      1$                : OK, CONTINUE
1451 016544 104423 016452      ERROR,   T42                :: *** MODE 7 IS FAILING ***
1452 016550 012770 125252 000000    1$: MOV      #125252,@0(R0)  : LOAD TEMP
1453 016556 017270 177774 000002    MOV      @-4(R2),@2(R0)   : TEMP => TEMP1
1454 016564 022767 125252 162216    CMP      #125252,TEMP1    : CHECK IT
1455 016572 001402                    BEQ      2$                : OK, CONTINUE
1456 016574 104423 016452      ERROR,   T42                :: *** INSTRUCTIONS FAILED IN MODE 7 ***
1457
1458 016600                    2$:
1459
(2)
(2)
(2) 016600 104420 000043
1460 016604 012700 001006
1461 016610 012701 001010
1462 016614 000277
1463 016616 105010
1464 016620 104404
1465 016622 104423 016600
1466 016626 105710
1467 016630 104404
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       : LOAD ADDRESSES OF ADDRESSES
      MOV      #ADR1,R1
      MOV      #ADR2,R2
      MOVB     #252,@0(R0)   : LOAD TEMP
      MOVB     @-4(R2),@2(R0) : TEMP => TEMP1
      CMPB     #252,TEMP1    : CHECK IT
      BEQ      1$           : OK, CONTINUE
      ERROR,   T42         :: *** MODE 7 IS FAILING ***
1$: MOV      #125252,@0(R0)  : LOAD TEMP
      MOV      @-4(R2),@2(R0) : TEMP => TEMP1
      CMP      #125252,TEMP1 : CHECK IT
      BEQ      2$           : OK, CONTINUE
      ERROR,   T42         :: *** INSTRUCTIONS FAILED IN MODE 7 ***
  
```

 .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 ***
  
```

PC 16644 = INCORRECT CC BITS.

1472	016650	105711		
1473	016652	104410		
1474	016654	104423	016600	
1475	016660	010002		
1476	016662	112762	000200	000000
1477	016670	112241		
1478	016672	026127	177777	100200
1479	016700	001402		
1480	016702	104423	016600	
1481	016706	020102		
1482	016710	001402		
1483	016712	104423	016600	
1484	016716			

```

TSTB (R1) ; CHECK IT
CHECKCC+10 ; CHECK FOR CC = 10
ERROR, T43 ; *** INCORRECT CC BITS ***
MOV R0,R2 ; R2 IS NOW POINTING TO LOCATION TEMP
MOVB #200,0(R2) ; PLACE #200 IN LOCATION TEMP
MOVB (R2)+,-(R1) ; MOVE #200 TO LOCATION TEMP+1
CMP -1(R1),#100200 ; CHECK THE DATA IN LOCATION TEMP
BEQ 4$
ERROR, T43 ; *** MOVB INSTRUCTION FAILED ***
4$: CMP R1,R2 ; CHECK THE REGISTERS FOR PROPER VALUE
BEQ 5$
ERROR, T43 ; *** MOVB INSTRUCTION FAILED ***
5$:

```

1485				
1486				
(2)				
(2)				
(2)	016716	104420	000044	
1487	016722	012701	001012	
1488	016726	012702	001006	
1489	016732	012711	000077	
1490	016736	112704	000377	
1491	016742	150412		
1492	016744	104410		
1493	016746	104423	016716	
1494	016752	120412		
1495	016754	001402		
1496	016756	104423	016716	
1497	016762	121112		
1498	016764	100002		
1499	016766	104423	016716	
1500	016772	121211		
1501	016774	100402		
1502	016776	104423	016716	
1503	017002			

```

;*****
.SBTTL T44 -- CMPB AND BISB
;*****
T44: BEGINTEST, 44 ; CHECK SWITCH REGISTER OPTIONS.
CMPB1: MOV #TEMP2,R1 ; LOAD ADDRESS
MOV #TEMP,R2 ; PLACE 77 IN LOCATION TEMP2
MOV #77,(R1) ; R4 SHOULD CONTAIN #177777
MOVB #377,R4 ; LOAD LOCATION
BISB R4,(R2) ; CHECK FOR CC = 10
CHECKCC+10 ; *** INCORRECT CC BITS ***
ERROR, T44 ; CHECK COMPARE
CMPB R4,(R2) ; CONTINUE IF OK
BEQ 2$ ; *** BISB OR CMPB INSTRUCTION FAILED ***
ERROR, T44 ; CHECK IT AGAIN
2$: CMPB (R1),(R2) ; CONTINUE IF OK
BPL 3$ ; *** CMPB INSTRUCTION FAILED (WRONG CC) ***
ERROR, T44 ; ONCE MORE
3$: CMPB (R2),(R1) ; CONTINUE IF OK
BMI 4$ ; *** INSTR FAILED ***
ERROR, T44
4$:

```

1504				
1505				
(2)				
(2)				
(2)	017002	104420	000045	
1506	017006	012703	001006	
1507	017012	112713	000377	
1508	017016	012700	001010	
1509	017022	010001		
1510	017024	112721	000252	
1511	017030	000277		
1512	017032	146013	000000	
1513	017036	104401		
1514	017040	104423	017002	
1515	017044	136113	177777	
1516	017050	001402		
1517	017052	104423	017002	
1518	017056	132713	000125	
1519	017062	104401		
1520	017064	104423	017002	
1521	017070	154113		

```

;*****
.SBTTL T45 -- BICB AND BITB
;*****
T45: BEGINTEST, 45 ; CHECK SWITCH REGISTER OPTIONS.
BICB1: MOV #TEMP,R3 ; LOAD ADDRESS
MOVB #377,(R3) ; LOAD LOCATION
MOV #TEMP1,R0 ; PLACE THE ADDRESS OF TEMP1 IN R0
MOV R0,R1 ; AND R1
MOVB #252,(R1)+ ; PLACE #252 IN TEMP1
SCC ; CLEAR EVERY OTHER BIT
BICB 0(R0),(R3) ; CHECK FOR CC = 1
CHECKCC+1 ; *** INCORRECT CC BITS ***
ERROR, T45 ; CHECK IT
BITB -1(R1),(R3) ; CONTINUE IF OK
BEQ 4$ ; *** BICB OR BITB INSTRUCTION FAILED ***
ERROR, T45 ; CHECK IT
4$: BITB #125,(R3) ; CHECK FOR CC = 1
CHECKCC+1 ; *** INCORRECT CC BITS ***
ERROR, T45 ; SET THE BITS THAT WERE CLEARED
BITB -(R1),(R3)

```

PC 17064 = INCORRECT CC BITS.

```

1522 017072 100402
1523 017074 104423 017002
1524 017100 012746 000177
1525 017104 142613
1526 017106 104411
1527 017110 104423 017002
1528 017114 132713 000377
1529 017120 104411
1530 017122 104423 017002
1531 017126 010300
1532 017130 012710 001010
1533 017134 012730 000377
1534 017140 000263
1535 017142 145070 000000
1536
1537 017146 104405
1538 017150 104423 017002
1539 017154 022027 001010
1540 017160 001402
1541 017162 104423 017002
1542 017166 005750
1543 017170 001402
1544 017172 104423 017002
1545 017176 000257
1546 017200 141010
1547 017202 104404
1548 017204 104423 017002
1549
1550
(2)
(2)
(2) 017210 104420 000046
1551 017214 012704 001006
1552 017220 112714 000177
1553 017224 000261
1554 017226 105214
1555 017230 104413
1556 017232 104423 017210
1557 017236 012714 000376
1558 017242 105224
1559 017244 104411
1560 017246 104423 017210
1561 017252 105744
1562 017254 005746
1563 017256 010426
1564 017260 000241
1565 017262 105256
1566 017264 104404
1567 017266 104423 017210
1568 017272 123634
1569 017274 000261
1570 017276 105264 177777
1571 017302 104401
1572 017304 104423 017210
1573 017310 124427 000001
1574 017314 001402

```

```

6$: BMI 6$ ; CONTINUE IF OK
ERROR, T45 ; *** BITB OR BISB INSTRUCTION FAILED ***
MOV #177, -(SP) ; STORE #177 ON THE STACK
BICB (SP)+, (R3) ; CLEAR ALL THE BITS EXCEPT SIGN BIT
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T45 ; *** INCORRECT CC BITS ***
BITB #377, (R3) ; CHECK IT
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T45 ; *** INCORRECT CC BITS ***
MOV R3, R0 ; PLACE THE ADDRESS OF TEMP IN R0
MOV #TEMP1, (R0) ; PLACE THE ADDRESS OF TEMP1 IN TEMP
MOV #377, a(R0)+ ; WRITE A 377 IN LOCATION TEMP1
SEVC ; SET V & C BITS
BICB a-(R0), a(R0) ; BIT CLEAR THE CONTENTS
; OF TEMP1 TO THE CONTENTS OF TEMP1
CHECKCC+5 ; CHECK FOR CC = 5
ERROR, T45 ; *** INCORRECT CC BITS ***
CMP (R0)+, #TEMP1 ; MAKE SURE THAT (R0) IS POINTING TO LOCATION TEMP1
BEQ 8$
8$: ERROR, T45 ; *** BICB OR CMP INSTRUCTION FAILED IN THE SPECIFIC MO
TST a-(R0) ; TEST LOCATION TEMP1
BEQ 10$
10$: ERROR, T45 ; *** BICB INSTRUCTION FAILED ***
CCC ;
BICB (R0), (R0) ; CLEAR THE LOCATION TEMP
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, T45 ; *** INCORRECT CC BITS ***

```

```

:*****
.SBTTL T46 -- INCB AND DECB
:*****

```

```

T46: BEGINTEST, 46 ; CHECK SWITCH REGISTER OPTIONS.
INCB1: MOV #TEMP, R4 ; LOAD ADDRESS
MOV #177, (R4) ; TEMP LOCATION=177
SEC ;
INCB (R4) ; ADD ONES INTO LOCATION
CHECKCC+13 ; CHECK FOR CC = 13
ERROR, T46 ; *** INCORRECT CC BITS ***
MOV #376, (R4) ;
INCB (R4)+ ;
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T46 ; *** INCORRECT CC BITS ***
TSTB -(R4) ; DECREMENT R4 BY 1
TST -(SP) ; AND SP BY 2
MOV R4, (SP)+ ; PLACE THE ADDRESS OF TEMP ON THE STACK
CLC ; CLEAR C BIT
INCB a-(SP) ; INCREMENT THE CONTENTS OF LOCATION TEMP
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, T46 ; *** INCORRECT CC BITS ***
CMPB a(SP)+, a(R4)+ ; RESTORE STACK POINTER
SEC ; SET C BIT
INCB -1(R4) ;
CHECKCC+1 ; CHECK FOR CC = 1
ERROR, T46 ; *** INCORRECT CC BITS ***
CMPB -(R4), #1 ; CHECK IT
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          :*****
T47: BEGINTEST, 47          ;; CHECK SWITCH REGISTER OPTIONS.
COMB1: MOV #TEMP,R3          ; LOAD ADDRESS
MOV #TEMP1,R4
MOV #252,(R4)
MOVB (R4)+,(R3)          ; LOAD EVERY OTHER BIT
SCC
COMB (R3)          ; 1'S COMPLEMENT
CHECKCC+1          ; CHECK FOR CC = 1
1603 017440 104423 017410          ERROR, T47          ;; *** INCORRECT CC BITS ***
1604 017444 122713 000125          CMPB #125,(R3)          ; CHECK IT
1605 017450 001402          BEQ 2$          ; CONTINUE IF OK
1606 017452 104423 017410          ERROR, T47          ;; *** COMB INSTRUCTION FAILED ***
1607 017456 000277          2$: SCC
COMB (R3)          ; COMPLEMENT BACK
CHECKCC+11         ; CHECK FOR CC = 11
1610 017464 104423 017410          ERROR, T47          ;; *** INCORRECT CC BITS ***
1611 017470 010400          MOV R4,R0
1612 017472 126013 177777          CMPB -1(R0),(R3)          ; CHECK IT
1613 017476 001402          BEQ 3$          ; CONTINUE IF OK
1614 017500 104423 017410          ERROR, T47          ;; *** COMB INSTRUCTION FAILED ***
1615 017504 112724 000377          3$: MOVB #377,(R4)+
1616 017510 114413          MOVB -(R4),(R3)          ; PLACE #377 IN (R3)
1617 017512 000277          SCC
COMB (R3)
CHECKCC+5          ; CHECK FOR CC = 5
1620 017520 104423 017410          ERROR, T47          ;; *** INCORRECT CC BITS ***
1621
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
```

PC 17540 = INCORRECT CC BITS.

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
1638
1639
(2)
(2)
(2) 017604 104420 000050
1640 017610 012701 001010
1641 017614 112711 000040
1642 017620 000257
1643 017622 106111
1644 017624 106111
1645 017626 104412
1646 017630 104423 017604
1647 017634 122711 000200
1648 017640 001402
1649 017642 104423 017604
1650 017646 106111
1651 017650 104407
1652 017652 104423 017604
1653 017656 106111
1654 017660 122711 000001
1655 017664 001402
1656 017666 104423 017604
1657
1658 017672 012702 001010
1659 017676 112712 000004
1660 017702 000257
1661 017704 106012
1662 017706 106012
1663 017710 122712 000001
1664 017714 001402
1665 017716 104423 017672
1666 017722 106012
1667 017724 104407
1668 017726 104423 017672
1669 017732 106012
1670 017734 104412
1671 017736 104423 017672
1672 017742 122712 000200
1673 017746 001402
1674 017750 104423 017672
1675 017754
1676
1677
(2)
(2)
(2) 017754 104420 000051

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\$:
;*****
.SBTTL T50 -- ROLB AND RORB
;*****
T50: BEGINTEST, 50 ; CHECK SWITCH REGISTER OPTIONS.
ROLB1: MOV #TEMP1,R1 ; LOAD ADDRESS
MOVB #40,(R1) ; LOAD LOCATION
CCC ; CLEAR FLAGS
ROLB (R1) ; SHIFT
ROLB (R1)
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, T50 ; ** INCORRECT CC BITS **
CMPB #200,(R1) ; CHECK IT
BEQ 1\$; CONTINUE IF OK
ERROR, T50 ; ** ROLB INSTRUCTION FAILED **
1\$: ROLB (R1) ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, T50 ; ** INCORRECT CC BITS **
ROLB (R1) ; SHIFT
CMPB #1,(R1) ; CHECK IT
BEQ RORB1 ; CONTINUE IF OK
ERROR, T50 ; ** ROLB FAILED **

RORB1: MOV #TEMP1,R2 ; LOAD ADDRESS
MOVB #4,(R2) ; LOAD LOCATION
CCC ; CLEAR FLAGS
RORB (R2) ; SHIFT
RORB (R2)
CMPB #1,(R2) ; CHECK IT
BEQ 1\$; CONTINUE IF OK
ERROR, RORB1 ; ** RORB INSTRUCTION FAILED **
1\$: RORB (R2) ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, RORB1 ; ** INCORRECT CC BITS **
RORB (R2) ; SHIFT
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, RORB1 ; ** INCORRECT CC BITS **
CMPB #200,(R2) ; CHECK IT
BEQ 2\$; CONTINUE IF OK
ERROR, RORB1 ; ** RORB FAILED **

2\$:
;*****
.SBTTL T51 -- ASLB AND ASRB
;*****
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
        MOVB #40,(R3)
        CCC
        ASLB (R3)
        ASLB (R3)
        CHECKCC+12
        ERROR, T51
        CMPB #200,(R3)
        BEQ 4$
        ERROR, T51
4$: ASLB (R3)
        CHECKCC+7
        ERROR, T51
        ASLB (R3)
        CHECKCC+4
        ERROR, T51

ASRB1: MOV #TEMP1,R4
        MOV #TEMP2,R3
        MOVB #4,(R4)
        CCC
        ASRB (R4)
        ASRB (R4)
        CMPB #1,(R4)
        BEQ 2$
        ERROR, ASRB1
2$: ASRB (R4)
        CHECKCC+7
        ERROR, ASRB1
        ASRB (R4)
        CHECKCC+4
        ERROR, ASRB1
        MOVB #202,(R3)
        ASRB (R3)
        ASRB (R3)
        CHECKCC+11
        ERROR, ASRB1
        CMPB #340,(R3)
        BEQ 3$
        ERROR, ASRB1
3$:
  
```

```

: LOAD ADDRESS
: LOAD LOCATION
: CLEAR FLAGS
: SHIFT
:
: CHECK FOR CC = 12
: *** INCORRECT CC BITS ***
: CHECK IT
: CONTINUE IF OK
: *** ASLB INSTRUCTION FAILED ***
: SHIFT
: CHECK FOR CC = 7
: *** INCORRECT CC BITS ***
: SHIFT
: CHECK FOR CC = 4
: *** INCORRECT CC BITS ***
:
: LOAD ADDRESSES
:
: LOAD LOCATION
: CLEAR FLAGS
: SHIFT
:
: CHECK IT
: CONTINUE IF OK
: *** ASRB 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
: *** ASRB FAILED ***
  
```

```

:*****
:SBTTL T52 -- ADCB AND SBCB
:*****
T52: BEGINTEST, 52
ADCB1: MOV #TEMP2,R0
        CLRB (R0)
        CCC
        ADCB (R0)
        CHECKCC+4
        ERROR, T52
        SEC
        ADCB (R0)
        SEC
        ADCB (R0)
  
```

```

: 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
  
```

PC 20162 = INCORRECT CC BITS.

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 #177,(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		2\$: 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		3\$: 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						

:SBTTL T53 -- TST,-CLR, AND MOV

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,a(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 024227 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$: CMP #77,-(R2) ;: CHECK IT AGAIN
      CHECKCC+1 ;: CHECK FOR CC = 1
      ERROR, T54 ;: *** INCORRECT CC BITS ***
      CMP #77777,(R2)+ ;:
      CHECKCC+13 ;: CHECK FOR CC = 13
      ERROR, T54 ;: *** INCORRECT CC BITS ***
      CMP -(R2),#77777 ;: ONCE MORE
      CHECKCC+10 ;: CHECK FOR CC = 10
      ERROR, T54 ;: *** INCORRECT CC BITS ***
      MOV #52525,TEMP2 ;: SET EVERY OTHER BIT IN TEMP2
      MOV #TEMP2,TEMP1 ;: PLACE THE ADDRESS OF TEMP2 IN TEMP1
      MOV #ADR,R4 ;:
      MOV #ADR1,(R4) ;: PLACE THE ADDRESS OF ADR1 IN ADR POINTED BY R4
      MOV #125252,a(R4)+ ;: PLACE THE #125252 IN LOCATION ADR1
      BIS a-2(R4),a(R2)+ ;: SET EVERY OTHER BIT AT LOCATION TEMP2
      ;: AND INCREMENT R2 BY 2
      MOV R2,R0 ;: PLACE ADDRESS OF TEMP2 IN R0
      CMP a-(R0),#177777 ;: TEMP2 SHOULD CONTAIN ALL 1'S
      BEQ 4$
*****

```



```

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$: CLR -(R0)          ; PLACE A 0 IN TEMP
1842 020712 010067 160074          MOV R0,TEMP2        ; PLACE ADDRESS OF TEMP IN TEMP2
1843 020716 022020                    CMP (R0)+,(R0)+     ; BUMP R0 BY 4
1844 020720 055070 000002          BIS @-(R0),@2(R0)   ; PLACE THE CONTENTS OF TEMP2 AT TEMP
1845 020724 022767 001006          160054 CMP #TEMP,TEMP      ; TEMP SHOULD CONTAIN ITS OWN ADDRESS
1846 020732 001402                    BEQ 6$
1847 020734 104423 020512          ERROR, T54          ;; *** CMP OR BIS INSTRUCTIONS FAILED ***
1848 020740                    6$:
1849
1850
1850 (2)
1850 (2)
1850 (2)
*****
.SBTTL T55 -- BIC AND BIT
*****
1851 020740 104420 000055          T55: BEGINTEST, 55 ; CHECK SWITCH REGISTER OPTIONS.
1852 020744 012703 001006          BIC1: MOV #TEMP,R3   ; LOAD ADDRESS
1853 020750 012713 177777          MOV #177777,(R3)   ; LOAD LOCATION
1854 020754 012704 001000          MOV #ADR,R4        ; PLACE THE ADDRESS OF ADR IN R4
1855 020760 012714 001002          MOV #ADR1,(R4)     ; PLACE THE ADDRESS OF ADR1 IN ADR
1856 020764 011334                    MOV (R3),@2(R4)+   ; LOAD LOCATION ADR1 WITH #177777
1857 020766 012700 001010          MOV #TEMP1,R0      ; PLACE THE ADDRESS OF TEMP1 IN R0
1858 020772 012710 125252          MOV #125252,(R0)   ; SET EVERY OTHER BIT AT LOCATION TEMP1
1859 020776 000277                    SCC
1860 021000 042013                    BIC (R0)+,(R3)     ; CLEAR EVERY OTHER BIT
1861 021002 104401                    CHECKCC+1          ; CHECK FOR CC = 1
1862 021004 104423 020740          ERROR, T55          ; *** INCORRECT CC BITS ***
1863 021010 034013                    BIT -(R0),(R3)     ; CHECK IT
1864 021012 001402                    BEQ 1$             ; CONTINUE IF OK
1865 021014 104423 020740          ERROR, T55          ; *** BIC OR BIT INSTRUCTION FAILED ***
1866 021020 032713 052525          1$: BIT #52525,(R3) ; CHECK IT
1867 021024 104401                    CHECKCC+1          ; CHECK FOR CC = 1
1868 021026 104423 020740          ERROR, T55          ; *** INCORRECT CC BITS ***
1869 021032 056013 000000          BIS 0(R0),(R3)     ; SET THE BITS THAT WERE CLEARED
1870 021036 100402                    BMI 2$             ; CONTINUE IF OK
1871 021040 104423 020740          ERROR, T55          ; *** BIT OR BIS INSTRUCTION FAILED ***
1872 021044 012720 077777          2$: MOV #77777,(R0)+ ; SET ALL THE BITS AT TEMP1 EXCEPT SIGN
1873 021050 010002                    MOV R0,R2
1874 021052 046213 177776          BIC -2(R2),(R3)    ; TRY CLEARING THE OTHER BITS
1875 021056 104411                    CHECKCC+11         ; CHECK FOR CC = 11
1876 021060 104423 020740          ERROR, T55          ; *** INCORRECT CC BITS ***
1877 021064 020027 001012          CMP R0,#TEMP1+2    ; R0 SHOULD CONTAIN THE ADDRESS OF TEMP1+2
1878 021070 001402                    BEQ 3$
1879 021072 104423 020740          ERROR, T55          ; *** R0 WRONG ***
1880 021076 010010                    3$: MOV R0,(R0)      ; PLACE ADDRESS OF TEMP2 IN TEMP2.
1881 021100 000263                    SEVC               ; SET V AND C BITS.
1882 021102 044000                    BIC -(R0),R0       ; SHOULD CLEAR R0.
1883 021104 104405                    CHECKCC+5          ; CHECK FOR CC = 5
1884 021106 104423 020740          ERROR, T55          ; *** INCORRECT CC BITS ***
1885 021112 037413 177776          BIT @-2(R4),(R3)   ; CHECK IT
1886 021116 104411                    CHECKCC+11         ; CHECK FOR CC = 11
1887 021120 104423 020740          ERROR, T55          ; *** INCORRECT CC BITS ***
1888 021124 012746 125252          MOV #125252,-(SP)  ; SET EVERY OTHER BIT ON THE STACK
1889 021130 017423 177776          MOV @-2(R4),(R3)+  ; SET ALL THE BITS AT LOCATION TEMP
1890 021134 046643 000000          BIC 0(SP),-(R3)   ; CLEAR EVERY OTHER BIT AT LOCATION TEMP

```

PC 21120 = INCORRECT CC BITS.

1890 021140 022327 052525
 1891 021144 001402
 1892 021146 104423 020740
 1893 021152 012700 001014
 1894 021156 010340
 1895 021160 014330
 1896 021162 000263
 1897 021164 035026
 1898 021166 104405
 1899 021170 104423 020740
 1900 021174 020627 003776
 1901 021200 001402
 1902 021202 104423 020740
 1903 021206
 1904
 1905
 (2)
 (2)
 (2) 021206 104420 000056
 1906 021212 012704 001010
 1907 021216 012714 077777
 1908 021222 000261
 1909 021224 005214
 1910 021226 104413
 1911 021230 104423 021206
 1912 021234 012714 177776
 1913 021240 012700 001006
 1914 021244 005214
 1915 021246 104411
 1916 021250 104423 021206
 1917 021254 005214
 1918 021256 104405
 1919 021260 104423 021206
 1920 021264 005214
 1921 021266 104401
 1922 021270 104423 021206
 1923 021274 026427 000000 000001
 1924 021302 001402
 1925 021304 104423 021206
 1926 021310 000261
 1927 021312 005314
 1928 021314 104405
 1929 021316 104423 021206
 1930 021322 005314
 1931 021324 104411
 1932 021326 104423 021206
 1933 021332 012714 100000
 1934 021336 005314
 1935 021340 104403
 1936 021342 104423 021206
 1937 021346 005314
 1938 021350 104401
 1939 021352 104423 021206
 1940
 1941
 (2)

```

4$: CMP (R3)+,#52525 ; TEMP SHOULD CONTAIN # 52525
    BEQ 4$
    ERROR, T55 ;: *** BIC FAILED IN MODE 6 ***
    MOV #TEMP2+2,R0 ;: PLACE THE ADDRESS OF TEMP2+2 IN R0
    MOV R3,-(R0) ;: PLACE THE ADDRESS OF TEMP1 IN TEMP2
    MOV -(R3),a(R0)+ ;: MOVE # 52525 IN LOCATION TEMP1
    SEC ;: SET V & C BITS
    BIT a-(R0),(SP)+ ;: BIT TEST TEMP1 WITH STACK AND RESTORE STACK POINTER
    CHECKCC+5 ;: CHECK FOR CC = 5
    ERROR, T55 ;: *** INCORRECT CC BITS ***
    CMP SP,#STACK ;: MAKE SURE THAT THE SP IS OK
    BEQ 5$
    ERROR, T55 ;: *** STACK POINTER FOULED UP ***

5$: ;*****
.SBTTL T56 -- INC AND DEC
;*****
T56: BEGINTEST, 56 ;: CHECK SWITCH REGISTER OPTIONS.
INC1: MOV #TEMP1,R4 ;: LOAD ADDRESS
      MOV #77777,(R4) ;: TEMP1 = 77777
      SEC
      INC (R4) ;: ADD ONES INTO LOCATION
      CHECKCC+13 ;: CHECK FOR CC = 13
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      MOV #177776,(R4)
      MOV #TEMP,R0 ;: R0 IS POINTING TO LOCATION TEMP
      INC (R4)
      CHECKCC+11 ;: CHECK CC = 11.
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      INC (R4)
      CHECKCC+5 ;: CHECK FOR CC = 5
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      INC (R4)
      CHECKCC+1 ;: CHECK FOR CC = 1
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      CMP 0(R4),#1 ;: CHECK IT
      BEQ 4$ ;: CONTINUE IF OK
      ERROR, T56 ;: *** INC INSTRUCTION FAILED ***

4$: SEC
      DEC (R4) ;: SUBTRACT ONES FROM LOCATION
      CHECKCC+5 ;: CHECK FOR CC = 5
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      DEC (R4)
      CHECKCC+11 ;: CHECK CC = 11.
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      MOV #100000,(R4)
      DEC (R4)
      CHECKCC+3 ;: CHECK FOR CC = 3
      ERROR, T56 ;: *** INCORRECT CC BITS ***
      DEC (R4)
      CHECKCC+1 ;: CHECK FOR CC = 1
      ERROR, T56 ;: *** INCORRECT CC BITS ***

;*****
.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 005414
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
*****
```

PC 21600 = INCORRECT CC BITS.

1993 021610 001402
 1994 021612 104423 021554
 1995 021616 006161 000000
 1996 021622 104407
 1997 021624 104423 021554
 1998 021630 010102
 1999 021632 006112
 2000 021634 022711 000001
 2001 021640 001402
 2002 021642 104423 021554
 2003
 2004 021646 012702 001012
 2005 021652 012712 000004
 2006 021656 000257
 2007 021660 006012
 2008 021662 006012
 2009 021664 022712 000001
 2010 021670 001402
 2011 021672 104423 021646
 2012 021676 006012
 2013 021700 104407
 2014 021702 104423 021646
 2015 021706 006012
 2016 021710 104412
 2017 021712 104423 021646
 2018 021716 022712 100000
 2019 021722 001402
 2020 021724 104423 021646
 2021 021730
 2022
 2023
 (2)
 (2)
 (2) 021730 104420 000061
 2024 021734 012703 001012
 2025 021740 012713 020000
 2026 021744 000257
 2027 021746 006313
 2028 021750 006313
 2029 021752 104412
 2030 021754 104423 021730
 2031 021760 022713 100000
 2032 021764 001402
 2033 021766 104423 021730
 2034 021772 006313
 2035 021774 104407
 2036 021776 104423 021730
 2037 022002 006313
 2038 022004 104404
 2039 022006 104423 021730
 2040
 2041 022012 012704 001012
 2042 022016 012703 001006
 2043 022022 012714 000004
 2044 022026 000257
 2045 022030 006214

```

1$: BEQ 1$ : CONTINUE IF OK
      ERROR, T60 : *** ROL INSTRUCTION FAILED ***
      ROL 0(R1) : SHIFT
      CHECKCC+7 : CHECK FOR CC = 7
      ERROR, T60 : *** INCORRECT CC BITS ***
      MOV R1,R2 : R2 IS NOW POINTING TO LOCATION TEMP2
      ROL (R2) : SHIFT
      CMP #1,(R1) : CHECK IT
      BEQ ROR1 : CONTINUE IF OK
      ERROR, T60 : *** ROL FAILED. ***

ROR1: MOV #TEMP2,R2 : LOAD ADDRESS
      MOV #4,(R2) : LOAD LOCATION
      CCC : CLEAR FLAGS
      ROR (R2) : SHIFT
      ROR (R2) :
      CMP #1,(R2) : CHECK IT
      BEQ 1$ : CONTINUE IF OK
      ERROR, ROR1 : *** ROR INSTRUCTION FAILED ***
1$: ROR (R2) : SHIFT
      CHECKCC+7 : CHECK FOR CC = 7
      ERROR, ROR1 : *** INCORRECT CC BITS ***
      ROR (R2) : SHIFT
      CHECKCC+12 : CHECK FOR CC = 12
      ERROR, ROR1 : *** INCORRECT CC BITS ***
      CMP #100000,(R2) : CHECK IT
      BEQ 2$ : CONTINUE IF OK
      ERROR, ROR1 : *** ROR FAILED. ***

2$:

:*****
.SBTTL T61 -- ASL AND ASR
:*****
T61: BEGINTEST, 61 : CHECK SWITCH REGISTER OPTIONS.
ASL1: MOV #TEMP2,R3 : LOAD ADDRESS
      MOV #20000,(R3) : LOAD LOCATION
      CCC : CLEAR FLAGS
      ASL (R3) : SHIFT
      ASL (R3) :
      CHECKCC+12 : CHECK FOR CC = 12
      ERROR, T61 : *** INCORRECT CC BITS ***
      CMP #100000,(R3) : CHECK IT
      BEQ 4$ : CONTINUE IF OK
      ERROR, T61 : *** ASL INSTRUCTION FAILED ***
4$: ASL (R3) : SHIFT
      CHECKCC+7 : CHECK FOR CC = 7
      ERROR, T61 : *** INCORRECT CC BITS ***
      ASL (R3) : SHIFT
      CHECKCC+4 : CHECK FOR CC = 4
      ERROR, T61 : *** INCORRECT CC BITS ***

ASR1: MOV #TEMP2,R4 : LOAD ADDRESSES
      MOV #TEMP,R3 :
      MOV #4,(R4) : LOAD LOCATION
      CCC : CLEAR FLAGS
      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
 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 012710 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

```

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$:
:*****
:SBTTL T62 -- ADC AND SBC
:*****
T62:

```

```

ADC1: MOV #TEMP,R0
CLR (R0)
CCC
ADC (R0)
CHECKCC+4
ERROR, T62
SEC
ADC (R0)
SEC
ADC (R0)
CHECKCC+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

```

```

: 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 ***

```

```

6$:
SBC1: MOV #TEMP,R1
MOV #3,(R1)
CCC

```

```

: LOAD ADDRESS
: LOAD LOCATION
: CLEAR FLAGS

```

PC 22234 = INCORRECT CC BITS.

2099 022252 005611
 2100 022254 104400
 2101 022256 104423 022240
 2102 022262 022711 000003
 2103 022266 001402
 2104 022270 104423 022240
 2105 022274 000261
 2106 022276 005611
 2107 022300 000261
 2108 022302 005611
 2109 022304 104400
 2110 022306 104423 022240
 2111 022312 022711 000001
 2112 022316 001402
 2113 022320 104423 022240
 2114 022324 000261
 2115 022326 005611
 2116 022330 104404
 2117 022332 104423 022240
 2118 022336 000261
 2119 022340 005611
 2120 022342 104411
 2121 022344 104423 022240
 2122 022350 022711 177777
 2123 022354 001402
 2124 022356 104423 022240
 2125 022362 012711 100000
 2126 022366 000261
 2127 022370 005611
 2128 022372 104402
 2129 022374 104423 022240
 2130
 2131
 (2)
 (2)
 (2) 022400 104420 000063
 2132 022404 012702 001010
 2133 022410 005012
 2134 022412 000277
 2135 022414 000254
 2136 022416 006712
 2137 022420 104405
 2138 022422 104423 022400
 2139 022426 005712
 2140 022430 001402
 2141 022432 104423 022400
 2142 022436 000273
 2143 022440 006712
 2144 022442 104411
 2145 022444 104423 022400
 2146 022450 022712 177777
 2147 022454 001402
 2148 022456 104423 022400
 2149
 2150 022462 012703 001012
 2151 022466 012713 125125

```

SBC (R1) ; SUBTRACT C BIT=0
CHECKCC+0 ; CHECK FOR CC = 0
ERROR, SBC1 ; *** INCORRECT CC BITS ***
CMP #3,(R1) ; CHECK IT
BEQ 2$ ; CONTINUE IF OK
ERROR, SBC1 ; *** SBC INSTRUCTION FAILED ***
2$: SEC ; C=1
SBC (R1) ; SUBTRACT C BIT=1
SEC ; C=1
SBC (R1)
CHECKCC+0 ; CHECK FOR CC = 0
ERROR, SBC1 ; *** INCORRECT CC BITS ***
CMP #1,(R1) ; CHECK IT
BEQ 3$ ; *** SBC INSTRUCTION FAILED ***
3$: SEC ; C=1
SBC (R1) ; SUBTRACT C BIT=1
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, SBC1 ; *** INCORRECT CC BITS ***
SEC ; C=1
SBC (R1) ; SUBTRACT C BIT = 1
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, SBC1 ; *** INCORRECT CC BITS ***
CMP #-1,(R1) ; CHECK IT
BEQ 4$ ; CONTINUE IF OK
ERROR, SBC1 ; *** 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, SBC1 ; *** INCORRECT CC BITS ***

```

```

;*****
.SBTTL T63 -- SXT, SWAB, AND XOR
;*****
T63: BEGINTEST, 63 ; CHECK SWITCH REGISTER OPTIONS.
SXT1: MOV #TEMP1,R2 ; LOAD ADDRESS
CLR (R2) ; CLEAR LOCATIONS
SCC
CLNZ
SXT (R2) ; SIGN EXTEND
CHECKCC+5 ; CHECK FOR CC = 5
ERROR, T63 ; *** INCORRECT CC BITS ***
TST (R2) ; LOCATION SHOULD STILL BE 0
BEQ 2$ ; CONTINUE IF OK
ERROR, T63 ; *** SXT INSTRUCTION FAILED ***
2$: SENVC ; SET N, V & C BITS
SXT (R2) ; SIGN EXTEND
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, T63 ; *** INCORRECT CC BITS ***
CMP #-1,(R2) ; LOCATION SHOULD NEW HAVE -1
BEQ SWAB1 ; CONTINUE IF OK
ERROR, T63 ; *** SXT FAILED ***

SWAB1: MOV #TEMP2,R3 ; LOAD ADDRESS
MOV #125125,(R3) ; LOAD BIT PATTERN INTO LOCATION

```

```

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 1$:
2193
2194
(2) ;*****
(2) .SBTTL T64 -- ADD, SUB, AND SOB ;*****
(2) T64: BEGINTEST, 64 ; CHECK SWITCH REGISTER OPTIONS.
2195 022672 012700 001012 ADD1: MOV #TEMP2,R0 ; LOAD ADDRESSES
2196 022676 012701 001006 MOV #TEMP,R1 ;
2197 022702 012767 021421 156102 MOV #21421,TEMP2 ; LOAD LOCATIONS
2198 022710 011011 MOV (R0),(R1) ;
2199 022712 061011 ADD (R0),(R1) ; ADD
2200 022714 104400 CHECKCC+0 ; CHECK FOR CC = 0
2201 022716 104423 022666 ERROR, T64 ; *** INCORRECT CC BITS ***
2202 022722 022767 043042 156056 CMP #43042,TEMP ; CHECK IT
2203 022730 001402 BEQ 1$ ; CONTINUE IF OK
2204 022732 104423 022666 ERROR, T64 ; *** ADD INSTRUCTION FAILED ***

```

PC 22732 = ADD INSTRUCTION FAILED.

```

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

```


PC 23240 = SUB INSTRUCTION FAILED.

```

2261 023244 012767 177777 155536 2$: MOV #1,TEMP1 ; LOAD LOCATIONS
2262 023252 012767 077777 155526 MOV #77777,TEMP ; LOAD LOCATIONS
2263 023260 161312 SUB (R3),(R2) ; RESULT SHOULD GIVE 10000 AND OVERFLOW
2264 023262 104413 CHECKCC+13 ; CHECK FOR CC = 13
2265 023264 104423 023156 ERROR, SUB1 ; *** INCORRECT CC BITS ***
2266 023270 022767 100000 155510 CMP #100000,TEMP ; CHECK IT
2267 023276 001402 BEQ 3$ ; CONTINUE IF OK
2268 023300 104423 023156 ERROR, SUB1 ; *** SUB INSTRUCTION FAILED ***
2269 023304 012712 177777 3$: MOV #-1,(R2)
2270 023310 161312 SUB (R3),(R2)
2271 023312 104404 CHECKCC+4 ; CHECK FOR CC = 4
2272 023314 104423 023156 ERROR, SUB1 ; *** INCORRECT CC BITS ***
2273 023320 012767 077777 155460 MOV #77777,TEMP
2274 023326 162767 077777 155452 SUB #77777,TEMP
2275 023334 104404 CHECKCC+4 ; CHECK FOR CC=4
2276 023336 104423 023156 ERROR, SUB1 ; *** INCORRECT CC BITS ***
2277 023342 005767 155440 TST TEMP ; TEMP SHOULD BE =0
2278 023346 001402 BEQ SOB1 ; *** SUB INSTRUCTION FAILED ***
2279 023350 104423 023156 ERROR, SUB1
2280 SOB1: MOV #10.,R0 ; LOAD REGISTERS
2281 023354 012700 000012 CLR R1 ;
2282 023360 005001 1$: INC R1 ; KEEP COUNT
2283 023362 005201 CMP R1,#10.
2284 023364 020127 000012 BLE 2$
2285 023370 003402 ERROR, SOB1 ; *** SOB INSTRUCTION FAILED ***
2286 023372 104423 023354 2$: SCC
2287 023376 000277 SOB R0,1$ ; SUB. 1 FROM REG. 0, GO BACK TO 1$
2288 023400 077010 CHECKCC+17 ; CHECK FOR CC = 17
2289 023402 104417 ERROR, SOB1 ; *** INCORRECT CC BITS ***
2290 023404 104423 023354 TST R0 ; REG. 0 = 0 ?
2291 023410 005700 BEQ 3$ ; NO, FAILED
2292 023412 001402 ERROR, SOB1 ; *** SOB INSTRUCTION FAILED ***
2293 023414 104423 023354 3$: CMP #10.,R1 ; DID IT GO THRU 10 TIMES ?
2294 023420 022701 000012 BEQ 4$ ; CONTINUE IF OK
2295 023424 001402 ERROR, SOB1 ; *** SOB INSTRUCTION FAILED ***
2296 023426 104423 023354 4$: MOV #10,R4 ; PLACE #10 IN R4
2297 023432 012704 000010 5$: SOB R4,5$ ; STAY HERE UNTILL R4 = 0
2298 023436 077401 TST R4
2299 023440 005704 BEQ 6$ ; CONTINUE IF OK
2300 023442 001402 ERROR, SOB1 ; *** SOB FAILED ***
2301 023444 104423 023354 6$:
2302 023450
2303
2304

```

```

;*****
.SBTTL T65 -- MTPS AND MFPS
;*****

```

```

(2) 023450 104420 000065 T65: BEGINTEST, 65 ; CHECK SWITCH REGISTER OPTIONS.
2305 023454 012700 001006 T65: MOV #TEMP,R0 ; PUT THE ADDRESS OF TEMP IN R0
2306 023460 012701 001010 T65: MOV #TEMP1,R1 ; PUT THE ADDRESS OF TEMP1 IN R1
2307 023464 012711 177777 T65: MOV #177777,(R1) ; TEMP1 = 177777
2308 023470 005010 T65: CLR (R0) ; TEMP = 0
2309 023472 106410 T65: MTPS (R0) ; PSW = 0
2310 023474 104400 T65: CHECKCC+0 ; CHECK FOR CC = 0
2311 023476 104423 023450 T65: ERROR, T65 ; *** INCORRECT CC BITS ***
2312 023502 106711 T65: MFPS (R1) ; MOVE PSW TO TEMP1
2313 023504 104404 T65: CHECKCC+4 ; CHECK FOR CC = 4

```

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		1\$:	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	155230	CMP #317,TEMP1	
2327	023560	001402			BEQ 2\$	
2328	023562	104423	023450		ERROR, T65	:: *** MFPS INSTRUCTION FAILED IN MODE 6 ***
2329	023566			2\$:		
2330						
2331						
(2)						
(2)						
(2)	023566	104420	000066			
2332	023572	005000			T66: BEGINTEST, 66	:: CHECK SWITCH REGISTER OPTIONS.
2333	023574	000277			BTWRD: CLR R0	
2334	023576	112700	000200		SCC	
2335					MOVB #200,R0	: SET THE HIGHEST BIT OF THE
2336	023602	104411				: LOWER BYTE
2337	023604	104423	023566		CHECKCC+11	: CHECK FOR CC=11
2338	023610	022700	177600		ERROR, T66	:: *** INCORRECT CC BITS ***
2339	023614	001402			CMP #177600,R0	: CHECK FOR SIGN EXTENSION IN R0
2340	023616	104423	023566		BEQ 1\$	
2341	023622	000277		1\$:	ERROR, T66	:: *** SIGN WAS NOT EXTENDED IN R0 ***
2342	023624	012700	177777		SCC	
2343	023630	112700	000000		MOV #177777,R0	
2344	023634	104405			MOVB #0,R0	: CLEAR THE LOWER BYTE OF R0.
2345	023636	104423	023566		CHECKCC+5	: CHECK FOR CC=5
2346	023642	005700			ERROR, T66	:: *** INCORRECT CC BITS ***
2347	023644	001402			TST R0	: CHECK R0 FOR SIGN EXTENTION
2348	023646	104423	023566		BEQ 2\$	
2349	023652	012704	001012	2\$:	ERROR, T66	:: *** SIGN WAS NOT EXTENDED IN R0. ***
2350	023656	012714	000377		MOV #TEMP2,R4	: R4 IS POINTING TO TEMP2
2351	023662	012706	003774		MOV #377,(R4)	: PLACE #377 IN LOCATION TEMP2
2352	023666	116426	000000		MOV #STACK-2,R6	
2353	023672	022706	003776		MOVB 0(R4),(R6)+	: PUSH # 377 ON STACK
2354	023676	001402			CMP #STACK,R6	
2355	023700	104423	023566		BEQ 3\$	
2356	023704	124627	000377	3\$:	ERROR, T66	:: *** R6 DID NOT GET INCREMENTED ***
2357					CMPB -(R6),#377	: CHECK LOCATION STACK-2 TO
2358	023710	001402				: CONTAIN PROPER DATA
2359	023712	104423	023566		BEQ 4\$	
2360	023716	022706	003774	4\$:	ERROR, T66	:: *** BYTE INSTRUCTION IS FAILING WITH R6 ***
2361					CMP #STACK-2,R6	: CHECK THAT R6 WAS DECREMENTED
2362	023722	001402				: BY 2 BY A BYTE INSTRUCTION
2363	023724	104423	023566		BEQ 5\$	
2364	023730	016467	000000	155050	ERROR, T66	:: *** R6 WAS NOT DECREMENTED ***
2365	023736	005726		5\$:	MOV 0(R4),TEMP	: SET THE LOWER BYTE OF LOCATION TEMP
2366	023740	000277			TST (R6)+	: RESTORE STACK POINTER
					SCC	

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          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

```

SECTION 2 -- T11 TRAPS AND INTERRUPTS.

.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 T67: BEGINTEST, 67 ;; CHECK SWITCH REGISTER OPTIONS.  
2391 024042 012700 001026 R6TST: MOV #K4,R0 ; USE K4 AS PSUEDO STACK...  
2392 024046 010006 MOV R0,R6 ; INIT SP.  
2393 024050 112667 154732 MOVB (R6)+,TEMP ;SIX SHOULD INCREMENT BY TWO  
2394 024054 020627 001030 CMP R6,#K4+2  
2395 024060 001402 BEQ 1$  
2396 024062 104423 024036 1$: ERROR, T67 ;; *** R6 DID NOT AUTO INCREMENT BY TWO ***  
2397 024066 010006 MOV R0,R6 ;SHOULD DECREMENT BY TWO  
2398 024070 114667 154712 MOVB -(R6),TEMP  
2399 024074 020627 001024 CMP R6,#K4-2  
2400 024100 001402 BEQ 2$  
2401 024102 104423 024036 2$: ERROR, T67 ;; *** R6 DID NOT AUTO DECREMENT BY 2 ***  
2402 024106 010006 MOV R0,R6 ;DOUBLE AUTO INCREMENT OF R6  
2403 024110 112626 MOVB (R6)+,(R6)+  
2404 024112 020627 001032 CMP R6,#K4+4  
2405 024116 001402 BEQ 3$  
2406 024120 104423 024036 3$: ERROR, T67 ;; *** WRONG AUTO INCREMENT OF R6 ***  
2407 024124 010006 MOV R0,R6  
2408 024126 005001 CLR R1 ;TEST INCREMENT OF R6 AND R1.  
2409 024130 122621 CMPB (R6)+,(R1)+  
2410 024132 020627 001030 CMP R6,#K4+2  
2411 024136 001402 BEQ 4$  
2412 024140 104423 024036 4$: ERROR, T67 ;; *** WRONG INCREMENT OF R6 ***  
2413 024144 020127 000001 CMP R1,#1  
2414 024150 001402 BEQ 5$  
2415 024152 104423 024036 5$: ERROR, T67 ;; *** WRONG INCREMENT OF R1 ***  
2416 024156 010006 MOV R0,R6 ; TEST INCREMENT OF R1 AND R6.  
2417 024160 005001 CLR R1  
2418 024162 122126 CMPB (R1)+,(R6)+  
2419 024164 020127 000001 CMP R1,#1  
2420 024170 001402 BEQ 6$  
2421 024172 104423 024036 6$: ERROR, T67 ;; *** WRONG INCREMENT OF R1 ***  
2422 024176 020627 001030 CMP R6,#K4+2  
2423 024202 001402 BEQ 7$  
2424 024204 104423 024036 7$: ERROR, T67 ;; *** WRONG INCREMENT OF R6 ***  
2425 024210  
2426  
2427 :*****  
(2) .SBTTL T70 -- TEST BYTE TRANSFERS USING STACK POINTER (R6)  
(2) :*****  
(2) 024210 104420 000070 T70: BEGINTEST, 70 ;; CHECK SWITCH REGISTER OPTIONS.  
2428 024214 012767 123456 154606 MOV #123456,K5  
2429 024222 012767 050505 154570 MOV #050505,K1  
2430 024230 012701 001020 MOV #K1,R1 ;R1=(050505)K1  
2431 024234 012706 001030 MOV #K5,R6 ;R6=(123456)K5  
2432 024240 112621 MOVB (R6)+,(R1)+ ;LOW .BYTE OF R6 0 R1  
2433 024242 022767 050456 154550 CMP #050456,K1  
2434 024250 001402 BEQ 1$  
2435 024252 104423 024210 ERROR, T70 ;; *** FALSE TRANSFER OF .BYTE ***
```

PC 24252 = FALSE TRANSFER OF .BYTE.

```
2436
2437 024256 012767 123456 154544 1$: MOV #123456,K5
2438 024264 012767 050505 154526 MOV #050505,K1
2439 024272 012701 001020 MOV #K1,R1 ;R1(050505)K1
2440 024276 012706 001032 MOV #K6,R6 ;R6(123456)K5
2441 024302 114621 MOVVB -(R6),(R1)+ ;LOW .BYTE OF R6 TO R1 (DECREMENT)
2442 024304 026727 154510 050456 CMP K1,#050456
2443 024312 001402 BEQ 2$
2444 024314 104423 024210 ERROR, T70 ;: *** FALSE R6 .BYTE TRANSFER ***
2445
2446 024320 012767 123456 154472 2$: MOV #123456,K1
2447 024326 012767 050505 154474 MOV #050505,K5
2448 024334 012701 001020 MOV #K1,R1 ;:(123456)
2449 024340 012706 001030 MOV #K5,R6 ;:(050505)
2450 024344 112126 MOVVB (R1)+(R6)+ ;LOW OF R1 TO LOW OF R6
2451 024346 022767 050456 154454 CMP #050456,K5
2452 024354 001402 BEQ 3$
2453 024356 104423 024210 ERROR, T70 ;: *** FALSE R6 .BYTE TRANSFER ***
2454
2455 024362 012767 123456 154430 3$: MOV #123456,K1
2456 024370 012767 050505 154432 MOV #050505,K5
2457 024376 012701 001021 MOV #K1+1,R1 ;:123456
2458 024402 012706 001030 MOV #K5,R6 ;:050505
2459 024406 112126 MOVVB (R1)+(R6)+ ;HIGH OF R1 TO LOW OF R6
2460 024410 026727 154414 050647 CMP K5,#050647
2461 024416 001402 BEQ 4$
2462 024420 104423 024210 ERROR, T70 ;: *** FALSE R6 .BYTE TRANSFER ***
2463
2464 024424 012767 123456 154366 4$: MOV #123456,K1
2465 024432 012767 050505 154370 MOV #050505,K5
2466 024440 012701 001021 MOV #K1+1,R1 ;:R1-123456-ODD ADDRESS
2467 024444 012706 001030 MOV #K5,R6 ;:R6-050505--.EVEN ADDRESS
2468 024450 112621 MOVVB (R6)+(R1)+ ;LOW OF R6 TO HIGH OF R1
2469 024452 022767 042456 154340 CMP #042456,K1
2470 024460 001402 BEQ 5$
2471 024462 104423 024210 ERROR, T70 ;: *** FAILED LOW OF 6 TO HIGH OF 1 ***
2472 024466 5$:
2473
2474 ;*****
(2) .SBTTL T71 -- TEST BYTE COMPARES ON SEQUENTIAL ODD/EVEN ADDRESSES
(2) ;*****
(2) T71: BEGINTEST, 71 ;: CHECK SWITCH REGISTER OPTIONS.
2475 024472 005067 154320 CLR K0 ;: K0 <= 000000
2476 024476 012767 052525 154314 MOV #52525,K1 ;: K1 <= 052525
2477 024504 012767 052400 154310 MOV #52400,K2 ;: K2 <= 052400
2478 024512 126767 154302 154301 CMPB K1,K1+1
2479 024520 001402 BEQ 1$
2480 024522 104423 024466 ERROR, T71 ;: *** LOW TO HIGH IN SAME WORD FAILS ***
2481 024526 126767 154267 154264 1$: CMPB K1+1,K1
2482 024534 001402 BEQ 2$
2483 024536 104423 024466 ERROR, T71 ;: *** HIGH TO LOW IN SAME WORD FAILS ***
2484 024542 126767 154255 154250 2$: CMPB K2+1,K1
2485 024550 001402 BEQ 3$
2486 024552 104423 024466 ERROR, T71 ;: *** HIGH TO LOW IN DIFFERENT WORDS FAILS ***
2487 024556 126767 154240 154232 3$: CMPB K2,K0
2488 024564 001402 BEQ 4$
```

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
2503
2504
(2)
(2)
(2) 024652 104420 000072
2505
2506
2507
2508
2509 024656 012706 003776          MOV    #STACK,SP    ; SET STACK POINTER.
2510 024662 012767 024712 153114  MOV    #3$,ERRVEC   ; SET TRAP VECTOR.
2511 024670 106427 000000          MTPS  #PRIO
2512 024674 005737 160000          1$: TST   @#160000    ; NEXM -- TIME-OUT AND TRAP.
2513 024700 000257          CCC
2514 024702 000240          2$: NOP
2515 024704 104423 024652          ERROR, T72          ; SHOULD EXECUTE THIS...
2516 024710 000414          BR    5$             ; ...THEN TRAP AND STACK THIS PC.
2517 024712 011600          3$: MOV    (SP),R0     ; GET STACKED PC...
2518 024714 016601 000002          MOV    2(SP),R1    ; ...AND PSW.
2519 024720 020027 024702          CMP   R0,#2$
2520 024724 001402          BEQ   4$             ; BR IF PC IS RIGHT.
2521 024726 104423 024652          ERROR, T72          ;; *** STACKED PC INCORRECT ON TIME-OUT TRAP ***
2522
2523 024732 005701          4$: TST   R1
2524 024734 001402          BEQ   5$             ; BR IF PSW IS RIGHT.
2525 024736 104423 024652          ERROR, T72          ;; *** STACKED PSW INCORRECT ON TIME-OUT TRAP ***
2526
2527 024742 012767 000006 153034 5$: MOV    #ERRVEC+2,ERRVEC ; RESET TRAP CATCHER.
2528
2529
(2)
(2)
(2) 024750 104420 000073
2530
2531
2532
2533
2534 024754 012700 025012          MOV    #1$,R0      ; 1ST PASS USES A "JMP R"
2535 024760 000402          SKP2
2536 024762 012700 025016          8$: MOV    #2$,R0      ; 2ND PASS USES A "JSR R,R"
2537 024766 012706 003776          MOV    #STACK,SP   ; STACK POINTER SETUP
2538 024772 012767 025030 153004  MOV    #3$,ERRVEC   ; SET TRAP PC...

;*****
;SBTTL      T72 -- TEST THAT BUS TIME-OUT TRAPS TO THE 'RESTART' ADDRESS
;*****
T72:  BEGINTEST, 72          ;; CHECK SWITCH REGISTER OPTIONS.
;
; T11 TRAPS TO ODT 'RESTART' ON TIME-OUT. THE CODE THERE WILL
; EMULATE A TRAP THRU THE TRADITIONAL BUS ERROR VECTOR (4).
;
;*****
;SBTTL      T73 -- TEST THAT AN "ILLEGAL" INSTRUCTION TRAPS TO 4
;*****
T73:  BEGINTEST, 73          ;; CHECK SWITCH REGISTER OPTIONS.
;
; DO BOTH "JMP R" AND "JSR R,R"
; ON ERROR, R0 POINTS TO THE OFFENDER.
;
;*****

```

T73 -- TEST THAT AN "ILLEGAL" INSTRUCTION TRAPS TO 4

```

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$:    CMP      SP,#STACK-4
2553 025044 001402          BEQ      5$         ; BR IF STACK IS RIGHT.
2554 025046 104423 024750          ERROR,  T73         ;; *** STACK POINTER INCORRECT ***
2555 025052 021627 025014          5$:    CMP      (SP),#1$+2
2556 025056 001405          BEQ      6$         ; BR IF STACKED PC IS RIGHT (JMP).
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...

```

```

:*****
.SBTTL      T74 -- TEST THAT A RESERVED (UNDEFINED) INSTRUCTION TRAPS TO 10
:*****

```

```

(2)
(2)
(2) 025122 104420 000074          T74:    BEGINTEST, 74    ; CHECK SWITCH REGISTER OPTIONS.
2569 025126 012706 003776          MOV      #STACK,SP    ; STACK POINTER SETUP
2570 025132 012767 025162 152650          MOV      #2$,RESVEC   ; SET TRAP PC...
2571 025140 005067 152646          CLR      RESVEC+2     ;...AND PSW.
2572 025144 106427 000317          MTPS     #PRI6!17     ; SET CURRENT (OLD) PSW.
2573 025150 000010          RESRVD
2574 025152 000240          1$:    NOP
2575 025154 104423 025122          ERROR,  T74         ;; *** RESERVED OPCODE DIDN'T TRAP ***
2576 025160 000424          BR      6$
2577
2578 025162 106700          2$:    MFPS     R0
2579 025164 001402          BEQ      3$         ; BR IF NEW PSW IS RIGHT.
2580 025166 104423 025122          ERROR,  T74         ;; *** NEW PSW INCORRECT AFTER TRAP TO 10 ***
2581 025172 020627 003772          3$:    CMP      SP,#STACK-4
2582 025176 001402          BEQ      4$         ; BR IF SP PUSHED DOWN RIGHT.
2583 025200 104423 025122          ERROR,  T74         ;; *** STACK POINTER INCORRECT ***
2584 025204 021627 025152          4$:    CMP      (SP),#1$
2585 025210 001402          BEQ      5$         ; BR IF STACKED PC IS RIGHT.
2586 025212 104423 025122          ERROR,  T74         ;; *** INCORRECT PC SAVED ON STACK ***
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)      .SBTTL      T75 -- TEST THAT A 'BPT' INSTRUCTION TRAPS TO 14
(2)      ;*****
(2)      T75:      BEGINTEST, 75          ;; CHECK SWITCH REGISTER OPTIONS.
2594      025240 104420 000075          MOV      #STACK,SP          ;STACK POINTER SETUP
2595      025244 012706 003776          MOV      #2$,BPTVEC        ; SET TRAP PC...
2596      025250 012767 025300 152536  CLR      BPTVEC+2          ;...AND PSW.
2597      025256 005067 152534          MTPS    #PRI6!17          ; SET CURRENT (OLD) PSW.
2598      025262 106427 000317          BPT                                ; BPT -- SHOULD TRAP.
2599      025266 000003
2600      025270 000240          1$:      NOP
2601      025272 104423 025240          ERROR,  T75          ;; *** BPT DIDN'T TRAP ***
2602      025276 000424          BR      6$
2603      025300 106700          2$:      MFPS    R0
2604      025302 001402          BEQ     3$          ; BR IF NEW PSW IS RIGHT.
2605      025304 104423 025240          ERROR,  T75          ;; *** PSW INCORRECT AFTER BPT TRAP TO 14 ***
2606      025310 020627 003772          3$:      CMP     SP,#STACK-4
2607      025314 001402          BEQ     4$          ; BR IF STACK IS RIGHT.
2608      025316 104423 025240          ERROR,  T75          ;; *** STACK POINTER INCORRECT ***
2609      025322 021627 025270          4$:      CMP     (SP),#1$
2610      025326 001402          BEQ     5$          ; BR IF STACKED PC IS RIGHT.
2611      025330 104423 025240          ERROR,  T75          ;; *** INCORRECT PC SAVED ON STACK ***
2612      025334 026627 000002 000317 5$:      CMP     2(SP),#PRI6!17
2613      025342 001402          BEQ     6$          ; BR IF STACKED (OLD) PSW IS RIGHT.
2614      025344 104423 025240          ERROR,  T75          ;; *** INCORRECT PSW SAVED ON STACK ***
2615
2616      025350 012767 000016 152436 6$:      MOV     #BPTVEC+2,BPTVEC ; RESET VECTOR.
2617
2618      ;*****
(2)      .SBTTL      T76 -- TEST THAT AN "IOT" INSTRUCTION TRAPS TO 20
(2)      ;*****
(2)      T76:      BEGINTEST, 76          ;; CHECK SWITCH REGISTER OPTIONS.
2619      025356 104420 000076          MOV      #STACK,SP          ;STACK POINTER SETUP
2620      025362 012706 003776          MOV      #2$,IOTVEC        ; SET TRAP PC...
2621      025366 012767 025416 152424  CLR      IOTVEC+2          ;...AND PSW.
2622      025374 005067 152422          MTPS    #PRI6!17          ; SET CURRENT (OLD) PSW.
2623      025400 106427 000317          IOT                                ; IOT -- SHOULD TRAP.
2624      025404 000004
2625      025406 000240          1$:      NOP
2626      025410 104423 025356          ERROR,  T76          ;; *** IOT DID NOT TRAP ***
2627      025414 000424          BR      6$
2628      025416 106700          2$:      MFPS    R0
2629      025420 001402          BEQ     3$          ; BR IF NEW PSW IS RIGHT.
2630      025422 104423 025356          ERROR,  T76          ;; *** PSW INCORRECT AFTER IOT TRAP TO 20 ***
2631      025426 020627 003772          3$:      CMP     SP,#STACK-4
2632      025432 001402          BEQ     4$          ; BR IS STACK IS RIGHT.
2633      025434 104423 025356          ERROR,  T76          ;; *** STACK POINTER INCORRECT ***
2634      025440 021627 025406          4$:      CMP     (SP),#1$
2635      025444 001402          BEQ     5$          ; BR IF STACKED PC IS RIGHT.
2636      025446 104423 025356          ERROR,  T76          ;; *** INCORRECT PC SAVED ON STACK ***
2637      025452 026627 000002 000317 5$:      CMP     2(SP),#PRI6!17
2638      025460 001402          BEQ     6$          ; BR IF STACKED (OLD) PSW IS RIGHT.
2639      025462 104423 025356          ERROR,  T76          ;; *** INCORRECT PSW SAVED ON STACK ***
2640
2641      025466 012767 000022 152324 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 1\$:
2665 025554 005014
2666 025556 104423 025474
2667 025562 000426
2668
2669 025564 106700 2\$:
2670 025566 005014
2671 025570 005700
2672 025572 001402
2673 025574 104423 025474
2674 025600 020627 003772 3\$:
2675 025604 001402
2676 025606 104423 025474
2677 025612 021627 025552 4\$:
2678 025616 001402
2679 025620 104423 025474
2680 025624 026627 000002 000300 5\$:
2681 025632 001402
2682 025634 104423 025474
2683
2684 025640 012767 000026 152156 6\$:
2685 025646 7\$:
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 EXERCISOR 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$,PWRVEC   ;; SET POWER TRAP PC...
      CLR      PWRVEC+2    ;; ...AND PSW.
      MTPS    #PRI6        ;; SET CURRENT (OLD) PSW.
      MOV      #140000,R0  ;; IF LOOPING, WE HAVE TO WAIT LONG...
      SOB     R0,,         ;; ...ENOUGH (==250MSEC) TO INSURE THE...
                          ;; ...PREVIOUS BPOK CYCLE WAS FINISHED...
                          ;; ... (IN QBX) BEFORE TRYING ANOTHER.
      MOV      #5,R0       ;; OK, SET A 25 USEC TIMER.
      BIS     #20000,(R4)  ;; SET PWR-CYCLE REQUEST...
      CCC     SOB          ;; ...SHOULD SEE IT AFTER THIS...
1$:   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 ***

3$:   CMP     SP,#STACK-4
      BEQ    4$          ;; BR IS STACK IS RIGHT.
      ERROR, T77         ;; *** STACK POINTER INCORRECT ***

4$:   CMP     (SP),#1$
      BEQ    5$          ;; BR IF STACKED PC IS RIGHT.
      ERROR, T77         ;; *** INCORRECT PC SAVED ON STACK ***

5$:   CMP     2(SP),#PRI6
      BEQ    6$          ;; 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
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$,TRTVEC      ; SET TRAP PC...
2754 026120 005067 151672          CLR      TRTVEC+2        ; ...AND NEW PSW.
2755 026124 012746 000020          MOV      #20,-(SP)       ; PUSH OLD PSW WITH T BIT...
2756 026130 012746 026136          MOV      #.+6,-(SP)      ; ...AND PC.
2757 026134 000002          RTI
2758 026136 000240          1$:  NOP                    ; ...SHOULD SET T BIT...
2759 026140 000240          2$:  NOP                    ; ...AND TRAP BEFORE THIS EXECUTES.
2760 026142 104423 026102          ERROR,  T102            ;; *** TRACE BIT DID NOT TRAP ***
2761 026146 000417          BR      6$
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
;
; NOW SET 'T' VIA RTT -- SHOULD ALLOW NEXT INSTRUCTION, THEN TRAP.
;
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
2779 026226 000240          NOP
2780 026230 000240          7$:  NOP                    ; ...SHOULD SET T BIT...
2781 026232 104423 026102          ERROR,  T102            ; ...EXECUTE THIS...
2782 026236 000417          BR      11$             ; ...AND TRAP BEFORE THIS.
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 104423 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
2794
2795
(2)
(2)
;*****
.SBTTL      T103 -- TEST THAT TRACE TRAP ON A TRAP IS INHIBITED
;*****
```

T103 -- TEST THAT TRACE TRAP ON A TRAP IS INHIBITED

```

(2) 026276 104420 000103
2796 026302 012706 003776
2797 026306 012767 026360 151500
2798 026314 005067 151476
2799 026320 012767 026370 151472
2800 026326 005067 151470
2801 026332 012746 000020
2802 026336 012746 026344
2803 026342 000006
2804 026344 000004
2805 026346 000240
2806 026350 000240
2807 026352 104423 026276
2808 026356 000404

```

```

T103: BEGINTEST, 103      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV #STACK,SP      ; SET TRACE VECTOR...
      MOV #1$,TRTVEC     ; ...AND PSW.
      CLR TRTVEC+2       ; SET IOT VECTOR...
      MOV #2$,IOTVEC     ; ...AND PSW.
      CLR IOTVEC+2       ; PUSH PSW WITH T BIT...
      MOV #20,-(SP)      ; ...AND PC.
      MOV #.+6,-(SP)     ; SET T BIT...
      RTT                ; ...AND XCT ANOTHER TRAP...
      IOT                ; THE IOT SHOULD WIN...
      NOP                ; ...AND INHIBIT THE TRACE.
      NOP                ; ** NEITHER TRAP OCCURRED ??? **
      ERROR, T103
      BR 2$

```

```

2809
2810 026360 000240
2811 026362 000240
2812 026364 104423 026276

```

```

1$:  NOP      ; IOT SHOULD HAVE CLEARED T...
     NOP      ; ...AND INHIBITED THE TRACE.
     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

```

```

;*****
.SBTTL      T104 -- TEST THAT RESET HAS NO EFFECT ON A TRACE TRAP
;*****

```

```

(2) 026376 104420 000104
2817 026402 012706 003776
2818 026406 012767 026446 151400
2819 026414 005067 151376
2820 026420 012746 000020
2821 026424 012746 026432
2822 026430 000006
2823 026432 000005
2824 026434 000240
2825 026436 000240
2826 026440 000240
2827 026442 104423 026376

```

```

T104: BEGINTEST, 104      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV #STACK,SP      ; SET STACK
      MOV #1$,TRTVEC     ; SET TRACE VECTOR.
      CLR TRTVEC+2
      MOV #20,-(SP)      ; PUSH PSW WITH T BIT...
      MOV #.+6,-(SP)     ; ...AND PC.
      RTT                ; SET 'T' BIT...
      RESET              ; ...AND XCT BUS RESET.
      NOP                ; SHOULD HAVE NO EFFECT...
      NOP                ; ... I.E. TRAP SHOULD STILL OCCUR...
      NOP                ; ...AFTER THE RESET INSTRUCTION.
      ERROR, T104       ; ** BUS-RESET DISABLED OR INHIBITED TRACE TRAP **

```

```

2828
2829 026446 106427 000000
2830 026452 012767 000016 151334

```

```

1$:  MTPS #PRIO          ; WE'RE ALL DONE.
     MOV #TRTVEC+2,TRTVEC ; RESET TRACE VECTOR.

```

```

2831
2832

```

```

;*****
.SBTTL      T105 -- TEST THAT ODD ADDRESS TRAPS ARE NOT IMPLEMENTED
;*****

```

```

(2) 026460 104420 000105
2833 026464 012706 003776
2834 026470 012767 026514 151306
2835 026476 000167 000007
2836 026502 000240
2837 026504 104423 026460
2838 026510 012707 026520

```

```

T105: BEGINTEST, 105      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV #STACK,SP      ; SET STACK.
      MOV #2$,ERRVEC     ; IN CASE ODD ADDRESS TRAPS.
      JMP 1$+1
      NOP                ; IT DIDN'T DO ANYTHING AT ALL !!
      ERROR, T105       ; ** JUMP TO AN ODD ADDRESS DID NOTHING **
1$:  MOV #3$,PC          ; ADJUST ODD PC AND EXIT.

```

```

2839 026514 104423 026460
(2) 026520 012767 000006 151256

```

```

2$:  ERROR, T105        ; ** JMP TO AN ODD ADDRESS TIMED OUT **
3$:  MOV #ERRVEC+2,ERRVEC ; RESET THE TRAP CATCHER.

```

```

2841
2842
(2)

```

```

;*****
.SBTTL      T106 -- TEST THE CPU TYPE INSTRUCTION (MFPT)
;*****

```

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

```

```

*****
T106: BEGINTEST, 106      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      #STACK,SP  ; SET STACK.
      MOV      #2$,RESVEC ; SET TRAP CATCHER.
      CLR      R0
      CCC
      MFPT     ; CLEAR CC BITS.
      MFPS     R1       ; MOVE FROM PROC TYPE (TO R0).
      BIT      #17,R1
      BEQ      1$
      ERROR,   T106     ;; *** CC BITS CHANGED ON MFPT INSTRUCTION ***
1$:   CMP      R0,#KXT11
      BEQ      3$
      ERROR,   T106     ;; *** RETURNED VALUE INCORRECT ON MFPT ***
      BR       3$
2$:   ERROR,   T106     ;; *** MFPT TRAPPED TO 10 ***
3$:   MOV      #RESVEC+2,RESVEC
*****

```

.SBTTL T107 -- TEST EXTERNAL Q-BUS INTERRUPT (REQUIRES Q-BUS EXERCISER)

```

(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 000240
2876 026704 000403
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

```

```

*****
T107: BEGINTEST, 107      ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      #STACK,SP  ; Q-BUS EXERCISOR AVAILABLE ??
      CALL     QBXIN      ; IF NOT, BYPASS TEST.
      BEQ      4$         ; YES, CSR1 POINTER => R4 AND FALL THRU.
      MOV      R0,R4
      ;
      ; 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$(R3)    ; SET Q-BUS VECTOR...
      MOV      #PRI4,2(R3) ; ...AND PRIORITY.
      MTPS     #PRI4      ; RAISE CPU TO Q-BUS LEVEL.
      MOV      #1,2(R4)   ; SET QBX GO (CLEAR DONE)...
      BIS      #20,2(R4)  ; ...ENABLE INT 4 ON DONE...
      BIC      #1,2(R4)   ; ...AND CLEAR GO (SET DONE).
      240
      BR       2$        ; INTERRUPT SHOULD BE HELD OFF.
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.

```

T110 -- TEST THE 'BEVNT' INTERRUPT (REQUIRES Q-BUS EXERCISER)

```

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$:
(2) 027006 104423 026742      ERROR, T110          ;; *** BEVNT INTERRUPT LEVEL INCORRECT ***
2903 027012 000410              BR    3$
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
  
```

SECTION 3 -- LOCAL 2K RAM TESTS.

```
2915 .SBTTL SECTION 3 -- LOCAL 2K RAM TESTS.
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 :*****
(2) .SBTTL T111 -- LOCAL 2K RAM ADDRESS TEST
(2) :*****
(2) 027074 104420 000111 T111: BEGINTEST, 111 ;; CHECK SWITCH REGISTER OPTIONS.
2922 :
2923 : WRITE EACH RAM LOCATION WITH ITS OWN ADDRESS, READ BACK AND VERIFY.
2924 : REPEAT A SECOND TIME USING BYTE-SWAPPED ADDRESS AS DATA.
2925 :
2926 027100 012706 003776 MOV #STACK,SP
2927 027104 012767 027206 150672 MOV #7$,ERRVEC ; SET TIME-OUT VECTOR.
2928 027112 016700 151670 MOV TEMP,R0 ; GET RAM ADDRESS...
2929 027116 016701 151666 MOV TEMP1,R1 ; ...AND SIZE.
2930 027122 010146 MOV R1,-(SP) ; SAVE A COPY.
2931
2932 027124 010010 1$: MOV R0,(R0) ; LOAD ADDRESSES (FROM BOTTOM UP).
2933 027126 005720 TST (R0)+ ; BUMP
2934 027130 077103 SOB R1,1$ ; LOOP 'TIL DONE.
2935
2936 027132 011602 MOV (SP),R2 ; RECOVER SIZE K.
2937 027134 014001 2$: MOV -(R0),R1 ; READ 'EM BACK (FROM TOP DOWN).
2938 027136 020100 CMP R1,R0 ; DATA = ADDRESS ??
2939 027140 001402 BEQ 3$ ; BR IF SO.
2940 027142 104423 027074 ERROR, T111 ;: *** RAM DATA INCORRECT AT ADDRESS IN R0 ***
2941 027146 077206 3$: SOB R2,2$ ; LOOP 'TIL DONE.
2942
2943 027150 011602 MOV (SP),R2 ; RECOVER SIZE K.
2944 027152 010001 4$: MOV R0,R1 ; ADDRESS => R1...
2945 027154 000301 SWAB R1 ; ...SWAP BYTES...
2946 027156 010120 MOV R1,(R0)+ ; ...AND LOAD SWAPPED ADDRESSES.
2947 027160 077204 SOB R2,4$
2948
2949 027162 011603 MOV (SP),R3 ; RECOVER SIZE K.
2950 027164 014001 5$: MOV -(R0),R1 ; GET DATA.
2951 027166 010002 MOV R0,R2
2952 027170 000302 SWAB R2
2953 027172 020102 CMP R1,R2 ; DATA = SWAPPED ADDRESS ??
2954 027174 001402 BEQ 6$ ; BR IF SO.
2955 027176 104423 027074 ERROR, T111 ;: *** RAM DATA (SWAPPED) INCORRECT AT ADDRESS IN R0 ***
2956 027202 077310 6$: SOB R3,5$
2957 027204 000402 BR 8$ ; DONE, EXIT.
2958
2959 027206 7$:
(2) 027206 104423 027074 ERROR, T111 ;: *** BUS TIME-OUT -- RAM ADDRESS IN R0 ***
2960 027212 8$:
2961
2962 :*****
(2) .SBTTL T112 -- LOCAL 2K RAM DATA TEST
(2) :*****
(2) 027212 104420 000112 T112: BEGINTEST, 112 ;: CHECK SWITCH REGISTER OPTIONS.
2963 :
```

T112 -- LOCAL 2K RAM DATA TEST

```

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$:  MOV      (SP),(R0)+      : FILL RAM WITH BACKGROUND DATA.
2980 027252 077102      SOB
2981
2982 027254 016700 151526      MOV      TEMP,R0        : RESET ADDRESS...
2983 027260 016701 151524      MOV      TEMP1,R1       : ...AND WORD COUNT...
2984 027264 011004 3$:  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$:  MOV      #1,R2          : *** INITIAL BACKGROUND INCORRECT AT ADDRESS IN R0 ***
2989 027302 011603      MOV      (SP),R3        : COPY CURRENT BACKGROUND TO R3.
2990 027304 074203      XOR     R2,R3          : SET OR CLEAR BIT 0.
2991 027306 012702 000020      MOV     #16.,R2        : SET BIT COUNTER.
2992 027312 000402      SKP2
2993 027314 006303 5$:  ASL     R3              : ROTATE DATA...
2994 027316 005503      ADC     R3             : ...BIT 15 => BIT 0.
2995 027320 010310 6$:  MOV     R3,(R0)        : WRITE FLOATING DATA...
2996 027322 011004      MOV     (R0),R4       : ...AND READ IT BACK.
2997 027324 020304      CMP     R3,R4
2998 027326 001402      BEQ     7$
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
3004 027342 016700 151440      MOV     TEMP,R0        : ...AND LOOP FOR ALL WORDS.
3005 027346 016701 151436      MOV     TEMP1,R1       : NOW, RESET POINTERS...
3006 027352 011004 8$:  MOV     (R0),R4        : ...AND VERIFY THAT RESTORED...
3007 027354 020416      CMP     R4,(SP)       : ...BACKGROUND IS STILL CORRECT.
3008 027356 001402      BEQ     9$
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
3013 027370 005726      TST     (SP)+         : ...AND LOOP 'TIL DONE.
3014 027372 001720      BEQ     1$            : FINALLY, IF BACKGROUND IS 0...
3015 027374 000402      BR     11$           : ...LOOP ONCE MORE...
3016
3017 027376 104423 027212 10$:  ERROR,  T112          : ...OTHERWISE, WE'RE ALL DONE.
3018 027402 11$:

```



```

3019
3020          :*****
          :SBTTL      T113 -- OPTIONAL DMA TEST (REQUIRES Q-BUS EXERCISER)
          :*****
          :T113:   BEGINTEST, 113          ;; CHECK SWITCH REGISTER OPTIONS.
3021 027402 104420 000113          :   MOV      #STACK,SP
3022 027412 004767 005644          :   CALL     QBXIN          ; Q-BUS EXERCISER AVAILABLE ??
3023 027416 001504          :   BEQ      10$          ; IF NOT, BYPASS TEST.
3024 027420 010004          :   MOV      R0,R4        ; YES, CSRI POINTER => R4 AND FALL THRU.
3025
3026          :
3027          :   TRANSFER EACH RAM WORD TO THE QBX AND BACK.
3028          :   VERIFY THAT RETURNED DATA MATCHES THAT WHICH WAS SENT.
3029          :
3029 027422 012767 027624 150354          :   MOV      #9$,ERRVEC    ; SET TRAP CATCHER.
3030 027430 016701 151352          :   MOV      TEMP,R1      ; RAM ADDRESS POINTER...
3031 027434 016702 151350          :   MOV      TEMP1,R2    ; ...AND WORD COUNT.
3032
3033          :
3033 027440 010111          2$:   MOV      R1,(R1)        ; SET 1ST/NEXT RAM WORD.
3034 027442 012714 140407          :   MOV      #140407,(R4) ; DATI (MEM => QBX) IN 'HOG' MODE.
3035 027446 010164 000004          :   MOV      R1,4(R4)     ; FROM RAM (R1) TO DATA REGISTER...
3036 027452 012764 177774 000006          :   MOV      #-4,6(R4)   ; ...4 TIMES.
3037 027460 012764 000401 000002          :   MOV      #401,2(R4)  ; GO, INHIBIT BA+.
3038 027466 005000          :   CLR      R0
3039 027470 105764 000002          3$:   TSTB     2(R4)
3040 027474 100401          :   BMI      33$
3041 027476 077004          :   SOB     R0,3$
3042 027500 032764 040000 000002          33$:  BIT      #40000,2(R4) ; ...AND CHECK FOR NXM ERROR.
3043 027506 001402          :   BEQ     4$
3044 027510 104423 027440          :   ERROR,  2$          ; BR IF OK.
3045 027514 026401 000010          4$:   CMP      10(R4),R1   ; *** NXM ERROR IN QBX ON DATI (MEM TO QBX) ***
3046 027520 001402          :   BEQ     5$          ; DATA REGISTER CORRECT ??
3047 027522 104423 027440          :   ERROR,  2$          ;: *** DATA INCORRECT IN QBX ON DATI (MEM TO QBX) ***
3048
3049 027526 012714 000601          5$:   MOV      #000601,(R4) ; DATO (QBX => MEM) IN 'HOG' MODE.
3050 027532 012764 001012 000004          :   MOV      #TEMP2,4(R4) ; FROM DATA REGISTER TO TEMP2...
3051 027540 012764 177774 000006          :   MOV      #-4,6(R4)   ; ...4 TIMES.
3052 027546 012764 000401 000002          :   MOV      #401,2(R4)  ; GO, INHIBIT BA+.
3053 027554 005000          :   CLR      R0
3054 027556 105764 000002          6$:   TSTB     2(R4)
3055 027562 100401          :   BMI      66$
3056 027564 077004          :   SOB     R0,6$
3057 027566 032764 040000 000002          66$:  BIT      #40000,2(R4) ; ...AND CHECK FOR NXM.
3058 027574 001402          :   BEQ     7$
3059 027576 104423 027526          :   ERROR,  5$          ; BR IF OK.
3060 027602 026701 151204          7$:   CMP      TEMP2,R1   ; *** NXM ERROR IN QBX ON DATO (QBX TO MEM) ***
3061 027606 001402          :   BEQ     8$          ; RETURNED DATA CORRECT ??
3062 027610 104423 027526          :   ERROR,  5$          ;: *** DATA INCORRECT IN TEMP2 ON DATO (QBX TO MEM) ***
3063
3064 027614 005721          8$:   TST      (R1)+
3065 027616 077270          :   SOB     R2,2$
3066 027620 005014          :   CLR     (R4)
3067 027622 000402          :   BR      10$
3068
3069 027624          9$:   ERROR,  T113
          :   ;; *** BUS TIME-OUT -- QBX (R4) OR RAM (R1) ***
3070

```

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

```
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
```

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

```

3076
3077
3078 027642 012767 036263 006330
3079 027650 012767 170000 151130
3080 027656 012767 002000 151124
3081
3082
(2)
(2)
(2) 027664 104420 000114
3083
3084
3085
3086
3087
3088
3089 027670 012706 003776
3090 027674 012767 027746 150102
3091 027702 016700 151100
3092 027706 016701 151076
3093
3094 027712 012702 123456
3095 027716 010210
3096 027720 021002
3097 027722 001006
3098 027724 005102
3099 027726 010210
3100 027730 021002
3101 027732 001002
3102 027734 104423 027712
3103
3104 027740 005720
3105 027742 077115
3106 027744 000402
3107
3108 027746
(2) 027746 104423 027664
3109
3110 027752 012767 000006 150024
3111 027760 004767 004744
3112
3113

```

```

S4:  MOV    #SEG4,SEGN    ; SEGMENT 4 TEXT.
      MOV    #ROMADR,TEMP  ; SAVE ROM ADDRESS...
      MOV    #ROMSIZ,TEMP1 ; ...AND SIZE.

:*****
.SBTTL      T114 -- LOCAL PROM/RAM ADDRESS TEST
:*****
T114:  BEGINTEST, 114      ;; CHECK SWITCH REGISTER OPTIONS.

:
: PROM/RAM SOCKETS CONTAIN THE 1KW MACRO ODT PROM SET.
: ATTEMPT TO WRITE (123456) INTO EACH LOCATION.
: IF AN ADDRESS HAPPENS TO CONTAIN THAT VALUE, THEN WRITE AGAIN
: WITH THE COMPLIMENT OF THAT VALUE.
:
      MOV    #STACK,SP
      MOV    #3$,ERRVEC    ; SET TIME-OUT VECTOR.
      MOV    TEMP,R0       ; SET ADDRESS...
      MOV    TEMP1,R1      ; ...AND SIZE.

1$:   MOV    #123456,R2
      MOV    R2,(R0)       ; TRY TO WRITE DATA (SHOULDN'T TRAP).
      CMP    (R0),R2      ; DID IT WRITE ??
      BNE    2$           ; BR IF NOT.
      COM    R2           ; MAYBE...
      MOV    R2,(R0)      ; ...TRY COMPLIMENT.
      CMP    (R0),R2      ; HOW NOW ??
      BNE    2$           ; BR IF OK.
      ERROR, 1$          ;; *** WRITE TO ROM ALTERED DATA ????? ***

2$:   TST    (R0)+
      SOB   R1,1$
      BR    4$           ; ...AND PROCEED.

3$:   ERROR,  T114       ;; *** BUS TIME-OUT -- ROM ADDRESS IN R0 ***

4$:   MOV    #ERRVEC+2,ERRVEC
      CALL  ENDSEG       ; CHECK FOR LOOP-IN-SEGMENT...
                          ; ...RETURN AND FALL THRU IF NOT.

.SBTTL

```

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

DLART BIT ASSIGNMENTS.

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

XMTBRK= 001
XMTIE= 100
XMTRDY= 200

RCVIE= 000100
RCVDUN= 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).

3134 027764 012767 036301 006206
3135 027772 012700 177560
3136 027776 012701 000060
3137 030002 012702 000200
3138 030006 012703 030152
3139 030012 000413
3140

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.

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

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

(2)
(2)
(2) 030042 104420 000115
3151 030046 012706 003776
3152 030052 010367 147726
3153 030056 010067 150762
3154 030062 005720
3155 030064 010067 150756
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

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.

T115 -- TEST SERIAL LINE REGISTER ADDRESSES

```

3168 030140 052700 000002
3169 030144 010067 150670
3170 030150 000432
3171
3172
3173
3174 030152
(1) 030152 104423 027764
3175 030156 000167 003270
3176 030162
(1) 030162 104423 030014
3177 030166 000167 003260
3178 030172
(1) 030172 104423 030236
3179 030176 000167 003250
3180 030202
(1) 030202 104423 030236
3181 030206 000167 003240
3182
3183
3184
3185 030212 015 012
3186 030214 001 002 004
3187 030224 376 375 373
3188 030234 000000
3189
3190 030236
3191
(2)
(2)
(2) 030236 104420 000116
3192
3193
3194
3195
3196 030242 106427 000300 150574
3197 030246 012777 000177 150570
3198 030254 012777 000377 150554
3199 030262 012777 000100
3200 030270 000240 000240
3201 030274 000005
3202 030276 000240 000240
3203 030302 017700 150536
3204 030306 017701 150534
3205 030312 017702 150532
3206 030316 017703 150530
3207 030322 016777 150512 150520
3208 030330 005077 150510
3209 030334 032700 000100
3210 030340 001402
3211 030342 104423 030236
3212 030346 032702 000105 1$:
3213 030352 001402
3214 030354 104423 030236
3215 030360 120227 000272 2$:
3216 030364 001402

```

```

BIS #2,R0 ; INSURE 'PBRE' IS SET...
MOV R0,DFPBR ;...AND SAVE AS THE DEFAULT BAUD RATE.
BR DLTSTS ; START 'EM UP.

; COME HERE IF BUS TIME-OUT OR UNEXPECTED INTERRUPT.
BADU1:
  ERROR, S5 ;: *** BUS TIME-OUT ON SLU1 ADDRESS ***
  JMP DLDUN
BADU2:
  ERROR, S6 ;: *** BUS TIME-OUT ON SLU2 ADDRESS ***
  JMP DLDUN
BADRI:
  ERROR, DLTSTS ;: *** UNEXPECTED RCVR INTERRUPT. ***
  JMP DLDUN
BADTI:
  ERROR, DLTSTS ;: *** UNEXPECTED XMTR INTERRUPT. ***
  JMP DLDUN

; FLOATING 1 AND 0 DATA TABLE, USED IN VARIOUS TESTS.
SYNC: .BYTE 015, 012 ; SYNC PAIR.
FLT1: .BYTE 001, 002, 004, 010, 020, 040, 100, 200
FLT0: .BYTE 376, 375, 373, 367, 357, 337, 277, 177
      .WORD 0 ; TERMINATOR.

DLTSTS:
;*****
;SBTTL T116 -- TEST THAT BUS-RESET CLEARS THE RIGHT BITS
;*****
T116: BEGINTEST, 116 ;: CHECK SWITCH REGISTER OPTIONS.

; SET ALL WRITABLE BITS AND RESET. CHECK THAT ONLY
; RCV-IE, XMIT-IE, MAINT, AND XMIT-BRK GET CLEARED.

MTPS #PRI6 ; RAISE CPU
MOV #177,@XCSR ; SET WRITABLE BITS IN XCSR...
MOV #377,@XBUF ;...AND XBUF...
MOV #100,@RCSR ;...AND RCSR.
      240,240 ; IN CASE WE NEED A DELAY.
RESET
      240,240
MOV @RCSR,R0 ; GET RCSR...
MOV @RBUF,R1 ;...RBUF (DUMMY READ)...
MOV @XCSR,R2 ;...XCSR...
MOV @XBUF,R3 ;...AND XBUF.
MOV DFPBR,@XCSR ; RESTORE DEFAULT XMIT...
CLR @RCSR ;...AND RCVR STATUS.
BIT #100,R0 ; NOW, DID RCV-IE CLEAR ??
BEQ 1$
ERROR, T116 ;: *** BUS-RESET DIDN'T CLEAR RCV-IE BIT IN RCSR ***
BIT #105,R2 ; DID XIE, MAINT AND XBRK CLEAR ??
BEQ 2$
ERROR, T116 ;: *** BUS-RESET DIDN'T CLEAR XIE, MAINT, AND/OR XBRK **
CMPB R2,#272 ; ALL OTHERS SET (INCLUDING XMTRDY) ??
BEQ 3$

```

PC 30366 = BUS-RESET CLEARED WRONG BITS IN XCSR.

```

3217 030366 104423 030236
3218 030372 120327 177777
3219 030376 001402
3220 030400 104423 030236
3221 030404 106427 000000
3222
3223
(2)
(2)
(2) 030410 104420 000117
3224 030414 017700 150424
3225 030420 017701 150422
3226 030424 017702 150420
3227 030430 017703 150416
3228 030434 032700 173477
3229 030440 001402
3230 030442 104423 030410
3231 030446 032701 013400
3232 030452 001402
3233 030454 104423 030410
3234 030460 032702 177400
3235 030464 001402
3236 030466 104423 030410
3237 030472 032703 177400
3238 030476 001402
3239 030500 104423 030410
3240 030504 105777 150340
3241 030510 100402
3242 030512 104423 030410
3243 030516 030516
3244 030516 000402
3245 030520 000167 002726
3246

```

```

3$: ERROR, T116 ;: *** BUS-RESET CLEARED WRONG BITS IN XCSR ***
CMPB R3,#-1 ;: ALL BITS STILL IN XBUF ??
BEQ 4$
ERROR, T116 ;: *** BUS-RESET CLEARED BITS IN XBUF ***
4$: MTPS #PRIO ;: LOWER THE CPU.

:*****
.SBTTL T117 -- TEST THAT ALL UNDEFINED BITS ARE ZERO
:*****
T117: BEGINTEST, 117 ;: CHECK SWITCH REGISTER OPTIONS.
MOV @RCSR,R0 ;: GET REGISTERS AGAIN.
MOV @RBUF,R1
MOV @XCSR,R2
MOV @XBUF,R3
BIT #173477,R0 ;: RCSR, UNDEFINED BITS SET ??
BEQ 1$
ERROR, T117 ;: *** UNDEFINED BITS SET IN RCVR STATUS ***
1$: BIT #13400,R1 ;: RBUF, UNDEFINED BITS SET ??
BEQ 2$
ERROR, T117 ;: *** UNDEFINED BITS SET IN RCVR DATA BUFFER ***
2$: BIT #177400,R2 ;: XCSR, UNDEFINED BITS SET ??
BEQ 3$
ERROR, T117 ;: *** UNDEFINED BITS SET IN XMTR STATUS ***
3$: BIT #177400,R3 ;: XBUF, UNDEFINED BITS SET ??
BEQ 4$
ERROR, T117 ;: *** UNDEFINED BITS SET IN XMTR DATA BUFFER ***
4$: TSTB @XCSR ;: XMTR READY ??
BMI 5$
ERROR, T117 ;: *** XMITTER NOT READY ***
.PRINT ;: *** DEBUG, NOP ***
5$: SKP2
JMP DLDUN
.SBTTL

```

TRANSMITTERS

3248
3249 030524
3250
(2)
(2)
(2) 030524 104420 000120
3251 030530 012700 030214
3252 030534 112001
3253 030536 010177 150310
3254 030542 017702 150304
3255 030546 120102
3256 030550 001402
3257 030552 104423 030536
3258 030556 112001
3259 030560 001366
3260
3261
(2)
(2)
(2) 030562 104420 000121
3262 030566 106427 000340
3263 030572 052777 000001 150250
3264 030600 017700 150244
3265 030604 042777 000001 150236
3266 030612 017701 150232
3267 030616 016777 150216 150224
3268 030624 106427 000000
3269 030630 032700 000001
3270 030634 001002
3271 030636 104423 030562
3272 030642 032701 000001
3273 030646 001402
3274 030650 104423 030562
3275 030654
3276
3277
(2)
(2)
(2) 030654 104420 000122
3278 030660 052777 000004 150162
3279 030666 017700 150156
3280 030672 042777 000004 150150
3281 030700 017701 150144
3282 030704 016777 150130 150136
3283 030712 032700 000004
3284 030716 001002
3285 030720 104423 030654
3286 030724 032701 000004
3287 030730 001402
3288 030732 104423 030654
3289 030736
3290
3291
(2)
(2)
(2) 030736 104420 000123

```
.SBTTL TRANSMITTERS
XMTRS: ; TRANSMITTER TEST SECTION.
:*****
.SBTTL T120 -- FLOAT 1 AND 0 THRU THE XMIT DATA BUFFER
:*****
T120: BEGINTEST, 120 ;: CHECK SWITCH REGISTER OPTIONS.
      MOV #FLT1,R0 ;: TABLE POINTER.
      MOVB (R0)+,R1 ;: GET 1ST CHARACTER.
1$: MOV R1,@XBUF ;: LOAD DATA INTO 'XMIT DATA BUF'...
     MOV @XBUF,R2 ;: ...AND READ IT BACK.
     CMPB R1,R2
     BEQ 2$
     ERROR, 1$ ;: *** FLOATING 1 OR 0 INCORRECT IN XBUF ***
2$: MOVB (R0)+,R1 ;: GET NEXT CHAR...
     BNE 1$ ;: ...AND LOOP 'TIL DONE.
:*****
.SBTTL T121 -- TEST THAT 'XMIT-BRK' CAN BE SET AND CLEARED
:*****
T121: BEGINTEST, 121 ;: CHECK SWITCH REGISTER OPTIONS.
      MTPS #PRI7 ;: RAISE CPU.
      BIS #1,@XCSR ;: SET 'XMIT-BRK' BIT...
      MOV @XCSR,R0 ;: ...AND SAVE IN R0.
      BIC #1,@XCSR ;: CLEAR IT...
      MOV @XCSR,R1 ;: ...AND SAVE IN R1.
      MOV DFPBR,@XCSR ;: RESTORE DEFAULT XMTR.
      MTPS #PRIO ;: AND PRIORITY.
      BIT #1,R0
      BNE 1$ ;:BR IF IT SET
      ERROR, T121 ;: *** XMIT-BRK BIT FAILED TO SET ***
1$: BIT #1,R1
     BEQ 2$ ;:BR IF IT CLEARED
     ERROR, T121 ;: *** XMIT-BRK BIT FAILED TO CLEAR ***
2$:
:*****
.SBTTL T122 -- TEST THAT 'XMIT-MAINT' CAN BE SET AND CLEARED
:*****
T122: BEGINTEST, 122 ;: CHECK SWITCH REGISTER OPTIONS.
      BIS #4,@XCSR ;: SET 'XMIT-MAINT' BIT...
      MOV @XCSR,R0 ;: ...AND SAVE IN R0.
      BIC #4,@XCSR ;: CLEAR IT...
      MOV @XCSR,R1 ;: ...AND SAVE IN R1.
      MOV DFPBR,@XCSR
      BIT #4,R0
      BNE 1$ ;:BR IF IT SET
      ERROR, T122 ;: *** XMIT-MAINT BIT FAILED TO SET ***
1$: BIT #4,R1
     BEQ 2$ ;:BR IF IT CLEARED
     ERROR, T122 ;: *** XMIT-MAINT BIT FAILED TO CLEAR ***
2$:
:*****
.SBTTL T123 -- TEST THAT 'XMIT-PBRE' CAN BE SET AND CLEARED
:*****
T123: BEGINTEST, 123 ;: CHECK SWITCH REGISTER OPTIONS.
```

```

3292 030742 052777 000002 150100
3293 030750 017700 150074
3294 030754 042777 000002 150066
3295 030762 017701 150062
3296 030766 016777 150046 150054
3297 030774 032700 000002
3298 031000 001002
3299 031002 104423 030736
3300 031006 032701 000002
3301 031012 001402
3302 031014 104423 030736
3303 031020
3304
3305

```

```

BIS #2,@XCSR ;SET 'XMIT-PBRE' BIT...
MOV @XCSR,R0 ;...AND SAVE IN R0.
BIC #2,@XCSR ; CLEAR IT...
MOV @XCSR,R1 ;...AND SAVE IN R1.
MOV DFPBR,@XCSR
BIT #2,R0
BNE 1$ ;BR IF IT SET
ERROR, T123 ;: *** XMIT-PBRE BIT FAILED TO SET ***
1$: BIT #2,R1
BEQ 2$ ;BR IF IT CLEARED
ERROR, T123 ;: *** XMIT-PBRE BIT FAILED TO CLEAR ***
2$:

```

```

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

```

```

:*****
.SBTTL T124 -- TEST THAT 'XMIT-PBRO' CAN BE SET AND CLEARED
:*****
T124: BEGINTEST, 124 ;: CHECK SWITCH REGISTER OPTIONS.
BIS #10,@XCSR ;SET 'XMIT-PBRO' 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-PBRO BIT FAILED TO SET ***
1$: BIT #10,R1
BEQ 2$ ;BR IF IT CLEARED
ERROR, T124 ;: *** XMIT-PBRO BIT FAILED TO CLEAR ***
2$:

```

```

3320 031102 104420 000125 147734
3321 031106 052777 000020
3322 031114 017700 147730
3323 031120 042777 000020 147722
3324 031126 017701 147716
3325 031132 016777 147702 147710
3326 031140 032700 000020
3327 031144 001002
3328 031146 104423 031102
3329 031152 032701 000020
3330 031156 001402
3331 031160 104423 031102
3332 031164
3333

```

```

:*****
.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
ERROR, T125 ;: *** XMIT-PBR1 BIT FAILED TO CLEAR ***
2$:

```

```

3334 031164 104420 000126 147652
3335 031170 052777 000040
3336 031176 017700 147646
3337 031202 042777 000040 147640
3338 031210 017701 147634
3339 031214 016777 147620 147626

```

```

:*****
.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

```


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

```

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

```

```

:*****
.SBTTL      T127 -- TEST THAT 'XMIT-IE' CAN BE SET AND CLEARED
:*****

```

```

(2)
(2)
(2) 031246 104420 000127          T127: BEGINTEST, 127      ;; CHECK SWITCH REGISTER OPTIONS.
3348 031252 106427 000300          MTPS   #PRI6      ;RAISE CPU PRIORITY
3349 031256 052777 000100 147564          BIS    #100,@XCSR ;SET 'XMIT-IE' BIT...
3350 031264 017700 147560          MOV    @XCSR,R0   ;...AND SAVE IN R0.
3351 031270 042777 000100 147552          BIC    #100,@XCSR ;CLEAR IT...
3352 031276 017701 147546          MOV    @XCSR,R1   ;...AND SAVE IN R1.
3353 031302 016777 147532 147540          MOV    DFPBR,@XCSR ;RESTORE XMTR...
3354 031310 106427 000000          MTPS   #PRI0      ;...AND CPU TOO.
3355 031314 032700 000100          BIT    #100,R0
3356 031320 001002          BNE    1$      ;BR IF IT SET
3357 031322 104423 031246          ERROR, T127     ;; *** XMIT-IE BIT FAILED TO SET ***
3358 031326 032701 000100          1$:  BIT    #100,R1
3359 031332 001402          BEQ    2$      ; BR IF IT CLEARED.
3360 031334 104423 031246          ERROR, T127     ;; *** XMIT-IE BIT FAILED TO CLEAR ***
3361 031340
3362
3363

```

```

:*****
.SBTTL      T130 -- TEST THAT 'XMIT-RDY' CAN BE SET AND CLEARED
:*****

```

```

(2)
(2)
(2) 031340 104420 000130          T130: BEGINTEST, 130      ;; CHECK SWITCH REGISTER OPTIONS.
3364 031344 016777 147470 147476          MOV    DFPBR,@XCSR
3365 031352 005001          CLR    R1        ; DELAY COUNT.
3366 031354 005002          CLR    R2        ; DITTO
3367 031356 105777 147466          1$:  TSTB   @XCSR    ;WAIT FOR 'XMIT-RDY' TO SET
3368 031362 100403          BMI    2$
3369 031364 077104          SOB    R1,1$     ;KEEP TRYING.
3370 031366 104423 031340          ERROR, T130     ;; *** XMIT-RDY FAILED TO SET ***
3371 031372 012777 000000 147452          2$:  MOV    #0,@XBUF ;TRANSMIT A NULL CHARACTER
3372 031400 017700 147444          MOV    @XCSR,R0  ;READ XMIT STATUS REG
3373 031404 032700 000200          BIT    #XMTRDY,R0 ;READY SHOULD BE GONE.
3374 031410 001402          BEQ    3$      ; BR IF SO.
3375 031412 104423 031340          ERROR, T130     ;; *** XMIT-RDY BIT FAILED TO CLEAR ***
3376 031416 105777 147426          3$:  TSTB   @XCSR    ;WAIT FOR 'XMIT-RDY' TO SET
3377 031422 100403          BMI    4$
3378 031424 077204          SOB    R2,3$    ; KEEP TRYING.
3379 031426 104423 031340          ERROR, T130     ;; *** XMIT-RDY FAILED TO SET AFTER XMITTING CHARACTER *
3380 031432
3381
3382

```

```

:*****
.SBTTL      T131 -- TEST THAT WE CAN INTERRUPT ON 'XMT-RDY'
:*****

```

```

(2)
(2)
(2) 031432 104420 000131          T131: BEGINTEST, 131      ;; CHECK SWITCH REGISTER OPTIONS.
3383 031436 016777 147376 147404          MOV    DFPBR,@XCSR
3384 031444 016701 147406          MOV    XVEC,R1   ; GET VECTOR POINTER.
3385 031450 106461 000002          MTPS   2(R1)     ; RAISE CPU TO XMTR LEVEL.

```

T131 -- TEST THAT WE CAN INTERRUPT ON 'XMT-RDY'

```

3386 031454 012711 031500      MOV      #2$, (R1)      ; SET XMITTER VECTOR.
3387 031460 105777 147364      1$: 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      2$: CMP     (SP)+, (SP)+  ; IT WASN'T.
3393 031502 042777 000100 147340  BIC     #100, @XCSR    ; REMOVE 'XMIT-IE'
3394 031510 104423 031432      ERROR,  T131         ;: *** XMT-RDY INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
3395
3396 031514 012711 031542      3$: MOV     #4$, (R1)    ; CHANGE THE VECTOR.
3397 031520 106427 000000      MTPS   #PRIO        ; LOWER CPU PRIORITY
3398 031524 000240                240      ; INTERRUPT SHOULD HAVE COME IN.
3399 031526 042777 000100 147314  BIC     #100, @XCSR    ; IT DIDN'T, REMOVE 'XMIT-IE' BIT
3400 031534 104423 031432      ERROR,  T131         ;: *** XMT-RDY INTERRUPT NOT RECEIVED AT CPU LEVEL 0 ***
3401 031540 000415                BR      6$

```

... NOW, ON INTERRUPT, WE'RE HERE AT CPU LEVEL 4 (5 IF SLU2).
... LOWER CPU AGAIN AND CHECK THAT WE DON'T GET ANOTHER INTERRUPT.

```

3402
3403
3404
3405
3406 031542 022626      4$: CMP     (SP)+, (SP)+  ; CHANGE THE VECTOR.
3407 031544 012711 031560      MOV     #5$, (R1)    ; LOWER THE CPU PRIORITY AGAIN.
3408 031550 106427 000000      MTPS   #PRIO        ; SHOULDN'T GET ANOTHER...
3409 031554 000240                240      ; ...OK, EXIT.
3410 031556 000406                BR      6$
3411 031560 022626      5$: CMP     (SP)+, (SP)+  ; 2ND INTERRUPT -- BAD NEWS.
3412 031562 042777 000100 147260  BIC     #100, @XCSR    ; REMOVE 'XMIT-IE'
3413 031570 104423 031432      ERROR,  T131         ;: *** XMT INTERRUPT ACK FAILED TO REMOVE THE REQUEST **
3414
3415 031574 042777 000100 147246  6$: BIC     #100, @XCSR  ; BE SURE THE ENABLE IS REMOVED.
3416 031602 012711 030202      MOV     #BADTI, (R1) ; RESET FALSE INTR VECTOR
3417

```

.SBTTL T132 -- TEST THAT EACH XMIT BAUD RATE SETS A DIFFERENT SPEED

```

(2) 031606 104420 000132 T132: BEGINTEST, 132 ;: CHECK SWITCH REGISTER OPTIONS.
(2)
(2)
3419
3420 ; FIRST DETERMINE THE RELATIVE SPEED OF EACH BAUD RATE.
3421 ; TRANSMIT A CHARACTER AND COUNT CPU BUS TRANSACTION CYCLES UNTIL
3422 ; TRANSMISSION IS COMPLETE (XMT-RDY ASSERTS).
3423 ; SAVE THE OBSERVED CYCLE COUNT IN THE SPEED TABLE.
3424 ; IF NO FLAG BY THE TIME THE COUNT OVERFLOWS (65536. CYCLES), SAVE
3425 ; THE ZERO, WHICH IMPLIES XMT RDY NEVER SET AT THAT SPEED.
3426
3427 ; NOTE: ALL FOLLOWING RECEIVER TESTS WILL USE THE SPEED VALUES
3428 ; OBSERVED HERE AS LOOP CONTROL COUNTERS.
3429 ; NOTE:NOTE: IF THERE'S A TTY ATTACHED, YOU'LL SEE SOME GARBAGE
3430 ; CHARACTERS ON THE SCREEN.
3431

```

```

3432 031612 012700 000002      MOV     #2, R0        ; START WITH SPEED 0 (300 BAUD).
3433 031616 012701 001016      MOV     #SPDO, R1    ; TABLE POINTER.
3434 031622 005002      1$: CLR     R2          ; CLEAR THE ACCUMULATOR.
3435 031624 010077 147220      MOV     R0, @XCSR    ; SET PBR BITS.
3436 031630 012777 000000 147214  MOV     #0, @XBUF    ; XMIT A CHAR.
3437 031636 105777 147206      2$: TSTB   @XCSR         ; ON READY, CHAR HAS BEEN COPIED...
3438 031642 100375                BPL     2$           ; ...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$:   MOV    R2,(R1)+    ; OK, R2 = TX TIME IN BUS CYCLES...
3447                                     ; ...AT APPROX 3 USEC/COUNT...
3448                                     ; ...SAVE IT IN THE SPEED TABLE.
3449 031670 020067 147144      CMP    R0,DFPBR      ; CURRENT RATE = DEFAULT ???
3450 031674 001002              BNE    5$            ; 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.

```

```

: NOW THE RELATIVE SPEED AT EACH SETTING HAS BEEN DETERMINED.
: CHECK THAT THE DIFFERENTIAL OF EACH ONE TO THE NEXT LOWER
: IS 2:1 (MORE OR LESS). THIS IS A GROSS SPEED CHECK SINCE
: WE CAN'T RELY ON A REAL CLOCK FOR TIMING.

```

```

: DIFFERENTIAL IS CALCULATED BY:
: (LOWER/2)-HIGHER = 0 +/- 25.0% OF THE LOWER.

```

```

: 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$          ; MAKE ABSOLUTE IF NECESSARY.
3481 031762 005403      NEG    R3
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      .SBTTL

```

```

3495 .SBTTL RECEIVERS
3496 032012 RCVRS: ; RECEIVER TEST SECTION.
3497 ;*****
(2) .SBTTL T133 -- TEST THAT 'RCV-IE' CAN BE SET AND CLEARED
(2) ;*****
(2) 032012 104420 000133 T133: BEGINTEST, 133 ;: CHECK SWITCH REGISTER OPTIONS.
3498 032016 106427 000300 MTPS #PRI6 ;:RAISE CPU PRIORITY
3499 032022 012777 000100 147014 MOV #100,@RCSR ;:SET 'RCV-IE' BIT...
3500 032030 017700 147010 MOV @RCSR,R0 ;:...AND SAVE IN R0.
3501 032034 005077 147004 CLR @RCSR ;: CLEAR IT...
3502 032040 017701 147000 MOV @RCSR,R1 ;:...AND SAVE IN R1.
3503 032044 032700 000100 BIT #100,R0
3504 032050 001002 BNE 1$ ;: BR IF IT SET.
3505 032052 104423 032012 ERROR, T133 ;: *** RCV-IE BIT FAILED TO SET ***
3506 032056 032701 000100 1$: BIT #100,R1
3507 032062 001402 BEQ 2$ ;: BR IF IT CLEARED.
3508 032064 104423 032012 ERROR, T133 ;: *** RCV-IE BIT FAILED TO CLEAR ***
3509 032070 106427 000000 2$: MTPS #PRI0

;*****
(2) .SBTTL T134 -- TEST THAT 'RCV-ACT' AND 'RCV-DONE' CAN SET AND CLEAR
(2) ;*****
(2) 032074 104420 000134 T134: BEGINTEST, 134 ;: CHECK SWITCH REGISTER OPTIONS.
3512 ;:
3513 ;: THE TEST ENSURES THAT:
3514 ;: 1. THE INTERNAL 'MAINT' PATH IS FUNCTIONAL
3515 ;: 2. RCV-ACT CLEARS BEFORE RCV-DONE SETS.
3516 ;: 3. RCV-ACT CAN SET AND CLEAR
3517 ;: 4. READING RCV DATA REGISTER CLEARS RCV DONE
3518 ;:
3519 032100 016777 146734 146742 MOV DFPBR,@XCSR
3520 032106 052777 000004 146734 BIS #4,@XCSR ;: SET MAINT BIT.
3521 032114 105777 146730 1$: TSTB @XCSR ;:INSURE WE'RE 'XMIT-RDY'
3522 032120 100375 BPL 1$
3523 032122 016700 146714 MOV DFSPD,R0
3524 032126 006300 ASL R0 ;: SET A 2 CHAR (20 BITS) DELAY COUNT...
3525 032130 010001 MOV R0,R1 ;:...AND SAVE 3 COPIES FOR LATER.
3526 032132 010002 MOV R0,R2
3527 032134 010003 MOV R0,R3
3528 032136 077001 SOB R0 ;: WAIT 20 BIT TIMES...
3529 032140 017700 146702 MOV @RBUF,R0 ;:...AND DUMMY READ.
3530 032144 017700 146674 MOV @RCSR,R0 ;: GET RCVR STATUS.
3531 032150 032700 004200 BIT #RCVACT!RCVDUN,R0 ; BOTH ACTIVE AND DONE SHOULD BE CLEAR
3532 032154 001402 BEQ 2$
3533 032156 104423 032074 ERROR, T134 ;: *** RCV-ACT AND/OR RCV-DUN ARE SET IN ERROR ***
3534
3535 032162 012777 000000 146662 2$: MOV #0,@XBUF ;:NOW, XMIT A NULL CHARACTER
3536 032170 000240 240
3537 032172 017700 146646 3$: MOV @RCSR,R0 ;:GET RCVR STATUS
3538 032176 032700 004000 BIT #RCVACT,R0
3539 032202 001003 BNE 4$ ;:BR IF AND WHEN ACTIVE SETS.
3540 032204 077106 SOB R1,3$
3541 032206 104423 032074 ERROR, T134 ;: *** RCV-ACT FAILED TO SET ***
3542 032212 032700 000200 4$: BIT #RCVDUN,R0 ;:TEST THAT 'RCV-DUN' IS NOT SET YET.
3543 032216 001402 BEQ 5$
3544 032220 104423 032074 ERROR, T134 ;: *** RCV-DUN SET WHEN RCV-ACT WAS SET ***

```

```
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 ;*****
(2) .SBTTL T135 -- TEST THE 'RCV-BRK', 'RCV-ERR', AND 'FR-ERR' BITS
(2) ;*****
(2) 032306 104420 000135 T135: BEGINTEST, 135 ;: CHECK SWITCH REGISTER OPTIONS.
3563 ;
3564 ; GENERATE A FRAMING ERROR BY SETTING THE RCV-BRK BIT.
3565 ; ON SLU1 -- EXPECT THAT BRK INVOKES A MASKABLE LEVEL 7 HALT REQUEST.
3566 ;
3567 032312 013746 000140 1$: MOV @#BHALT,-(SP) ; SAVE BREAK VECTOR.
3568 032316 106427 000340 MTPS #PRI7 ; RAISE CPU.
3569 032322 016777 146512 146520 MOV DFPBR,@XCSR
3570 032330 052777 000004 146512 BIS #4,@XCSR ; SET MAINT BIT.
3571 032336 017700 146504 MOV @RBUF,R0 ; DUMMY READ.
3572 032342 012767 032476 145570 MOV #6$,@BHALT ; SET THE HALT VECTOR (FOR SLU1).
3573 032350 012737 000340 000142 MOV #PRI7,@#BHALT+2
3574 032356 016701 146460 MOV DFSPD,R1
3575 032362 006301 ASL R1 ; SET A 2 CHAR (20 BITS) DELAY.
3576 032364 052777 000001 146456 BIS #XMTBRK,@XCSR ; SET 'XBRK'
3577 032372 012777 000025 146452 MOV #25,@XBUF ;...START XMITTER...
3578 032400 077101 SOB R1,.. ;...AND DELAY. IF SLU1, INTERRUPT...
3579 ;...SHOULD BE HELD OFF (CPU IS AT 7).
3580 032402 105777 146436 TSTB @RCSR ; DONE SHOULD BE THERE BY NOW.
3581 032406 100402 BMI 2$ ; BR IF SO.
3582 032410 104423 032316 176540 2$: ERROR, 1$ ;: *** NO RCV-DUN AFTER XMIT WITH XMT-BRK SET ***
3583 032414 026727 146424 176540 CMP RCSR,#SLU2 ; IF SLU2...
3584 032422 001430 BEQ 7$ ;...JUST VERIFY STATUS/DATA.
3585 ;
3586 ; SLU1, VERIFY THAT 'BHALT' INTERRUPT DOES THE RIGHT THINGS.
3587 ;
3588 032424 012767 032446 145506 3$: MOV #4$,@BHALT ; SO FAR, SO GOOD, CHANGE THE VECTOR...
3589 032432 106427 000300 MTPS #PRI6 ;...AND LOWER CPU TO 6...
3590 032436 000240 240 ;...INTERRUPT SHOULD COME IN.
3591 032440 104423 032316 4$: ERROR, 1$ ;: *** HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6 ***
3592 032444 000417 BR 7$
3593 032446 022626 4$: CMP (SP)+,(SP)+ ; OK, INT RECEIVED, FIX STACK.
3594 032450 012767 032466 145462 MOV #5$,@BHALT ; NOW, CHANGE THE VECTOR...
3595 032456 106427 000300 MTPS #PRI6 ;...LOWER CPU AGAIN...
3596 032462 000240 240 ;...AND SEE THAT WE DON'T GET ANOTHER.
3597 032464 000407 BR 7$ ; TERRIFIC -- PROCEED.
```

PC 32440 = HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6.

```

3598 032466 022626          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$: TSTB    R0          ; RCDV DATA SHOULD BE ZERO.
3616 032546 105700          BEQ      11$
3617 032550 001402          11$: ERROR,  1$      ;: *** RECEIVED DATA NON-ZERO ON 'BRK' ***
3618 032552 104423 032316
3619
3620 ;: AND FINALLY, WE SHOULD BE ABLE TO RESTORE A NORMAL RECEIVER.
3621
3622 032556 042777 000001 146264 11$: BIC      #XMTBRK,@XCSR ; NOW, CLEAR THE BREAK BIT...
3623 032564 016700 146252  MOV      DFSPD,R0
3624 032570 077001          SOB      R0          ; ...DELAY...
3625 032572 017700 146250  MOV      @RBUF,R0    ; ...DUMMY READ.
3626 032576 012777 000025 146246  MOV      #25,@XBUF   ; XMIT A CHAR...
3627 032604 105777 146234  12$: TSTB    @RCSR
3628 032610 100375          BPL      12$        ; ...WAIT FOR RCVR DONE...
3629 032612 017700 146230  MOV      @RBUF,R0    ; ...AND READ.
3630 032616 032700 124000  BIT      #RCVERR!FRERR!RCVBRK,R0 ; ERRORS SHOULD BE GONE.
3631 032622 001402          BEQ      13$
3632 032624 104423 032556  ERROR,  11$      ;: *** RCV-ERR, FR-ERR, AND/OR BRK BITS DIDN'T CLEAR ***
3633 032630 120027 000025  13$: CMPB    R0,#25
3634 032634 001402          BEQ      14$
3635 032636 104423 032556  ERROR,  11$      ; AND CORRECT DATA RECEIVED.
3636 032642 106427 000000  14$: MTPS   #PRIO      ;: *** DATA INCORRECT AFTER 'BRK' SEQUENCE ***
3637
3638 ;: *****
(2) .SBTTL  T136 -- TEST THAT 'RCV-ERR' AND 'OR-ERR' BITS SET AND CLEAR
(2) ;: *****
(2) 032646 104420 000136 1136: BEGINTEST, 136 ;: CHECK SWITCH REGISTER OPTIONS.
3639 032652 016777 146162 146170  MOV      DFPBR,@XCSR
3640 032660 052777 000004 146162  BIS      #4,@XCSR   ; SET MAINT BIT.
3641 032666 017700 146154  MOV      @RBUF,R0   ; DUMMY READ.
3642 032672 012701 000004  MOV      #4,R1
3643 032676 012777 000000 146146  1$:  MOV      #0,@XBUF   ; XMIT A CHARACTER
3644 032704 105777 146140  2$:  TSTB    @XCSR     ; WAIT FOR XMIT-RDY
3645 032710 100375          BPL      2$
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...

```

T136 -- TEST THAT 'RCV-ERR' AND 'OR-ERR' BITS SET AND CLEAR

```

3650 032722 017700 146120      MOV    @RBUF,R0      ;...AND READ RCV DATA BUFFER.
3651 032726 100003              BPL    3$            ; BR IF NO 'RCV-ERR'.
3652 032730 032700 040000      BIT    #ORERR,R0    ; 'OR-ERR' SHOULD BE SET AS WELL.
3653 032734 001002              BNE    4$            ; BOTH SET -- TERRIFIC.
3654 032736              3$:      ERROR,  T136      ;; *** RCV-ERR AND/OR OR-ERR NOT SET ON FORCED OVER-RUN
(2) 032736 104423 032646
3655
3656 032742 012777 000000 146102 4$:      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 146064      MOV    @RBUF,R0    ;...AND READ, ERROR SHOULD BE GONE.
3660 032762 032700 140000      BIT    #RCVERR!ORERR,R0
3661 032766 001402              BEQ    6$
3662 032770 104423 032646      6$:      ERROR,  T136      ;; *** RCV-ERR AND/OR OR-ERR BITS FAILED TO CLEAR ***
3663 032774
3664
3665

```

 .SBTTL T137 -- TEST THAT WE CAN INTERRUPT ON 'RCV-DUN'

```

(2)
(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
3676 033054 000406              BR     3$           ; INTERRUPT SHOULD BE HELD OFF...
3677 033056 022626              2$:      CMP    (SP)+,(SP)+   ;...BR IF SO.
3678 033060 042777 000100 145756      BIC    #100,@RCSR   ; BUT IT WASN'T,
3679 033066 104423 032774      ERROR,  T137       ; REMOVE 'RCV-IE'
;; *** RCV-DUN INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
3680
3681 033072 012711 033120      3$:      MOV    #4$,(R1)     ; CHANGE VECTOR.
3682 033076 106427 000000      MTPS  #PRIO        ; LOWER CPU PRIORITY
3683 033102 000240              240
3684 033104 042777 000100 145732      BIC    #100,@RCSR   ; INTERRUPT SHOULD COME IN.
3685 033112 104423 032774      ERROR,  T137       ; BUT IT DIDN'T, REMOVE 'RCV-IE' BIT
3686 033116 000415              BR     6$           ; *** RCV-DUN INTERRUPT NOT RECEIVED AT ***
3687
3688
3689
3690

```

;; NOW, ON INTERRUPT, WE'RE HERE AT LEVEL 4 (5 IF SLU2).
 ;; 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  #PRIO        ; LOWER THE CPU PRIORITY AGAIN.
3694 033132 000240              240
3695 033134 000406              BR     6$           ; SHOULDN'T GET ANOTHER...
3696 033136 022626              5$:      CMP    (SP)+,(SP)+   ;...OK, EXIT.
3697 033140 042777 000100 145676      BIC    #100,@RCSR   ; 2ND INTERRUPT -- TOO BAD !!
3698 033146 104423 032774      ERROR,  T137       ; REMOVE 'RCV-IE'
;; *** INTERRUPT ACK FAILED TO REMOVE THE REQUEST ***
3699
3700 033152 042777 000100 145664 6$:      BIC    #100,@RCSR   ; INSURE THE ENABLE IS OFF.
3701 033160 017700 145662      MOV    @RBUF,R0    ; READ

```



```

3752
3753 033406 017700 145434      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
      BPL    6$
3758 033430 100375                MOV    @RBUF,R2      ;...AND READ IT BACK.
3759 033432 017702 145410      CMPB  R1,R2
3760 033436 120102                BEQ    7$
3761 033440 001402                ERROR, 5$
3762 033442 104423 033420      7$:  TSTB  R1             ;; *** RCV DATA INCORRECT, EXTERNAL DATA PATH FAILURE *
3763 033446 105701                BNE   4$             ; TERMINATOR DONE ??
3764 033450 001362                ; NOT YET, LOOP.
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        ; CHECK FOR LOOP-IN-SEGMENT...
3770                                ;...RETURN AND FALL THRU IF NOT.
3771 033476 026727 145342 176540  CMP    RCSR,#SLU2    ; LINE 2 ??
3772 033504 001402                BEQ    S7            ; BR IF SO...
3773 033506 000167 174302        JMP    S6            ;...ELSE, GO 'ROUND ONCE.
3774                                .SBTTL

```

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

3776
3777
3778
3779
3780
3781 000233
3782 000264
3783
3784 000200
3785 000040
3786 000020
3787 000010
3788
3789 000004
3790 000002
3791 000001
3792
3793 033512 012767 036335 002460
3794
3795
(2)
(2)
(2) 033520 104420 000142
3796 033524 000240 000240
3797 033530 012700 176200
3798 033534 012701 000130
3799 033540 012702 000240
3800 033544 012767 033634 144232
3801 033552 012706 003776
3802 033556 010067 145276
3803 033562 005720
3804 033564 010067 145272
3805 033570 005720
3806 033572 010067 145266
3807 033576 005720
3808 033600 010067 145262
3809 033604 005710
3810 033606 010167 145260
3811 033612 012721 033644
3812 033616 010221
3813 033620 010167 145244
3814 033624 012721 033654
3815 033630 010221
3816 033632 000414
3817
3818
3819
3820 033634
(2) 033634 104423 033520
3821 033640 000167 001050
3822 033644
(1) 033644 104423 033664
3823 033650 000167 001040
3824 033654
(1) 033654 104423 033664
3825 033660 000167 001030

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

: PORTS A, B, AND C MUST BE INTERCONNECTED THRU A SPECIAL
: LOOP-BACK CONNECTOR. IF NOT THESE TESTS BREAK.

PPM0= 233 ; CONTROL WORD TO SET MODE 0 (ALL INPUT).
PPM1= 264 ; ANOTHER FOR MODE 1, A IN, B OUT, C<7:6> OUT.
LEDBIT= 200 ; LED CONTROL BIT (0 = ON, 1 = OFF).
INRDY= 040 ; INPUT DATA READY (IBF).
INENA= 020 ; ENABLE INTERRUPT ON "INRDY"
ININT= 010 ; INPUT INTERRUPT BIT.
OUTENA= 004 ; ENABLE INTERRUPT ON "OUTRDY".
OUTRDY= 002 ; OUTPUT PORT READY (-OBF).
OUTINT= 001 ; OUTPUT INTERRUPT BIT.

S7: MOV #SEG7,SEGN ; SEGMENT 7 TEXT.

.SBTTL T142 -- TEST PIO REGISTER ADDRESSES

T142: BEGINTEST, 142 ;: CHECK SWITCH REGISTER OPTIONS.
240,240 ;: *** TRY A RESET HERE ***
MOV #PIOA,R0 ;: A IS 1ST OF 4 REGISTERS.
MOV #PIOB,R1 ;: B IS 1ST OF 2 VECTOR PAIRS.
MOV #PIOP,R2 ;: PRIORITY (5) FOR BOTH.
MOV #BADPIO,ERRVEC ;: SET TIME-OUT VECTOR.
MOV #STACK,SP
MOV R0,PPA ;: SET REGISTER POINTERS.
TST (R0)+ ;: IF ANY TRAP -- TROUBLE !@!
MOV R0,PPB
TST (R0)+
MOV R0,PPC
TST (R0)+
MOV R0,PPCW
TST (R0)
MOV R1,PPBV ;: SAVE B VECTOR POINTER.
MOV #BADBI,(R1)+ ;: SET TO CATCH UNEXPECTED INTERRUPTS.
MOV R2,(R1)+
MOV R1,PPAV ;: DITTO FOR PORT A VECTOR.
MOV #BADAI,(R1)+
MOV R2,(R1)+
BR PIOTST

: COME HERE IF TIME-OUT OR UNEXPECTED PIO INTERRUPTS.

BADPIO: ERROR, T142 ;: *** BUS TIME-OUT ON PIO ADDRESS ***
JMP PIODUN
BADBI: ERROR, PIOTST ;: *** UNEXPECTED PORT B INTERRUPT ***
JMP PIODIN
BADAI: ERROR, PIOTST ;: *** UNEXPECTED PORT A INTERRUPT ***
JMP PIODUN

PC 33654 = UNEXPECTED PORT A INTERRUPT.

```

3826 033664
3827
(2)
(2)
(2) 033664 104420 000143
3828
3829
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)
(2)
(2) 033720 104420 000144
3844 033724 112777 000264 145134
3845 033732 112777 000016 145126
3846 033740 112777 000014 145120
3847 033746 012700 000012
3848 033752 077001
3849 033754 117703 145100
3850 033760 117703 145100
3851 033764 120327 000002
3852 033770 001402
3853 033772 104423 033720
3854
3855 033776 005001
3856 034000 012702 177777
3857 034004 005000
3858 034006 110177 145050
3859 034012 117703 145046
3860 034016 032703 000002
3861 034022 001402
3862 034024 104423 034004
3863 034030 117703 145030
3864 034034 120327 000042
3865 034040 001403
3866 034042 077006
3867 034044 104423 034004
3868 034050 117702 145004
3869 034054 117703 145004
3870 034060 120327 000002
3871 034064 001402
3872 034066 104423 034004
3873 034072 120201
3874 034074 001402
3875 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$:  MOVB @PPC,R3  ;; GET STATUS.
: BIT #OUTRDY,R3 ;; OUTRDY SHOULD BE GONE.
: BEQ 4$
: ERROR, 2$     ;; *** OUTRDY STILL SET AFTER OUTPUT XFER ***
4$:  MOVB @PPC,R3  ;; GET STATUS AGAIN...
: CMPB R3,#INRDY!OUTRDY ;; ...AND WAIT FOR THIS.
: BEQ 5$
: SOB R0,4$    ;; WONDERFUL, MOVE ON.
: ERROR, 2$     ;; *** STATUS INCORRECT AFTER OUTPUT XFER ***
5$:  MOVB @PPA,R2  ;; DATA INPUT (THRU A).
: MOVB @PPC,R3  ;; GET STATUS
: CMPB R3,#OUTRDY ;; EXPECT THIS STATUS.
: BEQ 6$
: ERROR, 2$    ;; BR IF SO.
: ERROR, 2$    ;; *** STATUS INCORRECT AFTER INPUT XFER ***
6$:  CMPB R2,R1   ;; DATA LOOP INTACT ??
: BEQ 7$
: ERROR, 2$    ;; BR IF SO.
: ERROR, 2$    ;; *** DATA INCORRECT AFTER OUT/IN EXCHANGE ***
  
```

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!OUTINI
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 034202 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          5$:   CMP    (SP)+,(SP)+
3906 034224 104423 034106   ERROR, T145
3907
3908 034230 112777 000004 144630 6$:   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$:   MOVB  @PPC,R3
3913 034256 001403
3914 034260 077006
3915 034262 104423 034106   BIT   #OUTINT,R3
3916
3917 034266 132777 000040 144570 8$:   SOB   R0,7$
3918 034274 001001
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$(R4)          ;; ...AND SET VECTOR.
      MOVB  #<2_1.>!1,@PPCW  ;; SET C<2> B INTR ENABLE.
      240                    ;; OUTRDY IS ALREADY THERE, SO...
      BR    2$               ;; ...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!OUTINI
      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$:   CMP    (SP)+,(SP)+
      MOV   #5$(R4)
      MTPS  #PRI4
      240
      BR    6$
      5$:   CMP    (SP)+,(SP)+
      ERROR, T145           ;; BUT WE DID -- TROUBLE.
                          ;; *** DOUBLE PORT B INTERRUPT ***
      6$:   MOVB  #<2 1.>!0,@PPCW
      MOVB  #1,@PPB
      CLR   R0
      7$:   MOVB  @PPC,R3
      BIT   #OUTINT,R3
      BEQ   8$
      SOB   R0,7$
      ERROR, T145           ;; OK, CLEAR THE ENABLE...
                          ;; ...WRITE -- SHOULD LOWER INT REQUEST.
      8$:   BITB  #INRDY,@PPC
      BNE   9$
      SOB   R0,8$
      9$:   MOVB  @PPA,R0
      MOV   #BADBI,(R4)

```

```

:*****
.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.

```

T146 -- TEST THE PORT A (INPUT) INTERRUPT

```
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$
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
3949 034430 104423 034310      ERROR,  T146                ;...INTERRUPT SHOULD COME IN.
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
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$:    MOVB    @PPC,R3         ; READ STATUS.
3964 034502 032703 000010      BIT     #ININT,R3
3965 034506 001403                      BEQ     9$
3966 034510 077006                      SOB    R0,8$
3967 034512 104423 034310      ERROR,  T146                ; BR WHEN INT REQUEST CLEARS.
3968
3969 034516 012714 033654      9$:    MOV     #BADAI,(R4)    ; RESET PORT A VECTOR.
3970
3971
(2)
(2)
(2) 034522 104420 000147      ;*****
.SBTTL    T147 -- TEST THAT B PRIORITY IS HIGHER THAN A
;*****
T147:    BEGINTEST, 147
MOV     PPBV,R4           ; CHECK SWITCH REGISTER OPTIONS.
MTPS   2(R4)            ; GET B VECTOR POINTER.
MOV     #3$(R4)          ; RAISE CPU TO DEVICE LEVEL.
MOV     #2$,4(R4)        ; SET B VECTOR.
MOV     #<2,1.>!1,@PPCW ; SET A VECTOR.
MOV     #<4,1.>!1,@PPCW ; ENABLE B...
MOV     #1,@PPB          ;...AND A INTERRUPTS.
MOV     #1,@PPB          ; OUTPUT DATA...
```

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

T147 -- TEST THAT B PRIORITY IS HIGHER THAN A

```

3979 034572 005000          CLR      RO
3980 034574 132777 000040 144262 1$:  BITB   #INRDY,@PPC    ;...AND WAIT 'TIL DONE.
3981 034602 001001          BNE     .+4
3982 034604 077005          SOB     RO,1$        ; 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,RO
4002 034702 012714 033644          MOV   #BADBI,(R4)  ; RESET VECTORS.
4003 034706 012764 033654 000004          MOV   #BADAI,4(R4)
4004
4005 034714 012767 000006 143062 PIODUN: 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

```

.SBTTL END OF PASS ROUTINES.

: AT LAST -- WE'VE HIT THE END OF THE TRAIL, PODNAH !!!
 : REPORT END-PASS, TAKE A LITTLE NAP, AND THEN DO IT ALL AGAIN.

: SOB RO,, EXECUTES IN 3.66 USEC IN KXT11.
 : IF Q-BUS TRANSACTION REQUIRED, ADD 609 NSEC (CYCLE SLIP).
 : THEREFORE, SOB FOR TIMING TAKES ABOUT 4.27 USEC/COUNT.

4010
 4011
 4012
 4013
 4014
 4015
 4016
 4017
 4018
 4019 034730 005767 000730
 4020 034734 001001
 4021 034736 000207
 4022
 4023 034740 104007
 4024 034742 016700 001232
 4025 034746 000402
 4026 034750 012700 035130
 4027 034754 104003
 4028 034756 012700 035141
 4029 034762 104003
 4030 034764 005267 147012
 4031 034770 026727 147004
 4032 034774 000000
 4033 034776 001002
 4034 035000 005267 147000
 4035 035004 016767 146770 177762
 4036 035012 016700 146764
 4037 035016 104424
 4038 035020 012700 035155
 4039 035024 104003
 4040 035026 016700 146746
 4041 035032 104424
 4042 035034 104007
 4043 035036 032767 010000 143132
 4044 035044 001402
 4045 035046 004767 000124
 4046 035052 106427 000000
 4047 035056 005000
 4048 035060 012701 000004
 4049 035064 077001
 4050 035066 077102
 4051
 4052 035070 013700 000042
 4053 035074 001405
 4054 035076 000240
 4055 035100 004710
 4056 035102 000240 000240 000240
 4057
 4058 035110 016700 000550
 4059 035114 001002
 4060 035116 012700 010336
 4061 035122 012706 003776
 4062 035126 000110
 4063
 4064 035130 005015 045516 040530
 4065 035141 054 042440 042116

```

ENDSEG: TST      INSEG      : LOOP-IN-SEGMENT ??
         BNE     1$         : YES.
         RETURN    : JUST RETURN IF NOT.

1$:      EMT+7          : <CRLF>
         MOV      SEGN,R0   : CURRENT SEGMENT DONE.
         SKP2
ENDPAS: MOV      #EOP,R0   : ALL SEGMENTS DONE.
         EMT+3
         MOV      #EOP1,R0
         EMT+3          : END PASS.
         INC      TPASS     : BUMP TOTAL PASS COUNT.
         CMP      ERRCNT,(PC)+ : ANY ERRORS THIS PASS ??
1$:      0
         BNE     2$         : SKIP IF SO.
         INC      GPASS     : NO, BUMP CONSECUTIVE-GOOD PASS COUNT.
2$:      MOV      ERRCNT,1$ : UPDATE SAVED COUNT.
         MOV      TPASS,R0  : PASS COUNT...
         TRAP+24
         MOV      #EOPE,R0
         EMT+3
         MOV      ERRCNT,R0
         TRAP+24
         EMT+7          : ...AND TOTAL ERROR COUNT.
         BIT      #10000,SWR : <CRLF>
         BEQ     3$         : ERROR SUMMARY ??
         CALL    ERRSUM     : PRINT IT IF SO.
3$:      MTPS     #PRI0     : LOWER THE CPU.
         CLR     R0
         MOV     #4,R1
4$:      SOB     R0,,       : ENDPASS DELAY...
         SOB     R1,4$      : ...ABOUT 1 SECOND.
         MOV     #42,R0
         BEQ     5$         : BR IF NOT CHAINING.
         NOP
         CALL    (R0)       : OR RESET.
         CALL    240,240,240 : SYSMAC STYLE CHAIN RETURN.
         MOV     INSEG,R0   : GET ITERATION ADDRESS.
5$:      BNE     6$         : IF NONE, BACK TO SQUARE 1.
         MOV     #DOALL,R0  : RESET STACK...
6$:      MOV     #STACK,SP  : ...AND LOOP FOREVER.
         JMP     (R0)
EOP:    .ASCIZ  <15><12>'NKXAA0'
EOP1:   .ASCIZ  ', END PASS '
  
```

END OF PASS ROUTINES.

4066 035155 054 052040 052117 EOPE: .ASCIZ ', TOTAL ERRORS '
4067 035176 .EVEN

: ERROR SUMMARY REPORT.
: CALLED IN 'END-PASS' IF SWR<12> IS SET.

4073 035176 012702 004006 ERRSUM: MOV #EPCTBL,R2
4074 035202 005712 TST (R2)
4075 035204 001414 BEQ 2\$; BR IF NO ERROR DATA LOGGED.
4076 035206 012700 035240 MOV #ERHEAD,R0 ; HEADER.
4077 035212 104003 EMT+3
4078 035214 1\$: EMT+10 ;: <TAB>
4082 035214 104010 MOV (R2)+,R0 ;: PRINT (R2)+
(1) 035216 012200 TRAP+24 ;: <TAB>
(1) 035220 104424 EMT+10 ;: PRINT (R2)+
(1) 035222 104010 MOV (R2)+,R0 ;: <CRLF>
(1) 035224 012200 TRAP+24 ; LOOP 'TIL DONE.
(1) 035226 104424 EMT+7
4083 035230 104007 TST (R2)
4084 035232 005712 BNE 1\$
4085 035234 001367 RETURN
4086 035236 000207 2\$:

4088 035240 005015 042411 051122 ERHEAD: .ASCIZ <15><12><11>'ERRPC'<11>' TIMES'<15><12>
4089 .EVEN

: SUBROUTINE TO CHECK FOR Q-BUS EXERCISOR.
: RETURN QBX CSR1 ADDRESS IN R0 IF IT'S THERE.

4095 035262 005767 143550 QBXIN: TST QBXA ; QBX INSTALLED ??
4096 035266 001020 BNE 3\$; BR IF SO.
4097 035270 012767 035314 142506 MOV #1\$,ERRVEC ; DON'T KNOW YET, SET TRAP CATCHER...
4098 035276 012700 174020 MOV #QBXCSR,R0 ; ...AND A POINTER.
4099 035302 005010 CLR (R0) ; CLEAR QBX CSR1 (IF POSSIBLE)...
4100 035304 000240 240 ; ...TRAP TO 1\$ IF NOT.
4101 035306 010067 143524 MOV R0,QBXA ; IT'S THERE, SET FLAG = CSR ADDRESS...
4102 035312 000403 BR 2\$; ...AND RETURN.
4103 1\$: CLR QBXA ; NOT THERE, CLEAR FLAG.
4104 035314 005067 143516 CMP (SP)+,(SP)+ ; FIX STACK.
4105 035320 022626 2\$: MOV #ERRVEC+2,ERRVEC ; RESET ERROR VECTOR.
4106 035322 012767 000006 142454 3\$: MOV QBXA,R0 ; RETURN CSR POINTER OR ZERO.
4107 035330 016700 143502 RETURN
4108 035334 000207


```

4110      .SBTTL TRAP HANDLER
4111      :*****
4112      : TRAP HANDLER
4113      :
4114 035336 010046 TRAPHAN: MOV R0,-(SP)      ; SAVE CALLERS R0...
4115 035340 010146      MOV R1,-(SP)      ; ...AND R1.
4116 035342 016600 000004      MOV 4(SP),R0      ; GET TRAP PC.
4117 035346 116000 177776      MOV#B -2(R0),R0    ; GET TRAP CODE.
4118 035352 010001      MOV R0,R1      ; WORKING COPY.
4119 035354 162701 000017      SUB #17,R1
4120 035360 100001      BPL 1$
4121 035362 005001      CLR R1      ; BR IF TRAP+20 OR HIGHER.
4122 035364 006301      1$: ASL R1      ; TRAP+0 THRU TRAP+17 ARE COMMON.
4123 035366 016107 035372      MOV 2$(R1),PC    ; SHIFT 17 TO WORD OFFSET...
4124      : ...AND DISPATCH THERE.
4125 035372 035406      2$: CCBIT      ; TRAP0 - TRAP17 = CC BIT CHECK.
4126 035374 035436      BGNTST      ; TRAP20 = BEGIN NEW TEST.
4127 035376 035460      CHKSWR      ; TRAP21 = CHECK SWR.
4128 035400 035470      NEWSWR      ; TRAP22 = GET NEW SWR.
4129 035402 035666      LOGERR      ; TRAP23 = LOG AND REPORT ERROR.
4130 035404 036464      ITOA      ; TRAP24 = CONVERT INTEGER TO ASCII AND PRINT.
4131      :
4132      : ADD MORE AS NEEDED.
4133      :
4134      :
4135      :*****
4136      : CONDITION CODE BIT CHECKER.
4137      : CALL: TRAP+N, WHERE N IS THE CONDITION CODE TO EXPECT (0 - 17).
4138      :
4139      : PSW TO EXAMINE IS AT 6(SP), N IS IN R0.
4140      : RETURN TO CALL+2 IF CC N.E. N
4141      : RETURN TO CALL+6 OTHERWISE.
4142      :
4143 035406 016601 000006 CCBIT: MOV 6(SP),R1      ; GET PSW TO CHECK.
4144 035412 042701 177760      BIC #^C17,R1      ; KEEP CC BITS ONLY.
4145 035416 020001      CMP R0,R1
4146 035420 001003      BNE 1$      ; BR IF CC BITS INCORRECT.
4147 035422 062766 000004 000004      ADD #4,4(SP)      ; OK, ADJUST RETURN PC.
4148 035430 012601      1$: MOV (SP)+,R1
4149 035432 012600      MOV (SP)+,R0
4150 035434 000002      RTI
4151      :
4152      :*****
4153      : BEGIN TEST ROUTINE.
4154      : CALL VIA TRAP+20
4155      :
4156 035436 104421      BGNTST: CHECKSWR
4157 035440 012601      MOV (SP)+,R1      ; RESTORE CALLERS REGISTERS.
4158 035442 012600      MOV (SP)+,R0
4159 035444 017667 000000 000524      MOV @ (SP),TSTNUM ; SAVE TEST NUMBER...
4160      : ...FOR THE ERROR HANDLER...
4161 035452 062716 000002      ADD #2,(SP)      ; ...BUMP RETURN PC...
4162 035456 000002      RTI      ; ...AND RETURN.

```

```

4164
4165
4166
4167
4168
4169 035460 104005
4170 035462 120027 000007
4171 035466 001024
4172
4173 035470 005737 000042
4174 035474 001021
4175 035476 012700 035546
4176 035502 104003
4177 035504 016700 142466
4178 035510 104424
4179 035512 012700 035557
4180 035516 104003
4181 035520 104000
4182 035522 104011
4183 035524 000405
4184 035526 120067 142444
4185 035532 001016
4186 035534 010067 142436
4187 035540 012601
4188 035542 012600
4189 035544 000002
4190
4191 035546 005015 053523 020122 OSWR:
4192 035557 040 047040 053505 NSWR:
4193
4194
4195
4196
4197 035570 010067 142402
4198 035574 042700 177770
4199 035600 006300
4200 035602 016067 035644 000054
4201 035610 012700 004000
4202 035614 012701 002000
4203 035620 005020
4204 035622 077102
4205 035624 016700 000034
4206 035630 001002
4207 035632 012700 010336
4208 035636 012706 003776
4209 035642 000110
4210 035644 000000
4211 035646 010336 024030 027052
4212 035654 027642 027764 030014
4213 035662 033512
4214 035664 000000
4215

:*****
: SWITCH REGISTER HANDLER.
: CALLED VIA TRAP+21 (CHECKSWR) AND TRAP+22 (GETSWR).
:
CHKSWR: .ENABL LSB
          EMT+5 : CHECK KEYBOARD ( => R0 ).
          CMPB R0,#7 : <^G> ??
          BNE 1$ : JUST RETURN IF NOT.

NEWSWR: TST @#42 : CHAINING ??
          BNE 1$ : DON'T IF SO.
          MOV #OSWR,R0 : OLD SWR...
          EMT+3 :
          MOV SWR,R0 : ...= THIS.
          TRAP+24 :
          MOV #NSWR,R0 : NEW SWR ...
          EMT+3 : ...AND WAIT FOR INPUT.
          EMT+0 : ...PARSE OCTAL NUMBER.
          EMT+11 : JUST EXIT IF NO CHANGE.
          BR 1$ : NEW SEGMENT SELECTED ??
          CMPB R0,SWR : BR IF SO.
          BNE NEWSEG : OTHERWISE, SAVE NEW SWR...
          MOV R0,SWR : ...RESTORE...
          1$: MOV (SP)+,R1 :
          MOV (SP)+,R0 : ...AND RETURN TO CALLER.
          RTI :

          .DSABL LSB
          .ASCIZ <15><12>'SWR = '
          .ASCIZ ' NEW = '
          .EVEN

: NEW SEGMENT REQUIRED. GET ADDRESS AND DO A RESTART-IN-SEGMENT.
:
NEWSEG: MOV R0,SWR : SAVE NEW SWR.
          BIC #^C7,R0 : STRIP OFF SEGMENT NUMBER.
          ASL R0 : SHIFT NUMBER UP TO WORD OFFSET...
          MOV 3$(R0),INSEG : ...AND SET/CLEAR "INSEG" FLAG.
          MOV #ERRCNT,R0
          MOV #1024.,R1
          1$: CLR (R0)+ : CLEAR THE ERROR TABLE AND COUNTERS.
          SOB R1,1$ :
          MOV INSEG,R0 : GET LOOP-SEGMENT ADDRESS.
          BNE 2$ :
          MOV #DOALL,R0 : IF ZERO, RESTORE "RUN-ALL" MODE.
          2$: MOV #STACK,SP : NOW, RESET STACK...
          JMP (R0) : ...AND RESTART-IN-SEGMENT.
          3$: 0 : SEGMENT = 0 = RUN 'EM ALL...
          S1,S2,S3 : ...OTHERWISE, RUN JUST ONE OF THESE.
          S4,S5,S6
          S7
INSEG: 0 : ZERO = RUN ALL...
          : ...NZ = ADDRESS FOR LOOP-IN-SEGMENT.

```

4217
 4218
 4219
 4220
 4221
 4222
 4223
 4224
 4225
 4226
 4227
 4228
 4229
 4230
 4231
 4232
 4233
 4234
 4235
 4236
 4237
 4238
 4239
 4240
 4241
 4242
 4243
 4244
 4245
 4246
 4247
 4248
 4249
 4250
 4251
 4252
 4253
 4254
 4255
 4256
 4261
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)

035666 012700 004000
 035672 016601 000004
 035676 005220
 035700 122020
 035702 005020
 035704 020110
 035706 001410
 035710 005710
 035712 001405
 035714 020027 007772
 035720 103005
 035722 022020
 035724 000767
 035726 010110
 035730 005260 000002
 035734
 035734 032767 020000 142234
 035742 001062
 035744 010667 000224
 035750 062767 000010 000216
 035756 104007
 035760 016700 000214
 035764 104003
 035766 012700 036357
 035772 104003
 035774 016700 000176
 036000 104424
 036002 012700 036377
 036006 104003
 036010 016600 000004
 036014 104424
 036016 104010
 036020 016600 000006
 036024 104424
 036026 104010
 036030 016600 000002
 036034 104424
 036036 104010
 036040 011600
 036042 104424
 036044 104010
 036046 010200
 036050 104424
 036052 104010
 036054 010300

```

*****
: ERROR HANDLER.
: CALLED VIA TRAP+23.
: ACCUMULATE ERROR DATA (PC AND NUMBER OF OCCURRANCES) IN THE ERROR LOG.
: PRINT THE STATE OF THE MACHINE AT TIME OF ERROR (UNLESS INHIBITED).
: TEST THE LOOP (SWR<9>) AND HALT (SWR<15>) SWITCHES
: AND EXIT ACCORDINGLY.
LOGERR: MOV #ERRCNT,R0 ; GET TABLE POINTER...
: MOV 4(SP),R1 ; ...AND CURRENT (ERROR) PC.
: INC (R0)+ ; BUMP TOTAL ERROR COUNT.
: CMPB (R0)+,(R0)+ ; SKIP OVER TOTAL PASS COUNT...
: CLR (R0)+ ; ...AND CLEAR "CONSECUTIVE-GOOD" COUNT.
1$: CMP R1,(R0) ; THIS PC ALREADY LOGGED ??
: BEQ 3$ ; JUST BUMP ITS COUNT IF SO.
: TST (R0) ; NO, END OF TABLE ??
: BEQ 2$ ; STORE PC HERE IF SO.
: CMP R0,#LOGEND-4 ; TABLE FULL ??
: BHIS 4$ ; YES, FORGET IT !!!
: CMP (R0)+,(R0)+ ; NONE OF THE ABOVE, BUMP POINTER...
: BR 1$ ; ...AND LOOP 'TIL WE FIND ONE...
: ; ...OR THE OTHER.
2$: MOV R1,(R0) ; STORE ERROR PC...
3$: INC 2(R0) ; ...AND BUMP ITS COUNTER...
4$: ; ...AND FALL THRU.

: ERROR MESSAGES INHIBITED ??
: BR IF SO.
: GET STACK POINTER...
: ADJUST TO SP AT TIME OF ERROR.
: <CRLF>
: CURRENT SEGMENT...
: ...AND TEST...
: ...NUMBER.
: ...HEADER.
:: PRINT 4(SP)
:: <TAB>
: MOV 6(SP),R0 ; PRINT 6(SP)
: TRAP+24 ; <TAB>
: EMT+10 ; PRINT 2(SP)
: MOV 2(SP),R0 ; <TAB>
: TRAP+24 ; PRINT (SP)
: EMT+10 ; <TAB>
: MOV (SP),R0 ; PRINT R2
: TRAP+24 ; <TAB>
: EMT+10 ; PRINT R2
: MOV R2,R0 ; <TAB>
: TRAP+24 ; <TAB>
: EMT+10 ; <TAB>
: MOV R3,R0 ; <TAB>

```

(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,SAVTTY	:: 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 SAVTTY,@#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				SAVTTY: 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: .ASCII	<15><12>' PC+2'<11>
4291	036407	040	050040	053523		.ASCII	' PSW'<11>
4292	036415	040	051040	004460		.ASCII	' R0'<11>
4293	036422	020040	030522	011		.ASCII	' R1'<11>
4294	036427	040	051040	004462		.ASCII	' R2'<11>
4295	036434	020040	031522	011		.ASCII	' R3'<11>
4296	036441	040	051040	004464		.ASCII	' R4'<11>
4297	036446	020040	032522	011		.ASCII	' R5'<11>
4298	036453	122	024066	050123		.ASCIZ	'R6(SP)'<15><12>
4299						.EVEN	

4301
 4302
 4303
 4304
 4305
 4306
 4307
 4308
 4309 036464 016600 000002
 4310 036470 012701 036522
 4311 036474 005011
 4312 036476 104030
 4313 036500 005767 000016
 4314 036504 001403
 4315 036506 012700 036522
 4316 036512 104003
 4317 036514 012601
 4318 036516 012600
 4319 036520 000002
 4320
 4321 036522 031061 032063 033065
 4322 036532
 4323
 4324 036532 036532
 4325
 4326 010000

```

*****
: CONVERT INTEGER TO ASCII AND PRINT.
: CALL VIA TRAP+24, NUMBER TO CONVERT IS 2ND WORD ON THE STACK.
:
: CODE ADDED BECAUSE EMT+30 (INTEGER TO ASCII) FUNCTIONALITY IS
: DIFFERENT BETWEEN XDP+ V1.0 AND V1.1. THIS ROUTINE MAKES THAT
: DIFFERENCE TRANSPARENT TO THE CALLER.
:
: ITOA:  MOV      2(SP),R0      ; NUMBER => R0.
:        MOV      #2$,R1      ; IF V1.1, PACK THE ASCII STRING AT 2$.
:        CLR      (R1)
:        EMT+30
:        TST      2$          ; CONVERT I TO A.
:        BEQ      1$          ; IF V1.0, IT'S ALREADY BEEN PRINTED.
:        MOV      #2$,R0      ; IF V1.1, PRINT IT NOW.
:        EMT+3
: 1$:    MOV      (SP)+,R1     ; RESTORE...
:        MOV      (SP)+,R0
:        RTI
:        ;...AND RETURN.
:
: 2$:    .ASCIZ  /123456/
:        .EVEN
:
: $LSTAD: .
:
:        .END  BEGIN
  
```

CROSS REFERENCE TABLE -- USER SYMBOLS

ADCBO	012536	736#														
ADCB1	020146	1721#														
ADCO	014142	1020#														
ADC1	022122	2067#														
ADDO	014656	1144#														
ADD1	022672	2195#														
ADR =	001000	141#	142	143	144	145	146	147	148	149	150	151	152	153		
		154	155	157	167	168	170	171	172	173	174	175	177	178		
		179	180	181	182	1288*	1291	1306	1373*	1376	1398	1441*	1444	1829		
		1853														
ADR1 =	001002	142#	1289*	1292	1307	1374*	1377	1399	1442*	1445	1830	1854				
ADR2 =	001004	143#	1290*	1375*	1378	1400	1443*	1446								
ALLCC	010604	388#														
ASLCO	012364	696#														
ASLB1	017760	1678#														
ASLO	013770	980#														
ASL1	021734	2024#														
ASRBO	012436	712#	718	721	724	729	732									
ASRB1	020036	1695#	1703	1706	1709	1714	1717									
ASRO	014042	996#	1002	1005	1008	1013	1016									
ASR1	022012	2041#	2049	2052	2055	2060	2063									
BADAI	033654	3814	3824#	3969	4003											
BADBI	033644	3811	3822#	3921	4002											
BADPIO	033634	3800	3820#													
BADRI	030172	3161	3178#	3702												
BADTI	030202	3164	3180#	3416												
BADU1	030152	3138	3174#													
BADU2	030162	3147	3176#													
BEGIN	010000	219#	243	4326												
BEGINT=	104420	92#	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		
BEVNT =	000100	118#	2897*	2904*	2908*											
BGNTST	035436	4126	4156#													
BHALT =	000140	109	119#	246*	247*	3567	3572*	3573*	3588*	3594*	3607*					
BICBO	011610	565#														
BICB1	017006	1506#														
BICO	013162	839#														
BIC1	020744	1851#														
BPTVEC=	000014	66#	67	2595*	2596*	2616*										
BRANCH	010350	312#														
BTWRD	023572	2332#														
CBIT	010514	360#														
CCBIT	035406	4125	4143#													
CHECKC=	104400	91#	389	393	419	431	503	527	530	533	536	542	569	575		
		581	584	591	595	598	601	608	611	615	618	625	632	640		
		645	652	664	670	685	688	700	706	709	720	723	728	739		
		745	753	761	767	776	783	787	795	801	804	808	811	818		
		825	844	850	856	860	864	875	879	882	885	892	895	899		
		902	909	916	924	929	936	948	954	969	972	984	990	993		
		1004	1007	1012	1023	1029	1037	1045	1051	1060	1067	1071	1079	1087		

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*	
TSTBO	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				

CROSS REFERENCE TABLE -- USER SYMBOLS

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
3892	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

CROSS REFERENCE TABLE -- MACRO NAMES

RUN-TIME RATIO: 77/39=1.9
CORE USED: 10K (19 PAGES)