

SBC11/21  
11/21+

KXT-11 PROC DIAG  
CNKXABO

COPYRIGHT (c) 1982-83  
AH-S943B-MC  
FICHE 01 OF 01

APR 1984  
digital  
Made In USA

**IDENTIFICATION**  
-----

**PRODUCT CODE:** AC-S942B-MC  
**PRODUCT NAME:** CNXABO KXT-11 PROCESSOR DIAGNOSTIC  
**PRODUCT DATE:** DECEMBER 5, 1983  
**MAINTAINER:** ISS DIAGNOSTIC SERVICES  
**AUTHOR:** G. PASQUANTONIO  
**REVISED BY:** JAMES S. DOUCETTE

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  
COPYRIGHT (C) 1983 DIGITAL EQUIPMENT CORPORATION

## 1.0 GENERAL INFORMATION

-----

THIS DIAGNOSTIC IS A RAM-BASED VERSION OF "CIKXACO" AND IS INTENDED FOR FIELD SUPPORT OF THE KXT-11 SBC-11/21 AND SBC-11/21+.

## 1.1 ABSTRACT

-----

THE KXT11 IS A T11-BASED SINGLE BOARD COMPUTER (SBC-11). IT INCLUDES A T11 CPU, UP TO 12 KW 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 RAM TESTS.
4. LOCAL 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 MACRO-ODT ROM -OR-  
 KXT11-AB (M7676) MODULE UNDER TEST, WITH MACRO-ODT ROM.  
 KXT11-LP (M3275) DUAL LOOP-BACK CONNECTOR.  
 16KW (OR MORE) Q-BUS MEMORY (MSV11-D).  
 XXDP+ LOAD DEVICE AND MEDIA (RX02, RL02, ETC.).  
 CONSOLE TERMINAL (VT52, VT100, ETC.).  
 Q-BUS EXERCISER (G5281) MODULE (OPTIONAL).

## 1.3 RELATED DOCUMENTS AND STANDARDS

-----

KXT11-AA ENGINEERING SPEC.  
 M8063 FALCON SBC-11/21 SINGLE BOARD COMPUTER USER'S GUIDE.  
 KXT11-AB ENGINEERING SPEC.  
 M7676 FALCON-PLUS SBC-11/21+ USER'S GUIDE.  
 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 FOR SBC-11/21) OR
LOCAL ROM ADDRESS/SIZE	164000	2KW	(MACRO-ODT FOR SBC-11/21+)
LOCAL RAM ADDRESS/SIZE	160010	2KW-4	(FOR SBC-11/21) OR
LOCAL RAM ADDRESS/SIZE	100000	12KW	(FOR SBC-11/21+) OR
LOCAL RAM ADDRESS/SIZE	140000	4KW	(FOR SBC-11/21+)
BEVNT VECTOR	000100	PRI6	
BHALT VECTOR	000140	PRI7	(MASKABLE)
SERIAL LINE 1 ADDRESS	177560		(CONSOLE TERMINAL)
RCVR VECTOR	000060	PRI4	
XMITR VECTOR	000064	PRI4	
SERIAL LINE 2 ADDRESS	176540		(WITH LOOP-BACK...)
RCVR VECTOR	000120	PRI5	(...CLOCK DISABLED)
XMITR 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:

SMR<15> 100000 HALT ON ERROR.  
SMR<13> 020000 INHIBIT ERROR MESSAGES.  
SMR<12> 010000 PRINT ERROR SUMMARY AT END-PASS.  
SMR<09> 001000 LOOP ON ERROR.  
SMR<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 FROM 10 TO 20 SECONDS PER PASS (DEPENDING ON THE HARDWARE) 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 SHR<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 SHR<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 PSM CONTEXT IS MAINTAINED.

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

**5.3 SEGMENT 3 -- LOCAL RAM TESTS**  
-----

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

NOTE: FOR THE SBC-11/21 (NOT SBC-11/21+), MACRO-ODT UTILIZES A PORTION OF THE LOCAL RAM. IF A BREAK-AND-PROCEED SEQUENCE IS SERVICED DURING THIS SEGMENT, ERRORS MAY OCCUR DUE TO THE OVERWRITING OF RAM BACKGROUND DATA PATTERNS (BY ODT).

**5.4 SEGMENT 4 -- LOCAL 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 AN END-PASS MESSAGE (AND OPTIONAL ERROR SUMMARY) WILL BE PRINTED ON THE CONSOLE TERMINAL.

**6.0 PROGRAM LISTING**  
-----

THE PROGRAM LISTING FOLLOWS.

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  
65  
80  
81  
82

.TITLE CNKXAB -- KXT-11 (FALCON(-PLUS)) VERIFICATION TESTS (FIELD VERSION).

THE FOLLOWING CHANGES WERE MADE TO CNKXAA IN PRODUCING CNKXAB. THE MOST SIGNIFICANT CHANGES WERE MADE TO ACCOMODATE THE FALCON-PLUS ARCHITECTURE. REVISION B RELEASE DATE IS DECEMBER 5, 1983. CHANGES WERE MADE BY JAMES S. DOUCETTE (CHANGES ARE MARKED BY ;JSD REV B).

1. CHANGES TO SECTION 3 (LOCAL RAM TESTS) AND SECTION 4 (LOCAL PROM/ROM TESTS) TO ACCOMODATE MEMORY MAP 0 FOR BOTH FALCON AND FALCON-PLUS. ALSO RENAMED LITERALS DEFINING THE STARTING ADDRESS AND SIZE OF LOCAL RAM AND ROM.

2. MINOR CODING CHANGES TO ACCOMODATE VAX/VMS HOST DEVELOPMENT, AS OPPOSED TO TOPS-20. (EXAMPLES; .ABS BECOMES .ENABL ABS, INSTRUCTIONS USING <N,N> (WHERE N = NUMBER) CHANGED TO THE EQUIVALENT VALUE IN IMMEDIATE MODE).

NOTE: THREE ASSEMBLY ERRORS (TYPE "Z") WILL RESULT IF THIS DIAGNOSTIC IS ASSEMBLED UNDER VAX/VMS USING "MCR MAC". THESE ERRORS ARE NON-FATAL AND SHOULD BE IGNORED. IN SOME CASES, THE "ERRONEOUS" INSTRUCTION IS REQUIRED FOR THE TEST WHICH IS TAKING PLACE.

- PART 1 -- TEST THE T11 INSTRUCTION SET.
- PART 2 -- TEST THE T11 TRAPS AND INTERRUPTS.
- PART 3 -- TEST THE LOCAL 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 ;JSD REV B
.ENABL ABS ;JSD REV B
.NLIST MD,MC,CND,TOC
.LIST ME
```

```
*****
;PRGSIZ=<?H<#LSTAD>>*1 ; PROGRAM SIZE IN 1/8 K UNITS. ;JSD REV B
```

```
.SBTTL SWITCH REGISTER OPTIONS
SMR= 176 ; THE STANDARD SOFT SWITCH REGISTER ADDRESS.
; SMR<15> 100000 HALT ON ERROR.
; SMR<14>
; SMR<13> 020000 INHIBIT ERROR MESSAGES.
; SMR<12> 010000 PRINT ERROR SUMMARY AT END-PASS.
; SMR<11>
; SMR<10>
; SMR<09> 001000 LOOP ON ERROR.
; SMR<08>
; SMR<02:00> N RUN SEGMENT N (1 THRU 7) ONLY.
```

```
*****
TN#= 0 ; INIT TEST NUMBER SEQUENCE.
```

```
; T11 CPU EQUATES.
```

000000

000176

000000



SWITCH REGISTER OPTIONS

83	000006	R6=	SP		
84	000007	R7=	PC		
85	000004	ERRVEC=	4	:	ILLEGAL INSTRUCTION VECTOR.
86	000010	RESVEC=	10	:	RESERVED OPCODE VECTOR.
87	000014	BPTVEC=	14	:	BREAKPOINT VECTOR...
88	000014	TRTVEC=	BPTVEC	:	...TRACE TRAP TOO.
89	000020	IOTVEC=	20	:	IOT VECTOR.
90	000024	PMRVEC=	24	:	POWER FAIL VECTOR.
91	000030	EMTVEC=	30	:	EMT VECTOR.
92	000034	TRPVEC=	34	:	TRAP VECTOR.
93	000340	PRI7=	340		
94	000300	PRI6=	300		
95	000240	PRI5=	240		
96	000200	PRI4=	200		
97	000140	PRI3=	140		
98	000100	PRI2=	100		
99	000040	PRI1=	040		
100	000000	PRI0=	000		
101					
102	000010	RESRVD=	000010	:	RESERVED (UNDEFINED) OPCODE.
103	000007	MFPT=	000007	:	MFPT OPCODE (NEW INSTRUCTION)...
104	000004	KXT11=	4	:	...SHOULD RETURN #4 IN R0.
105	000254	CLNZ=	CLN CLZ		
106	000263	SEVC=	SEV SEC		
107	000273	SENV=	SEN SEV SEC		
108	000260	NOP1=	260		
109	000401	SKP1=	BR+1		
110	000402	SKP2=	BR+2		
111	000403	SKP3=	BR+3		
112	104400	CHECKCC=	TRAP	:	CC BIT CHECK CALL.
113	104420	BEGINTEST=	TRAP+20		
114	104421	CHECKSMR=	TRAP+21	:	CHECK FOR <+G>.
115	104422	GETSMR=	TRAP+22	:	GET NEW SMR.
116	104423	ERROR=	TRAP+23	:	ERROR CALL.
117					
118		.SBTTL KXT-11 (FALCON) DEFAULT CONFIGURATION			
119					
120	172000	START=	ODT	:	ON POWER-UP, START MACRO-ODT IN PROM...
121	172004	RESTART=	ODTRE	:	...TRAP TO "RESTART" ON TIME-OUT OR HALT...
122	000000	EHALT=	HALT	:	...AND HALT ON PROGRAM ERROR.
123					
124					
125		;ROMADR=	170000	:	LOCAL ROM SOCKETS STRAPPED... ;JSD REV B
126		;ROMSIZ=	1024.	:	...TO RESPOND AT 30K TO 31K... ;JSD REV B
127				:	...MACRO-ODT PROM MUST BE INSTALLED.
128	170000	ROMADR1=	170000	:	LOCAL ROM SOCKETS STRAPPED... ;JSD REV B
129	002000	ROMSZ1=	1024.	:	...TO RESPOND AT 30K TO 31K... ;JSD REV B
130	164000	ROMADR2=	164000	:	ROM ADDRESS FOR FALCON-PLUS ;JSD REV B
131	004000	ROMSZ2=	2048.	:	2KW OF ROM SPACE FOR FALCON-PLUS ;JSD REV B
132					
133	172000	ODT=	172000	:	IT STARTS HERE...
134	172004	ODTRE=	172004	:	...AND RESTARTS HERE.
135	170000	ODTBRK=	170000	:	"BREAK" HALT IS SERVICED HERE...
136	000140	ODTIN=	BHALT	:	...AND THE VECTOR SERVES AS A HANDY FLAG.
137					
138		;RAMADR=	160010	:	LOCAL 2K RAM STRAPPED... ;JSD REV B
139		;RAMSIZ=	2048.-4.-8.	:	...TO RESPOND AT 28K(+4) TO 30K... ;JSD REV B

KXT-11 (FALCON) DEFAULT CONFIGURATION

```

140
141
142
143      167776      TRAP4= 167776
144      160010      RAMAD1=160010
145      003764      RAMSZ1=2048.-4.-8.
146      100000      RAMAD2=100000
147      024000      RAMSZ2=10240.
148
149      000100      BEVNT= 100
150      000140      BHALT= 140
151
152      177560      SLU1= 177560
153      000060      SLU1V= 60
154      000200      SLU1P= PRI4
155      176540      SLU2= 176540
156      000120      SLU2V= 120
157      000240      SLU2P= PRI5
158
159      176200      PIOA= 176200
160      000134      PIOAV= 134
161      176202      PIOB= 176202
162      000130      PIOBV= 130
163      000240      PIOP= PRI5
164      176204      PIOC= 176204
165      176206      PIOMC= 176206
166
167      174020      QBXCSR= 174020
168      000110      QBXVEC= 110
169
170      ;
171      ; NOW, EQUATE SOME GENERAL REGISTERS.
172      ;
172      001000      ADR= 1000
173      001002      ADR1= ADR+2
174      001004      ADR2= ADR+4
175      001006      TEMP= ADR+6
176      001010      TEMP1= ADR+10
177      001012      TEMP2= ADR+12
178      001014      TEMP3= ADR+14
179      001016      K0= ADR+16
180      001020      K1= ADR+20
181      001022      K2= ADR+22
182      001024      K3= ADR+24
183      001026      K4= ADR+26
184      001030      K5= ADR+30
185      001032      K6= ADR+32
186      001034      K7= ADR+34
187
188      001036      QBXA= ADR+36
189
190      001016      SPD0= K0
191      001020      SPD1= K1
192      001022      SPD2= K2
193      001024      SPD3= K3
194      001026      SPD4= K4
195      001030      SPD5= K5
196      001032      SPD6= K6

```

```

;...1ST 4 WORDS ARE NON-EX SO THAT...
;...CONVENTIONAL SIZERS WILL WORK RIGHT...
;...AND LAST 8 WORDS BELONG TO MACRO-ODT...
;...(THE LAST BEING THE TRAP4 EMULATE FLAG).
; LOCAL 2K RAM STRAPPED... ;JSD REV B
;...TO RESPOND AT 28K(+4) TO 30K... ;JSD REV B
; RAM START ADDRESS FOR FALCON-PLUS ;JSD REV B
; MAX RAM SPACE (WORDS) FOR FALCON-PLUS ;JSD REV B
;...(AVOID MONITOR CODE AT 15000-OCT) ;JSD REV B
; "LINE CLOCK" (ON SLU2) VECTOR.
; "BREAK" (ON SLU1) OR Q-BUS "BHALT" VECTOR.

; SLU (DLART) 1 (CONSOLE TTY)...
;...1ST VECTOR (OF 2)...
;...AND PRIORITY.
; SLU (DLART) 2...
;...1ST VECTOR (OF 2)...
;...AND PRIORITY.

; PIO PORT A (INPUT)...
;...AND VECTOR.
; PIO PORT B (OUTPUT)...
;...AND VECTOR.
; INTERRUPT PRIORITY (FOR BOTH).
; PIO PORT C (PORT CONTROL)...
;...AND MODE CONTROL.

; Q-BUS EXERCISER CSR1 ADDRESS (OPTIONAL)...
;...AND VECTOR (AT PRI4).

;
; INSTRUCTION TEST SCRATCH LOCATIONS.

; Q-BUS EXER CSR ADDRESS (IF INSTALLED).
; DLART, VARIABLE SPEED COUNTERS.

```



INITIAL START AND CLEAR VECTORS.

```

229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249 010000
250 010000 010000
251 010000 000240
252 010002 013746 000030
253 010006 013746 000040
254 010012 013746 000042
255 010016 012700 000002
256 010022 005001
257 010024 012702 000200
258 010030 010021
259 010032 012721 000000
260 010036 022020
261 010040 077205
262 010042 012637 000042
263 010046 012637 000040
264 010052 012637 000030
265 010056 012737 000137 000200
266 010064 012737 010072 000202
267
268
269
270
271
272 010072 012700 001000
273 010076 005020
274 010100 020027 010000
275 010104 103774
276 010106 012706 003776
277 010112 012737 170000 000140
278 010120 012737 000340 000142
279 010126 012737 000001 167776
280
281 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
      MOV   @#30,-(SP)         ; RESET ???
      MOV   @#40,-(SP)         ; SAVE XDP EMT VECTOR...
      MOV   @#42,-(SP)         ; ...LOAD DEV ID AND UNIT...
      MOV   @2,R0              ; ...AND CHAIN RETURN.
      CLR   R1                 ; COAT THE ENTIRE VECTOR AREA...
      MOV   @200,R2            ; ...WITH A TRADITIONAL TRAP CATCHER.
1@:   MOV   RO,(R1)+           ;
      MOV   @EHALT,(R1)+       ;           .+2
      CMP   (RO)+,(RO)+       ;           EHALT
      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   RO,@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.
:
: .IIF P2, .PRINT -.2
: *** DEBUG, 100000 *** ;JSD REV B
MTPS  @PRIO                 ; AND LOWER THE CPU LEVEL.
    
```

QUICK-VERIFY EMT AND TRAP.

```

283
284
285
286
287
288
289
290
291
292 010140 016746 167664
293 010144 012767 010204 167656
294 010152 012767 000036 167654
295 010160 012700 104000
296 010164 000402
297 010166 012700 104400
298 010172 010067 000000
299 010176 104000
300 010200 000240
301 010202 000000
307 010204 012716 010216
308 010210 000002
309 010212 000240
310 010214 000000
316 010216 105200
317 010220 001364
318 010222 020027 104400
319 010226 001407
320 010230 012767 000032 167572
321 010236 012767 010204 167570
322 010244 000750
323
324 010246 012667 167556
325 010252 012767 035464 167554
326 010260 012700 010300
327 010264 104003
328 010266 104422
329 010270 016700 025516
330 010274 001420
331 010276 010007
332
333 010300 015 012 116
    010303 113 130 101
    010306 102 060 040
    010311 113 130 124
    010314 055 061 061
    010317 040 104 111
    010322 101 107 116
    010325 117 123 124
    010330 111 103 015
    010333 012 000

```

```

.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.
;
EMTTRP: MOV     EMTVEC, -(SP)      ; SAVE MONITORS EMT VECTOR.
        MOV     #4#, EMTVEC      ; SET EMT...
        MOV     #TRPVEC+2, TRPVEC ; ...AND RESET TRAP VECTORS.
        MOV     #EMT+0, R0       ; TEST EMT'S FIRST...
        BR      2#
1#:     MOV     #TRAP+0, R0      ; ...THEN TRAP'S.
2#:     MOV     R0, 3#
3#:     EMT+0                    ; EXECUTE EMT/TRAP.
        NOP                    ; DIDN'T WORK -- BIG TROUBLE...
        HALT                    ; ... (R0) HOLDS THE OFFENDER !!!
4#:     MOV     #5#, (SP)        ; EMT/TRAP SEEMS OK, TWEAK RETURN PC...
        RTI                    ; ...AND TEST THE "RTI" TOO.
        NOP                    ; W-T-F ???
        HALT                    ; RTI DOESN'T WORK !!!
5#:     INCB     R0              ; WE'RE HERE IF EVERYTHING WORKED...
        BNE     2#              ; ...BUMP THE OPCODE AND LOOP 'TIL DONE.
        CMP     R0, #TRAP       ; OPCODE = TRAP ??
        BEQ     6#              ; WE'RE DONE IF SO.
        MOV     #EMTVEC+2, EMTVEC ; OTHERWISE, RESET EMT...
        MOV     #4#, TRPVEC     ; ...AND SET TRAP VECTORS...
        BR      1#             ; ...AND GO 'ROUND ONCE.
6#:     MOV     (SP)+, EMTVEC    ; DONE, RESTORE EMT FOR MONITOR CALLS...
        MOV     #TRAPHAN, TRPVEC ; ...AND SET-UP OUR OWN TRAP HANDLER.
        MOV     #HELLO, R0
        EMT+3                    ; "GREETINGS".
        GETSMR                    ; GET INITIAL SWITCH SETTING...
        MOV     INSEG, R0        ; SINGLE SEGMENT SELECTED ??
        BEQ     DOALL            ; ...IF NOT, RUN 'EM ALL...
        MOV     R0, PC          ; ...ELSE, GO THERE.

```

```

HELLO: .ASCIZ <15><12>'NKXABO KXT-11 DIAGNOSTIC'<15><12>

```

```

.EVEN
.SBTTL

```

```

334
335

```

SECTION 1 -- T11 BASIC INSTRUCTION TESTS.

```

337          .SBTTL SECTION 1 -- T11 BASIC INSTRUCTION TESTS.
338
339 010336    DOALL:
340 010336 012767 036330 025762 S1:  MOV    #SEG1,SEGN    ; SEGMENT 1 TEXT.
341
342          ;*****
          .SBTTL      T1 -- BRANCHES, FORWARD, BACKWARD, AND CONDITIONAL
          ;*****
010344 104420 000001    T1:  BEGINTEST, 1          ; CHECK SWITCH REGISTER OPTIONS.
343 010350 000405    BRANCH: BR      2#
344 010352 000405    1#:   BR      3#
345 010354 104423 010344    4#:   ERROR,  T1          ; ** FORWARD BRANCH FAILED **
346 010360 012707 010374    2#:   MOV    #5#,PC
347 010364 000772    3#:   BR      1#
348 010366 000774    5#:   BR      4#
349 010370 104423 010344    ERROR,  T1          ; ** BACKWARD BRANCH FAILED **
350 010374
351 010374 000257    NOBIT: CCC
352 010376 103414    BCS      1#
353 010400 102413    BVS      1#
354 010402 001412    BEQ      1#
355 010404 100411    BHI      1#
356 010406 000260    NOP1
357 010410 103407    BCS      1#          ; CHECK NOP1 INSTRUCTION (OPCODE 260).
358 010412 102406    BVS      1#
359 010414 001405    BEQ      1#
360 010416 100404    BHI      1#
361 010420 002403    BLT      1#
362 010422 003402    BLE      1#
363 010424 101401    BLOS     1#
364 010426 101002    BHI      2#
365 010430
010430 104423 010344    1#:   ERROR,  T1          ; ** BRANCH FAILURE OR CC NOT 0000 **
366 010434
367 010434 000270    2#:   SEN
368 010436 100007    NOBIT: BPL      1#
369 010440 001406    BEQ      1#
370 010442 002005    BGE      1#
371 010444 003004    BGT      1#
372 010446 103403    BCS      1#
373 010450 101402    BLOS     1#
374 010452 103401    BLO      1#
375 010454 003402    BLE      2#
376 010456
010456 104423 010344    1#:   ERROR,  T1          ; ** BRANCH FAILURE OR CC NOT 1000 **
377 010462
378 010462 000270    2#:   SEN
379 010464 000262    VBIT: SEV
380 010466 102010    BVC      1#
381 010470 001407    BEQ      1#
382 010472 100006    BPL      1#
383 010474 103405    BCS      1#
384 010476 002404    BLT      1#
385 010500 003403    BLE      1#
386 010502 101402    BLOS     1#
387 010504 103401    BLO      1#
388 010506 003002    BGT      2#

```

PC 10456 = BRANCH FAILURE OR CC NOT 1000.

```

389 010510          10:
      010510 104423 010344      ERROR, T1          ;; *** BRANCH FAILURE OR CC NOT 1010 ***
390 010514          20:
391 010514 000270      CBIT:  SEN          ; SET N...
392 010516 000262      SEV          ;...V...
393 010520 000261      SEC          ;...AND C, NZVC=1011
394 010522 001406      BEQ          10
395 010524 100005      BPL          10
396 010526 102004      BVC          10
397 010530 002403      BLT          10
398 010532 003402      BLE          10
399 010534 101001      BHI          10
400 010536 002002      BGE          20
401 010540          10:
      010540 104423 010344      ERROR, T1          ;; *** BRANCH FAILURE OR CC NOT 1011 ***
402 010544          20:
403 010544 000270      ZBIT:  SEN          ; SET N...
404 010546 000262      SEV          ;...V...
405 010550 000261      SEC          ;...C...
406 010552 000264      SEZ          ;...AND Z, NZVC = 1111.
407 010554 001007      BNE          10
408 010556 100006      BPL          10
409 010560 102005      BVC          10
410 010562 103004      BCC          10
411 010564 002403      BLT          10
412 010566 003002      BGT          10
413 010570 101001      BHI          10
414 010572 001402      BEQ          20
415 010574          10:
      010574 104423 010344      ERROR, T1          ;; *** BRANCH FAILURE OR CC NOT 1111 ***
416 010600          20:
417
418
;*****
.SBTTL          T2 -- SET AND CLEAR ALL CONDITION CODES
;*****
419 010600 104420 000002      T2:  BEGINTEST, 2          ;; CHECK SWITCH REGISTER OPTIONS.
420 010606 104417          ALLCC:  SCC          ; NZVC = 1111
421 010610 104423 010600      CHECKCC+17          ; CHECK ALL BITS SET.
422          ERROR, T2          ;; *** CC BITS WRONG AFTER SCC ***
423 010614 000257          CCC          ; NZVC = 0000
424 010616 104400          CHECKCC+0
425 010620 104423 010600      ERROR, T2          ;; *** CC BITS WRONG AFTER CCC ***
426
427
;*****
.SBTTL          T3 -- TEST REGISTER SELECTION
;*****
428 010624 104420 000003      T3:  BEGINTEST, 3          ;; CHECK SWITCH REGISTER OPTIONS.
429 010630 012700 000001      REGS:  MOV          #1,R0          ; LOAD UP R0 THRU R4.
430 010634 012701 000004      MOV          #4,R1
431 010640 012702 000020      MOV          #20,R2
432 010644 012703 000100      MOV          #100,R3
433 010650 012704 000400      MOV          #400,R4
434 010654 012705 001000      MOV          #1000,R5
435 010660 060600          ADD          R6,R0          ; R0 = #STACK...
436 010664 060500          ADD          R5,R0          ;...+ 1000...
          ADD          R4,R0          ;...+ 400...

```

T3 -- TEST REGISTER SELECTION

```

437 010666 050300          ADD      R3,R0          ;.... 100...
438 010670 060200          ADD      R2,R0          ;.... 20...
439 010672 060100          ADD      R1,R0          ;.... 4...
440 010674 020027 005523    CMP      R0,#STACK+1525 ;...SHOULD = THIS.
441 010700 001402          BEQ      10             ; BR IF OK.
442 010702 104423 010624    ERROR,   T3           ;; *** REGISTER SELECTION FAILURE ***
443 010706
444
445

```

10: ;\*\*\*\*\*  
.SBTTL T4 -- JMP INSTRUCTION -- MODE 1  
;\*\*\*\*\*

```

010706 104420 000004
446 010712 012700 010726    T4:      BEGINTEST, 4      ;; CHECK SWITCH REGISTER OPTIONS.
447 010716 000277          JMP1:    MOV      #10,R0    ; TEST JMP INSTRUCTION MODE 1
448 010720 000110          SCC
449 010722 104423 010706    JMP      (R0)
450 010726 104417          ERROR,   T4             ;; *** MODE 1 JMP FAILED ***
451 010730 104423 010706    10:     CHECKCC.17
452 010734 020027 010726    ERROR,   T4             ;; *** CC BITS ALTERED ON JMP ***
453 010740 001402          CMP      R0,#10
454 010742 104423 010706    BEQ      20
455 010746          ERROR,   T4             ; CONTINUE IF R0 IS OK
456
457          ; *** RO INCORRECT AFTER JMP ***

```

20: ;\*\*\*\*\*  
.SBTTL T5 -- JMP INSTRUCTION -- MODES 2 AND 3  
;\*\*\*\*\*

```

010746 104420 000005
458 010752 012700 010766    T5:      BEGINTEST, 5      ;; CHECK SWITCH REGISTER OPTIONS.
459 010756 000277          JMP2:    MOV      #10,R0    ; TEST JMP INSTRUCTION MODE 2
460 010760 000120          SCC
461 010762 104423 010746    JMP      (R0).           ; POSSIBLE "Z" ASSEMBLY ERROR O.K. ;JSD REV B
462 010766 104417          ERROR,   T5             ;; *** MODE 2 JMP FAILED ***
463 010770 104423 010746    10:     CHECKCC.17
464 010774 020027 010770    ERROR,   T5             ;; *** CC BITS ALTERED AFTER JMP ***
465 011000 001402          CMP      R0,#10+2       ; IS THERE AUTO INC.?
466 011002 104423 010746    BEQ      JMP3
467          ERROR,   T5             ;; *** MODE 2 FAILED ON JMP ***
468 011006 012767 011036 167772 JMP3:    MOV      #30,TEMP
469 011014 012767 011052 167766    MOV      #40,TEMP+2
470 011022 012700 001006    MOV      #TEMP,R0
471 011026 000277          SCC
472 011030 000130          JMP      B(R0).
473 011032 104423 010746    ERROR,   T5             ;; *** MODE 3 JMP FAILED ***
474 011036 027067 000000 000006 30:     CMP      B(R0),40
475 011044 001402          BEQ      40             ; IS THERE AUTO INC.?
476 011046 104423 010746    ERROR,   T5             ;; *** MODE 3 JMP FAILED ***
477 011052
478
479

```

40: ;\*\*\*\*\*  
.SBTTL T6 -- JMP INSTRUCTION -- MODES 4 AND 5  
;\*\*\*\*\*

```

011052 104420 000006
480 011056 012700 011074    T6:      BEGINTEST, 6      ;; CHECK SWITCH REGISTER OPTIONS.
481 011062 000277          JMP4:    MOV      #30,R0    ; TEST JMP INSTRUCTION MODE 4
482 011064 000140          SCC
483 011066 104423 011052    JMP      -(R0)
484 011072 000402          ERROR,   T6             ;; *** DIDN'T JMP AT ALL ***
                                BR      40             ; JMP SHOULD LAND HERE

```



PC 11066 = DIDN'T JUMP AT ALL.

```

485 011074          30:
    011074 104423 011052      ERROR, T6          ;; *** MODE 4 JUMP FAILED. ***
486 011100 022700 011072      CMP      030-2,R0      ; CHECK R0
487 011104 001402          BEQ      JMP5
488 011106 104423 011052      ERROR, T6          ;; *** R0 INCORRECT AFTER JUMP ***
489
490 011112 012767 011140 167670 JMP5:  MOV      030,TEMP1      ; TEST JUMP INSTRUCTION MODE 5
491 011120 012700 001010      MOV      0TEMP1,R0
492 011124 012767 011144 167654      MOV      040,TEMP1-2
493 011132 000150          JMP      0-(R0)
494 011134 104423 011052      ERROR, T6          ;; *** MODE 5 JUMP DIDN'T ***
495 011140          30:
    011140 104423 011052      ERROR, T6          ;; *** AUTO-DECR FAILED ON MODE 5 JUMP ***
496 011144 022700 001006      CMP      0TEMP1-2,R0 ; CHECK R0
497 011150 001402          BEQ      50
498 011152 104423 011052      ERROR, T6          ;; *** R0 INCORRECT AFTER JMP5 ***
499 011156          50:
500
501

```

```

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

```

```

011156 104420 000007      T7:  BEGINTEST, 7          ;; CHECK SWITCH REGISTER OPTIONS.
502 011162 012703 011204      JMP6:  MOV      010+6,R3
503 011166 000163 177772      JMP      -6(R3)
504 011172 104423 011156      ERROR, T7          ;; *** MODE 6 JUMP FAILED ***
505 011176 020327 011204      10:  CMP      R3,010+6      ; CHECK R3
506 011202 001402          BEQ      20
507 011204 104423 011156      ERROR, T7          ;; *** R3 WRONG AFTER JMP6 ***
508 011210 000167 000004      20:  JMP      30--4(PC)      ; TEST JUMP INSTRUCTION MODE 6
509 011214 104423 011156      ERROR, T7          ;; *** JUMP FAILED ***
510
511 011220 012703 011234      30:  MOV      0JMP7,R3
512 011224 000163 000000      JMP      0(R3)
513 011230 104423 011156      ERROR, T7          ;; *** JUMP FAILED ***
514
515 011234 012703 001006      JMP7:  MOV      0TEMP,R3
516 011240 012713 011254      MOV      010,(R3)
517 011244 000173 000000      JMP      0(R3)
518 011250 104423 011156      ERROR, T7          ;; *** MODE 7 JUMP FAILED ***
519 011254 012713 011274      10:  MOV      030,(R3)      ; TEST JUMP INSTRUCTION MODE 7
520 011260 012700 001002      MOV      0TEMP-4,R0
521 011264 000170 000004      JMP      04(R0)
522 011270 104423 011156      ERROR, T7          ;; *** JUMP FAILED ***
523 011274 012767 011316 167504 30:  MOV      040,TEMP
524 011302 012700 001006      MOV      0TEMP,R0
525 011306 000170 000000      JMP      00(R0)
526 011312 104423 011156      ERROR, T7          ;; *** MODE 7 JUMP FAILED. ***
527 011316          40:
528
529

```

```

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

```

```

011316 104420 000010      T10:  BEGINTEST, 10          ;; CHECK SWITCH REGISTER OPTIONS.
530 011322 012706 003776      JSRTST: MOV     0STACK,SP ; SET UP STACK POINTER.
531 011326 000277
532 011330 004767 000004      SCC
533 011334          JSR      PC,20
          10:

```

PC 11334 = JSR INSTRUCTION FAILED.

011334 104423 011316  
 534 011340 104417  
 535 011342 104423 011316  
 536 011346 022706 003774  
 537 011352 001402  
 538 011354 104423 011316  
 539 011360 022716 011334  
 540 011364 001402  
 541 011366 104423 011316  
 542  
 543 011372 012716 011404  
 544 011376 000207  
 545 011400 104423 011316  
 546 011404 022706 003774  
 547 011410 001402  
 548 011412 104423 011316  
 549 011416  
 550  
 551  
 552  
 553  
 554

```

20:  ERROR, T10          ;: *** JSR INSTRUCTION FAILED ***
      CHECKCC+17
40:  ERROR, T10          ;: *** CC BITS CHANGED ON JSR ***
      CMP  #STACK-2,SP
      BEQ  50            ; BR IF STACK IS RIGHT.
      ERROR, T10        ;: *** STACK WRONG AFTER JSR ***
50:  CMP  #10,(SP)
      BEQ  60            ; BR IF STACKED PC IS RIGHT.
      ERROR, T10        ;: *** SP DID NOT HAVE CORRECT RETURN ADDRESS ***
60:  MOV  #70,(SP)      ; CHANGE RETURN PC.
      RTS  PC
      ERROR, T10        ;: *** RTS INSTRUCTION FAILED ***
70:  CMP  #STACK,SP
      BEQ  80            ; BR IF STACK PROPERLY RESTORED.
      ERROR, T10        ;: *** SP WAS NOT RESTORED BY RTS INSTRUCTION ***
80:

```

```

; .SBTTL
; .SBTTL  DESTINATION MODE 0
;
; *****
; .SBTTL  T11 -- TSTB, CLRB, AND MOVB
; *****

```

011416 104420 000011  
 555 011422 012706 003774  
 556 011426 000277  
 557 011430 105000  
 558 011432 104404  
 559 011434 104423 011416  
 560 011440 105700  
 561 011442 104404  
 562 011444 104423 011416  
 563 011450 112701 000377  
 564 011454 104410  
 565 011456 104423 011416  
 566 011462 105701  
 567 011464 104410  
 568 011466 104423 011416  
 569  
 570

```

T11:  BEGINTEST, 11      ;: CHECK SWITCH REGISTER OPTIONS.
TSTB0: MOV  #STACK,SP   ;: INIT THE STACK.
      SCC
      CLRB R0           ;: CLEAR THE REGISTER
      CHECKCC+4         ;: CHECK FOR CC = 4
      ERROR, T11        ;: *** CLRB FAILED ***
      TSTB R0           ;: CHECK IT
      CHECKCC+4         ;: CHECK FOR CC = 4
      ERROR, T11        ;: *** TSTB FAILED. ***
      MOVB #577,R1      ;: LOAD THE REGISTER
      CHECKCC+10        ;: CHECK FOR CC = 10
      ERROR, T11        ;: *** INCORRECT CC BITS ***
      TSTB R1           ;: CHECK IT
      CHECKCC+10        ;: CHECK FOR CC = 10
      ERROR, T11        ;: *** INCORRECT CC BITS ***

```

```

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

```

011472 104420 000012  
 571 011476 000277  
 572 011500 152702 000377  
 573 011504 104411  
 574 011506 104423 011472  
 575 011512 122702 000377  
 576 011516 001402  
 577 011520 104423 011472  
 578 011524 112700 000077  
 579 011530 120002  
 580 011532 100002  
 581 011534 104423 011472  
 582 011540 120200  
 583 011542 100402

```

T12:  BEGINTEST, 12      ;: CHECK SWITCH REGISTER OPTIONS.
CMPB0: SCC
      BISB #577,R2      ;: LOAD REGISTER
      CHECKCC+11        ;: CHECK FOR CC = 11
      ERROR, T12        ;: *** INCORRECT CC BITS ***
      CMPB #577,R2      ;: CHECK COMPARE
      BEQ  20            ;: CONTINUE IF OK
      ERROR, T12        ;: *** BISB OR CMPB INSTRUCTION FAILED ***
20:  MOVB #77,R0
      CMPB R0,R2        ;: CHECK IT AGAIN
      BPL  30            ;: CONTINUE IF OK
      ERROR, T12        ;: *** CMPB INSTRUCTION FAILED (WRONG CC) ***
30:  CMPB R2,R0
      BMI  40            ;: CONTINUE IF OK

```

PC 11544 = CNPB INSTRUCTION FAILED (WRONG CC).

```

584 011544 104423 011472          ERROR,  T12          ;; *** CNPB INSTRUCTION FAILED (WRONG CC) ***
585 011550 112702 000377          40:  MOVB   #377,R2    ; LOAD REGISTER, SIGN EXTEND
586 011554 122702 000377          CNPB   #377,R2    ; CHECK IF BYTE INSTRUCTION
587 011560 001402          BEQ    50          ; CONTINUE IF OK
588 011562 104423 011472          ERROR,  T12          ;; *** CNPB BECAME CNP INSTRUCTION ***
589 011566 112702 000377          50:  MOVB   #377,R2    ; LOAD REGISTER, SIGN EXTEND
590 011572 120227 000377          CNPB   R2,#377    ; CHECK IF BYTE INSTRUCTION
591 011576 001402          BEQ    60          ; CONTINUE IF OK
592 011600 104423 011472          60:  ERROR,  T12          ;; *** MOVB OR CNPB FAILED ***
593 011604
594
595

```

\*\*\*\*\*  
.SBTTL T13 -- BICB AND BITB  
\*\*\*\*\*

```

011604 104420 000013          T13:  BEGINTEST, 13    ;; CHECK SWITCH REGISTER OPTIONS.
596 011610 112703 000377          BICB0: MOVB   #377,R3 ; LOAD REGISTER
597 011614 112700 000252          MOVB   #252,R0    ; PLACE #252 IN R0
598 011620 000277          SCC
599 011622 140003          BICB   R0,R3      ; CLEAR EVERY OTHER BIT
600 011624 104401          CHECKCC+1        ; CHECK FOR CC = 1
601 011626 104423 011604          ERROR,  T13      ;; *** INCORRECT CC BITS ***
602 011632 130003          BITB   R0,R3      ; CHECK IT
603 011634 001402          BEQ    40          ; CONTINUE IF OK
604 011636 104423 011604          40:  ERROR,  T13      ;; *** BICB OR BITB INSTRUCTION FAILED ***
605 011642 132703 000125          BITB   #125,R3   ; CHECK IT
606 011646 104401          CHECKCC+1        ; CHECK FOR CC = 1
607 011650 104423 011604          ERROR,  T13      ;; *** INCORRECT CC BITS ***
608 011654 150003          BISB   R0,R3      ; SET THE BITS THAT WERE CLEARED
609 011656 100402          BMI   60
610 011660 104423 011604          60:  ERROR,  T13      ;; *** BISB INSTRUCTION FAILED ***
611 011664 142703 000177          BICB   #177,R3   ; CLEAR ALL THE BITS EXCEPT FOR SIGN
612 011670 104411          CHECKCC+11      ; CHECK FOR CC = 11
613 011672 104423 011604          ERROR,  T13      ;; *** INCORRECT CC BITS ***
614 011676 132703 000377          BITB   #377,R3   ; CHECK IT
615 011702 104411          CHECKCC+11      ; CHECK FOR CC = 11
616 011704 104423 011604          ERROR,  T13      ;; *** INCORRECT CC BITS ***
617
618

```

\*\*\*\*\*  
.SBTTL T14 -- INCB AND DECB  
\*\*\*\*\*

```

011710 104420 000014          T14:  BEGINTEST, 14    ;; CHECK SWITCH REGISTER OPTIONS.
619 011714 112704 000177          INCB0: MOVB   #177,R4 ; R4 = 177
620 011720 000261          SEC
621 011722 105204          INCB   R4         ; ADD ONES INTO REG. 4
622 011724 104413          CHECKCC+13      ; CHECK FOR CC = 13
623 011726 104423 011710          ERROR,  T14      ;; *** INCORRECT CC BITS ***
624 011732 112704 000376          MOVB   #376,R4
625 011736 105204          INCB   R4
626 011740 104411          CHECKCC+11      ; CHECK FOR CC = 11
627 011742 104423 011710          ERROR,  T14      ;; *** INCORRECT CC BITS ***
628 011746 105204          INCB   R4
629 011750 104405          CHECKCC+5       ; CHECK FOR CC = 5
630 011752 104423 011710          ERROR,  T14      ;; *** INCORRECT CC BITS ***
631 011756 105204          INCB   R4
632 011760 104401          CHECKCC+1       ; CHECK FOR CC = 1
633 011762 104423 011710          ERROR,  T14      ;; *** INCORRECT CC BITS ***
634 011766 122704 000001          CNPB   #1,R4     ; CHECK IT

```

PC 11762 = INCORRECT CC BITS.

635 011772 001402  
 636 011774 104423 011710  
 637 012000 000261  
 638 012002 105304  
 639 012004 104405  
 640 012006 104423 011710  
 641 012012 105304  
 642 012014 104411  
 643 012016 104423 011710  
 644 012022 012704 000200  
 645 012026 105304  
 646 012030 104403  
 647 012032 104423 011710  
 648 012036 105304  
 649 012040 104401  
 650 012042 104423 011710  
 651  
 652

```

20: BEQ 20 ; CONTINUE IF OK
      ERROR, T14 ; ; *** INCB INSTRUCTION FAILED ***
      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  
 \*\*\*\*\*

012046 104420 000015  
 653 012052 112703 000252  
 654 012056 000277  
 655 012060 105103  
 656 012062 104401  
 657 012064 104423 012046  
 658 012070 122703 000125  
 659 012074 001402  
 660 012076 104423 012046  
 661 012102 000277  
 662 012104 105103  
 663 012106 104411  
 664 012110 104423 012046  
 665 012114 122703 000252  
 666 012120 001402  
 667 012122 104423 012046  
 668 012126 012703 000377  
 669 012132 000277  
 670 012134 105103  
 671 012136 104405  
 672 012140 104423 012046  
 673  
 674 012144 112700 000001  
 675 012150 105400  
 676 012152 104411  
 677 012154 104423 012144  
 678 012160 122700 000377  
 679 012164 001402  
 680 012166 104423 012144  
 681 012172 012700 000200  
 682 012176 105400  
 683 012200 104413  
 684 012202 104423 012144  
 685 012206 122700 000200  
 686 012212 001402  
 687 012214 104423 012144  
 688 012220

```

T15: BEGINTEST, 15 ; ; CHECK SWITCH REGISTER OPTIONS.
COMB0: 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 20 ; CONTINUE IF OK
      ERROR, T15 ; ; *** COMB INSTRUCTION FAILED ***
20: SCC
      COMB R3 ; COMPLEMENT BACK
      CHECKCC+11 ; CHECK FOR CC = 11
      ERROR, T15 ; ; *** INCORRECT CC BITS ***
      CMPB #252,R3 ; CHECK IT
      BEQ 30 ; CONTINUE IF OK
      ERROR, T15 ; ; *** COMB INSTRUCTION FAILED ***
30: 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 20 ; CONTINUE IF OK
      ERROR, NEGB0 ; ; *** NEGB INSTRUCTION FAILED ***
20: MOV #200,R0
      NEGB R0 ; 2'S COMPLEMENT
      CHECKCC+13 ; CHECK FOR CC = 13
      ERROR, NEGB0 ; ; *** INCORRECT CC BITS ***
      CMPB #200,R0 ; CHECK IT
      BEQ 30 ; CONTINUE IF OK
      ERROR, NEGB0 ; ; *** NEGB FAILED ***
30:
  
```

PC 12214 = NEGB FAILED.

689  
690

```

012220 104420 000016
691 012224 112701 000040
692 012230 000257
693 012232 106101
694 012234 106101
695 012236 104412
696 012240 104423 012220
697 012244 122701 000200
698 012250 001402
699 012252 104423 012220
700 012256 106101
701 012260 104407
702 012262 104423 012220
703 012266 106101
704 012270 122701 000001
705 012274 001402
706 012276 104423 012220
707 012302
708 012302 112702 000004
709 012306 000257
710 012310 106002
711 012312 106002
712 012314 122702 000001
713 012320 001402
714 012322 104423 012302
715 012326 106002
716 012330 104407
717 012332 104423 012302
718 012336 106002
719 012340 104412
720 012342 104423 012302
721 012346 122702 000200
722 012352 001402
723 012354 104423 012302
724 012360
725
726

```

```

;*****
.SBTTL      T16 -- ROLB AND RORB
;*****
T16:  BEGINTEST, 16      ; CHECK SWITCH REGISTER OPTIONS.
ROLB0: MOVB    #40,R1    ; LOAD REGISTER
      CCC      ; CLEAR FLAGS
      ROLB    R1        ; SHIFT
      CHECKCC+12      ; CHECK FOR CC = 12
      ERROR,  T16      ; *** INCORRECT CC BITS ***
      CPB     #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
      CPB     #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        ;
      CPB     #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 ***
      CPB     #200,R2   ; CHECK IT
      BEQ     2#       ; CONTINUE IF OK
      ERROR,  RORB0    ; *** RORB FAILED ***
2#:

```

```

012360 104420 000017
727 012364 112703 000040
728 012370 000257
729 012372 106303
730 012374 106303
731 012376 104412
732 012400 104423 012360
733 012404 122703 000200
734 012410 001402
735 012412 104423 012360
736 012416 106303
737 012420 104407
738 012422 104423 012360
739 012426 106303

```

```

;*****
.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
      ERROR,  T17      ; *** INCORRECT CC BITS ***
      CPB     #200,R3   ; CHECK IT
      BEQ     4#       ; CONTINUE IF OK
      ERROR,  T17      ; *** ASLB INSTRUCTION FAILED ***
4#:   ASLB    R3        ; SHIFT
      CHECKCC+7      ; CHECK FOR CC = 7
      ERROR,  T17      ; *** INCORRECT CC BITS ***
      ASLB    R3        ; SHIFT

```

PC 12422 = INCORRECT CC BITS.

```

740 012430 104404          CHECKCC+4          ; CHECK FOR CC = 4
741 012432 104423 012360  ERROR, T17          ;; *** INCORRECT CC BITS ***
742
743 012436 112704 000004  ASRBO: MOVB #4,R4          ; LOAD REGISTER
744 012442 000257          CCC          ; CLEAR FLAGS
745 012444 106204          ASRB R4          ; SHIFT
746 012446 106204          ASRB R4          ;
747 012450 122704 000001  CMPB #1,R4          ; CHECK IT
748 012454 001402          BEQ 21          ; CONTINUE IF OK
749 012456 104423 012436  ERROR, ASRBO          ;; *** ASRB INSTRUCTION FAILED ***
750 012462 106204          ASRB R4          ; SHIFT
751 012464 104407          CHECKCC+7          ; CHECK FOR CC = 7
752 012466 104423 012436  ERROR, ASRBO          ;; *** INCORRECT CC BITS ***
753 012472 106204          ASRB R4          ; SHIFT
754 012474 104404          CHECKCC+4          ; CHECK FOR CC = 4
755 012476 104423 012436  ERROR, ASRBO          ;; *** INCORRECT CC BITS ***
756 012502 112703 000202  MOVB #202,R3        ; LOAD REGISTER
757 012506 106203          ASRB R3          ; SHIFT
758 012510 106203          ASRB R3          ;
759 012512 104411          CHECKCC+11         ; CHECK FOR CC = 11
760 012514 104423 012436  ERROR, ASRBO          ;; *** INCORRECT CC BITS ***
761 012520 122703 000340  CMPB #340,R3        ; CHECK IT
762 012524 001402          BEQ 31          ; CONTINUE IF OK
763 012526 104423 012436  ERROR, ASRBO          ;; *** ASRB FAILED ***
764 012532
765
766
;*****
.SBTTL T20 -- ADCB AND SBCB
;*****
T20: BEGINTEST, 20          ;; CHECK SWITCH REGISTER OPTIONS.
ADCB0: CLRB R0          ; CLEAR THE REGISTER
       CCC          ; CLEAR FLAGS
       ADCB R0          ; ADD C BIT = 0
       CHECKCC+4          ; CHECK FOR CC = 4
       ERROR, T20          ;; *** INCORRECT CC BITS ***
       SEC          ; C=1
       ADCB R0          ; ADD C BIT=1
       SEC          ; C=1
       ADCB R0          ; AGAIN
       CHECKCC+0          ; CHECK FOR CC = 0
       ERROR, T20          ;; *** INCORRECT CC BITS ***
       CMPB #2,R0          ; CHECK IT
       BEQ 41          ; CONTINUE IF OK
       ERROR, T20          ;; *** ADCB INSTRUCTION FAILED ***
41: MOVB #177,R0          ; LOAD LARGEST POSITIVE NUMBER
     SEC          ; C=1
     ADCB R0          ; ADD C BIT=1
     CHECKCC+12         ; CHECK FOR CC = 12
     ERROR, T20          ;; *** INCORRECT CC BITS ***
     CMPB #200,R0          ; CHECK IT
     BEQ 61          ; CONTINUE IF OK
     ERROR, T20          ;; *** ADCB INSTRUCTION FAILED ***
61: MOVB #377,R0          ; LOAD -1
     SEC          ; C=1
     ADCB R0          ; ADD C BIT=1
     CHECKCC+5          ; CHECK FOR CC = 5
     ERROR, T20          ;; *** INCORRECT CC BITS ***

```

PC 12644 = INCORRECT CC BITS.

794  
795 012650 112701 000003  
796 012654 000257  
797 012656 105601  
798 012660 104400  
799 012662 104423 012650  
800 012666 122701 000003  
801 012672 001402  
802 012674 104423 012650  
803 012700 000261  
804 012702 105601  
805 012704 000261  
806 012706 105601  
807 012710 104400  
808 012712 104423 012650  
809 012716 122701 000001  
810 012722 001402  
811 012724 104423 012650  
812 012730 000261  
813 012732 105601  
814 012734 104404  
815 012736 104423 012650  
816 012742 000261  
817 012744 105601  
818 012746 104411  
819 012750 104423 012650  
820 012754 122701 000377  
821 012760 001402  
822 012762 104423 012650  
823 012766 112701 000200  
824 012772 000261  
825 012774 105601  
826 012776 104402  
827 013000 104423 012650  
828  
829

```

SBCB0:  MOVB  #3,R1      ; LOAD REGISTER
        CCC                    ; CLEAR FLAGS
        SBCB  R1             ; SUBTRACT C BIT=0
        CHECKCC*0            ; CHECK FOR CC = 0
        ERROR, SBCB0         ; *** INCORRECT CC BITS ***
        CMPS #3,R1          ; CHECK IT
        BEQ  2:              ; CONTINUE IF OK
        ERROR, SBCB0         ; *** SBCB INSTRUCTION FAILED ***
2:      SEC                    ; C=1
        SBCB  R1             ; SUBTRACT C BIT=1
        SEC                    ; C=1
        SBCB  R1             ;
        CHECKCC*0            ; CHECK FOR CC = 0
        ERROR, SBCB0         ; *** INCORRECT CC BITS ***
        CMPS #1,R1          ; CHECK IT
        BEQ  3:              ; CONTINUE IF OK
        ERROR, SBCB0         ; *** SBCB INSTRUCTION FAILED ***
3:      SEC                    ; C=1
        SBCB  R1             ; SUBTRACT C BIT=1
        CHECKCC*4            ; CHECK FOR CC = 4
        ERROR, SBCB0         ; *** INCORRECT CC BITS ***
        SEC                    ; C=1
        SBCB  R1             ; SUBTRACT C BIT = 1
        CHECKCC*11           ; CHECK FOR CC = 11
        ERROR, SBCB0         ; *** INCORRECT CC BITS ***
        CMPS #377,R1        ; CHECK IT
        BEQ  4:              ; CONTINUE IF OK
        ERROR, SBCB0         ; *** SBCB INSTRUCTION FAILED ***
4:      MOVB  #200,R1        ; LOAD R1
        SEC                    ; C=1
        SBCB  R1             ; SUBTRACT C BIT = 1
        CHECKCC*2            ; CHECK FOR CC = 2
        ERROR, SBCB0         ; *** INCORRECT CC BITS ***

```

```

;*****
.SBTL   T21 -- TST, CLR, AND MOV
;*****
T21:   BEGINTEST, 21      ; CHECK SWITCH REGISTER OPTIONS.
MOV0:  SCC
        CLR  R0            ; CLEAR THE REGISTER
        CHECKCC*4          ; CHECK FOR CC = 4
        ERROR, T21         ; *** INCORRECT CC BITS ***
        TST  R0            ; CHECK IT
        CHECKCC*4          ; CHECK FOR CC = 4
        ERROR, T21         ; *** INCORRECT CC BITS ***
        MOV  #177777,R4    ; LOAD THE REGISTER
        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

```

013004 104420 000021  
830 013010 000277  
831 013012 005000  
832 013014 104404  
833 013016 104423 013004  
834 013022 005700  
835 013024 104404  
836 013026 104423 013004  
837 013032 012704 177777  
838 013036 010401  
839 013040 104410  
840 013042 104423 013004  
841 013046 005701  
842 013050 104410  
843 013052 104423 013004  
844 013056 020401  
845 013060 001402  
846 013062 104423 013004  
847 013066 000263

PC 13062 = R1 WRONG AFTER MOV.

848 013070 010000  
 849 013072 104405  
 850 013074 104423 013004  
 851  
 852

MOV R0,R0  
 CHECKCC.5  
 ERROR, T21 ;: \*\*\* INCORRECT CC BITS \*\*\*

;\*\*\*\*\*  
 .SBTTL T22 -- CMP AND BIS  
 ;\*\*\*\*\*

013100 104420 000022  
 853 013104 000277  
 854 013106 012700 177777  
 855 013112 050002  
 856 013114 104411  
 857 013116 104423 013100  
 858 013122 020002  
 859 013124 001402  
 860 013126 104423 013100  
 861 013132 022702 000077  
 862 013136 100002  
 863 013140 104423 013100  
 864 013144 020227 000077  
 865 013150 100402  
 866 013152 104423 013100  
 867 013156  
 868  
 869

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 20 ; CONTINUE IF OK  
 ERROR, T22 ;: \*\*\* BIS OR CMP INSTRUCTION FAILED \*\*\*  
 20: CMP #77,R2 ; CHECK IT AGAIN  
 BPL 30 ; CONTINUE IF OK  
 ERROR, T22 ;: \*\*\* CMP INSTRUCTION FAILED (WRONG CC) \*\*\*  
 30: CMP R2,#77 ; ONCE MORE  
 BMI 40 ; CONTINUE IF OK  
 ERROR, T22 ;: \*\*\* BIC OR CMP FAILED \*\*\*  
 40:

;\*\*\*\*\*  
 .SBTTL T23 -- BIC AND BIT  
 ;\*\*\*\*\*

013156 104420 000023  
 870 013162 012703 177777  
 871 013166 012700 001006  
 872 013172 012710 125252  
 873 013176 000277  
 874 013200 041003  
 875 013202 104401  
 876 013204 104423 013156  
 877 013210 031003  
 878 013212 001402  
 879 013214 104423 013156  
 880 013220 032703 052525  
 881 013224 104401  
 882 013226 104423 013156  
 883 013232 052703 125252  
 884 013236 100402  
 885 013240 104423 013156  
 886 013244 042703 077777  
 887 013250 104411  
 888 013252 104423 013156  
 889 013256 012700 177777  
 890 013262 030003  
 891 013264 104411  
 892 013266 104423 013156  
 893 013272 000263  
 894 013274 040000  
 895 013276 104405  
 896 013300 104423 013156  
 897 013304 005700  
 898 013306 001402

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 10 ; CONTINUE IF OK  
 ERROR, T23 ;: \*\*\* BIC OR BIT INSTRUCTION FAILED \*\*\*  
 10: 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 20 ; CONTINUE IF OK  
 ERROR, T23 ;: \*\*\* BIT OR BIS INSTRUCTION FAILED \*\*\*  
 20: BIC #77777,R3 ; CLEAR ALL THE BITS EXCEPT FOR SIGN  
 CHECKCC.11 ; CHECK FOR CC = 11  
 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 30 ;



PC 13310 = BIC FAILED.

899 013310 104423 013156  
900 013314  
901  
902

30: ERROR, T23 ;: \*\*\* BIC FAILED \*\*\*

;\*\*\*\*\*  
.SBTTL T24 -- INC AND DEC  
;\*\*\*\*\*

013314 104420 000024  
903 013320 012704 077777  
904 013324 000261  
905 013326 005204  
906 013330 104413  
907 013332 104423 013314  
908 013336 012704 177776  
909 013342 005204  
910 013344 104411  
911 013346 104423 013314  
912 013352 005204  
913 013354 104405  
914 013356 104423 013314  
915 013362 005204  
916 013364 104401  
917 013366 104423 013314  
918 013372 022704 000001  
919 013376 001402  
920 013400 104423 013314  
921 013404 000261  
922 013406 005304  
923 013410 104405  
924 013412 104423 013314  
925 013416 005304  
926 013420 104411  
927 013422 104423 013314  
928 013426 012704 100000  
929 013432 005304  
930 013434 104405  
931 013436 104423 013314  
932 013442 005304  
933 013444 104401  
934 013446 104423 013314  
935  
936

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 #0 ;: FAILED  
ERROR, T24 ;: \*\*\* INC INSTRUCTION FAILED \*\*\*  
40: 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  
;\*\*\*\*\*

013452 104420 000025  
937 013456 012703 125252  
938 013462 000277  
939 013464 005103  
940 013466 104401  
941 013470 104423 013452  
942 013474 022703 052525  
943 013500 001402  
944 013502 104423 013452  
945 013506 000277  
946 013510 005103  
947 013512 104411  
948 013514 104423 013452  
949 013520 022703 125252

T25: BEGINTEST, 25 ;: CHECK SWITCH REGISTER OPTIONS.  
COMO: MOV #125252,R3 ;: LOAD EVERY OTHER BIT  
SCC  
COM R3 ;: 1'S COMPLEMENT  
CHECKCC.1 ;: CHECK FOR CC = 1  
ERROR, T25 ;: \*\*\* INCORRECT CC BITS \*\*\*  
CMP #52525,R3 ;: CHECK IT  
BEQ #0 ;: CONTINUE IF OK  
ERROR, T25 ;: \*\*\* COM INSTRUCTION FAILED \*\*\*  
20: SCC  
COM R3 ;: COMPLEMENT BACK  
CHECKCC.11 ;: CHECK FOR CC = 11  
ERROR, T25 ;: \*\*\* INCORRECT CC BITS \*\*\*  
CMP #125252,R3 ;: CHECK IT

PC 13514 = INCORRECT CC BITS.

```

950 013524 001402
951 013526 104423 013452
952 013532 012703 177777
953 013536 000277
954 013540 005103
955 013542 104405
956 013544 104423 013452
957
958 013550 012700 000001
959 013554 005400
960 013556 104411
961 013560 104423 013550
962 013564 022700 177777
963 013570 001402
964 013572 104423 013550
965 013576 012700 100000
966 013602 005400
967 013604 104413
968 013606 104423 013550
969 013612 022700 100000
970 013616 001402
971 013620 104423 013550
972 013624
973
974
    
```

```

30: BEQ 30 ; CONTINUE IF OK
      ERROR, T25 ; ** COM INSTRUCTION FAILED **
      MOV #177777,R3
      SCC
      COM R3
      CHECKCC+5 ; CHECK FOR CC = 5
      ERROR, T25 ; ** INCORRECT CC BITS **

NEGO: MOV #1,R0 ; LOAD THE REGISTER
      NEG R0 ; 2'S COMPLEMENT
      CHECKCC+11 ; CHECK FOR CC = 11
      ERROR, NEGO ; ** INCORRECT CC BITS **
      CMP #177777,R0 ; CHECK IT
      BEQ 20 ; CONTINUE IF OK
      ERROR, NEGO ; ** NEG INSTRUCTION FAILED **

20: MOV #100000,R0
      NEG R0 ; 2'S COMPLEMENT
      CHECKCC+13 ; CHECK FOR CC = 13
      ERROR, NEGO ; ** INCORRECT CC BITS **
      CMP #100000,R0 ; CHECK IT
      BEQ 30 ; CONTINUE IF OK
      ERROR, NEGO ; ** NEG FAILED **

30:
    
```

\*\*\*\*\*  
.SBTTL T26 -- ROL AND ROR  
\*\*\*\*\*

```

013624 104420 000026
975 013630 012701 020000
976 013634 000257
977 013636 006101
978 013640 006101
979 013642 104412
980 013644 104423 013624
981 013650 022701 100000
982 013654 001402
983 013656 104423 013624
984 013662 006101
985 013664 104407
986 013666 104423 013624
987 013672 006101
988 013674 022701 000001
989 013700 001402
990 013702 104423 013624
991 013706
992 013706 012702 000004
993 013712 000257
994 013714 006002
995 013716 006002
996 013720 022702 000001
997 013724 001402
998 013726 104423 013706
999 013732 006002
1000 013734 104407
1001 013736 104423 013706
1002 013742 006002
1003 013744 104412
    
```

```

T26: BEGINTEST, 26 ; CHECK SWITCH REGISTER OPTIONS.
ROLO: MOV #20000,R1 ; LOAD REGISTER
      CCC ; CLEAR FLAGS
      ROL R1 ; SHIFT
      CHECKCC+12 ; CHECK FOR CC = 12
      ERROR, T26 ; ** INCORRECT CC BITS **
      CMP #100000,R1 ; CHECK IT
      BEQ 10 ; CONTINUE IF OK
      ERROR, T26 ; ** ROL INSTRUCTION FAILED **

10: ROL R1 ; SHIFT
      CHECKCC+7 ; CHECK FOR CC = 7
      ERROR, T26 ; ** INCORRECT CC BITS **
      ROL R1 ; SHIFT
      CMP #1,R1 ; CHECK IT
      BEQ 20 ; CONTINUE IF OK
      ERROR, T26 ; ** ROL FAILED **

20: RORO: MOV #4,R2 ; LOAD REGISTER
      CCC ; CLEAR FLAGS
      ROR R2 ; SHIFT
      ROR R2
      CMP #1,R2 ; CHECK IT
      BEQ 10 ; CONTINUE IF OK
      ERROR, RORO ; ** ROR INSTRUCTION FAILED **

10: ROR R2 ; SHIFT
      CHECKCC+7 ; CHECK FOR CC = 7
      ERROR, RORO ; ** INCORRECT CC BITS **
      ROR R2 ; SHIFT
      CHECKCC+12 ; CHECK FOR CC = 12
    
```

PC 13746 = INCORRECT CC BITS.

1004 013746 104423 013706  
 1005 013752 022702 100000  
 1006 013756 001402  
 1007 013760 104423 013706  
 1008 013764  
 1009  
 1010

```

20:
    ERROR, RORO          ;; *** INCORRECT CC BITS ***
    CMP    #100000,R2    ; CHECK IT
    BEQ    20            ; CONTINUE IF OK
    ERROR, RORO          ;; *** ROR FAILED ***
    
```

```

;*****
.SBTTL      T27 -- ASL AND ASR
;*****
    
```

1011 013764 104420 000027  
 1012 013770 012703 020000  
 1013 013774 000257  
 1014 013776 006303  
 1015 014000 006303  
 1016 014002 104412  
 1017 014004 104423 013764  
 1018 014010 022703 100000  
 1019 014014 001402  
 1020 014016 104423 013764  
 1021 014022 006303  
 1022 014024 104407  
 1023 014026 104423 013764  
 1024 014032 006303  
 1025 014034 104404  
 1026 014036 104423 013764  
 1027 014042 012704 000004  
 1028 014046 000257  
 1029 014050 006204  
 1030 014052 006204  
 1031 014054 022704 000001  
 1032 014060 001402  
 1033 014062 104423 014042  
 1034 014066 006204  
 1035 014070 104407  
 1036 014072 104423 014042  
 1037 014076 006204  
 1038 014100 104404  
 1039 014102 104423 014042  
 1040 014106 012703 100002  
 1041 014112 006203  
 1042 014114 006203  
 1043 014116 104411  
 1044 014120 104423 014042  
 1045 014124 022703 160000  
 1046 014130 001402  
 1047 014132 104423 014042  
 1048 014136  
 1049  
 1050

```

T27:  BEGINTEST, 27      ;; CHECK SWITCH REGISTER OPTIONS.
ASL0:  MOV    #20000,R3  ; LOAD REGISTER
        CCC                    ; CLEAR FLAGS
        ASL   R3          ; SHIFT
        CHECKCC+12        ; CHECK FOR CC = 12
        ERROR, T27        ;; *** INCORRECT CC BITS ***
        CMP    #100000,R3 ; CHECK IT
        BEQ    40         ; CONTINUE IF OK
        ERROR, T27        ;; *** ASL INSTRUCTION FAILED ***
40:    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          ; SHIFT
        CMP    #1,R4      ; CHECK IT
        BEQ    20         ; CONTINUE IF OK
        ERROR, ASR0       ;; *** ASR INSTRUCTION FAILED ***
20:    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          ; SHIFT
        CHECKCC+11        ; CHECK FOR CC = 11
        ERROR, ASR0       ;; *** INCORRECT CC BITS ***
        CMP    #160000,R3 ; CHECK IT
        BEQ    30         ; CONTINUE IF OK
        ERROR, ASR0       ;; *** ASR FAILED ***
    
```

```

;*****
.SBTTL      T30 -- ADC AND SBC
;*****
    
```

1051 014136 104420 000030  
 1052 014142 005000  
 1053 014144 000257  
 1054 014146 005500  
 1055 014150 104404

```

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
    
```

PC 14152 = INCORRECT CC BITS.

1055	014152	104423	014136		ERROR, T30	;; *** INCORRECT CC BITS ***
1056	014156	000261			SEC	; C=1
1057	014160	005500			ADC R0	; ADD C BIT=1
1058	014162	000261			SEC	; C=1
1059	014164	005500			ADC R0	; AGAIN
1060	014166	104400			CHECKCC=0	; CHECK FOR CC = 0
1061	014170	104423	014136		ERROR, T30	;; *** INCORRECT CC BITS ***
1062	014174	022700	000002		CMP #2,R0	; CHECK IT
1063	014200	001402			BEG 4#	; CONTINUE IF OK
1064	014202	104423	014136		ERROR, T30	;; *** ADC INSTRUCTION FAILED ***
1065	014206	012700	077777	4#:	MOV #77777,R0	; LOAD LARGEST POSITIVE NUMBER
1066	014212	000261			SEC	; C=1
1067	014214	005500			ADC R0	; ADD C BIT=1
1068	014216	104412			CHECKCC=12	; CHECK FOR CC = 12
1069	014220	104423	014136		ERROR, T30	;; *** INCORRECT CC BITS ***
1070	014224	022700	100000		CMP #100000,R0	; CHECK IT
1071	014230	001402			BEG 6#	; FAILED
1072	014232	104423	014136		ERROR, T30	;; *** ADC INSTRUCTION FAILED ***
1073	014236	012700	177777	6#:	MOV #-1,R0	; LOAD -1
1074	014242	000261			SEC	; C=1
1075	014244	005500			ADC R0	; ADD C BIT=1
1076	014246	104405			CHECKCC=5	; CHECK FOR CC = 5
1077	014250	104423	014136		ERROR, T30	;; *** INCORRECT CC BITS ***
1078						
1079	014254	012701	000003	SBC0:	MOV #3,R1	; LOAD REGISTER
1080	014260	000257			CCC	; CLEAR FLAGS
1081	014262	005601			SBC R1	; SUBTRACT C BIT=0
1082	014264	104400			CHECKCC=0	; CHECK FOR CC = 0
1083	014266	104423	014254		ERROR, SBC0	;; *** INCORRECT CC BITS ***
1084	014272	022701	000003		CMP #3,R1	; CHECK IT
1085	014276	001402			BEG 2#	; CONTINUE IF OK
1086	014300	104423	014254		ERROR, SBC0	;; *** SBC INSTRUCTION FAILED ***
1087	014304	000261		2#:	SEC	; C=1
1088	014306	005601			SBC R1	; SUBTRACT C BIT=1
1089	014310	000261			SEC	; C=1
1090	014312	005601			SBC R1	;
1091	014314	104400			CHECKCC=0	; CHECK FOR CC = 0
1092	014316	104423	014254		ERROR, SBC0	;; *** INCORRECT CC BITS ***
1093	014322	022701	000001		CMP #1,R1	; CHECK IT
1094	014326	001402			BEG 3#	; CONTINUE IF OK
1095	014330	104423	014254		ERROR, SBC0	;; *** SBC INSTRUCTION FAILED ***
1096	014334	000261		3#:	SEC	; C=1
1097	014336	005601			SBC R1	; SUBTRACT C BIT=1
1098	014340	104404			CHECKCC=4	; CHECK FOR CC = 4
1099	014342	104423	014254		ERROR, SBC0	;; *** INCORRECT CC BITS ***
1100	014346	000261			SEC	; C=1
1101	014350	005601			SBC R1	; SUBTRACT C BIT = 1
1102	014352	104411			CHECKCC=11	; CHECK FOR CC = 11
1103	014354	104423	014254		ERROR, SBC0	;; *** INCORRECT CC BITS ***
1104	014360	022701	177777		CMP #-1,R1	; CHECK IT
1105	014364	001402			BEG 4#	; CONTINUE IF F OK
1106	014366	104423	014254		ERROR, SBC0	;; *** SBC INSTRUCTION FAILED ***
1107	014372	012701	100000	4#:	MOV #100000,R1	; LOAD R1
1108	014376	000261			SEC	; C=1
1109	014400	005601			SBC R1	; SUBTRACT C BIT = 1
1110	014402	104402			CHECKCC=2	; CHECK FOR CC = 2
1111	014404	104423	014254		ERROR, SBC0	;; *** INCORRECT CC BITS ***

PC 14404 = INCORRECT CC BITS.

1112  
1113

```

1114 014410 104420 000031
1115 014414 005002
1116 014416 000277
1117 014420 000254
1118 014422 006702
1119 014424 104405
1120 014426 104423 014410
1121 014432 005702
1122 014434 001402
1123 014436 104423 014410
1124 014442 000273
1125 014444 006702
1126 014446 104411
1127 014450 104423 014410
1128 014454 022702 177777
1129 014460 001402
1130 014462 104423 014410
1131 014466 012703 125125
1132 014472 000277
1133 014474 000250
1134 014476 000303
1135 014500 104410
1136 014502 104423 014466
1137 014506 022703 052652
1138 014512 001402
1139 014514 104423 014466
1140 014520 012703 000377
1141 014524 000277
1142 014526 000244
1143 014530 000303
1144 014532 104404
1145 014534 104423 014466
1146 014540 022703 177400
1147 014544 001402
1148 014546 104423 014466
1149
1150 014552 012704 177777
1151 014556 012703 177777
1152 014562 000277
1153 014564 074403
1154 014566 104405
1155 014570 104423 014552
1156 014574 012703 077777
1157 014600 010400
1158 014602 000263
1159 014604 000244
1160 014606 074003
1161 014610 104411
1162 014612 104423 014552
1163 014616 012702 125252
1164 014622 012704 052525
1165 014626 000277

```

```

;*****
.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      20      ; CONTINUE IF OK
      ERROR,  T31      ; *** SXT INSTRUCTION FAILED ***
20:   SEVNC
      SXT      R2      ; SIGN EXTEND
      CHECKCC.11     ; CHECK FOR CC = 11
      ERROR,  T31      ; *** INCORRECT CC BITS ***
      CMP      0-1,R2  ; REG. 2 SHOULD NOW HAVE -1
      BEQ      SWAB0    ; CONTINUE IF OK
      ERROR,  T31      ; *** SXT FAILED ***

SWAB0: MOV      0125125,R3 ; LOAD BIT PATTERN INTO REGISTER
      SCC
      CLN
      SWAB     R3      ; SWAB BYTES OF REGISTER
      CHECKCC.10     ; CHECK FOR CC = 10
      ERROR,  SWAB0    ; *** INCORRECT CC BITS ***
      CMP      052652,R3 ; CHECK IT
      BEQ      10      ; CONTINUE IF OK
      ERROR,  SWAB0    ; *** SWAB INSTRUCTION FAILED ***
10:   MOV      0377,R3
      SCC
      CLZ
      SWAB     R3      ; CHECK FOR CC = 4
      CHECKCC.4      ; CHECK FOR CC = 4
      ERROR,  SWAB0    ; *** INCORRECT CC BITS ***
      CMP      0177400,R3
      BEQ      XOR0
      ERROR,  SWAB0    ; *** SWAB FAILED ***

XOR0: MOV      0-1,R4
      MOV      0-1,R3
      SCC
      XOR      R4,R3   ; SHOULD PRODUCE 0'S IN REG. 3
      CHECKCC.5      ; CHECK FOR CC = 5
      ERROR,  XOR0     ; *** INCORRECT CC BITS ***
      MOV      077777,R3
      MOV      R4,R0
      SEVNC
      CLZ
      XOR      R0,R3   ; PLACE A -1 IN R0
      CHECKCC.11     ; SET V & C BITS
      ERROR,  XOR0     ; CHECK FOR CC = 11
      MOV      0125252,R2 ; *** INCORRECT CC BITS ***
      MOV      052525,R4 ; LOAD REGISTERS
      SCC

```

PC 14612 = INCORRECT CC BITS.

1166 014630 074204  
 1167 014632 104411  
 1168 014634 104423 014552  
 1169 014640 022704 177777  
 1170 014644 001402  
 1171 014646 104423 014552  
 1172 014652  
 1173  
 1174

XOR R2,R4  
 CHECKCC=11  
 ERROR, XOR0  
 CMP 0-1,R4  
 BEQ 10  
 ERROR, XOR0

;; SHOULD PRODUCE ALL 1'S IN REG. 4  
 ;; CHECK FOR CC = 11  
 ;; \*\*\* INCORRECT CC BITS \*\*\*  
 ;; CHECK IT  
 ;; CONTINUE IF OK  
 ;; \*\*\* XOR FAILED \*\*\*

10:

\*\*\*\*\*  
 .SBTTL T32 -- ADD AND SUB  
 \*\*\*\*\*

1175 014652 104420 000032  
 1176 014656 012701 021421  
 1177 014662 060101  
 1178 014664 104400  
 1179 014666 104423 014652  
 1180 014672 022701 043042  
 1181 014676 001402  
 1182 014700 104423 014652  
 1183 014704 012700 156357  
 1184 014710 060000  
 1185 014712 104411  
 1186 014714 104423 014652  
 1187 014720 022700 134736  
 1188 014724 001402  
 1189 014726 104423 014652  
 1190 014732 012702 100000  
 1191 014736 060202  
 1192 014740 104407  
 1193 014742 104423 014652  
 1194 014746 012704 021421  
 1195 014752 012701 156357  
 1196 014756 060401  
 1197 014760 001402  
 1198 014762 104423 014652  
 1199 014766 005404  
 1200 014770 012701 021421  
 1201 014774 060104  
 1202 014776 001402  
 1203 015000 104423 014652  
 1204 015004 012702 021421  
 1205 015010 012703 156357  
 1206 015014 160203  
 1207 015016 104410  
 1208 015020 104423 015004  
 1209 015024 022703 134736  
 1210 015030 001402  
 1211 015032 104423 015004  
 1212 015036 012703 021421  
 1213 015042 010204  
 1214 015044 160403  
 1215 015046 001402  
 1216 015050 104423 015004  
 1217 015054 012703 177777  
 1218 015060 012702 077777  
 1219 015064 160302

T32: BEGINTEST, 32  
 ADD0: MOV 021421,R1  
 ADD R1,R1  
 CHECKCC=0  
 ERROR, T32  
 CMP 043042,R1  
 BEQ 10  
 ERROR, T32  
 10: MOV 0-21421,R0  
 ADD R0,R0  
 CHECKCC=11  
 ERROR, T32  
 CMP 0-43042,R0  
 BEQ 20  
 ERROR, T32  
 20: MOV 0100000,R2  
 ADD R2,R2  
 CHECKCC=7  
 ERROR, T32  
 MOV 021421,R4  
 MOV 0-21421,R1  
 ADD R4,R1  
 BEQ 30  
 ERROR, T32  
 30: NEG R4  
 MOV 021421,R1  
 ADD R1,R4  
 BEQ SUB0  
 ERROR, T32  
 SUB0: MOV 021421,R2  
 MOV 0-21421,R3  
 SUB R2,R3  
 CHECKCC=10  
 ERROR, SUB0  
 CMP 0-43042,R3  
 BEQ 40  
 ERROR, SUB0  
 40: MOV 021421,R3  
 MOV R2,R4  
 SUB R4,R3  
 BEQ 60  
 ERROR, SUB0  
 60: MOV 0-1,R3  
 MOV 077777,R2  
 SUB R3,R2

;; CHECK SWITCH REGISTER OPTIONS.  
 ;; LOAD REGISTERS  
 ;; ADD  
 ;; CHECK FOR CC = 0  
 ;; \*\*\* INCORRECT CC BITS \*\*\*  
 ;; CHECK IT  
 ;; CONTINUE IF OK  
 ;; \*\*\* ADD INSTRUCTION FAILED \*\*\*  
 ;; LOAD REGISTERS  
 ;; ADD  
 ;; CHECK FOR CC = 11  
 ;; \*\*\* INCORRECT CC BITS \*\*\*  
 ;; CHECK IT  
 ;; CONTINUE IF OK  
 ;; \*\*\* ADD INSTRUCTION FAILED \*\*\*  
 ;; LOAD REGISTERS  
 ;; ADD SHOULD RESULT AS 0'S  
 ;; CHECK FOR CC = 7  
 ;; \*\*\* INCORRECT CC BITS \*\*\*  
 ;; LOAD REGISTERS  
 ;; ADD SHOULD RESULT AS 0'S  
 ;; CONTINUE IF OK  
 ;; \*\*\* ADD INSTRUCTION FAILED \*\*\*  
 ;; SWITCH SOURCE AND DESTINATION  
 ;; SHOULD RESULT AS 0'S  
 ;; CONTINUE IF OK  
 ;; \*\*\* ADD FAILED \*\*\*  
 ;; LOAD REGISTERS  
 ;; RESULT SHOULD--43042  
 ;; CHECK FOR CC = 10  
 ;; \*\*\* INCORRECT CC BITS \*\*\*  
 ;; CHECK IT  
 ;; CONTINUE IF OK  
 ;; \*\*\* SUB INSTRUCTION FAILED \*\*\*  
 ;; LOAD REGISTER  
 ;; NOW R4 = 021421  
 ;; RESULT SHOULD=0  
 ;; \*\*\* SUB NSTRUCTION FAILED \*\*\*  
 ;; LOAD REGISTERS  
 ;; LOAD REGISTERS  
 ;; RESULT SHOULD BE 100000 AND OVERFLOW

PC 15050 = SUB NSTRUCTION FAILED.

```

1220 015066 104413
1221 015070 104423 015004
1222 015074 022702 100000
1223 015100 001402
1224 015102 104423 015004
1225 015106 012704 177777
1226 015112 160304
1227 015114 104404
1228 015116 104423 015004
1229
1230

```

```

CHECKCC+13 ; CHECK FOR CC = 13
ERROR, SUBO ; *** INCORRECT CC BITS ***
CMP #100000,R2 ; CHECK IT
BEQ #0 ; CONTINUE IF OK
ERROR, SUBO ; *** SUB INSTRUCTION FAILED ***
80: MOV #1,R4
SUB R3,R4
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, SUBO ; *** INCORRECT CC BITS ***

```

```

;*****
.SBTTL T33 -- MTPS AND MFPS
;*****

```

```

015122 104420 000033
1231 015126 012701 177777
1232 015132 005000
1233 015134 106400
1234 015136 104400
1235 015140 104423 015122
1236 015144 106701
1237 015146 001402
1238 015150 104423 015122
1239 015154 104404
1240 015156 104423 015122
1241 015162 012700 000337
1242 015166 106400
1243 015170 104417
1244 015172 104423 015122
1245 015176 106701
1246 015200 104411
1247 015202 104423 015122
1248 015206 022701 177717
1249 015212 001402
1250 015214 104423 015122
1251 015220
1252
1253
1254
1255
1256

```

```

T33: BEGINTEST, 33 ; CHECK SWITCH REGISTER OPTIONS.
TPSW: MOV #177777,R1
CLR R0
MTPS R0 ; SET PSM TO 0
CHECKCC+0 ; CHECK FOR CC = 0
ERROR, T33 ; *** INCORRECT CC BITS ***
MFPS R1 ; MOVE PSM TO R1
BEQ #0 ; BR IF BIT 8 OF PSM WAS EXTENDED IN R1
ERROR, T33 ; *** MTPS OR MFPS INSTRUCTION FAILED ***
20: CHECKCC+4 ; CHECK FOR CC = 4
ERROR, T33 ; *** INCORRECT CC BITS ***
MOV #337,R0 ; EXPECT 317 SINCE MTPS DOESN'T SET T BIT
MTPS R0 ; CHECK FOR CC = 17
CHECKCC+17 ; *** INCORRECT CC BITS ***
ERROR, T33 ; MOVE PSM TO R1
MFPS R1 ; CHECK FOR CC = 11 (C BIT SHOULD NOT BE EFFECTED BY MFPS)
CHECKCC+11 ; *** INCORRECT CC BITS ***
ERROR, T33 ; CHECK TO SEE IF BIT 8 OF PSM WAS EXTENDED THRU R1
CMP #177717,R1 ; *** MTPS OR MFPS INSTRUCTION FAILED ***
BEQ #0
ERROR, T33

```

```

30:
;
.SBTTL
.SBTTL NON-ZERO ADDRESS MODES
;

```

```

;*****
.SBTTL T34 -- TEST MODES 0 AND 1 USING MOVB AND MOV
;*****

```

```

015220 104420 000034
1257 015224 112700 000252
1258 015230 110001
1259 015232 110102
1260 015234 122702 000252
1261 015240 001402
1262 015242 104423 015220
1263 015246 012700 125252
1264 015252 010001
1265 015254 010102
1266 015256 022702 125252
1267 015262 001402
1268 015264 104423 015220
1269
1270 015270 012700 001006

```

```

T34: BEGINTEST, 34 ; CHECK SWITCH REGISTER OPTIONS.
MODE0: MOVB #252,R0 ; LOAD REGISTERS
MOVB R0,R1
MOVB R1,R2
CPB #252,R2 ; CHECK IT
BEQ #0 ; OK, CONTINUE
ERROR, T34 ; *** MOVB INSTRUCTION FAILED IN MODE 0 ***
10: MOV #125252,R0 ; LOAD REGISTERS
MOV R0,R1
MOV R1,R2
CMP #125252,R2 ; CHECK IT
BEQ MODE1 ; OK, CONTINUE
ERROR, T34 ; *** MOV INSTRUCTION FAILED IN MODE 0 ***
MODE1: MOV #TEMP,R0 ; LOAD ADDRESSES INTO REGS.

```

PC 15264 = MOV INSTRUCTION FAILED IN MODE 0.

```

1271 015274 012701 001010      MOV      @TEMP1,R1      ;
1272 015300 012702 001012      MOV      @TEMP2,R2      ;
1273 015304 005067 163502      CLR      TEMP2          ; START CLEAN
1274 015310 112710 000125      MOVB    @125,(R0)       ; LOAD THE LOCATIONS
1275 015314 111011              MOVB    (R0),(R1)       ; TEMP => TEMP1
1276 015316 111112              MOVB    (R1),(R2)       ; TEMP1 => TEMP2
1277 015320 122767 000125 163464  CPB     @125,TEMP2      ; CHECK IT
1278 015326 001402              BEQ     1#              ; OK, CONTINUE
1279 015330 104423 015220      ERROR,  T34            ; *** MOV INSTRUCTION FAILED IN MODE 1 ***
1280 015334 012710 052525 1#:   MOV     @52525,(R0)     ; LOAD THE LOCATIONS
1281 015340 011011              MOV     (R0),(R1)       ; TEMP => TEMP1
1282 015342 011112              MOV     (R1),(R2)       ; TEMP1 => TEMP2
1283 015344 022767 052525 163440  CP      @52525,TEMP2    ; CHECK IT
1284 015352 001402              BEQ     2#              ; OK, CONTINUE
1285 015354 104423 015220      ERROR,  T34            ; *** MOV INSTRUCTION FAILED IN MODE 1 ***
1286 015360 2#:
1287
1288

```

```

;*****
.SBTTL      T35 -- TEST MODE 2 USING MOVB AND MOV
;*****

```

```

1289 015360 104420 000035      T35:   BEGINTEST, 35      ; CHECK SWITCH REGISTER OPTIONS.
1290 015364 012700 001006      MODE2: MOV     @TEMP,R0      ; LOAD ADDRESSES
1291 015370 012701 001010      MOV     @TEMP1,R1      ;
1292 015374 012702 001012      MOV     @TEMP2,R2      ;
1293 015400 105022              CLRB   (R2)*           ; START CLEAN
1294 015402 112710 000252      MOVB   @252,(R0)       ; LOAD THE LOCATIONS
1295 015406 112021              MOVB   (R0)*,(R1)*     ; TEMP => TEMP1
1296 015410 105201              INCB   R1              ; MAKE IT EVEN
1297 015412 111167 163370      MOVB   (R1),TEMP       ; MORE 0'S INTO TEMP
1298 015416 105200              INCB   R0              ; MAKE IT EVEN
1299 015420 112021              MOVB   (R0)*,(R1)*     ; TEMP1 => TEMP2
1300 015422 124227 000252      CPB    -(R2),@252      ; CHECK IT
1301 015426 001003              BNE    1#              ; FAILED
1302 015430 105767 163352      TSTB   TEMP           ; CHECK IT
1303 015434 001402              BEQ    2#              ; OK, CONTINUE
1304 015436 104423 015360      1#:   ERROR,  T35          ; *** INSTRUCTIONS FAILED IN MODE 2 ***

```

```

1305 015442 005741 2#:   TST    -(R1)           ;
1306 015444 005022              CLR    (R2)*           ; START CLEAN
1307 015446 012740 125252      MOV    @125252,-(R0)   ; LOAD LOCATIONS
1308 015452 012020              MOV    (R0)*,(R0)*     ; TEMP => TEMP1
1309 015454 011067 163326      MOV    (R0),TEMP       ; 0 => TEMP
1310 015460 012121              MOV    (R1)*,(R1)*     ; 125252 => TEMP2
1311 015462 024227 125252      CP     -(R2),@125252   ; CHECK IT
1312 015466 001003              BNE    3#              ; FAILED
1313 015470 005767 163312      TST    TEMP           ; CHECK IT
1314 015474 001402              BEQ    4#              ; OK, CONTINUE
1315 015476 104423 015360      3#:   ERROR,  T35          ; *** INSTRUCTIONS FAILED IN MODE 2 ***

```

```

1316 015502 4#:
1317
1318
;*****
.SBTTL      T36 -- TEST MODE 3 USING MOVB AND MOV
;*****

```

```

1319 015502 104420 000036      T36:   BEGINTEST, 36      ; CHECK SWITCH REGISTER OPTIONS.
1319 015506 012767 001006 163264  MODE3: MOV     @TEMP,ADR  ; LOAD ADDRESSES

```



T36 -- TEST MODE 3 USING MOVB AND MOV

```

1320 015514 012767 001010 163260      MOV    #TEMP1,ADR1      ;
1321 015522 012767 001012 163254      MOV    #TEMP2,ADR2      ;
1322 015530 012700 001000              MOV    #ADR,R0          ; LOAD ADDRESSES OF ADDRESSES
1323 015534 012701 001002              MOV    #ADR1,R1         ;
1324 015540 105067 163246              CLR    TEMP2            ; START CLEAN
1325 015544 112767 000125 163234      MOV    #125,TEMP        ;
1326 015552 113031              MOV    B(R0),B(R1)     ; TEMP => TEMP1
1327 015554 113167 163226              MOV    B(R1),TEMP      ; TEMP2 => TEMP
1328 015560 113030              MOV    B(R0),B(R0)     ; TEMP1 => TEMP2
1329 015562 122767 000125 163222      CMP    #125,TEMP2      ; CHECK IT
1330 015570 001003              BNE    1#              ; FAILED
1331 015572 105767 163210              TST    TEMP            ; CHECK IT
1332 015576 001402              BEQ    2#              ; OK, CONTINUE
1333 015600              1#:
1334 015600 104423 015502              ERROR, T36             ; *** INSTRUCTIONS FAILED IN MODE 3 ***
1335 015604 005067 163202              CLR    TEMP2            ; START CLEAN
1336 015610 012767 052525 163170      2#: MOV    #52525,TEMP    ; LOAD LOCATIONS
1337 015616 012700 001000              MOV    #ADR,R0          ; LOAD ADDRESSES OF ADDRESSES
1338 015622 012701 001002              MOV    #ADR1,R1         ;
1339 015626 013030              MOV    B(R0),B(R0)     ; TEMP => TEMP1
1340 015630 013067 163152              MOV    B(R0),TEMP      ; TEMP2 => TEMP
1341 015634 013131              MOV    B(R1),B(R1)     ; TEMP1 => TEMP2
1342 015636 022767 052525 163146      CMP    #52525,TEMP2    ; CHECK IT
1343 015644 001003              BNE    3#              ; FAILED
1344 015646 005767 163134              TST    TEMP            ; CHECK IT
1345 015652 001402              BEQ    4#              ; OK, CONTINUE
1346 015654              3#:
1347 015654 104423 015502              4#: ERROR, T36             ; *** INSTRUCTIONS FAILED IN MODE 3 ***
1348 015660
1349

```

```

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

```

```

1350 015660 104420 000037              T37: BEGINTEST, 37      ; CHECK SWITCH REGISTER OPTIONS.
1351 015664 105067 163116      MODE4: CLR    TEMP        ; START CLEAN
1352 015670 012700 001006              MOV    #TEMP,R0        ; LOAD ADDRESSES
1353 015674 012701 001010              MOV    #TEMP1,R1       ;
1354 015700 012702 001012              MOV    #TEMP2,R2       ;
1355 015704 005202              INC    R2              ; ADJUST THE POINTER
1356 015706 021267 163101              CMP    (R2),TEMP2+1    ;
1357 015712 001402              BEQ    1#              ;
1358 015714 104423 015660              1#: ERROR, T37             ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1359 015720 112742 000252              MOV    #252,-(R2)     ; LOAD TEMP2
1360 015724 005201              INC    R1              ; ADJUST THE POINTERS
1361 015726 005202              INC    R2              ;
1362 015730 114241              MOV    -(R2),-(R1)    ; TEMP2 => TEMP1
1363 015732 005200              INC    R0              ; ADJUST THE POINTERS
1364 015734 005202              INC    R2              ;
1365 015740 114042              MOV    -(R0),-(R2)    ; TEMP => TEMP2
1366 015742 105200              INCB   R0              ; ADJUST THE POINTERS
1367 015744 021067 163041              CMP    (R0),TEMP+1    ;
1368 015746 001402              BEQ    2#              ;
1369 015750 104423 015660              2#: ERROR, T37             ; *** INSTRUCTIONS FAILED IN MODE 4 ***
1370 015754 105201              INCB   R1              ;
1371 015756 114140              MOV    -(R1),-(R0)    ; TEMP1 => TEMP
1371 015760 122767 000252 163020      CMP    #252,TEMP      ; CHECK IT

```

PC 15750 = INSTRUCTIONS FAILED IN MODE 4.

```

1372 015766 001003      BNE      30      ; FAILED
1373 015770 105767 163016  TSTB    TEMP2   ; CHECK IT
1374 015774 001402      BEQ      40      ; OK, CONTINUE
1375 015776      30:   ERROR,   T37      ;: *** INSTRUCTIONS FAILED IN MODE 4 ***
      015776 104423 015660
1376
1377 016002 005067 163000  40:   CLR     TEMP    ; START CLEAN
1378 016006 012700 001006  MOV     @TEMP,R0  ; LOAD ADDRESSES
1379 016012 012701 001010  MOV     @TEMP1,R1 ;
1380 016016 012702 001012  MOV     @TEMP2,R2 ;
1381 016022 005722      TST     (R2)+    ; ADJUST THE POINTER
1382 016024 021267 162764  CMP     (R2),TEMP2+2
1383 016030 001402      BEQ      50
1384 016032 104423 015660  50:   ERROR,   T37      ;: *** INSTRUCTIONS FAILED IN MODE 4 ***
1385 016036 012742 125252  MOV     @125252,-(R2) ; LOAD TEMP2
1386 016042 005721      TST     (R1)+    ; ADJUST THE POINTERS
1387 016044 005722      TST     (R2)+    ;
1388 016046 014241      MOV     -(R2),-(R1) ; TEMP2 => TEMP1
1389 016050 005720      TST     (R0)+    ; ADJUST POINTERS
1390 016052 005722      TST     (R2)+    ;
1391 016054 014042      MOV     -(R0),-(R2) ; TEMP => TEMP2
1392 016056 005720      TST     (R0)+    ; ADJUST THE POINTERS
1393 016060 005721      TST     (R1)+    ;
1394 016062 014140      MOV     -(R1),-(R0) ; TEMP1 => TEMP
1395 016064 022767 125252 162714  CMP     @125252,TEMP ; CHECK IT
1396 016072 001003      BNE      60      ; FAILED
1397 016074 005767 162712  TST     TEMP2   ; CHECK IT
1398 016100 001402      BEQ      70      ; OK, CONTINUE
1399 016102      60:   ERROR,   T37      ;: *** INSTRUCTIONS FAILED IN MODE 4 ***
      016102 104423 015660
1400 016106      70:
1401
1402

```

```

;*****
.SBTTL      T40 -- TEST MODE 5 USING MOVB AND MOV
;*****

```

```

1403 016106 104420 000040  T40:  BEGINTEST, 40 ;: CHECK SWITCH REGISTER OPTIONS.
1404 016112 105067 162670  MODE5: CLRB    TEMP    ; START CLEAN
1405 016116 012767 001006 162654  MOV     @TEMP,ADR  ; LOAD ADDRESSES
1406 016124 012767 001010 162650  MOV     @TEMP1,ADR1 ;
1407 016132 012767 001012 162644  MOV     @TEMP2,ADR2 ;
1408 016140 012700 001000      MOV     @ADR,R0    ; LOAD ADDRESSES OF ADDRESSES
1409 016144 012701 001002      MOV     @ADR1,R1   ;
1410 016150 012702 001004      MOV     @ADR2,R2   ;
1411 016154 005722      TST     (R2)+    ; ADJUST THE POINTER
1412 016156 112752 000125  MOVB    @125,B-(R2) ; LOAD TEMP2
1413 016162 022122      CMP     (R1)+,(R2)+ ; ADJUST THE POINTERS
1414 016164 115251      MOVB    B-(R2),B-(R1) ; TEMP2 => TEMP1
1415 016170 022022      CMP     (R0)+,(R2)+ ; ADJUST THE POINTERS
1416 016172 115052      MOVB    B-(R0),B-(R2) ; TEMP => TEMP2
1417 016174 022022      CMP     (R0)+,(R2)+ ; ADJUST THE POINTERS
1418 016176 125052      CPB     B-(R0),B-(R2) ; CHECK IT
1419 016176 001402      BEQ      10
1419 016200 104423 016106  10:   ERROR,   T40      ;: *** INSTRUCTION FAILED IN MODE 5 ***
1420 016204 022120      CMP     (R1)+,(R0)+ ; ADJUST THE POINTERS
1421 016206 115150      MOVB    B-(R1),B-(R0) ; TEMP1 => TEMP
1422 016210 122767 000125 162570  CPB     @125,TEMP  ; CHECK IT
1423 016216 001003      BNE      20      ; FAILED

```

PC 16200 = INSTRUCTION FAILED IN MODE 5.

```

1424 016220 105767 162566      TSTB   TEMP2      ; CHECK IT
1425 016224 001402              BEQ     30         ; OK, CONTINUE
1426 016226 104423 016106      20:   ERROR,   T40      ; *** INSTRUCTIONS FAILED IN MODE 5 ***
1427 016226 104423 016106
1428 016232 005067 162550      30:   CLR     TEMP      ; START CLEAN
1429 016236 012700 001000      MOV    @ADR,R0     ; LOAD ADDRESSES OF ADDRESSES
1430 016242 012701 001002      MOV    @ADR1,R1
1431 016246 012702 001004      MOV    @ADR2,R2
1432 016252 005722              TST     (R2)+
1433 016254 012752 052525      MOV    #52525,0-(R2) ; ADJUST THE POINTER
1434 016260 022122              CMP    (R1)+,(R2)+ ; LOAD TEMP2
1435 016262 015251              MOV    0-(R2),0-(R1) ; ADJUST THE POINTERS
1436 016264 022022              CMP    (R0)+,(R2)+ ; TEMP2 => TEMP1
1437 016266 015052              MOV    0-(R0),0-(R2) ; ADJUST THE POINTERS
1438 016270 022021              CMP    (R0)+,(R1)+ ; TEMP => TEMP2
1439 016272 015150              MOV    0-(R1),0-(R0) ; ADJUST THE POINTERS
1440 016274 022767 052525 162504  CMP    #52525,TEMP ; TEMP1 => TEMP
1441 016302 001003              BNE    40         ; CHECK IT
1442 016304 005767 162502      TST    TEMP2      ; FAILED
1443 016310 001402              BEQ     50         ; CHECK IT
1444 016312 104423 016106      40:   ERROR,   T40      ; OK, CONTINUE
1445 016316 104423 016106      50:   ERROR,   T40      ; *** INSTRUCTIONS FAILED IN MODE 5 ***
1446
1447

```

```

;*****
.SBTTL      T41 -- TEST MODE 6 USING MOVB AND MOV
;*****

```

```

1448 016316 104420 000041      T41:   BEGINTEST, 41 ; CHECK SWITCH REGISTER OPTIONS.
1449 016322 005067 162464      MODE6: CLR     TEMP2      ; START CLEAN
1450 016326 012700 001006      MOV    @TEMP,R0     ; LOAD ADDRESSES
1451 016332 012701 001010      MOV    @TEMP1,R1
1452 016336 012702 001012      MOV    @TEMP2,R2
1453 016342 112760 000252 000000  MOVB   #252,0(R0)   ; LOAD TEMP (LOW BYTE)
1454 016350 112760 000252 000001  MOVB   #252,1(R0)   ; LOAD TEMP (HIGH BYTE)
1455 016356 022767 125252 162422  CMP    #125252,TEMP ; CHECK IT
1456 016364 001012              BNE    10         ; FAILED
1457 016374 116062 000001 000000  MOVB   1(R0),0(R2)  ; TEMP(H) => TEMP2(L)
1458 016402 022767 125252 162402  MOVB   2(R1),5(R0)  ; TEMP2(L) => TEMP2(H)
1459 016410 001402              CMP    #125252,TEMP2 ; CHECK IT
1460 016412 001402              BEQ     20         ; OK, CONTINUE
1461 016412 104423 016316      10:   ERROR,   T41      ; *** INSTRUCTIONS FAILED IN MODE 6 ***
1462 016416 005067 162366      20:   CLR     TEMP1      ; START CLEAN
1463 016422 012760 052525 000000  MOV    #52525,0(R0) ; LOAD TEMP
1464 016430 016260 177774 000002  MOV    -4(R2),2(R0) ; TEMP => TEMP1
1465 016436 022767 052525 162344  CMP    #52525,TEMP1 ; CHECK IT
1466 016444 001402              BEQ     30         ; OK, CONTINUE
1467 016446 104423 016316      30:   ERROR,   T41      ; *** INSTRUCTIONS FAILED IN MODE 6 ***
1468 016452
1469
1470

```

```

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

```

```

1471 016452 104420 000042      T42:   BEGINTEST, 42 ; CHECK SWITCH REGISTER OPTIONS.
1471 016456 005067 162326      MODE7: CLR     TEMP1      ; START CLEAN

```

T42 -- TEST MODE 7 USING MOV8 AND MOV

```

1472 016462 012767 001006 162310      MOV      @TEMP,ADR      ; LOAD ADDRESSES
1473 016470 012767 001010 162304      MOV      @TEMP1,ADR1   ;
1474 016476 012767 001012 162300      MOV      @TEMP2,ADR2   ;
1475 016504 012700 001000              MOV      @ADR,R0       ; LOAD ADDRESSES OF ADDRESSES
1476 016510 012701 001002              MOV      @ADR1,R1      ;
1477 016514 012702 001004              MOV      @ADR2,R2      ;
1478 016520 112770 000252 000000      MOV8     @252,80(R0)    ; LOAD TEMP
1479 016526 112770 177774 000002      MOV8     @-4(R2),82(R0) ; TEMP => TEMP1
1480 016534 122767 000252 162246      CMP8     @252,TEMP1    ; CHECK IT
1481 016542 001402                    BEQ      10            ; OK, CONTINUE
1482 016544 104423 016452              ERROR,   T42          ; *** MODE 7 IS FAILING ***
1483 016550 012770 125252 000000 10:  MOV      @125252,80(R0) ; LOAD TEMP
1484 016556 012770 177774 000002      MOV8     @-4(R2),82(R0) ; TEMP => TEMP1
1485 016564 022767 125252 162216      CMP      @125252,TEMP1 ; CHECK IT
1486 016572 001402                    BEQ      20            ; OK, CONTINUE
1487 016574 104423 016452              ERROR,   T42          ; *** INSTRUCTIONS FAILED IN MODE 7 ***
1488
1489 016600
1490
20:
;*****
.SBTTL      T43 -- TSTB, CLRB, AND MOV8
;*****
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 ***
      MOV8     @377,(R1)    ; LOAD THE LOCATION
      CHECKCC+10            ; CHECK FOR CC = 10
      ERROR,   T43         ; *** INCORRECT CC BITS ***
      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
      MOV8     @200,0(R2)   ; PLACE #200 IN LOCATION TEMP
      MOV8     (R2),-(R1)   ; MOVE #200 TO LOCATION TEMP+1
      CMP      -1(R1),#100200 ; CHECK THE DATA IN LOCATION TEMP
      BEQ      40
      ERROR,   T43         ; *** MOV8 INSTRUCTION FAILED ***
40:  CMP      R1,R2          ; CHECK THE REGISTERS FOR PROPER VALUE
      BEQ      50
      ERROR,   T43         ; *** MOV8 INSTRUCTION FAILED ***
50:
;*****

```

```

      016600 104420 000043
1491 016604 012700 001006
1492 016610 012701 001010
1493 016614 000277
1494 016616 105010
1495 016620 104404
1496 016622 104423 016600
1497 016626 105710
1498 016630 104404
1499 016632 104423 016600
1500 016636 112711 000377
1501 016642 104410
1502 016644 104423 016600
1503 016650 105711
1504 016652 104410
1505 016654 104423 016600
1506 016660 010002
1507 016662 112762 000200 000000
1508 016670 112241
1509 016672 026127 177777 100200
1510 016700 001402
1511 016702 104423 016600
1512 016706 020102
1513 016710 001402
1514 016712 104423 016600
1515 016716
1516
1517

```

```

;*****
.SBTTL      T44 -- CMP8 AND BIS8
;*****
T44:  BEGINTEST, 44          ; CHECK SWITCH REGISTER OPTIONS.
CMP81: MOV      @TEMP2,R1   ; LOAD ADDRESS
      MOV      @TEMP,R2    ; PLACE 77 IN LOCATION TEMP2
      MOV      @77,(R1)    ; R4 SHOULD CONTAIN #177777
      MOV8     @377,R4     ; LOAD LOCATION
      BIS8     R4,(R2)

```

```

      016716 104420 000044
1518 016722 012701 001012
1519 016726 012702 001006
1520 016732 012711 000077
1521 016736 112704 000377
1522 016742 150412

```

T44 -- CMPB AND BISB

1523	016744	104410		CHECKCC+10		CHECK FOR CC = 10
1524	016746	104423	016716	ERROR, T44		*** INCORRECT CC BITS ***
1525	016752	120412		CMPB R4,(R2)		CHECK COMPARE
1526	016754	001402		BEG 20		CONTINUE IF OK
1527	016756	104423	016716	ERROR, T44		*** BISB OR CMPB INSTRUCTION FAILED ***
1528	016762	121112		20: CMPB (R1),(R2)		CHECK IT AGAIN
1529	016764	100002		BPL 30		CONTINUE IF OK
1530	016766	104423	016716	ERROR, T44		*** CMPB INSTRUCTION FAILED (WRONG CC) ***
1531	016772	121211		30: CMPB (R2),(R1)		ONCE MORE
1532	016774	100402		BMI 40		CONTINUE IF OK
1533	016776	104423	016716	ERROR, T44		*** INSTR FAILED ***
1534	017002			40:		
1535						
1536						

\*\*\*\*\*  
 .SBTTL T45 -- BICB AND BITB  
 \*\*\*\*\*

1537	017002	104420	000045	T45: BEGINTEST, 45		CHECK SWITCH REGISTER OPTIONS.
1538	017006	012703	001006	BICB1: MOV #TEMP,R3		LOAD ADDRESS
1539	017012	112713	000377	MOV #377,(R3)		LOAD LOCATION
1540	017016	012700	001010	MOV #TEMP1,R0		PLACE THE ADDRESS OF TEMP1 IN R0
1541	017022	010001		MOV R0,R1		AND R1
1542	017024	112721	000252	MOV #252,(R1)		PLACE #252 IN TEMP1
1543	017030	000277		SCC		
1544	017032	146013	000000	BICB 0(R0),(R3)		CLEAR EVERY OTHER BIT
1545	017036	104401		CHECKCC+1		CHECK FOR CC = 1
1546	017040	104423	017002	ERROR, T45		*** INCORRECT CC BITS ***
1547	017044	136113	177777	BITB -1(R1),(R3)		CHECK IT
1548	017050	001402		BEG 40		CONTINUE IF OK
1549	017052	104423	017002	ERROR, T45		*** BICB OR BITB INSTRUCTION FAILED ***
1550	017056	132713	000125	40: BITB #125,(R3)		CHECK IT
1551	017062	104401		CHECKCC+1		CHECK FOR CC = 1
1552	017064	104423	017002	ERROR, T45		*** INCORRECT CC BITS ***
1553	017070	154113		BISB -(R1),(R3)		SET THE BITS THAT WERE CLEARED
1554	017072	100402		BMI 60		CONTINUE IF OK
1555	017074	104423	017002	ERROR, T45		*** BITB OR BISB INSTRUCTION FAILED ***
1556	017100	012746	000177	60: MOV #177,-(SP)		STORE #177 ON THE STACK
1557	017104	142613		BICB (SP),(R3)		CLEAR ALL THE BITS EXCEPT SIGN BIT
1558	017106	104411		CHECKCC+11		CHECK FOR CC = 11
1559	017110	104423	017002	ERROR, T45		*** INCORRECT CC BITS ***
1560	017114	132713	000377	BITB #377,(R3)		CHECK IT
1561	017120	104411		CHECKCC+11		CHECK FOR CC = 11
1562	017122	104423	017002	ERROR, T45		*** INCORRECT CC BITS ***
1563	017126	010300		MOV R3,R0		PLACE THE ADDRESS OF TEMP IN R0
1564	017130	012710	001010	MOV #TEMP1,(R0)		PLACE THE ADDRESS OF TEMP1 IN TEMP
1565	017134	012730	000377	MOV #377,B(R0)		WRITE A 377 IN LOCATION TEMP1
1566	017140	000263		SEVC		SET V & C BITS
1567	017142	145070	000000	BICB 8-(R0),B(R0)		BIT CLEAR THE CONTENTS
1568	017146	104405		CHECKCC+5		OF TEMP1 TO THE CONTENTS OF TEMP1
1569	017150	104423	017002	ERROR, T45		CHECK FOR CC = 5
1570	017154	022027	001010	CMP (R0),#TEMP1		*** INCORRECT CC BITS ***
1571	017160	001402		BEG 80		MAKE SURE THAT (R0) IS POINTING TO LOCATION TEMP1
1572	017162	104423	017002	ERROR, T45		*** BICB OR CMP INSTRUCTION FAILED IN THE SPECIFIC MODE *
1573	017166	005750		80: TST 8-(R0)		TEST LOCATION TEMP1
1574	017170	001402		BEG 100		
1575	017172	104423	017002	ERROR, T45		*** BICB INSTRUCTION FAILED ***
1576	017176	000257		100: CCC		

PC 17172 = BICB INSTRUCTION FAILED.

1577 017200 141010  
 1578 017202 104404  
 1579 017204 104423 017002  
 1580  
 1581

BICB (R0),(R0) ; CLEAR THE LOCATION TEMP  
 CHECKCC+4 ; CHECK FOR CC = 4  
 ERROR, T45 ; \*\*\* INCORRECT CC BITS \*\*\*

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

1582 017210 104420 000046  
 1583 017214 012704 001006  
 1584 017220 112714 000177  
 1585 017224 000261  
 1586 017226 105214  
 1587 017230 104413  
 1588 017232 104423 017210  
 1589 017236 012714 000376  
 1590 017242 105224  
 1591 017244 104411  
 1592 017246 104423 017210  
 1593 017252 105744  
 1594 017254 005746  
 1595 017256 010426  
 1596 017260 000241  
 1597 017262 105256  
 1598 017264 104404  
 1599 017266 104423 017210  
 1600 017272 123634  
 1601 017274 000261  
 1602 017276 105264 177777  
 1603 017302 104401  
 1604 017304 104423 017210  
 1605 017310 124427 000001  
 1606 017314 001402  
 1607 017316 104423 017210  
 1608 017322 000261  
 1609 017324 105314  
 1610 017326 104405  
 1611 017330 104423 017210  
 1612 017334 105324  
 1613 017336 104411  
 1614 017340 104423 017210  
 1615 017344 112764 000200 177777  
 1616 017352 105344  
 1617 017354 104403  
 1618 017356 104423 017210  
 1619 017362 105364 000000  
 1620 017366 104401  
 1621 017370 104423 017210  
 1622 017374 126427 000000 000176  
 1623 017402 001402  
 1624 017404 104423 017210  
 1625  
 1626

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 B-(SP) ; INCREMENT THE CONTENTS OF LOCATION TEMP  
 CHECKCC+4 ; CHECK FOR CC = 4  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 CPB B(SP)+,B(R4) ; RESTORE STACK POINTER  
 SEC ; SET C BIT  
 INCB -1(R4)  
 CHECKCC+1 ; CHECK FOR CC = 1  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 CPB -(R4),#1 ; CHECK IT  
 BEQ #0 ; CONTINUE IF OK  
 ERROR, T46 ; \*\*\* INCB INSTRUCTION FAILED \*\*\*  
 20: SEC  
 DECB (R4) ; SUBTRACT ONES FROM LOCATION  
 CHECKCC+5 ; CHECK FOR CC = 5  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 DECB (R4) ;  
 CHECKCC+11 ; CHECK FOR CC = 11  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 MOV #200,-1(R4)  
 DECB -(R4) ;  
 CHECKCC+3 ; CHECK CC = 3  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 DECB 0(R4) ;  
 CHECKCC+1 ; CHECK FOR CC = 1  
 ERROR, T46 ; \*\*\* INCORRECT CC BITS \*\*\*  
 CPB 0(R4),#176  
 BEQ #0 ;  
 ERROR, T46 ; \*\*\* DECB INSTRUCTION FAILED \*\*\*

\*\*\*\*\*  
 .SBTTL T47 -- COMB AND NEGB  
 \*\*\*\*\*

1627 017410 104420 000047  
 1627 017414 012703 001006

T47: BEGINTEST, 47 ; CHECK SWITCH REGISTER OPTIONS.  
 COMB1: MOV #TEMP,R3 ; LOAD ADDRESS

T47 -- COMB AND NEGB

```

1628 017420 012704 001010
1629 017424 012714 000252
1630 017430 112413
1631 017432 000277
1632 017434 105113
1633 017436 104401
1634 017440 104423 017410
1635 017444 122713 000125
1636 017450 001402
1637 017452 104423 017410
1638 017456 000277
1639 017460 105113
1640 017462 104411
1641 017464 104423 017410
1642 017470 010400
1643 017472 126013 177777
1644 017476 001402
1645 017500 104423 017410
1646 017504 112724 000377
1647 017510 114413
1648 017512 000277
1649 017514 105113
1650 017516 104405
1651 017520 104423 017410
1652
1653 017524 012700 001006
1654 017530 112710 000001
1655 017534 105410
1656 017536 104411
1657 017540 104423 017524
1658 017544 122710 000377
1659 017550 001402
1660 017552 104423 017524
1661 017556 012710 000200
1662 017562 105410
1663 017564 104413
1664 017566 104423 017524
1665 017572 122710 000200
1666 017576 001402
1667 017600 104423 017524
1668 017604
1669
1670

```

```

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
ERROR,  T47           ; *** INCORRECT CC BITS ***
CMPB    #125,(R3)     ; CHECK IT
BEQ     2#            ; CONTINUE IF OK
ERROR,  T47           ; *** COMB INSTRUCTION FAILED ***
2#:     SCC
COMB    (R3)           ; COMPLEMENT BACK
CHECKCC+11            ; CHECK FOR CC = 11
ERROR,  T47           ; *** INCORRECT CC BITS ***
MOV     R4,R0
CMPB    -1(R0),(R3)   ; CHECK IT
BEQ     3#            ; CONTINUE IF OK
ERROR,  T47           ; *** COMB INSTRUCTION FAILED ***
3#:     MOVB #377,(R4)+
MOVB    -(R4),(R3)   ; PLACE #377 IN (R3)
SCC
COMB    (R3)
CHECKCC+5             ; CHECK FOR CC = 5
ERROR,  T47           ; *** INCORRECT CC BITS ***

NEGB1:  MOV     #TEMP,R0      ; LOAD ADDRESS
MOVB    #1,(R0)       ; LOAD THE LOCATION
NEGB    (R0)          ; 2'S COMPLEMENT
CHECKCC+11            ; CHECK FOR CC = 11
ERROR,  NEGB1        ; *** INCORRECT CC BITS ***
CMPB    #377,(R0)    ; CHECK IT
BEQ     2#            ; CONTINUE IF OK
ERROR,  NEGB1        ; *** NEGB INSTRUCTION FAILED ***
2#:     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
;*****

```

```

1671 017604 104420 000050
1672 017610 012701 001010
1673 017614 112711 000040
1674 017620 000257
1675 017622 106111
1676 017624 106111
1677 017626 104412
1678 017630 104423 017604
1679 017634 122711 000200
1680 017640 001402
1681 017642 104423 017604
1681 017646 106111

```

```

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

```

PC 17642 = ROLB INSTRUCTION FAILED.

```

1682 017650 104407
1683 017652 104423 017604
1684 017656 106111
1685 017660 122711 000001
1686 017664 001402
1687 017666 104423 017604
1688
1689 017672 012702 001010
1690 017676 112712 000004
1691 017702 000257
1692 017704 106012
1693 017706 106012
1694 017710 122712 000001
1695 017714 001402
1696 017716 104423 017672
1697 017722 106012
1698 017724 104407
1699 017726 104423 017672
1700 017732 106012
1701 017734 104412
1702 017736 104423 017672
1703 017742 122712 000200
1704 017746 001402
1705 017750 104423 017672
1706 017754
1707
1708

```

```

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
MOV #4,(R2) ; LOAD LOCATION
CCC ; CLEAR FLAGS
RORB (R2) ; SHIFT
RORB (R2) ;
CMPB #1,(R2) ; CHECK IT
BEQ 10 ; CONTINUE IF OK
ERROR, RORB1 ; ** RORB INSTRUCTION FAILED **
10: 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 20 ; CONTINUE IF OK
ERROR, RORB1 ; ** RORB FAILED **
20:

```

```

*****
.SBTTL T51 -- ASLB AND ASRB
*****

```

```

017754 104420 000051
1709 017760 012703 001010
1710 017764 112713 000040
1711 017770 000257
1712 017772 106313
1713 017774 106313
1714 017776 104412
1715 020000 104423 017754
1716 020004 122713 000200
1717 020010 001402
1718 020012 104423 017754
1719 020016 106313
1720 020020 104407
1721 020022 104423 017754
1722 020026 106313
1723 020030 104404
1724 020032 104423 017754
1725
1726 020036 012704 001010
1727 020042 012703 001012
1728 020046 112714 000004
1729 020052 000257
1730 020054 106214
1731 020056 106214
1732 020060 122714 000001
1733 020064 001402
1734 020066 104423 020036
1735 020072 106214

```

```

T51: BEGINTEST, 51 ; CHECK SWITCH REGISTER OPTIONS.
ASLB1: MOV #TEMP1,R3 ; LOAD ADDRESS
MOV #40,(R3) ; LOAD LOCATION
CCC ; CLEAR FLAGS
ASLB (R3) ; SHIFT
ASLB (R3) ;
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, T51 ; ** INCORRECT CC BITS **
CMPB #200,(R3) ; CHECK IT
BEQ 40 ; CONTINUE IF OK
ERROR, T51 ; ** ASLB INSTRUCTION FAILED **
40: ASLB (R3) ; SHIFT
CHECKCC+7 ; CHECK FOR CC = 7
ERROR, T51 ; ** INCORRECT CC BITS **
ASLB (R3) ; SHIFT
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, T51 ; ** INCORRECT CC BITS **

ASRB1: MOV #TEMP1,R4 ; LOAD ADDRESSES
MOV #TEMP2,R3 ;
MOV #4,(R4) ; LOAD LOCATION
CCC ; CLEAR FLAGS
ASRB (R4) ; SHIFT
ASRB (R4) ;
CMPB #1,(R4) ; CHECK IT
BEQ 20 ; CONTINUE IF OK
ERROR, ASRB1 ; ** ASRB INSTRUCTION FAILED **
20: ASRB (R4) ; SHIFT

```



PC 20066 = ASRB INSTRUCTION FAILED.

```

1736 020074 104407
1737 020076 104423 020036
1738 020102 106214
1739 020104 104404
1740 020106 104423 020036
1741 020112 112713 000202
1742 020116 106213
1743 020120 106213
1744 020122 104411
1745 020124 104423 020036
1746 020130 122713 000340
1747 020134 001402
1748 020136 104423 020036
1749 020142
1750
1751

```

```

CHECKCC+7 ; CHECK FOR CC = 7
ERROR, ASRB1 ; *** INCORRECT CC BITS ***
ASRB (R4) ; SHIFT
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, ASRB1 ; *** INCORRECT CC BITS ***
MOVB #202,(R3) ; LOAD LOCATION
ASRB (R3) ; SHIFT
ASRB (R3) ;
CHECKCC+11 ; CHECK FOR CC = 11
ERROR, ASRB1 ; *** INCORRECT CC BITS ***
CMPB #340,(R3) ; CHECK IT
BEQ 31 ; CONTINUE IF OK
ERROR, ASRB1 ; *** ASRB FAILED ***

```

31:

```

;*****
.SBTTL TS2 -- ADCB AND SBCB
;*****

```

```

020142 104420 000052
1752 020146 012700 001012
1753 020152 105010
1754 020154 000257
1755 020156 105510
1756 020160 104404
1757 020162 104423 020142
1758 020166 000261
1759 020170 105510
1760 020172 000261
1761 020174 105510
1762 020176 104400
1763 020200 104423 020142
1764 020204 122710 000002
1765 020210 001402
1766 020212 104423 020142
1767 020216 112710 000177
1768 020222 000261
1769 020224 105510
1770 020226 104412
1771 020230 104423 020142
1772 020234 122710 000200
1773 020240 001402
1774 020242 104423 020142
1775 020246 112710 000377
1776 020252 000261
1777 020254 105510
1778 020256 104405
1779 020260 104423 020142
1780
1781 020264 012701 001012
1782 020270 112711 000003
1783 020274 000257
1784 020276 105611
1785 020300 104400
1786 020302 104423 020264
1787 020306 122711 000003
1788 020312 001402
1789 020314 104423 020264

```

```

TS2: BEGINTEST, 52 ; CHECK SWITCH REGISTER OPTIONS.
ADCB1: MOV #TEMP2,R0 ; LOAD ADDRESS
CLRB (R0) ; CLEAR THE LOCATION
CCC ; CLEAR FLAGS
ADCB (R0) ; ADD C BIT = 0
CHECKCC+4 ; CHECK FOR CC = 4
ERROR, TS2 ; *** INCORRECT CC BITS ***
SEC ; C-1
ADCB (R0) ; ADD C BIT-1
SEC ; C-1
ADCB (R0) ; AGAIN
CHECKCC+0 ; CHECK FOR CC = 0
ERROR, TS2 ; *** INCORRECT CC BITS ***
CMPB #2,(R0) ; CHECK IT
BEQ 41 ; CONTINUE IF OK
ERROR, TS2 ; *** ADCB INSTRUCTION FAILED ***
41: MOVB #177,(R0) ; LOAD LARGEST POSITIVE BYTE
SEC ; C-1
ADCB (R0) ; ADD C BIT-1
CHECKCC+12 ; CHECK FOR CC = 12
ERROR, TS2 ; *** INCORRECT CC BITS ***
CMPB #200,(R0) ; CHECK IT
BEQ 61 ; CONTINUE IF OK
ERROR, TS2 ; *** ADCB INSTRUCTION FAILED ***
61: MOVB #377,(R0) ; LOAD -1
SEC ; C-1
ADCB (R0) ; ADD C BIT-1
CHECKCC+5 ; CHECK FOR CC = 5
ERROR, TS2 ; *** INCORRECT CC BITS ***

SBCB1: MOV #TEMP2,R1 ; LOAD ADDRESS
MOVB #3,(R1) ; LOAD LOCATION
CCC ; CLEAR FLAGS
SBCB (R1) ; SUBTRACT C BIT=0
CHECKCC+0 ; CHECK FOR CC = 0
ERROR, SBCB1 ; *** INCORRECT CC BITS ***
CMPB #3,(R1) ; CHECK IT
BEQ 21 ; CONTINUE IF OK
ERROR, SBCB1 ; *** SBCB INSTRUCTION FAILED ***

```

PC 20314 = SBCB INSTRUCTION FAILED.

```

1790 020320 000261
1791 020322 105611
1792 020324 000261
1793 020326 105611
1794 020330 104400
1795 020332 104423 020264
1796 020336 122711 000001
1797 020342 001402
1798 020344 104423 020264
1799 020330 000261
1800 020332 105611
1801 020334 104404
1802 020336 104423 020264
1803 020362 000261
1804 020364 105611
1805 020366 104411
1806 020370 104423 020264
1807 020374 122711 000377
1808 020400 001402
1809 020402 104423 020264
1810 020406 112711 000200
1811 020412 000261
1812 020414 105611
1813 020416 104402
1814 020420 104423 020264
1815
1816

```

```

20: SEC C-1
SBCB (R1) SUBTRACT C BIT=1
SEC C-1
SBCB (R1)
CHECKCC=0 CHECK FOR CC = 0
ERROR, SBCB1 ; *** INCORRECT CC BITS ***
CMPB #1,(R1) CHECK IT
BEQ 30 CONTINUE IF OK
ERROR, SBCB1 ; *** SBCB INSTRUCTION FAILED ***

30: SEC C-1
SBCB (R1) SUBTRACT C BIT=1
CHECKCC=4 CHECK FOR CC = 4
ERROR, SBCB1 ; *** INCORRECT CC BITS ***
SEC C-1
SBCB (R1) SUBTRACT C BIT = 1
CHECKCC=11 CHECK FOR CC = 11
ERROR, SBCB1 ; *** INCORRECT CC BITS ***
CMPB #377,(R1) CHECK IT
BEQ 40 CONTINUE IF OK
ERROR, SBCB1 ; *** SBCB INSTRUCTION FAILED ***

40: MOVB #200,(R1) LOAD R1
SEC C-1
SBCB (R1) SUBTRACT C BIT = 1
CHECKCC=2 CHECK FOR CC = 2
ERROR, SBCB1 ; *** INCORRECT CC BITS ***

```

```

;*****
.SBTTL T53 -- TST, CLR, AND MOV
;*****

```

```

1817 020424 104420 000053
1818 020430 012701 001006
1819 020434 012700 001010
1820 020440 000277
1821 020442 005010
1822 020444 104404
1823 020446 104423 020424
1824 020452 005720
1825 020454 104404
1826 020456 104423 020424
1827 020462 010040
1828 020464 012730 177777
1829 020470 016011 177776
1830 020474 104410
1831 020476 104423 020424
1832 020502 005711
1833 020504 104410
1834 020506 104423 020424
1835

```

```

T53: BEGINTEST, 53 ; CHECK SWITCH REGISTER OPTIONS.
MOV1: MOV #TEMP,R1 ; LOAD ADDRESSES
MOV #TEMP1,R0
SEC
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) ; POSSIBLE "Z" ASSEMBLY ERROR O.K.
MOV #177777,(R0) ; LOAD THE LOCATION.
MOV -2(R0),(R1) ; CHECK FOR CC = 10
CHECKCC=10 ; *** INCORRECT CC BITS ***
ERROR, T53 ; CHECK IT
TST (R1) ; CHECK FOR CC = 10
CHECKCC=10 ; *** INCORRECT CC BITS ***
ERROR, T53

```

;JSD REV B

```

;*****
.SBTTL T54 -- CMP AND BIS
;*****

```

```

1836 020512 104420 000054
1837 020516 012702 001010
1838 020522 012700 001006
1839 020526 000257
1840 020530 012720 177777
1840 020534 054012

```

```

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

```

Z

T54 -- CMP AND BIS

```

1841 020536 104410 CHECKCC+10 ; CHECK FOR CC = 10
1842 020540 104423 020512 ERROR, T54 ; ; *** INCORRECT CC BITS ***
1843 020544 022227 177777 CMP (R2),#177777 ; CHECK COMPARE
1844 020550 001402 BEQ 20 ; CONTINUE IF OK
1845 020552 104423 020512 ERROR, T54 ; ; *** CMP OR BIS INSTRUCTION FAILED ***
1846 020556 020227 001012 20: CMP R2,#TEMP1+2 ; CHECK R2 TO CONTAIN ADDRESS OF TEMP1+2
1847 020562 001402 BEQ 30
1848 020564 104423 020512 ERROR, T54 ; ; *** NO AUTO INCREMENT ***
1849 020570 022742 000077 30: CMP #77,-(R2) ; CHECK IT AGAIN
1850 020574 104401 CHECKCC+1 ; CHECK FOR CC = 1
1851 020576 104423 020512 ERROR, T54 ; ; *** INCORRECT CC BITS ***
1852 020602 022722 077777 CMP #77777,(R2)+
1853 020606 104413 CHECKCC+13 ; CHECK FOR CC = 13
1854 020610 104423 020512 ERROR, T54 ; ; *** INCORRECT CC BITS ***
1855 020614 024227 077777 CMP -(R2),#77777 ; ONCE MORE
1856 020620 104410 CHECKCC+10 ; CHECK FOR CC = 10
1857 020622 104423 020512 ERROR, T54 ; ; *** INCORRECT CC BITS ***
1858 020626 012767 052525 160156 MOV #52525,TEMP2 ; SET EVERY OTHER BIT IN TEMP2
1859 020634 012767 001012 160146 MOV #TEMP2,TEMP1 ; PLACE THE ADDRESS OF TEMP2 IN TEMP1
1860 020642 012704 001000 MOV #ADR,R4
1861 020646 012714 001002 MOV #ADR1,(R4) ; PLACE THE ADDRESS OF ADR1 IN ADR POINTED BY R4
1862 020652 012734 125252 MOV #125252,B(R4)+ ; PLACE THE #125252 IN LOCATION ADR1
1863 020656 057432 177776 BIS B-2(R4),B(R2)+ ; SET EVERY OTHER BIT AT LOCATION TEMP2
1864
1865 020662 010200 MOV R2,R0 ; AND INCREMENT R2 BY 2
1866 020664 025027 177777 CMP B-(R0),#177777 ; PLACE ADDRESS OF TEMP2 IN R0
1867 020670 001402 BEQ 40 ; TEMP2 SHOULD CONTAIN ALL 1'S
1868 020672 104423 020512 ERROR, T54 ; ; *** CMP OR BIS INSTUCTIONS FAILED IN MODES OTHER THAN 0 *
**
1869 020676 020227 001012 40: CMP R2,#TEMP1+2 ; R2 SHOULD CONTAIN THE ADDRESS OF TEMP2
1870 020702 001402 BEQ 50
1871 020704 104423 020512 ERROR, T54 ; ; *** MODE 5 IS FAILING ***
1872 020710 005040 50: CLR -(R0) ; PLACE A 0 IN TEMP
1873 020712 010067 160074 MOV R0,TEMP2 ; PLACE ADDRESS OF TEMP IN TEMP2
1874 020716 022020 CMP (R0),B(R0)+ ; BUMP R0 BY 4
1875 020720 055070 000002 BIS B-(R0),B2(R0) ; PLACE THE CONTENTS OF TEMP2 AT TEMP
1876 020724 022767 001006 160054 CMP #TEMP,TEMP ; TEMP SHOULD CONTAIN ITS OWN ADDRESS
1877 020732 001402 BEQ 60
1878 020734 104423 020512 ERROR, T54 ; ; *** CMP OR BIS INSTRUCTIONS FAILED ***
1879 020740
1880
1881
;*****
.SBTTL T55 -- BIC AND BIT
;*****
T55: BEGINTEST, 55 ; ; CHECK SWITCH REGISTER OPTIONS.
BIC1: MOV #TEMP,R3 ; LOAD ADDRESS
MOV #177777,(R3) ; LOAD LOCATION
MOV #ADR,R4 ; PLACE THE ADDRESS OF ADR IN R4
MOV #ADR1,(R4) ; PLACE THE ADDRESS OF ADR1 IN ADR
MOV (R3),B(R4)+ ; LOAD LOCATION ADR1 WITH #177777
MOV #TEMP1,R0 ; PLACE THE ADDRESS OF TEMP1 IN R0
MOV #125252,(R0) ; SET EVERY OTHER BIT AT LOCATION TEMP1
SCC
BIC (R0),B(R3) ; CLEAR EVERY OTHER BIT
CHECKCC+1 ; CHECK FOR CC = 1
1882 020744 012703 001006 ERROR, T55 ; ; *** INCORRECT CC BITS ***
1883 020750 012713 177777 BIT -(R0),B(R3) ; CHECK IT
1884 020754 012704 001000 BEQ 10 ; CONTINUE IF OK
1885 020760 012714 001002
1886 020764 011334
1887 020766 012700 001010
1888 020772 012710 125252
1889 020776 000277
1890 021000 042013
1891 021002 104401
1892 021004 104423 020740
1893 021010 034013
1894 021012 001402

```

PC 21014 = BIC OR BIT INSTRUCTION FAILED.

```

1895 021014 104423 020740      ERROR, T55      ;; *** BIC OR BIT INSTRUCTION FAILED ***
1896 021020 032713 052525      10: BIT #52525,(R3) ; CHECK IT
1897 021024 104401              CHECKCC+1       ; CHECK FOR CC = 1
1898 021026 104423 020740      ERROR, T55      ;; *** INCORRECT CC BITS ***
1899 021032 056013 000000      BIS 0(R0),(R3) ; SET THE BITS THAT WERE CLEARED
1900 021036 100402              BMI 20          ; CONTINUE IF OK
1901 021040 104423 020740      ERROR, T55      ;; *** BIT OR BIS INSTRUCTION FAILED ***
1902 021044 012720 077777      20: MOV #77777,(R0); ; SET ALL THE BITS AT TEMP1 EXCEPT SIGN
1903 021050 010002              MOV R0,R2
1904 021052 046213 177776      BIC -2(R2),(R3) ; TRY CLEARING THE OTHER BITS
1905 021056 104411              CHECKCC+11     ; CHECK FOR CC = 11
1906 021060 104423 020740      ERROR, T55      ;; *** INCORRECT CC BITS ***
1907 021064 020027 001012      CMP R0,#TEMP1+2 ; R0 SHOULD CONTAIN THE ADDRESS OF TEMP1+2
1908 021070 001402              BEQ 30
1909 021072 104423 020740      ERROR, T55      ;; *** R0 WRONG ***
1910 021076 010010              30: MOV R0,(R0)    ; PLACE ADDRESS OF TEMP2 IN TEMP2.
1911 021100 000263              SEVC           ; SET V AND C BITS.
1912 021102 044000              BIC -(R0),R0   ; SHOULD CLEAR R0.
1913 021104 104405              CHECKCC+5      ; CHECK FOR CC = 5
1914 021106 104423 020740      ERROR, T55      ;; *** INCORRECT CC BITS ***
1915 021112 037413 177776      BIT 0-2(R4),(R3) ; CHECK IT
1916 021116 104411              CHECKCC+11     ; CHECK FOR CC = 11
1917 021120 104423 020740      ERROR, T55      ;; *** INCORRECT CC BITS ***
1918 021124 012746 125252      MOV #125252,-(SP) ; SET EVERY OTHER BIT ON THE STACK
1919 021130 017423 177776      MOV 0-2(R4),(R3); ; SET ALL THE BITS AT LOCATION TEMP
1920 021134 046643 000000      BIC 0(SP),-(R3) ; CLEAR EVERY OTHER BIT AT LOCATION TEMP
1921 021140 022327 052525      CMP (R3),#52525 ; TEMP SHOULD CONTAIN # 52525
1922 021144 001402              BEQ 40
1923 021146 104423 020740      ERROR, T55      ;; *** BIC FAILED IN MODE 6 ***
1924 021152 012700 001014      40: MOV #TEMP2+2,R0 ; PLACE THE ADDRESS OF TEMP2+2 IN R0
1925 021156 010340              MOV R3,-(R0)   ; PLACE THE ADDRESS OF TEMP1 IN TEMP2
1926 021160 014330              MOV -(R3),0(R0); ; MOVE # 52525 IN LOCATION TEMP1
1927 021162 000263              SEVC           ; SET V & C BITS
1928 021164 035026              BIT 0-(R0),(SP); ; BIT TEST TEMP1 WITH STACK AND RESTORE STACK POINTER
1929 021166 104405              CHECKCC+5      ; CHECK FOR CC = 5
1930 021170 104423 020740      ERROR, T55      ;; *** INCORRECT CC BITS ***
1931 021174 020627 003776      CMP SP,#STACK   ; MAKE SURE THAT THE SP IS OK
1932 021200 001402              BEQ 50
1933 021202 104423 020740      ERROR, T55      ;; *** STACK POINTER FOULED UP ***
1934 021206
1935
1936
;*****
.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
1937 021212 012704 001010      ERROR, T56      ;; *** INCORRECT CC BITS ***
1938 021216 012714 077777      MOV #177776,(R4) ; R0 IS POINTING TO LOCATION TEMP
1939 021222 000261              MOV #TEMP,R0
1940 021224 005214              INC (R4)
1941 021226 104413              CHECKCC+11     ; CHECK CC = 11.
1942 021230 104423 021206      ERROR, T56      ;; *** INCORRECT CC BITS ***
1943 021234 012714 177776      INC (R4)
1944 021240 012700 001006      CHECKCC+11
1945 021244 005214              ERROR, T56
1946 021246 104411              INC (R4)
1947 021250 104423 021206      ERROR, T56
1948 021254 005214              INC (R4)

```

PC 21250 = INCORRECT CC BITS.

1949	021256	104405		CHECKCC=5		;	CHECK FOR CC = 5
1950	021260	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1951	021264	005214		INC (R4)			
1952	021266	104401		CHECKCC=1		;	CHECK FOR CC = 1
1953	021270	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1954	021274	026427	000000 000001	CMP 0(R4),#1		;	CHECK IT
1955	021302	001402		BEQ #1		;	CONTINUE IF OK
1956	021304	104423	021206	ERROR, T56		;;	*** INC INSTRUCTION FAILED ***
1957	021310	000261		SEC	41:		
1958	021312	005314		DEC (R4)		;	SUBTRACT ONES FROM LOCATION
1959	021314	104405		CHECKCC=5		;	CHECK FOR CC = 5
1960	021316	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1961	021322	005314		DEC (R4)			
1962	021324	104411		CHECKCC=11		;	CHECK CC = 11.
1963	021326	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1964	021332	012714	100000	MOV #100000,(R4)			
1965	021336	005314		DEC (R4)			
1966	021340	104403		CHECKCC=3		;	CHECK FOR CC = 3
1967	021342	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1968	021346	005314		DEC (R4)			
1969	021350	104401		CHECKCC=1		;	CHECK FOR CC = 1
1970	021352	104423	021206	ERROR, T56		;;	*** INCORRECT CC BITS ***
1971							
1972							
;*****							
				.SBTTL	T57 -- COM AND NEG		
				;*****			
	021356	104420	000057	T57: BEGINTEST, 57		;;	CHECK SWITCH REGISTER OPTIONS.
1973	021362	012703	001010	COM1: MOV #TEMP1,R3		;	LOAD ADDRESS
1974	021366	012713	125252	MOV #125252,(R3)		;	LOAD EVERY OTHER BIT
1975	021372	000277		SCC			
1976	021374	005163	000000	COM 0(R3)		;	1'S COMPLEMENT
1977	021400	104401		CHECKCC=1		;	CHECK FOR CC = 1
1978	021402	104423	021356	ERROR, T57		;;	*** INCORRECT CC BITS ***
1979	021406	022713	052525	CMP #52525,(R3)		;	CHECK IT
1980	021412	001402		BEQ #1		;	CONTINUE IF OK
1981	021414	104423	021356	ERROR, T57		;;	*** COM INSTRUCTION FAILED ***
1982	021420	000277		SCC	21:		
1983	021422	005123		COM (R3)		;	COMPLEMENT BACK
1984	021424	104411		CHECKCC=11		;	CHECK FOR CC = 11
1985	021426	104423	021356	ERROR, T57		;;	*** INCORRECT CC BITS ***
1986	021432	022743	125252	CMP #125252,-(R3)		;	CHECK IT
1987	021436	001402		BEQ #1		;	CONTINUE IF OK
1988	021440	104423	021356	ERROR, T57		;;	*** COM INSTRUCTION FAILED ***
1989	021444	010300		MOV R3,R0	31:	;	R0 IS NOW POINTING TO LOCATION TEMP1
1990	021446	012710	177777	MOV #177777,(R0)			
1991	021452	000277		SCC			
1992	021454	005110		COM (R0)			
1993	021456	104405		CHECKCC=5		;	CHECK FOR CC = 5
1994	021460	104423	021356	ERROR, T57		;;	*** INCORRECT CC BITS ***
1995							
1996	021464	012704	001010	NEG1: MOV #TEMP1,R4		;	LOAD ADDRESS
1997	021470	012724	000001	MOV #1,(R4)		;	LOAD THE LOCATION
1998	021474	010402		MOV R4,R2			
1999	021476	012762	100000 000000	MOV #100000,0(R2)			
2000	021504	005444		NEG -(R4)		;	2'S COMPLEMENT
2001	021506	104411		CHECKCC=11		;	CHECK FOR CC = 11
2002	021510	104423	021464	ERROR, NEG1		;;	*** INCORRECT CC BITS ***

PC 21510 = INCORRECT CC BITS.

2003 021514 022724 177777  
 2004 021520 001402  
 2005 021522 104423 021464  
 2006 021526 016444 000000  
 2007 021532 005414  
 2008 021534 104413  
 2009 021536 104423 021464  
 2010 021542 026214 000000  
 2011 021546 001402  
 2012 021550 104423 021464  
 2013 021554  
 2014  
 2015

```

20:  CMP      #177777,(R4)  ; CHECK IT
     BEQ      20          ; CONTINUE IF OK
     ERROR,  NEG1        ; *** NEG INSTRUCTION FAILED ***
     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      30          ; CONTINUE IF OK
     ERROR,  NEG1        ; *** NEG FAILED. ***
  
```

```

;*****
.SBTTL      T60 -- ROL AND ROR
;*****
  
```

2016 021554 104420 000060  
 2017 021560 012701 001012  
 2018 021564 012711 020000  
 2019 021570 000257  
 2020 021572 006121  
 2021 021574 006141  
 2022 021576 104412  
 2023 021600 104423 021554  
 2024 021604 022711 100000  
 2025 021610 001402  
 2026 021612 104423 021554  
 2027 021616 006161 000000  
 2028 021622 104407  
 2029 021624 104423 021554  
 2030 021630 010102  
 2031 021632 006112  
 2032 021634 022711 000001  
 2033 021640 001402  
 2034 021642 104423 021554  
 2035 021646 012702 001012  
 2036 021652 012712 000004  
 2037 021656 000257  
 2038 021660 006012  
 2039 021662 006012  
 2040 021664 022712 000001  
 2041 021670 001402  
 2042 021672 104423 021646  
 2043 021676 006012  
 2044 021700 104407  
 2045 021702 104423 021646  
 2046 021706 006012  
 2047 021710 104412  
 2048 021712 104423 021646  
 2049 021716 022712 100000  
 2050 021722 001402  
 2051 021724 104423 021646  
 2052 021730  
 2053  
 2054

```

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
       BEQ      10         ; CONTINUE IF OK
       ERROR,  T60         ; *** ROL INSTRUCTION FAILED ***
10:    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      10         ; CONTINUE IF OK
       ERROR,  ROR1       ; *** ROR INSTRUCTION FAILED ***
10:    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      20         ; CONTINUE IF OK
       ERROR,  ROR1       ; *** ROR FAILED. ***
20:
  
```

```

;*****
.SBTTL      T61 -- ASL AND ASR
;*****
  
```

T61 -- ASL AND ASR

```

021730 104420 000061
2055 021734 012703 001012
2056 021740 012713 020000
2057 021744 000257
2058 021746 006313
2059 021750 006313
2060 021752 104412
2061 021754 104423 021730
2062 021760 022713 100000
2063 021764 001402
2064 021766 104423 021730
2065 021772 006313
2066 021774 104407
2067 021776 104423 021730
2068 022002 006313
2069 022004 104404
2070 022006 104423 021730
2071
2072 022012 012704 001012
2073 022016 012703 001006
2074 022022 012714 000004
2075 022026 000257
2076 022030 006214
2077 022032 006214
2078 022034 022714 000001
2079 022040 001402
2080 022042 104423 022012
2081 022046 006214
2082 022050 104407
2083 022052 104423 022012
2084 022056 006214
2085 022060 104404
2086 022062 104423 022012
2087 022066 012713 100002
2088 022072 006213
2089 022074 006213
2090 022076 104411
2091 022100 104423 022012
2092 022104 022713 160000
2093 022110 001402
2094 022112 104423 022012
2095 022116
2096
2097
    
```

```

T61:  BEGINTEST, 61
ASL1:  MOV     #TEMP2,R3
        MOV     #20000,(R3)
        CCC
        ASL     (R3)
        ASL     (R3)
        CHECKCC+12
        ERROR,  T61
        CMP     #100000,(R3)
        BEQ     4#
        ERROR,  T61
4#:    ASL     (R3)
        CHECKCC+7
        ERROR,  T61
        ASL     (R3)
        CHECKCC+4
        ERROR,  T61

ASR1:  MOV     #TEMP2,R4
        MOV     #TEMP,R3
        MOV     #4,(R4)
        CCC
        ASR     (R4)
        ASR     (R4)
        CMP     #1,(R4)
        BEQ     2#
        ERROR,  ASR1
2#:    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
3#:
    
```

```

; ; CHECK SWITCH REGISTER OPTIONS.
; ; LOAD ADDRESS
; ; LOAD LOCATION
; ; CLEAR FLAGS
; ; SHIFT
; ;
; ; CHECK FOR CC = 12
; ; *** INCORRECT CC BITS ***
; ; CHECK IT
; ; CONTINUE IF OK
; ; *** ASL 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
; ; *** 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 ***
    
```

\*\*\*\*\*  
.SBTTL T62 -- ADC AND SBC  
\*\*\*\*\*

```

022116 104420 000062
2098 022122 012700 001006
2099 022126 005010
2100 022130 000257
2101 022132 005510
2102 022134 104404
2103 022136 104423 022116
2104 022142 000261
2105 022144 005510
2106 022146 000261
2107 022150 005510
    
```

```

T62:  BEGINTEST, 62
ADC1:  MOV     #TEMP,R0
        CLR     (R0)
        CCC
        ADC     (R0)
        CHECKCC+4
        ERROR,  T62
        SEC
        ADC     (R0)
        SEC
        ADC     (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 22136 = INCORRECT CC BITS.

2108	022152	104400		CHECKCC=0	;	CHECK FOR CC = 0	
2109	022154	104423	022116	ERROR, T62	;;	*** INCORRECT CC BITS ***	
2110	022160	022710	000002	CMP #2,(R0)	;	CHECK IT	
2111	022164	001402		BEG 4#	;	CONTINUE IF OK	
2112	022166	104423	022116	ERROR, T62	;;	*** ADC INSTRUCTION FAILED ***	
2113	022172	012710	077777	4#:	MOV #77777,(R0)	;	LOAD LARGEST POSITIVE NUMBER
2114	022176	000261		SEC	;	C=1	
2115	022200	005510		ADC (R0)	;	ADD C BIT=1	
2116	022202	104412		CHECKCC=12	;	CHECK FOR CC = 12	
2117	022204	104423	022116	ERROR, T62	;;	*** INCORRECT CC BITS ***	
2118	022210	022710	100000	CMP #100000,(R0)	;	CHECK IT	
2119	022214	001402		BEG 6#	;	CONTINUE IF OK	
2120	022216	104423	022116	ERROR, T62	;;	*** ADC INSTRUCTION FAILED ***	
2121	022222	012710	177777	6#:	MOV #-1,(R0)	;	LOAD -1
2122	022226	000261		SEC	;	C=1	
2123	022230	005510		ADC (R0)	;	ADD C BIT=1	
2124	022232	104405		CHECKCC=5	;	CHECK FOR CC = 5	
2125	022234	104423	022116	ERROR, T62	;;	*** INCORRECT CC BITS ***	
2126							
2127	022240	012701	001006	SBC1:	MOV #TEMP,R1	;	LOAD ADDRESS
2128	022244	012711	000003		MOV #3,(R1)	;	LOAD LOCATION
2129	022250	000257			CCC	;	CLEAR FLAGS
2130	022252	005611			SBC (R1)	;	SUBTRACT C BIT=0
2131	022254	104400		CHECKCC=0	;	CHECK FOR CC = 0	
2132	022256	104423	022240	ERROR, SBC1	;;	*** INCORRECT CC BITS ***	
2133	022262	022711	000003	CMP #3,(R1)	;	CHECK IT	
2134	022266	001402		BEG 2#	;	CONTINUE IF OK	
2135	022270	104423	022240	ERROR, SBC1	;;	*** SBC INSTRUCTION FAILED ***	
2136	022274	000261		2#:	SEC	;	C=1
2137	022276	005611			SBC (R1)	;	SUBTRACT C BIT=1
2138	022300	000261			SEC	;	C=1
2139	022302	005611			SBC (R1)	;	
2140	022304	104400		CHECKCC=0	;	CHECK FOR CC = 0	
2141	022306	104423	022240	ERROR, SBC1	;;	*** INCORRECT CC BITS ***	
2142	022312	022711	000001	CMP #1,(R1)	;	CHECK IT	
2143	022316	001402		BEG 3#	;	CONTINUE IF OK	
2144	022320	104423	022240	ERROR, SBC1	;;	*** SBC INSTRUCTION FAILED ***	
2145	022324	000261		3#:	SEC	;	C=1
2146	022326	005611			SBC (R1)	;	SUBTRACT C BIT=1
2147	022330	104404		CHECKCC=4	;	CHECK FOR CC = 4	
2148	022332	104423	022240	ERROR, SBC1	;;	*** INCORRECT CC BITS ***	
2149	022336	000261			SEC	;	C=1
2150	022340	005611			SBC (R1)	;	SUBTRACT C BIT = 1
2151	022342	104411		CHECKCC=11	;	CHECK FOR CC = 11	
2152	022344	104423	022240	ERROR, SBC1	;;	*** INCORRECT CC BITS ***	
2153	022350	022711	177777	CMP #-1,(R1)	;	CHECK IT	
2154	022354	001402		BEG 4#	;	CONTINUE IF OK	
2155	022356	104423	022240	ERROR, SBC1	;;	*** SBC INSTRUCTION FAILED ***	
2156	022362	012711	100000	4#:	MOV #100000,(R1)	;	LOAD R1
2157	022366	000261			SEC	;	C=1
2158	022370	005611			SBC (R1)	;	SUBTRACT C BIT = 1
2159	022372	104402		CHECKCC=2	;	CHECK FOR CC = 2	
2160	022374	104423	022240	ERROR, SBC1	;;	*** INCORRECT CC BITS ***	
2161							
2162							

```

;*****
.SBTTL      T63 -- SXT, SWAB, AND XOR
;*****

```



T63 -- SXT, SWAB, AND XOR

022400	104420	000063		T63:	BEGINTEST, 63	;; CHECK SWITCH REGISTER OPTIONS.
2163 022404	012702	001010		SXT1:	MOV #TEMP1,R2	;; LOAD ADDRESS
2164 022410	005012				CLR (R2)	;; CLEAR LOCATIONS
2165 022412	000277				SCC	
2166 022414	000254				CLNZ	
2167 022416	006712				SXT (R2)	;; SIGN EXTEND
2168 022420	104405				CHECKCC*5	;; CHECK FOR CC = 5
2169 022422	104423	022400			ERROR, T63	;; *** INCORRECT CC BITS ***
2170 022426	005712				TST (R2)	;; LOCATION SHOULD STILL BE 0
2171 022430	001402				BEQ 2#	;; CONTINUE IF OK
2172 022432	104423	022400			ERROR, T63	;; *** SXT INSTRUCTION FAILED ***
2173 022436	000273			2#:	SEVNC	;; SET N, V & C BITS
2174 022440	006712				SXT (R2)	;; SIGN EXTEND
2175 022442	104411				CHECKCC*11	;; CHECK FOR CC = 11
2176 022444	104423	022400			ERROR, T63	;; *** INCORRECT CC BITS ***
2177 022450	022712	177777			CMP #-1,(R2)	;; LOCATION SHOULD NOW HAVE -1
2178 022454	001402				BEQ SWAB1	;; CONTINUE IF OK
2179 022456	104423	022400			ERROR, T63	;; *** SXT FAILED ***
2180						
2181 022462	012703	001012		SWAB1:	MOV #TEMP2,R3	;; LOAD ADDRESS
2182 022466	012713	125125			MOV #125125,(R3)	;; LOAD BIT PATTERN INTO LOCATION
2183 022472	000277				SCC	
2184 022474	000250				CLN	
2185 022476	000313				SWAB (R3)	;; SWAP BYTES OF LOCATIONS
2186 022500	104410				CHECKCC*10	;; CHECK FOR CC = 10
2187 022502	104423	022462			ERROR, SWAB1	;; *** INCORRECT CC BITS ***
2188 022506	022713	052652			CMP #52652,(R3)	;; CHECK IT
2189 022512	001402				BEQ 1#	;; CONTINUE IF OK
2190 022514	104423	022462			ERROR, SWAB1	;; *** SWAB INSTRUCTION FAILED ***
2191 022520	012713	000377		1#:	MOV #377,(R3)	
2192 022524	000277				SCC	
2193 022526	000244				CLZ	
2194 022530	000363	000000			SWAB 0(R3)	
2195 022534	104404				CHECKCC*4	;; CHECK FOR CC = 4
2196 022536	104423	022462			ERROR, SWAB1	;; *** INCORRECT CC BITS ***
2197 022542	022713	177400			CMP #177400,(R3)	
2198 022546	001402				BEQ XOR1	
2199 022550	104423	022462			ERROR, SWAB1	;; *** SWAB FAILED. ***
2200						
2201 022554	012704	177777		XOR1:	MOV #-1,R4	;; LOAD LOCATIONS
2202 022560	012767	177777	156222		MOV #-1,TEMP1	
2203 022566	000277				SCC	
2204 022570	074467	156214			XOR R4,TEMP1	;; SHOULD PRODUCE 0'S IN TEMP1
2205 022574	104405				CHECKCC*5	;; CHECK FOR CC = 5
2206 022576	104423	022554			ERROR, XOR1	;; *** INCORRECT CC BITS ***
2207 022602	012767	077777	156200		MOV #77777,TEMP1	
2208 022610	012700	001010			MOV #TEMP1,R0	;; PLACE THE ADDRESS OF TEMP1 IN R0
2209 022614	000263				SEVNC	;; SET V & C BITS
2210 022616	000244				CLZ	
2211 022620	074410				XOR R4,(R0)	
2212 022622	104411				CHECKCC*11	;; CHECK FOR CC = 11
2213 022624	104423	022554			ERROR, XOR1	;; *** INCORRECT CC BITS ***
2214 022630	012701	125252			MOV #125252,R1	;; LOAD LOCATIONS
2215 022634	012720	052525			MOV #52525,(R0)	
2216 022640	000277				SCC	
2217 022642	074140				XOR R1,-(R0)	;; SHOULD PRODUCE ALL 1'S IN TEMP1
2218 022644	104411				CHECKCC*11	;; CHECK FOR CC = 11

PC 22646 = INCORRECT CC BITS.

2219 022646 104423 022554  
 2220 022652 022767 177777 156130  
 2221 022660 001402  
 2222 022662 104423 022554  
 2223 022666  
 2224  
 2225

```

ERROR, XOR1      ;: *** INCORRECT CC BITS ***
CMP      0-1,TEMP1 ; CHECK IT
BEQ      10       ; CONTINUE IF OK
ERROR, XOR1      ;: *** XOR FAILED. ***
  
```

10:

```

;*****
.SBTTL      T64 -- ADD, SUB, AND SOB
;*****
  
```

2226 022666 104420 000064  
 2227 022672 012700 001012  
 2228 022676 012701 001006  
 2229 022702 012767 021421 156102  
 2230 022710 011011  
 2231 022712 061011  
 2232 022714 104400  
 2233 022716 104423 022666  
 2234 022722 022767 043042 156056  
 2235 022730 001402  
 2236 022732 104423 022666  
 2237 022736 005010  
 2238 022740 060020

```

T64:      BEGINTEST, 64      ;: CHECK SWITCH REGISTER OPTIONS.
ADD1:     MOV      0TEMP2,R0 ;: LOAD ADDRESSES
          MOV      0TEMP,R1  ;:
          MOV      021421,TEMP2 ;: LOAD LOCATIONS
          MOV      (R0),(R1)  ;:
          ADD      (R0),(R1)  ;: ADD
          CHECKCC=0          ;: CHECK FOR CC = 0
          ERROR, T64         ;: *** INCORRECT CC BITS ***
          CMP      043042,TEMP ;: CHECK IT
          BEQ      10         ;: CONTINUE IF OK
          ERROR, T64         ;: *** ADD INSTRUCTION FAILED ***
          CLR      (R0)       ;: CLEAR LOCATION TEMP2
          ADD      R0,(R0)+   ;: PLACE THE ADDRESS OF TEMP2 IN TEMP2
          ;: POSSIBLE "Z" ASSEMBLY ERROR O.K.
  
```

10:

Z

2239 022742 024027 001014  
 2240 022746 001402  
 2241 022750 104423 022666  
 2242 022754 012767 156357 156030 20:  
 2243 022762 012011  
 2244 022764 064011  
 2245 022766 104411  
 2246 022770 104423 022666  
 2247 022774 022767 134736 156004  
 2248 023002 001402  
 2249 023004 104423 022666  
 2250 023010 012767 100000 155774 30:  
 2251 023016 011061 000000  
 2252 023022 066011 000000  
 2253 023026 104407  
 2254 023030 104423 022666  
 2255 023034 012767 021421 155746  
 2256 023042 012760 001010 000000  
 2257 023050 012711 156357  
 2258 023054 010004  
 2259 023056 067411 000000  
 2260 023062 104405  
 2261 023064 104423 022666  
 2262 023070 005430  
 2263 023072 012746 021421  
 2264 023076 063066 000000  
 2265 023102 104405  
 2266 023104 104423 022666  
 2267 023110 005726  
 2268  
 2269 023112 001402  
 2270 023114 104423 022666  
 2271 023120 012767 137777 155664 40:  
 2272 023126 062767 137777 155656

```

          CMP      -(R0),0TEMP2+2 ;: CHECK IT.
          BEQ      20
          ERROR, T64         ;: *** ADD INSTRUCTION FAILED IN MODE 2 ***
          MOV      0-21421,TEMP2 ;: LOAD LOCATIONS
          MOV      (R0),(R1)  ;:
          ADD      -(R0),(R1) ;: ADD
          CHECKCC=11        ;: CHECK FOR CC = 11
          ERROR, T64         ;: *** INCORRECT CC BITS ***
          CMP      0-43042,TEMP ;: CHECK IT
          BEQ      30         ;: CONTINUE IF OK
          ERROR, T64         ;: *** ADD INSTRUCTION FAILED ***
          MOV      0100000,TEMP2 ;: LOAD LOCATIONS
          MOV      (R0),0(R1)  ;:
          ADD      0(R0),(R1)  ;: ADD SHOULD RESULT AS 0'S
          CHECKCC=7         ;: CHECK FOR CC=7
          ERROR, T64         ;: *** INCORRECT CC BITS ***
          MOV      021421,TEMP1 ;: LOAD LOCATION TEMP1
          MOV      0TEMP1,0(R0) ;: PLACE THE ADDRESS OF TEMP1 IN TEMP2
          MOV      0-21421,(R1) ;: LOAD LOCATION TEMP
          MOV      R0,R4       ;: MAKE R4 POINT TO LOCATION TEMP2
          ADD      00(R4),(R1) ;: ADD SHOULD RESULT AS 0'S
          CHECKCC=5         ;: CHECK FOR CC=5
          ERROR, T64         ;: *** INCORRECT CC BITS ***
          NEG      0(R0)+     ;: NEGATE THE CONTENTS OF TEMP1
          MOV      021421,-(SP) ;: PLACE 0 21421 ON THE STACK
          ADD      0-(R0),0(SP) ;: ADD, SHOULD=0'S
          CHECKCC=5         ;: CHECK FOR CC=5
          ERROR, T64         ;: *** INCORRECT CC BITS ***
          TST      (SP)+     ;: CHECK THE STACK TO CONTAIN 0. ALSO
          ;: RESTORE THE STACK POINTER
          BEQ      40
          ERROR, T64         ;: *** ADD INSTRUCTION FAILED IN MODE 5 ***
          MOV      0137777,TEMP2 ;:
          ADD      0137777,TEMP2 ;:
  
```

40:

;JSD REV B

PC 23114 = ADD INSTRUCTION FAILED IN MODE 5.

2273	023134	104403				CHECKCC+3			:	CHECK CC=3
2274	023136	104423	022666			ERROR, T64			::	*** INCORRECT CC BITS ***
2275	023142	022767	077776	155642		CMP #77776,TEMP2				
2276	023150	001402				BEG SUB1				
2277	023152	104423	022666			ERROR, T64			::	*** ADD FAILED ***
2278										
2279	023156	012702	001006		SUB1:	MOV #TEMP,R2			:	LOAD ADDRESSES
2280	023162	012703	001010			MOV #TEMP1,R3			:	
2281	023166	012767	021421	155612		MOV #21421,TEMP			:	LOAD LOCATIONS
2282	023174	012767	156357	155606		MOV #-21421,TEMP1			:	
2283	023202	161213				SUB (R2),(R3)			:	RESULT SHOULD=-43042
2284	023204	104410				CHECKCC+10			:	CHECK FOR CC = 10
2285	023206	104423	023156			ERROR, SUB1			::	*** INCORRECT CC BITS ***
2286	023212	022767	134736	155570		CMP #-43042,TEMP1			:	CHECK IT
2287	023220	001402				BEG 1#			:	CONTINUE IF OK
2288	023222	104423	023156			ERROR, SUB1			::	*** SUB INSTRUCTION FAILED ***
2289	023226	012767	021421	155554	1#:	MOV #21421,TEMP1			:	LOAD LOCATION
2290	023234	161213				SUB (R2),(R3)			:	RESULT SHOULD=0
2291	023236	001402				BEG 2#				
2292	023240	104423	023156			ERROR, SUB1			::	*** SUB INSTRUCTION FAILED ***
2293	023244	012767	177777	155536	2#:	MOV #-1,TEMP1			:	LOAD LOCATIONS
2294	023252	012767	077777	155526		MOV #77777,TEMP			:	LOAD LOCATIONS
2295	023260	161312				SUB (R3),(R2)			:	RESULT SHOULD GIVE 100000 AND OVERFLOW
2296	023262	104413				CHECKCC+13			:	CHECK FOR CC = 13
2297	023264	104423	023156			ERROR, SUB1			::	*** INCORRECT CC BITS ***
2298	023270	022767	100000	155510		CMP #100000,TEMP			:	CHECK IT
2299	023276	001402				BEG 3#			:	CONTINUE IF OK
2300	023300	104423	023156			ERROR, SUB1			::	*** SUB INSTRUCTION FAILED ***
2301	023304	012712	177777		3#:	MOV #-1,(R2)				
2302	023310	161312				SUB (R3),(R2)				
2303	023312	104404				CHECKCC+4			:	CHECK FOR CC = 4
2304	023314	104423	023156			ERROR, SUB1			::	*** INCORRECT CC BITS ***
2305	023320	012767	077777	155460		MOV #77777,TEMP				
2306	023326	162767	077777	155452		SUB #77777,TEMP				
2307	023334	104404				CHECKCC+4			:	CHECK FOR CC=4
2308	023336	104423	023156			ERROR, SUB1			::	*** INCORRECT CC BITS ***
2309	023342	005767	155440			TST TEMP				
2310	023346	001402				BEG SOB1			:	TEMP SHOULD BE =0
2311	023350	104423	023156			ERROR, SUB1			::	*** SUB INSTRUCTION FAILED ***
2312										
2313	023354	012700	000012		SOB1:	MOV #10.,R0			:	LOAD REGISTERS
2314	023360	005001				CLR R1			:	
2315	023362	005201			1#:	INC R1			:	KEEP COUNT
2316	023364	020127	000012			CMP R1,#10.				
2317	023370	003402				BLE 2#				
2318	023372	104423	023354			ERROR, SOB1			::	*** SOB INSTRUCTION FAILED ***
2319	023376	000277			2#:	SCC				
2320	023400	077010				SOB R0,1#			:	SUB. 1 FROM REG. 0. GO BACK TO 1#
2321	023402	104417				CHECKCC+17			:	CHECK FOR CC = 17
2322	023404	104423	023354			ERROR, SOB1			::	*** INCORRECT CC BITS ***
2323	023410	005700				TST R0			:	REG. 0 = 0 ?
2324	023412	001402				BEG 3#			:	NO, FAILED
2325	023414	104423	023354			ERROR, SOB1			::	*** SOB INSTRUCTION FAILED ***
2326	023420	022701	000012		3#:	CMP #10.,R1			:	DID IT GO THRU 10 TIMES ?
2327	023424	001402				BEG 4#			:	CONTINUE IF OK
2328	023426	104423	023354			ERROR, SOB1			::	*** SOB INSTRUCTION FAILED ***
2329	023432	012704	000010		4#:	MOV #10,R4			:	PLACE #10 IN R4

PC 23426 = SOB INSTRUCTION FAILED.

2330 023436 077401  
 2331 023440 005704  
 2332 023442 001402  
 2333 023444 104423 023354  
 2334 023450  
 2335  
 2336

50: SOB R4,50 ; STAY HERE UNTILL R4 = 0  
 TST R4  
 BEQ 60 ; CONTINUE IF OK  
 ERROR, SOB1 ;: \*\*\* SOB FAILED \*\*\*

60:

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

023450 104420 000065  
 2337 023454 012700 001006  
 2338 023460 012701 001010  
 2339 023464 012711 177777  
 2340 023470 005010  
 2341 023472 106410  
 2342 023474 104400  
 2343 023476 104423 023450  
 2344 023502 106711  
 2345 023504 104404  
 2346 023506 104423 023450  
 2347 023512 022711 177400  
 2348  
 2349 023516 001402  
 2350 023520 104423 023450  
 2351 023524 005011  
 2352 023526 106427 000337  
 2353 023532 104417  
 2354 023534 104423 023450  
 2355 023540 106767 155244  
 2356 023544 104411  
 2357 023546 104423 023450  
 2358 023552 022767 000317 155230  
 2359 023560 001402  
 2360 023562 104423 023450  
 2361 023566  
 2362  
 2363

T65: BEGINTEST, 65 ;: CHECK SWITCH REGISTER OPTIONS.  
 TPSW1: MOV @TEMP,R0 ; PUT THE ADDRESS OF TEMP IN R0  
 MOV @TEMP1,R1 ; PUT THE ADDRESS OF TEMP1 IN R1  
 MOV @177777,(R1) ; TEMP1 = 177777  
 CLR (R0) ; TEMP = 0  
 MTPS (R0) ; PSW = 0  
 CHECKCC\*0 ; CHECK FOR CC = 0  
 ERROR, T65 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 MFPS (R1) ; MOVE PSW TO TEMP1  
 CHECKCC\*4 ; CHECK FOR CC = 4  
 ERROR, T65 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 CMP @177400,(R1) ; CHECK TEMP1 TO MAKE SURE THAT ONLY  
 ; THE LOWER BYTE WAS AFFECTED BY MFPS  
 BEQ 10  
 ERROR, T65 ;: \*\*\* MTPS OR MFPS INSTRUCTION FAILED \*\*\*  
 CLR (R1)  
 10: MTPS @337 ; SET PSW = 317 SINCE T BIT CAN NOT BE SET BY MTPS  
 CHECKCC\*17 ; CHECK FOR CC = 17  
 ERROR, T65 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 MFPS TEMP1 ; MOVE PSW TO TEMP1  
 CHECKCC\*11 ; CHECK FOR CC = 11 (C BIT SHOULD NOT BE EFFECTED BY MFPS)  
 ERROR, T65 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 CMP @317,TEMP1  
 BEQ 20  
 ERROR, T65 ;: \*\*\* MFPS INSTRUCTION FAILED IN MODE 6 \*\*\*

20:

;\*\*\*\*\*  
 .SBTTL T66 -- BYTE INSTRUCTIONS AFFECTING FULL WORDS  
 ;\*\*\*\*\*

023566 104420 000066  
 2364 023572 005000  
 2365 023574 000277  
 2366 023576 112700 000200  
 2367  
 2368 023602 104411  
 2369 023604 104423 023566  
 2370 023610 022700 177600  
 2371 023614 001402  
 2372 023616 104423 023566  
 2373 023622 000277  
 2374 023624 012700 177777  
 2375 023630 112700 000000  
 2376 023634 104405  
 2377 023636 104423 023566  
 2378 023642 005700  
 2379 023644 001402  
 2380 023646 104423 023566

T66: BEGINTEST, 66 ;: CHECK SWITCH REGISTER OPTIONS.  
 BTWRD: CLR R0  
 SCC  
 MOVB @200,R0 ; SET THE HIGHEST BIT OF THE  
 ; LOWER BYTE  
 CHECKCC\*11 ; CHECK FOR CC=11  
 ERROR, T66 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 CMP @177600,R0 ; CHECK FOR SIGN EXTENSION IN R0  
 BEQ 10  
 ERROR, T66 ;: \*\*\* SIGN WAS NOT EXTENDED IN R0 \*\*\*  
 10: SCC  
 MOV @177777,R0  
 MOVB @0,R0 ; CLEAR THE LOWER BYTE OF R0.  
 CHECKCC\*5 ; CHECK FOR CC=5  
 ERROR, T66 ;: \*\*\* INCORRECT CC BITS \*\*\*  
 TST R0 ; CHECK R0 FOR SIGN EXTENTION  
 BEQ 20  
 ERROR, T66 ;: \*\*\* SIGN WAS NOT EXTENDED IN R0. \*\*\*

PC 23646 = SIGN WAS NOT EXTENDED IN RO..

```

2381 023652 012704 001012      20:  MOV    #TEMP2,R4      ; R4 IS POINTING TO TEMP2
2382 023656 012714 000377      MOV    #377,(R4)      ; PLACE #377 IN LOCATION TEMP2
2383 023662 012706 003774      MOV    #STACK-2,R6
2384 023666 116426 000000      MOVB  0(R4),(R6)+    ; PUSH # 377 ON STACK
2385 023672 022706 003776      CMP    #STACK,R6
2386 023676 001402              BEQ    30
2387 023700 104423 023566      ERROR, T66           ;; *** R6 DID NOT GET INCREMENTED ***
2388 023704 124627 000377      30:  CPB    -(R6),#377    ; CHECK LOCATION STACK-2 TO
2389                                ; CONTAIN PROPER DATA
2390 023710 001402              BEQ    40
2391 023712 104423 023566      ERROR, T66           ;; *** BYTE INSTRUCTION IS FAILING WITH R6 ***
2392 023716 022706 003774      40:  CMP    #STACK-2,R6  ; CHECK THAT R6 WAS DECREMENTED
2393                                ; BY 2 BY A BYTE INSTRUCTION
2394 023722 001402              BEQ    50
2395 023724 104423 023566      ERROR, T66           ;; *** R6 WAS NOT DECREMENTED ***
2396 023730 016467 000000 155050 50:  MOV    0(R4),TEMP    ; SET THE LOWER BYTE OF LOCATION TEMP
2397 023736 005726              TST    (R6)+         ; RESTORE STACK POINTER
2398 023740 000277              SCC
2399 023742 114667 155041      MOVB  -(SP),TEMP+1  ; SET THE HIGHER BYTE OF LOCATION TEMP
2400 023746 104411              CHECKCC+11          ; CHECK FOR CC=11
2401 023750 104423 023566      ERROR, T66           ;; *** INCORRECT CC BITS ***
2402 023754 022767 177777 155024  CMP    #177777,TEMP  ; CHECK TEMP FOR THE CORRECT VALUE
2403 023762 001402              BEQ    60
2404 023764 104423 023566      ERROR, T66           ;; *** TEMP FOULED UP ***
2405 023770 005067 155012      60:  CLR    TEMP
2406 023774 000241              CLC
2407 023776 105167 155005      COMB  TEMP+1        ; WRITE 1'S IN THE HIGHER BYTE OF TEMP
2408 024002 104411              CHECKCC+11          ; CHECK FOR CC=11
2409 024004 104423 023566      ERROR, T66           ;; *** INCORRECT CC BITS ***
2410 024010 022767 177400 154770  CMP    #177400,TEMP
2411 024016 001402              BEQ    70
2412 024020 104423 023566      ERROR, T66           ;; *** TEMP FOULED UP ***
2413
2414 024024 004767 011026      70:  CALL  ENDSEG        ; CHECK FOR LOOP-IN-SEG...
2415                                ; ...RETURN AND FALL THRU IF NOT.
2416                                .SBTTL

```

SECTION 2 -- T11 TRAPS AND INTERRUPTS.

2418  
2419  
2420 024030 012767 036350 012270  
2421  
2422

.SBTTL SECTION 2 -- T11 TRAPS AND INTERRUPTS.

S2: MOV #SEG2,SEGN ; SEGMENT 2 TEXT.

;\*\*\*\*\*  
.SBTTL T67 -- TEST AUTO INCREMENT/DECREMENT OF STACK POINTER (R6)  
;\*\*\*\*\*

024036 104420 000067  
2423 024042 012700 001026  
2424 024046 010006  
2425 024050 112667 154732  
2426 024054 020627 001030  
2427 024060 001402  
2428 024062 104423 024036  
2429 024066 010006  
2430 024070 114667 154712  
2431 024074 020627 001024  
2432 024100 001402  
2433 024102 104423 024036  
2434 024106 010006  
2435 024110 112626  
2436 024112 020627 001032  
2437 024116 001402  
2438 024120 104423 024036  
2439 024124 010006  
2440 024126 005001  
2441 024130 122621  
2442 024132 020627 001030  
2443 024136 001402  
2444 024140 104423 024036  
2445 024144 020127 000001  
2446 024150 001402  
2447 024152 104423 024036  
2448 024156 010006  
2449 024160 005001  
2450 024162 122126  
2451 024164 020127 000001  
2452 024170 001402  
2453 024172 104423 024036  
2454 024176 020627 001030  
2455 024202 001402  
2456 024204 104423 024036  
2457 024210  
2458  
2459

T67: BEGINTEST, 67 ;; CHECK SWITCH REGISTER OPTIONS.  
R6TST: MOV #K4,R0 ;; USE K4 AS PSEUDO STACK...  
MOV R0,R6 ;; INIT SP.  
MOVSB (R6)+,TEMP ;;SIX SHOULD INCREMENT BY TWO  
CMP R6,#K4+2  
BEQ 10  
ERROR, T67 ;; \*\*\* R6 DID NOT AUTO INCREMENT BY TWO \*\*\*  
10: MOV R0,R6  
MOVSB -(R6),TEMP ;;SHOULD DECREMENT BY TWO  
CMP R6,#K4-2  
BEQ 20  
ERROR, T67 ;; \*\*\* R6 DID NOT AUTO DECREMENT BY 2 \*\*\*  
20: MOV R0,R6  
MOVSB (R6)+,(R6)+ ;;DOUBLE AUTO INCREMENT OF R6  
CMP R6,#K4+4  
BEQ 30  
ERROR, T67 ;; \*\*\* WRONG AUTO INCREMENT OF R6 \*\*\*  
30: MOV R0,R6  
CLR R1  
CMPB (R6)+,(R1)+ ;;TEST INCREMENT OF R6 AND R1.  
CMP R6,#K4+2  
BEQ 40  
ERROR, T67 ;; \*\*\* WRONG INCREMENT OF R6 \*\*\*  
40: CMP R1,#1  
BEQ 50  
ERROR, T67 ;; \*\*\* WRONG INCREMENT OF R1 \*\*\*  
50: MOV R0,R6  
CLR R1  
CMPB (R1)+,(R6)+ ;; TEST INCREMENT OF R1 AND R6.  
CMP R1,#1  
BEQ 60  
ERROR, T67 ;; \*\*\* WRONG INCREMENT OF R1 \*\*\*  
60: CMP R6,#K4+2  
BEQ 70  
ERROR, T67 ;; \*\*\* WRONG INCREMENT OF R6 \*\*\*  
70:

;\*\*\*\*\*  
.SBTTL T70 -- TEST BYTE TRANSFERS USING STACK POINTER (R6)  
;\*\*\*\*\*

024210 104420 000070  
2460 024214 012767 123456 154606  
2461 024222 012767 050505 154570  
2462 024230 012701 001020  
2463 024234 012706 001030  
2464 024240 112621  
2465 024242 022767 050456 154550  
2466 024250 001402  
2467 024252 104423 024210  
2468

T70: BEGINTEST, 70 ;; CHECK SWITCH REGISTER OPTIONS.  
MOV #123456,K5  
MOV #050505,K1  
MOV #K1,R1 ;;R1=(050505)K1  
MOV #K5,R6 ;;R6=(123456)K5  
MOVSB (R6)+,(R1)+ ;;LOW .BYTE OF R6 0 R1  
CMP #050456,K1  
BEQ 10  
ERROR, T70 ;; \*\*\* FALSE TRANSFER OF .BYTE \*\*\*

PC 24252 = FALSE TRANSFER OF .BYTE.

```

2469 024256 012767 123456 154544 10:  MOV    0123456,K5
2470 024264 012767 050505 154526  MOV    0050505,K1
2471 024272 012701 001020  MOV    0K1,R1          ;R1(050505)K1
2472 024276 012706 001032  MOV    0K6,R6          ;R6(123456)K5
2473 024302 114621  MOV    -(R6),(R1)+    ;LOW .BYTE OF R6 TO R1 (DECREMENT)
2474 024304 026727 154510 050456  CMB    K1,0050456
2475 024312 001402  BEQ    20
2476 024314 104423 024210  ERROR, T70          ;; *** FALSE R6 .BYTE TRANSFER ***
2477
2478 024320 012767 123456 154472 20:  MOV    0123456,K1
2479 024326 012767 050505 154474  MOV    0050505,K5
2480 024334 012701 001020  MOV    0K1,R1          ;(123456)
2481 024340 012706 001030  MOV    0K5,R6          ;(050505)
2482 024344 112126  MOV    (R1)+,(R6)+    ;LOW OF R1 TO LOW OF R6
2483 024346 022767 050456 154454  CMB    0050456,K5
2484 024354 001402  BEQ    30
2485 024356 104423 024210  ERROR, T70          ;; *** FALSE R6 .BYTE TRANSFER ***
2486
2487 024362 012767 123456 154430 30:  MOV    0123456,K1
2488 024370 012767 050505 154432  MOV    0050505,K5
2489 024376 012701 001021  MOV    0K1+1,R1       ;123456
2490 024402 012706 001030  MOV    0K5,R6         ;050505
2491 024406 112126  MOV    (R1)+,(R6)+    ;HIGH OF R1 TO LOW OF R6
2492 024410 026727 154414 050647  CMB    K5,0050647
2493 024416 001402  BEQ    40
2494 024420 104423 024210  ERROR, T70          ;; *** FALSE R6 .BYTE TRANSFER ***
2495
2496 024424 012767 123456 154366 40:  MOV    0123456,K1
2497 024432 012767 050505 154370  MOV    0050505,K5
2498 024440 012701 001021  MOV    0K1+1,R1       ;R1-123456-ODD ADDRESS
2499 024444 012706 001030  MOV    0K5,R6         ;R6-050505--.EVEN ADDRESS
2500 024450 112621  MOV    (R6)+,(R1)+    ;LOW OF R6 TO HIGH OF R1
2501 024452 022767 042456 154340  CMB    0042456,K1
2502 024460 001402  BEQ    50
2503 024462 104423 024210  ERROR, T70          ;; *** FAILED LOW OF 6 TO HIGH OF 1 ***
2504 024466
2505
2506
;*****
.SBTL      T71 -- TEST BYTE COMPARES ON SEQUENTIAL ODD/EVEN ADDRESSES
;*****
T71:      BEGINTEST, 71          ;; CHECK SWITCH REGISTER OPTIONS.
          CLR      K0
          MOV    052525,K1      ; K0 = 000000
          MOV    052400,K2      ; K1 = 052525
          CMB    K1,K1+1        ; K2 = 052400
          BEQ    10
          ERROR, T71          ;; *** LOW TO HIGH IN SAME WORD FAILS ***
2507 024472 005067 154320  MOV    052525,K1
2508 024476 012767 052525 154314  MOV    052400,K2
2509 024504 012767 052400 154310  CMB    K1,K1+1
2510 024512 126767 154302 154301  BEQ    20
          ERROR, T71          ;; *** HIGH TO LOW IN SAME WORD FAILS ***
2511 024520 001402  BEQ    30
2512 024522 104423 024466  MOV    052400,K1
2513 024526 126767 154267 154264 10:  CMB    K2+1,K1
          ERROR, T71          ;; *** HIGH TO LOW IN DIFFERENT WORDS FAILS ***
2514 024534 001402  BEQ    40
2515 024536 104423 024466  MOV    052400,K2
2516 024542 126767 154255 154250 20:  CMB    K2,K0
          ERROR, T71
2517 024550 001402  BEQ    40
2518 024552 104423 024466  MOV    052400,K2
2519 024556 126767 154240 154232 30:  CMB    K1+1,K2+1
          ERROR, T71
2520 024564 001402  BEQ    40
2521 024566 104423 024466  MOV    052400,K2
2522 024572 126767 154223 154223 40:  CMB    K1+1,K2+1
          ERROR, T71

```

PC 24566 = EVEN TO EVEN FAILED.

```

2523 024600 001402          BEQ      50
2524 024602 104423 024466   ERROR,   T71          ;; *** ODD TO ODD FAILED ***
2525 024606 126767 154210 154207 50:   CMPB    K2,K2+1
2526 024614 001002          BNE      60
2527 024616 104423 024466   ERROR,   T71          ;; *** LOW TO HIGH IN SAME WORD FAILED ***
2528 024622 126767 154175 154173 60:   CMPB    K2+1,K2+1
2529 024630 001402          BEQ      70
2530 024632 104423 024466   ERROR,   T71          ;; *** HIGH TO HIGH IN SAME WORD FAILED ***
2531 024636 126767 154160 154155 70:   CMPB    K2,K1+1
2532 024644 001002          BNE      80
2533 024646 104423 024466   ERROR,   T71          ;; *** EVEN TO ODD FAILED ***
2534 024652
2535
2536

```

```

;*****
.SBTTL      T72 -- TEST THAT BUS TIME-OUT TRAPS TO THE "RESTART" ADDRESS
;*****

```

```

2537 024652 104420 000072   T72:    BEGINTEST, 72          ;; CHECK SWITCH REGISTER OPTIONS.
2538
2539
2540
2541 024656 012706 003776          MOV      #STACK,SP          ; SET STACK POINTER.
2542 024662 012767 024712 153114   MOV      #30,ERRVEC        ; SET TRAP VECTOR.
2543 024670 106427 000000          MTPS    #PRIO
2544 024674 005737 160000          TST     #0160000           ; NEXT -- TIME-OUT AND TRAP.
2545 024700 000257          CCC
2546 024702 000240          NOP
2547 024704 104423 024652          ERROR,   T72              ; SHOULD EXECUTE THIS...
2548 024710 000414          BR      50                 ; ...THEN TRAP AND STACK THIS PC.
2549 024712 011600          MOV     (SP),R0            ;; *** BUS TIME-OUT DIDN'T TRAP AT ALL ***
2550 024714 016601 000002          MOV     2(SP),R1          ; GET STACKED PC...
2551 024720 020027 024702          CMP     R0,#020           ; ...AND PSM.
2552 024724 001402          BEQ     40                 ; BR IF PC IS RIGHT.
2553 024726 104423 024652          ERROR,   T72              ;; *** STACKED PC INCORRECT ON TIME-OUT TRAP ***
2554
2555 024732 005701          TST     R1
2556 024734 001402          BEQ     50                 ; BR IF PSM IS RIGHT.
2557 024736 104423 024652          ERROR,   T72              ;; *** STACKED PSM INCORRECT ON TIME-OUT TRAP ***
2558
2559 024742 012767 000006 153034 50:   MOV     #ERRVEC+2,ERRVEC  ; RESET TRAP CATCHER.
2560
2561

```

```

;*****
.SBTTL      T73 -- TEST THAT AN "ILLEGAL" INSTRUCTION TRAPS TO 4
;*****

```

```

2562 024750 104420 000073   T73:    BEGINTEST, 73          ;; CHECK SWITCH REGISTER OPTIONS.
2563
2564
2565
2566 024754 012700 025012          MOV     #10,R0            ; 1ST PASS USES A "JMP R"
2567 024760 000402          SKP2
2568 024762 012700 025016          MOV     #20,R0            ; 2ND PASS USES A "JSR R,R"
2569 024766 012706 003776          MOV     #STACK,SP        ; STACK POINTER SETUP
2570 024772 012767 025030 153004   MOV     #30,ERRVEC        ; SET TRAP PC...
2571 025000 005067 153002          CLR     ERRVEC+2          ; ...AND PSM.
2572 025004 106427 000317          MTPS    #PRI6117         ; SET CURRENT (OLD) PSM.
2573 025010 000110          JMP     (R0)              ; DISPATCH TO ONE OR THE OTHER.

```



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

```

2574 025012 000101      10:  JMP      R1          ; ILLEGAL JMP -- SHOULD TRAP.
2575 025014 000401      SKP1
2576 025016 004101      20:  JSR      R1,R1       ; ILLEGAL JSR -- SHOULD TRAP.
2577 025020 000240      NOP
2578 025022 104423 024750  ERROR,   T73        ;; *** ILLEGAL JMP/JSR DIDN'T TRAP ***
2579 025026 000427      BR       70
2580
2581 025030 106701      30:  MFPS     R1          ; OK, GET CURRENT (NEW) PSM.
2582 025032 001402      BEQ      40          ; BR IF IT'S RIGHT.
2583 025034 104423 024750  ERROR,   T73        ;; *** PSM INCORRECT AFTER ILLEGAL INSTR TRAP TO 4 ***
2584 025040 020627 003772  40:  CMP      SP,#STACK-4
2585 025044 001402      BEQ      50          ; BR IF STACK IS RIGHT.
2586 025046 104423 024750  ERROR,   T73        ;; *** STACK POINTER INCORRECT ***
2587 025052 021627 025014  50:  CMP      (SP),#10*2
2588 025056 001405      BEQ      60          ; BR IF STACKED PC IS RIGHT (JMP).
2589 025060 021627 025020  CMP      (SP),#20*2
2590 025064 001402      BEQ      60          ; BR IF STACKED PC IS RIGHT (JSR).
2591 025066 104423 024750  ERROR,   T73        ;; *** INCORRECT PC SAVED ON STACK ***
2592 025072 026627 000002 000317  60:  CMP      2(SP),#PRI6!17
2593 025100 001402      BEQ      70          ; BR IF STACKED (OLD) PSM IS RIGHT.
2594 025102 104423 024750  ERROR,   T73        ;; *** INCORRECT PSM SAVED ON STACK ***
2595
2596 025106 020027 025012  70:  CMP      R0,#10     ; IS THIS THE 1ST PASS ??
2597 025112 001723      BEQ      80          ; LOOP ONCE IF SO...
2598 025114 012767 000006 152662  MOV      #ERRVEC+2,ERRVEC ; ...OTHERWISE, RESET VECTOR...
2599
2600

```

```

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

```

```

2601 025122 104420 000074      T74:  BEGINTEST, 74      ;; CHECK SWITCH REGISTER OPTIONS.
2602 025126 012706 003776      MOV      #STACK,SP  ;STACK POINTER SETUP
2603 025132 012767 025162 152650  MOV      #20,RESVEC ; SET TRAP PC...
2604 025140 003067 152646      CLR      RESVEC+2   ; ...AND PSM.
2605 025144 106427 000317  MTPS     #PRI6!17   ; SET CURRENT (OLD) PSM.
2606 025150 000010      RESRVD          ; RESERVED INSTRUCTION -- SHOULD TRAP
2607 025152 000240      10:  NOP
2608 025154 104423 025122  ERROR,   T74        ;; *** RESERVED OPCODE DIDN'T TRAP ***
2609 025160 000424      BR       60
2610 025162 106700      20:  MFPS     R0
2611 025164 001402      BEQ      30          ; BR IF NEW PSM IS RIGHT.
2612 025166 104423 025122  ERROR,   T74        ;; *** NEW PSM INCORRECT AFTER TRAP TO 10 ***
2613 025172 020627 003772  30:  CMP      SP,#STACK-4
2614 025176 001402      BEQ      40          ; BR IF SP PUSHED DOWN RIGHT.
2615 025200 104423 025122  ERROR,   T74        ;; *** STACK POINTER INCORRECT ***
2616 025204 021627 025152  40:  CMP      (SP),#10
2617 025210 001402      BEQ      50          ; BR IF STACKED PC IS RIGHT.
2618 025212 104423 025122  ERROR,   T74        ;; *** INCORRECT PC SAVED ON STACK ***
2619 025216 026627 000002 000317  50:  CMP      2(SP),#PRI6!17
2620 025224 001402      BEQ      60          ; BR IF STACKED (OLD) PSM IS RIGHT.
2621 025226 104423 025122  ERROR,   T74        ;; *** INCORRECT PSM SAVED ON STACK ***
2622
2623 025232 012767 000012 152550  60:  MOV      #RESVEC+2,RESVEC ; RESET VECTOR.
2624
2625

```

```

;*****
.SBTTL      T75 -- TEST THAT A "BPT" INSTRUCTION TRAPS TO 14
;*****

```

T75 -- TEST THAT A "BPT" INSTRUCTION TRAPS TO 14

```

025240 104420 000075          T75:  BEGINTEST, 75          ;; CHECK SWITCH REGISTER OPTIONS.
2626 025244 012706 003776    152536  MOV     #STACK,SP      ;STACK POINTER SETUP
2627 025250 012767 025300    MOV     #20,BPTVEC     ; SET TRAP PC...
2628 025256 005067 152534    CLR     BPTVEC+2      ;...AND PSM.
2629 025262 106427 000317    MTPS   #PRI6!17      ; SET CURRENT (OLD) PSM.
2630 025266 000003          BPT                    ; BPT -- SHOULD TRAP.
2631 025270 000240          10:  NOP
2632 025272 104423 025240    ERROR, T75           ;; *** BPT DIDN'T TRAP ***
2633 025276 000424          BR     60
2634
2635 025300 106700          20:  MFPS   R0
2636 025302 001402          BEQ    30             ; BR IF NEW PSM IS RIGHT.
2637 025304 104423 025240    ERROR, T75           ;; *** PSM INCORRECT AFTER BPT TRAP TO 14 ***
2638 025310 020627 003772    30:  CMP     SP,#STACK-4
2639 025314 001402          BEQ    40             ; BR IF STACK IS RIGHT.
2640 025316 104423 025240    ERROR, T75           ;; *** STACK POINTER INCORRECT ***
2641 025322 021627 025270    40:  CMP     (SP),#10
2642 025326 001402          BEQ    50             ; BR IF STACKED PC IS RIGHT.
2643 025330 104423 025240    ERROR, T75           ;; *** INCORRECT PC SAVED ON STACK ***
2644 025334 026627 000002 000317 50:  CMP     2(SP),#PRI6!17
2645 025342 001402          BEQ    60             ; BR IF STACKED (OLD) PSM IS RIGHT.
2646 025344 104423 025240    ERROR, T75           ;; *** INCORRECT PSM SAVED ON STACK ***
2647
2648 025350 012767 000016 152436 60:  MOV     #BPTVEC+2,BPTVEC ; RESET VECTOR.
2649
2650

```

```

;*****
.SBTTL      T76 -- TEST THAT AN "IOT" INSTRUCTION TRAPS TO 20
;*****

```

```

025356 104420 000076          T76:  BEGINTEST, 76          ;; CHECK SWITCH REGISTER OPTIONS.
2651 025362 012706 003776    152424  MOV     #STACK,SP      ;STACK POINTER SETUP
2652 025366 012767 025416    MOV     #20,IOTVEC     ; SET TRAP PC...
2653 025374 005067 152422    CLR     IOTVEC+2      ;...AND PSM.
2654 025400 106427 000317    MTPS   #PRI6!17      ; SET CURRENT (OLD) PSM.
2655 025404 000004          IOT                    ; IOT -- SHOULD TRAP.
2656 025406 000240          10:  NOP
2657 025410 104423 025356    ERROR, T76           ;; *** IOT DID NOT TRAP ***
2658 025414 000424          BR     60
2659
2660 025416 106700          20:  MFPS   R0
2661 025420 001402          BEQ    30             ; BR IF NEW PSM IS RIGHT.
2662 025422 104423 025356    ERROR, T76           ;; *** PSM INCORRECT AFTER IOT TRAP TO 20 ***
2663 025426 020627 003772    30:  CMP     SP,#STACK-4
2664 025432 001402          BEQ    40             ; BR IS STACK IS RIGHT.
2665 025434 104423 025356    ERROR, T76           ;; *** STACK POINTER INCORRECT ***
2666 025440 021627 025406    40:  CMP     (SP),#10
2667 025444 001402          BEQ    50             ; BR IF STACKED PC IS RIGHT.
2668 025446 104423 025356    ERROR, T76           ;; *** INCORRECT PC SAVED ON STACK ***
2669 025452 026627 000002 000317 50:  CMP     2(SP),#PRI6!17
2670 025460 001402          BEQ    60             ; BR IF STACKED (OLD) PSM IS RIGHT.
2671 025462 104423 025356    ERROR, T76           ;; *** INCORRECT PSM SAVED ON STACK ***
2672
2673 025466 012767 000022 152324 60:  MOV     #IOTVEC+2,IOTVEC ; RESET VECTOR.
2674
2675

```

```

;*****
.SBTTL      T77 -- TEST THAT "POWER-FAIL" TRAPS TO 24 (REQUIRES Q-BUS EXERCISER)
;*****
T77:  BEGINTEST, 77          ;; CHECK SWITCH REGISTER OPTIONS.

```

025474 104420 000077

T77 -- TEST THAT "POWER-FAIL" TRAPS TO 24 (REQUIRES Q-BUS EXER

```

2676 025500 012706 003776          MOV    #STACK,SP      ;STACK POINTER SETUP
2677 025504 004767 007700          CALL  QBXIN           ; Q-BUS EXERCISOR AVAILABLE ??
2678 025510 001456                BEQ    7#              ; BYPASS TEST IF NOT.
2679 025512 010004                MOV    R0,R4          ; YES, MOVE POINTER TO R4 AND FALL THRU.
2680
2681                                ;
2682                                ; QBX CSR1<13> IS USED TO EMULATE A POWER DOWN/UP SEQUENCE.
2683                                ; CAUTION:
2684                                ; WE ONLY WANT TO RECOGNIZE THE "DOWN" HALF OF THE POWER SEQUENCE.
2685                                ; ABORT THE PMR-CYCLE AT THAT POINT SO WE DON'T CRASH TO ODT.
2686 025514 012767 025564 152302      MOV    #2#,PMRVEC     ; SET POWER TRAP PC...
2687 025522 005067 152300          CLR    PMRVEC+2       ;...AND PSW.
2688 025526 106427 000300          MTPS  #PRI6           ; SET CURRENT (OLD) PSW.
2689 025532 012700 140000          MOV    #140000,R0    ; IF LOOPING, WE HAVE TO WAIT LONG...
2690 025536 077001                SOB    R0,,           ;...ENOUGH (≈250MSEC) TO INSURE THE...
2691                                ;...PREVIOUS BPOK CYCLE WAS FINISHED...
2692                                ;...(IN QBX) BEFORE TRYING ANOTHER.
2693 025540 012700 000005          MOV    #5,R0         ; OK, SET A 25 USEC TIMER.
2694 025544 052714 020000          BIS    #20000,(R4)   ; SET PMR-CYCLE REQUEST...
2695 025550 000257                CCC                ;...SHOULD SEE IT AFTER THIS...
2696 025552 077001 1#           SOB    R0,,           ;...AND STACK THIS PC.
2697 025554 005014                CLR    (R4)          ; TRAP NOT RECEIVED, CLEAR THE REQUEST.
2698 025556 104423 025474          ERROR, T77          ;; *** POWER-FAIL DID NOT TRAP ***
2699 025562 000426                BR     6#
2700
2701 025564 106700 2#           MFPS  R0              ; OK, GET NEW PSW...
2702 025566 005014                CLR    (R4)          ;...AND CLEAR PMR-CYCLE REQUEST.
2703 025570 005700                TST   R0
2704 025572 001402                BEQ    3#           ; BR IF NEW PSW IS RIGHT.
2705 025574 104423 025474          ERROR, T77          ;; *** PSW INCORRECT AFTER PMR-FAIL TRAP TO 24 ***
2706 025600 020627 003772 3#           CMP    SP,#STACK-4   ; BR IS STACK IS RIGHT.
2707 025604 001402                BEQ    4#           ;...
2708 025606 104423 025474          ERROR, T77          ;; *** STACK POINTER INCORRECT ***
2709 025612 021627 025552 4#           CMP    (SP),#1#     ; BR IF STACKED PC IS RIGHT.
2710 025616 001402                BEQ    5#           ;...
2711 025620 104423 025474          ERROR, T77          ;; *** INCORRECT PC SAVED ON STACK ***
2712 025624 026627 000002 000300 5#           CMP    2(SP),#PRI6   ; BR IF STACKED (OLD) PSW IS RIGHT.
2713 025632 001402                BEQ    6#           ;...
2714 025634 104423 025474          ERROR, T77          ;; *** INCORRECT PSW SAVED ON STACK ***
2715
2716 025640 012767 000026 152156 6#           MOV    #PMRVEC+2,PMRVEC ; RESTORE VECTOR.
2717 025646 7#
2718
2719                                ;
2720                                ; *****
2721                                ; .SBTTL      T100 -- TEST THAT AN "EMT" INSTRUCTION TRAPS TO 30
2722                                ; *****
2723 025646 104420 000100          T100:  BEGINTEST, 100      ; CHECK SWITCH REGISTER OPTIONS.
2724                                ;
2725                                ; NOW EMT'S HAVE TO BE OK OR WE'D NEVER HAVE GOTTEN STARTED IN THE FIRST
2726                                ; PLACE. WE'VE ALREADY VERIFIED ALL VARIATIONS OF THE EMT OPCODE.
2727                                ; SO WE'LL JUST TEST FOR CORRECT STACK MANIPULATION HERE.
2728                                ;
2729 025652 012706 003776          MOV    #STACK,SP      ;STACK POINTER SETUP
2730 025656 016701 152146          MOV    EMTVEC,R1     ; SAVE EMT VECTOR IN R1.
2731 025662 012767 025706 152140          MOV    #2#,EMTVEC    ; SET TRAP PC...
2732 025670 005067 152136          CLR    EMTVEC+2       ;...AND PSW.
2733 025674 106427 000317          MTPS  #PRI6!17      ; SET CURRENT (OLD) PSW.

```

T100 -- TEST THAT AN "ENT" INSTRUCTION TRAPS TO 30

```

2730 025700 104000          EMT          ; ENT -- SHOULD TRAP...
2731 025702 000240      10:  NOP          ; ...AND STACK THIS PC.
2732 025704 000240          NOP          ; IF IT DOESN'T, JUST FALL THRU ANYWAY.
2733
2734 025706 106700      20:  MFPS        RO          ; GET NEW PSM...
2735 025710 010167 152114  MOV        R1,ENTVEC ; ...AND RESTORE VECTOR.
2736 025714 005700          TST        RO
2737 025716 001402          BEQ        30          ; BR IF NEW PSM IS RIGHT.
2738 025720 104423 025646  ERROR,    T100        ;; *** PSM INCORRECT AFTER ENT TRAP TO 30 ***
2739 025724 020627 003772  30:  CMP        SP,#STACK-4
2740 025730 001402          BEQ        40          ; BR IF SP IS RIGHT.
2741 025732 104423 025646  ERROR,    T100        ;; *** STACK POINTER INCORRECT ***
2742 025736 021627 025702  40:  CMP        (SP),#10
2743 025742 001402          BEQ        50          ; BR IF STACKED PC IS RIGHT.
2744 025744 104423 025646  ERROR,    T100        ;; *** INCORRECT PC SAVED ON STACK ***
2745 025750 026627 000002 000317 50:  CMP        2(SP),#PRI6!17
2746 025756 001402          BEQ        60          ; BR IF STACKED (OLD) PSM IS RIGHT.
2747 025760 104423 025646  ERROR,    T100        ;; *** INCORRECT PSM SAVED ON STACK ***
2748 025764
2749
2750

```

```

;*****
.SBTTL      T101 -- TEST THAT A "TRAP" INSTRUCTION TRAPS TO 34
;*****

```

```

025764 104420 000101  T101:  BEGINTEST, 101      ;; CHECK SWITCH REGISTER OPTIONS.
;
; WE ALREADY KNOW THAT ALL VARIATIONS OF THE TRAP OPCODE WORK.
; SO WE'LL JUST CHECK THAT THE STACK IS HANDLED PROPERLY.
;
2751
2752
2753
2754
2755 025770 012706 003776      MOV        #STACK,SP ;STACK POINTER SETUP
2756 025774 016701 152034      MOV        TRPVEC,R1 ; SAVE TRAP VECTOR IN R1.
2757 026000 012767 026024 152026  MOV        #2,TRPVEC ; SET TRAP PC...
2758 026006 005067 152024      CLR        TRPVEC+2 ; ...AND PSM.
2759 026012 106427 000317      MTPS      #PRI6!17  ; SET CURRENT (OLD) PSM.
2760 026016 104400          TRAP#0          ; TRAP -- SHOULD TRAP...
2761 026020 000240      10:  NOP          ; ...AND STACK THIS PC.
2762 026022 000240          NOP          ; IF IT DOESN'T, JUST FALL THRU ANYWAY.
2763
2764 026024 106700      20:  MFPS        RO          ; GET NEW PSM...
2765 026026 010167 152002  MOV        R1,TRPVEC ; ...AND RESTORE VECTOR.
2766 026032 005700          TST        RO
2767 026034 001402          BEQ        30          ; BR IF NEW PSM IS RIGHT.
2768 026036 104423 025764  ERROR,    T101        ;; *** PSM INCORRECT AFTER TRAP TO 34 ***
2769 026042 020627 003772  30:  CMP        SP,#STACK-4
2770 026046 001402          BEQ        40          ; BR IF STACK IS RIGHT.
2771 026050 104423 025764  ERROR,    T101        ;; *** STACK POINTER INCORRECT ***
2772 026054 021627 026020  40:  CMP        (SP),#10
2773 026060 001402          BEQ        50          ; BR IF STACKED PC IS RIGHT.
2774 026062 104423 025764  ERROR,    T101        ;; *** INCORRECT PC SAVED ON STACK ***
2775 026066 026627 000002 000317 50:  CMP        2(SP),#PRI6!17
2776 026074 001402          BEQ        60          ; BR IF STACKED (OLD) PSM IS RIGHT.
2777 026076 104423 025764  ERROR,    T101        ;; *** INCORRECT PSM SAVED ON STACK ***
2778 026102
2779
2780

```

```

;*****
.SBTTL      T102 -- TEST THAT THE "T" BIT CAUSES A TRACE TRAP
;*****

```

```

026102 104420 000102  T102:  BEGINTEST, 102      ;; CHECK SWITCH REGISTER OPTIONS.

```

T102 -- TEST THAT THE "T" BIT CAUSES A TRACE TRAP

```

2781
2782
2783
2784 026106 012706 003776
2785 026112 012767 026150 151674
2786 026120 005067 151672
2787 026124 012746 000020
2788 026130 012746 026136
2789 026134 000002
2790 026136 000240
2791 026140 000240
2792 026142 104423 026102
2793 026146 000417
2794
2795 026150 106700
2796 026152 001402
2797 026154 104423 026102
2798 026160 021627 026136
2799 026164 001402
2800 026166 104423 026102
2801 026172 026627 000002 000020
2802 026200 001402
2803 026202 104423 026102
2804
2805
2806
2807 026206 012767 026240 151600
2808 026214 012746 000020
2809 026220 012746 026226
2810 026224 000006
2811 026226 000240
2812 026230 000240
2813 026232 104423 026102
2814 026236 000417
2815
2816 026240 106700
2817 026242 001402
2818 026244 104423 026102
2819 026250 021627 026230
2820 026254 001402
2821 026256 104423 026102
2822 026262 026627 000002 000020
2823 026270 001402
2824 026272 104423 026102
2825 026276
2826
2827
; FIRST, SET "T" VIA RTI -- SHOULD TRAP TO 14 BEFORE NEXT INSTRUCTION.
;
MOV #STACK,SP
MOV #30,TRTVEC ; SET TRAP PC...
CLR TRTVEC+2 ; ...AND NEW PSM.
MOV #20,-(SP) ; PUSH OLD PSM WITH T BIT...
MOV #.6,-(SP) ; ...AND PC.
RTI ; SHOULD SET T BIT...
; ...AND TRAP BEFORE THIS EXECUTES.
10: NOP
20: NOP
;
; ** TRACE BIT DID NOT TRAP **
;
30: MFPS R0 ; TRAPPED OK, T SHOULD BE CLEAR NOW.
BEQ 40 ; BR IF SO.
; ** NEW PSM INCORRECT AFTER TRACE TRAP TO 14 **
; IT SHOULD HAVE TRAPPED BEFORE...
40: CMP (SP),#10 ; ...THE INSTRUCTION AT 10.
BEQ 50 ; ** TRACE TRAP HAPPENED AT WRONG TIME **
; AND T BIT SHOULD HAVE BEEN STACKED...
; ...ALONG WITH THE OLD PSM.
50: CMP 2(SP),#20 ; ** T BIT WASN'T SAVED IN STACKED PSM **
BEQ 60
ERROR, T102
BR 60
;
; NOW SET "T" VIA RTT -- SHOULD ALLOW NEXT INSTRUCTION, THEN TRAP.
;
60: MOV #80,TRTVEC ; CHANGE VECTOR.
MOV #20,-(SP) ; PUSH OLD PSM WITH T BIT...
MOV #.6,-(SP) ; ...AND PC.
RTT ; SHOULD SET T BIT...
NOP ; ...EXECUTE THIS...
70: NOP ; ...AND TRAP BEFORE THIS.
ERROR, T102 ; ** TRACE TRAP DIDN'T HAPPEN **
BR 110
;
80: MFPS R0 ; OK, T SHOULD BE CLEAR NOW.
BEQ 90
90: CMP (SP),#70 ; ** NEW PSM INCORRECT AFTER TRACE TRAP TO 14 **
; AND IT SHOULD HAVE ALLOWED...
; ...THE 1ST NOP TO EXECUTE.
100: ERROR, T102 ; ** RTT DIDN'T ALLOW NEXT INSTRUCTION TO EXECUTE **
; AND THE T BIT SHOULD BE ON STACK.
CMP 2(SP),#20
BEQ 110
110: ERROR, T102 ; ** T BIT WASN'T SAVED IN STACKED PSM **
;
;*****
;SBTTL T103 -- TEST THAT TRACE TRAP ON A TRAP IS INHIBITED
;*****
T103: BEGINTEST, 103 ; CHECK SWITCH REGISTER OPTIONS.
MOV #STACK,SP
MOV #10,TRTVEC ; SET TRACE VECTOR...
CLR TRTVEC+2 ; ...AND PSM.
MOV #20,IOTVEC ; SET IOT VECTOR...
CLR IOTVEC+2 ; ...AND PSM.
MOV #20,-(SP) ; PUSH PSM WITH T BIT...
MOV #.6,-(SP) ; ...AND PC.

```

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

```

2835 026342 000006          RTT          ; SET T BIT...
2836 026344 000004          IOT          ; ...AND XCT ANOTHER TRAP...
2837 026346 000240          NOP          ; THE IOT SHOULD WIN...
2838 026350 000240          NOP          ; ...AND INHIBIT THE TRACE.
2839 026352 104423 026276  ERROR, T103  ; ; *** NEITHER TRAP OCCURRED ??? ***
2840 026356 000404          BR          20
2841
2842 026360 000240          10: NOP          ; IOT SHOULD HAVE CLEARED T...
2843 026362 000240          NOP          ; ...AND INHIBITED THE TRACE.
2844 026364 104423 026276  ERROR, T103  ; ; *** TRACE TRAP ON TRAP WASN'T INHIBITED ***
2845
2846 026370 012767 000022 151422 20: MOV      @IOTVEC+2,IOTVEC ; RESET IOT VECTOR.
2847
2848

```

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

```

2849 026376 104420 000104          T104: BEGINTEST, 104 ; CHECK SWITCH REGISTER OPTIONS.
2850 026402 012706 003776          MOV      @STACK,SP ; SET STACK
2851 026406 012767 026446 151400  MOV      @10,TRTVEC ; SET TRACE VECTOR.
2852 026414 005067 151376          CLR      TRTVEC+2
2853 026420 012746 000020          MOV      @20,-(SP) ; PUSH PSM WITH T BIT...
2854 026424 012746 026432          MOV      @+.6,-(SP) ; ...AND PC.
2855 026430 000006          RTT          ; SET "T" BIT...
2856 026432 000005          RESET        ; ...AND XCT BUS RESET.
2857 026434 000240          NOP          ; SHOULD HAVE NO EFFECT...
2858 026436 000240          NOP          ; ... I.E. TRAP SHOULD STILL OCCUR...
2859 026440 000240          NOP          ; ...AFTER THE RESET INSTRUCTION.
2860 026442 104423 026376  ERROR, T104  ; ; *** BUS-RESET DISABLED OR INHIBITED TRACE TRAP ***
2861 026446 106427 000000          10: MTPS     @PRIO ; WE'RE ALL DONE.
2862 026452 012767 000016 151334  MOV      @TRTVEC+2,TRTVEC ; RESET TRACE VECTOR.
2863
2864

```

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

```

2865 026460 104420 000105          T105: BEGINTEST, 105 ; CHECK SWITCH REGISTER OPTIONS.
2866 026464 012706 003776          MOV      @STACK,SP ; SET STACK.
2867 026470 012767 026514 151306  MOV      @20,ERRVEC ; IN CASE ODD ADDRESS TRAPS.
2868 026476 000167 000007          JMP      10+1
2869 026502 000240          NOP          ; IT DIDN'T DO ANYTHING AT ALL !!
2870 026504 104423 026460          ERROR, T105  ; ; *** JMP TO AN ODD ADDRESS DID NOTHING ***
2871 026510 012707 026520          10: MOV      @30,PC ; ADJUST ODD PC AND EXIT.
2872 026514 104423 026460          20: ERROR, T105  ; ; *** JMP TO AN ODD ADDRESS TIMED OUT ***
2873 026520 012767 000006 151256  30: MOV      @ERRVEC+2,ERRVEC ; RESET THE TRAP CATCHER.
2874

```

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

```

2875 026526 104420 000106          T106: BEGINTEST, 106 ; CHECK SWITCH REGISTER OPTIONS.
2876 026532 012706 003776          MOV      @STACK,SP ; SET STACK.
2877 026536 012767 026602 151244  MOV      @20,RESVEC ; SET TRAP CATCHER.
2878 026544 005000          CLR      R0
2879 026546 000257          CCC          ; CLEAR CC BITS.
2880 026550 000007          MFPT         ; MOVE FROM PROC TYPE (TO R0).
2881 026552 106701          MFPS        R1
2882 026554 032701 000017          BIT      @17,R1

```

T106 -- TEST THE CPU TYPE INSTRUCTION (MFPT)

```

2882 026560 001402          BEQ      10
2883 026562 104423 026526  ERROR,   T106          ;; *** CC BITS CHANGED ON MFPT INSTRUCTION ***
2884 026566 020027 000004    10:     CMP      RO,#KXT11
2885 026572 001405          BEQ      30
2886 026574 104423 026526  ERROR,   T106          ;; *** RETURNED VALUE INCORRECT ON MFPT ***
2887 026600 000402          BR       30
2888 026602    20:
2889 026602 104423 026526  ERROR,   T106          ;; *** MFPT TRAPPED TO 10 ***
2890 026606 012767 000012 151174  30:     MOV      @RESVEC+2,RESVEC
2891

```

```

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

```

```

2892 026614 104420 000107  T107:   BEGINTEST, 107      ;; CHECK SWITCH REGISTER OPTIONS.
2893 026620 012706 003776    MOV      @STACK,SP
2894 026624 004767 006560    CALL    QBXIN        ; Q-BUS EXERCISER AVAILABLE ??
2895 026632 010004          BEQ      40          ; IF NOT, BYPASS TEST.
2896          MOV      RO,R4          ; YES, CSR1 POINTER => R4 AND FALL THRU.
2897          ;
2898          ; USE QBX TO FAKE A Q-BUS INTERRUPT (NO DMA).
2899          ;
2900          MOV      @QBXVEC,R3    ; GET VECTOR POINTER.
2901          CLR      (R4)          ; CLEAR CSR1 (NO DMA REQUIRED).
2902          MOV      @10,(R3)      ; SET Q-BUS VECTOR...
2903          MOV      @PRI4,2(R3)   ; ...AND PRIORITY.
2904          MTPS    @PRI4        ; RAISE CPU TO Q-BUS LEVEL.
2905          MOV      @1,2(R4)     ; SET QBX GO (CLEAR DONE)...
2906          BIS      @20,2(R4)    ; ...ENABLE INT 4 ON DONE...
2907          BIC      @1,2(R4)    ; ...AND CLEAR GO (SET DONE).
2908          BR       20          ; INTERRUPT SHOULD BE HELD OFF.
2909          10:
2910          ERROR,   T107          ;; *** Q-BUS INTERRUPTS AT WRONG LEVEL ***
2911          BR       30
2912          MOV      @30,(R3)     ; CHANGE THE VECTOR.
2913          MTPS    @PRIO        ; LOWER CPU...
2914          BR       240         ; ...INTERRUPT SHOULD COME IN.
2915          ERROR,   T107          ;; *** Q-BUS INTERRUPT NOT RECEIVED ***
2916          CLR      (R4)          ; THAT'S ALL THERE IS TO IT.
2917          MOV      @QBXVEC+2,(R3); RESET Q-BUS VECTOR.
2918          CLR      (R3)
2919          40:
2920

```

```

;*****
.SBTTL      T110 -- TEST THE "BEVNT" INTERRUPT (REQUIRES Q-BUS EXERCISER)
;*****

```

```

2921 026742 104420 000110  T110:   BEGINTEST, 110      ;; CHECK SWITCH REGISTER OPTIONS.
2922 026746 012706 003776    MOV      @STACK,SP
2923 026752 004767 006432    CALL    QBXIN        ; SET STACK POINTER.
2924 026760 010004          BEQ      40          ; Q-BUS EXERCISER AVAILABLE ??
2925          MOV      RO,R4          ; IF NOT, BYPASS TEST.
2926          ; YES, CSR1 POINTER => R4 AND FALL THRU.
2927          ;
2928          ; QBX CSR1<11> EMULATES A "BEVNT" INTERRUPT.
2929          ;
2930          MTPS    @PRI6        ; SET PRI 06
2931          MOV      @10,BEVNT    ; SET BEVNT VECTOR.
2932          CLR      RO

```

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

```

2931 026776 012714 004000      MOV    #4000,(R4)      ; SET "BEVNT" REQUEST.
2932 027002 077001              SOB    R0,..          ; DELAY ABOUT 250 MSEC, INTERRUPT...
2933 027004 000403              BR     2#             ; ...SHOULD BE MASKED AT THIS LEVEL.
2934 027006              1#:
      027006 104423 026742      ERROR, T110          ;; *** BEVNT INTERRUPT LEVEL INCORRECT ***
2935 027012 000410              BR     3#
2936 027014 012767 027034 151056 2#:  MOV    #3# ,BEVNT      ; OK, CHANGE VECTOR...
2937 027022 106427 000240      MTPS  #PRI5          ; ...AND LOWER CPU TO PRI 5.
2938 027026 077001              SOB    R0,..          ; DELAY AGAIN, INT SHOULD COME IN.
2939 027030 104423 026742      ERROR, T110          ;; *** BEVNT INTERRUPT NOT RECEIVED ***
2940 027034 012767 000102 151036 3#:  MOV    #BEVNT+2,BEVNT ; RESET VECTOR.
2941 027042 106427 000000      MTPS  #PRIO          ; LOWER CPU TO 0.
2942
2943 027046 004767 006004      4#:  CALL  ENDSEG        ; CHECK FOR LOOP-IN-SEGMENT...
2944
2945              .SBTTL  ; ...RETURN AND FALL THRU IF NOT.

```



SECTION 3 -- LOCAL RAM TESTS.

```

2947          .SBTTL SECTION 3 -- LOCAL RAM TESTS.
2948
2949 027052 012767 036373 007246 S3:  MOV    #SEG3,SEGN      ; SEGMENT 3 TEXT.
2950          ; MOV    #RAMADR,TEMP    ; SAVE RAM ADDRESS...
2951          ; MOV    #RAMSIZ,TEMP1   ; ...AND SIZE.
2952 027060 000401          BR      .+4                ; JSD REV 8
2953 027062 000000          FPLUS: .WORD          ; FLAG (0=FALCON, 1=FALCON+) ; JSD REV 8
2954 027064 005067 177772  CLR      FPLUS          ; ASSUME FALCON; CLEAR FLAG ; JSD REV 8
2955 027070 012767 160010 151710  MOV    #RAMAD1,TEMP    ; SET FALCON RAM ADDR ; JSD REV 8
2956 027076 012767 003764 151704  MOV    #RAMSZ1,TEMP1   ; ...AND SIZE. ; JSD REV 8
2957 027104 012767 027200 150672  MOV    #11#,ERRVEC    ; SET TIMEOUT VECTOR ; JSD REV 8
2958 027112 012700 177600  MOV    #177600,R0     ; START OF FALCON+ ODT SCRATCH ; JSD REV 8
2959 027116 012701 000100  MOV    #64.,R1        ; NUMBER OF WORDS ; JSD REV 8
2960 027122 005720          10#:  TST    (R0)+          ; TEST LOCATION (IF PRESENT) ; JSD REV 8
2961 027124 077102          SOB      R1,10#        ; (FALCON WILL TRAP) ; JSD REV 8
2962 027126 012767 000001 177726  MOV    #1,FPLUS       ; ELSE, SET FLAG FOR FALCON+ ; JSD REV 8
2963 027134 012767 100000 151644  MOV    #RAMAD2,TEMP    ; SET FALCON+ RAM ADDR ; JSD REV 8
2964 027142 012767 024000 151640  MOV    #RAMSZ2,TEMP1   ; ...AND SIZE. ; JSD REV 8
2965 027150 005077 151632  CLR      #TEMP        ; CHECK IF SOCKET 8 IS PRESENT ; JSD REV 8
2966 027154 027727 151626 177777  CMP    #TEMP,#-1      ; IF CLEARED LOC IS NOT ALL 1'S ; JSD REV 8
2967 027162 001006          BNE     11#           ; THEN THERE IS RAM SPACE HERE ; JSD REV 8
2968 027164 062767 040000 151614  ADD    #40000,TEMP     ; ELSE SKIP OVER SOCKET 8 ; JSD REV 8
2969 027172 162767 020000 151610  SUB    #20000,TEMP1    ; REDUCE RAM SIZE (# WORDS) ; JSD REV 8
2970 027200          11#:
2971          ;*****
2972          .SBTTL T111 -- LOCAL RAM ADDRESS TEST
                ;*****
027200 104420 000111          T111:  BEGINTEST, 111      ; CHECK SWITCH REGISTER OPTIONS.
                ;
                ; WRITE EACH RAM LOCATION WITH ITS OWN ADDRESS, READ BACK AND VERIFY.
                ; REPEAT A SECOND TIME USING BYTE-SWAPPED ADDRESS AS DATA.
                ;
2973          MOV    #STACK,SP
2974          MOV    #7#,ERRVEC    ; SET TIME-OUT VECTOR.
2975          MOV    TEMP,R0        ; GET RAM ADDRESS...
2976          MOV    TEMP1,R1       ; ...AND SIZE.
2977 027204 012706 003776          MOV    R1,-(SP)        ; SAVE A COPY.
2978 027210 012767 027312 150566  1#:  MOV    R0,(R0)        ; LOAD ADDRESSES (FROM BOTTOM UP).
2979 027216 016700 151564          TST    (R0)+          ; BUMP
2980 027222 016701 151562          SOB      R1,1#       ; LOOP 'TIL DONE.
2981 027226 010146
2982
2983 027230 010010          2#:  MOV    (SP),R2        ; RECOVER SIZE K.
2984 027232 005720          MOV    -(R0),R1      ; READ 'EM BACK (FROM TOP DOWN).
2985 027234 077103          CMP    R1,R0        ; DATA = ADDRESS ??
2986
2987 027236 011602          BEQ    3#           ; BR IF SO.
2988 027240 014001          ERROR, T111       ; *** RAM DATA INCORRECT AT ADDRESS IN R0 ***
2989 027242 020100          3#:  SOB      R2,2#   ; LOOP 'TIL DONE.
2990 027244 001402
2991 027246 104423 027200
2992 027252 077206
2993
2994 027254 011602          4#:  MOV    (SP),R2        ; RECOVER SIZE K.
2995 027256 010001          MOV    R0,R1        ; ADDRESS => R1...
2996 027260 000301          SWAB   R1           ; ...SWAP BYTES...
2997 027262 010120          MOV    R1,(R0)+     ; ...AND LOAD SWAPPED ADDRESSES.
2998 027264 077204          SOB      R2,4#
2999
3000 027266 011603          MOV    (SP),R3        ; RECOVER SIZE K.

```

PC 27246 = RAM DATA INCORRECT AT ADDRESS IN R0.

```

3001 027270 014001          50:  MOV    -(R0),R1      ; GET DATA.
3002 027272 010002          MOV    R0,R2
3003 027274 000302          SWAB   R2
3004 027276 020102          CMP    R1,R2        ; DATA = SWAPPED ADDRESS ??
3005 027300 001402          BEQ   60            ; BR IF SO.
3006 027302 104423 027200   ERROR, T111        ;; *** RAM DATA (SWAPPED) INCORRECT AT ADDRESS IN R0 ***
3007 027306 077310          SOB   R3,50
3008 027310 000402          BR    80            ; DONE, EXIT.
3009
3010 027312          70:
3011 027312 104423 027200   ERROR, T111        ;; *** BUS TIME-OUT -- RAM ADDRESS IN R0 ***
3012
3013
;*****
;SBTTL      T112 -- LOCAL RAM DATA TEST
;*****
027316 104420 000112   T112:  BEGINTEST, 112      ;; CHECK SWITCH REGISTER OPTIONS.
;
; FILL RAM WITH A BACKGROUND PATTERN OF 0'S.
; FLOAT AND VERIFY A 1 BIT THRU EACH WORD, AND WHEN DONE
; RESTORE THAT WORD TO THE BACKGROUND.
; WHEN ALL WORDS DONE, VERIFY THAT THE ENTIRE RAM STILL CONTAINS
; THE CORRECT BACKGROUND DATA.
;
; REPEAT A SECOND TIME WITH BACKGROUND 1, AND A FLOATING 0.
;
3023 027322 012706 003776   MOV    #STACK,SP
3024 027326 012767 027502 150450  MOV    #100,ERRVEC    ; SET TIME-OUT VECTOR.
3025 027334 005046          CLR    -(SP)          ; BACKGROUND = 0'S FIRST...
3026 027336 000402          SKP2
3027 027340 012746 177777   10:  MOV    @-1,-(SP)      ; ...THEN 1'S.
3028 027344 016700 151436   MOV    TEMP,R0        ; SET RAM ADDRESS...
3029 027350 016701 151434   MOV    TEMP1,R1       ; ...AND SIZE.
3030 027354 011620          MOV    (SP),(R0)+     ; FILL RAM WITH BACKGROUND DATA.
3031 027356 077102          SOB   R1,20
3032
3033 027360 016700 151422   MOV    TEMP,R0        ; RESET ADDRESS...
3034 027364 016701 151420   MOV    TEMP1,R1       ; ...AND WORD COUNT...
3035 027370 011004          30:  MOV    (R0),R4        ; ...AND EXERCISE 1ST/NEXT LOCATION.
3036 027372 020416          CMP    R4,(SP)
3037 027374 001402          BEQ   40
3038 027376 104423 027370   ERROR, 30            ; BR IF BACKGROUND IS RIGHT.
3039 027402 012702 000001   40:  MOV    @1,R2          ;; *** INITIAL BACKGROUND INCORRECT AT ADDRESS IN R0 ***
3040 027406 011603          MOV    (SP),R3
3041 027410 074203          XOR   R2,R3
3042 027412 012702 000020   MOV    @16.,R2
3043 027416 000402          SKP2
3044 027420 006303          50:  ASL   R3
3045 027422 005503          ADC   R3
3046 027424 010310          60:  MOV    R3,(R0)
3047 027426 011004          MOV    (R0),R4
3048 027430 020304          CMP   R3,R4
3049 027432 001402          BEQ   70
3050 027434 104423 027424   ERROR, 60
3051 027440 077211          70:  SOB   R2,50
3052 027442 011620          MOV    (SP),(R0)+
3053 027444 077127          SOB   R1,30
; ...AND LOOP FOR ALL WORDS.

```

PC 27434 = FLOATING DATA INCORRECT AT ADDRESS IN R0.

```

3054
3055 027446 016700 151334      MOV     TEMP,R0      ; NOW, RESET POINTERS...
3056 027452 016701 151332      MOV     TEMP1,R1    ; ...AND VERIFY THAT RESTORED...
3057 027456 011004      80:    MOV     (R0),R4    ; ...BACKGROUND IS STILL CORRECT.
3058 027460 020416      CMP     R4,(SP)
3059 027462 001402      BEQ     90
3060 027464 104423 027456      ERROR, 80           ; BR IF SO.
3061 027470 005720      90:    TST     (R0),      ; ... BACKGROUND CHANGED AT ADDRESS IN R0 ...
3062 027472 077107      SOB     R1,80       ; BUMP...
3063
3064 027474 005726      TST     (SP),      ; FINALLY, IF BACKGROUND IS 0...
3065 027476 001720      BEQ     10
3066 027500 000402      BR      110        ; ...LOOP ONCE MORE...
3067
3068 027502      100:   ERROR, T112       ; ...OTHERWISE, WE'RE ALL DONE.
3069 027506 104423 027316      110:
3070
3071
;*****
.SBTTL      T113 -- OPTIONAL DMA TEST (REQUIRES Q-BUS EXERCISER)
;*****
T113:      BEGINTEST, 113      ; CHECK SWITCH REGISTER OPTIONS.
          MOV     @STACK,SP
          CALL   GBXIN        ; Q-BUS EXERCISER AVAILABLE ??
          BEQ    100         ; IF NOT, BYPASS TEST.
          MOV    R0,R4       ; YES, CSR1 POINTER => R4 AND FALL THRU.

;
; TRANSFER EACH RAM WORD TO THE GBX AND BACK.
; VERIFY THAT RETURNED DATA MATCHES THAT WHICH WAS SENT.
;
3080 027526 012767 027730 150250      MOV     @90,ERRVEC   ; SET TRAP CATCHER.
3081 027534 016701 151246      MOV     TEMP,R1     ; RAM ADDRESS POINTER...
3082 027540 016702 151244      MOV     TEMP1,R2    ; ...AND WORD COUNT.
3083
3084 027544 010111      20:    MOV     R1,(R1)     ; SET 1ST/NEXT RAM WORD.
3085 027546 012714 140407      MOV     @140407,(R4) ; DATI (MEM => GBX) IN "HOG" MODE.
3086 027552 010164 000004      MOV     R1,4(R4)    ; FROM RAM (R1) TO DATA REGISTER...
3087 027556 012764 177774 000006      MOV     @-4,6(R4)   ; ...4 TIMES.
3088 027564 012764 000401 000002      MOV     @401,2(R4)  ; GO, INHIBIT BA..
3089 027572 005000      CLR     R0
3090 027574 105764 000002      30:    TSTB   2(R4)
3091 027600 100401      BMI    330
3092 027602 077004      SOB    R0,30       ; PROCEED WHEN DONE...
3093 027604 032764 040000 000002      330:   BIT    @40000,2(R4) ; ...AND CHECK FOR NON ERROR.
3094 027612 001402      BEQ    40
3095 027614 104423 027544      ERROR, 20         ; BR IF OK.
3096 027620 026401 000010      40:    CMP    10(R4),R1  ; ... NON ERROR IN GBX ON DATI (MEM TO GBX) ...
3097 027624 001402      BEQ    50         ; DATA REGISTER CORRECT ??
3098 027626 104423 027544      ERROR, 20       ; ... DATA INCORRECT IN GBX ON DATI (MEM TO GBX) ...
3099
3100 027632 012714 000601      50:    MOV     @000601,(R4) ; DATO (GBX => MEM) IN "HOG" MODE.
3101 027636 012764 001012 000004      MOV     @TEMP2,4(R4) ; FROM DATA REGISTER TO TEMP2...
3102 027644 012764 177774 000006      MOV     @-4,6(R4)   ; ...4 TIMES.
3103 027652 012764 000401 000002      MOV     @401,2(R4)  ; GO, INHIBIT BA..
3104 027660 005000      CLR     R0
3105 027662 105764 000002      60:    TSTB   2(R4)
3106 027666 100401      BMI    660        ; PROCEED WHEN DONE...

```

PC 27626 = DATA INCORRECT IN GBX ON DATI (MEM TO GBX).

```

3107 027670 077004          SOB      R0,60
3108 027672 032764 040000 000002 660:  BIT      @40000,2(R4) ;...AND CHECK FOR NCM.
3109 027700 001402          BEQ      70      ; BR IF OK.
3110 027702 104423 027632  ERROR.  50      ;: *** NCM ERROR IN GBX ON DATO (GBX TO MEM) ***
3111 027706 026701 151100 70:      CMP      TEMP2,R1 ; RETURNED DATA CORRECT ??
3112 027712 001402          BEQ      80
3113 027714 104423 027632  ERROR.  50      ;: *** DATA INCORRECT IN TEMP2 ON DATO (GBX TO MEM) ***
3114
3115 027720 005721          80:      TST      (R1),
3116 027722 077270          SOB      R2,20      ; SET NEXT ADDRESS...
3117 027724 005014          CLR      (R4)      ;...AND LOOP 'TIL DONE.
3118 027726 000402          BR       100      ; CLEAR GBX...
3119
3120 027730          90:
3120 027730 104423 027506  ERROR.  T113 ;: *** BUS TIME-OUT -- GBX (R4) OR RAM (R1) ***
3121
3122 027734 012767 000006 150042 100:  MOV      @ERRVEC+2,ERRVEC ; RESET ERROR VECTOR.
3123 027742 004767 005110  CALL     ENDSEG    ; CHECK FOR LOOP-IN-SEGMENT...
3124
3125          .SBTTL ;...RETURN AND FALL THRU IF NOT.

```

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

```

3127          .SBTTL SECTION 4 -- LOCAL PROM/RAM TESTS (EMPTY SOCKETS).
3128
3129 027746 012767 036405 006352 S4:  MOV  #SEG4,SEGN      ; SEGMENT 4 TEXT.
3130          ;          #ROMADR,TEMP    ; SAVE ROM ADDRESS...
3131          ;          #ROMSZ,TEMP1   ; ...AND SIZE.           ;JSD REV 8
3132 027754 012767 170000 151024          ;          #ROMAD1,TEMP    ; ASSUME FALCON; SAVE ROM ADDR ;JSD REV 8
3133 027762 012767 002000 151020          ;          #ROMSZ1,TEMP1   ; ...AND SIZE           ;JSD REV 8
3134 027770 005767 177066          TST  FPLUS        ; IS THIS FALCON?       ;JSD REV 8
3135 027774 001406          BEQ   S4           ; NO - FALCON - READY TO START ;JSD REV 8
3136 027776 012767 164000 151002          MOV  #ROMAD2,TEMP    ; SAVE FALCON ROM ADDRESS ;JSD REV 8
3137 030004 012767 004000 150776          MOV  #ROMSZ2,TEMP1   ; ...AND SIZE           ;JSD REV 8
3138 030012          S0:
3139
3140          ;*****
          .SBTTL      T114 -- LOCAL PROM/RAM ADDRESS TEST
          ;*****
          T114:  BEGINTEST, 114          ;; CHECK SWITCH REGISTER OPTIONS.
          ;
          ; PROM/RAM SOCKETS CONTAIN THE 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.
          ;
3141          MOV  #STACK,SP
3142          MOV  #S1,ERRVEC          ; SET TIME-OUT VECTOR.
3143          MOV  TEMP,R0            ; SET ADDRESS...
3144          MOV  TEMP1,R1          ; ...AND SIZE.
3145
3146          10:  MOV  #123456,R2
3147 030016 012706 005776          ; TRY TO WRITE DATA (SHOULDN'T TRAP).
3148 030022 012767 030074 147754          MOV  R2,(R0)          ; DID IT WRITE ??
3149 030030 016700 150752          CMP  (R0),R2         ; BR IF NOT.
3150 030034 016701 150750          BNE  20             ; MAYBE...
3151          COM  R2
3152 030040 012702 123456          ; ...TRY COMPLIMENT.
3153 030044 010210          MOV  R2,(R0)
3154 030046 021002          CMP  (R0),R2
3155 030050 001006          ; HOW NOW ??
3156 030052 005102          BNE  20             ; BR IF OK.
3157 030054 010210          ; ** WRITE TO ROM ALTERED DATA ????? **
3158 030056 021002          ERROR, 10
3159 030060 001002
3160 030062 104423 030040
3161
3162 030066 005720          20:  TST  (R0)
3163 030070 077115          SOB  R1,10
3164 030072 000402          BR   40
3165
3166 030074          30:  ERROR, T114          ;; ** BUS TIME-OUT -- ROM ADDRESS IN R0 **
          030074 104423 030012
3167
3168 030100 012767 000006 147676 40:  MOV  #ERRVEC+2,ERRVEC
3169 030106 004767 004744          CALL ENDSEG
3170          ; CHECK FOR LOOP-IN-SEGMENT...
3171          ; ...RETURN AND FALL THRU IF NOT.
          .SBTTL

```

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

```

3173 .SBTTL SECTIONS 5 AND 6 -- SERIAL LINES 1 AND 2 TESTS.
3174 ;
3175 ; DLART BIT ASSIGNMENTS.
3176 ;
3177 XMTBRK= 001
3178 XMTIE= 100
3179 XMTROY= 200
3180
3181 RCVIE= 000100
3182 RCVDUN= 000200
3183 RCVACT= 004000
3184 RCVBRK= 004000
3185 FRERR= 020000
3186 ORERR= 040000
3187 RCVERR= 100000
3188 ;
3189 ; THE FOLLOWING DLART TESTS ARE RUN TWICE PER PASS.
3190 ; FIRST, ON SLU1 (THE CONSOLE SLOT).
3191 ;
3192 030112 012767 036417 006206 S5: MOV #SEG5,SEGN ; SEGMENT 5 TEXT.
3193 030120 012700 177560 MOV #SLU1,R0 ; 1ST UNIT CSR...
3194 030124 012701 000060 MOV #SLU1V,R1 ; ...AND VECTOR(S).
3195 030130 012702 000200 MOV #SLU1P,R2 ; SLU1 INTERRUPTS AT LEVEL 4.
3196 030134 012703 030300 MOV #BADU1,R3 ; TIME-OUT VECTOR.
3197 030140 000413 BR DLSET ; SET-UP POINTERS AND XCT.
3198 ;
3199 ; AND THEN ON SLU2.
3200 ;
3201 030142 012767 036435 006156 S6: MOV #SEG6,SEGN ; SEGMENT 6 TEXT.
3202 030150 012700 176540 MOV #SLU2,R0 ; 2ND UNIT CSR...
3203 030154 012701 000120 MOV #SLU2V,R1 ; ...AND VECTOR.
3204 030160 012702 000240 MOV #SLU2P,R2 ; SLU2 INTERRUPTS AT LEVEL 5.
3205 030164 012703 030310 MOV #BADU2,R3 ; TIME-OUT VECTOR.
3206
3207 030170 DLSET:
3208 ;
;*****
;***** T115 -- TEST SERIAL LINE REGISTER ADDRESSES *****
;*****
T115: BEGINTEST, 115 ; CHECK SWITCH REGISTER OPTIONS.
MOV #STACK,SP ; INIT THE STACK.
MOV R3,ERRVEC ; SET BUS TIME-OUT TRAP CATCHER.
MOV R0,RCSR ; LOAD UP THE REGISTER POINTERS.
TST (R0) ; IF ANY OF THESE TRAP, WE'RE DEAD !!
MOV R0,RBUF
TST (R0)
MOV R0,XCSR
TST (R0)
MOV R0,XBUF
MOV R1,RVEC ; SAVE RCVR VECTOR POINTER...
MOV #BADRI,(R1) ; ...AND INIT VECTOR.
MOV R2,(R1)
MOV R1,XVEC ; SAVE XMTR VECTOR POINTER...
MOV #BADTI,(R1) ; ...AND INT VECTOR.
MOV R2,(R1)
MOV R0,EXCSR,R0 ; NOW GET THE DEFAULT XMIT STATUS...
BIC #C72,R0 ; ...AND STRIP THE BAUD RATE BITS.
BIS #2,R0 ; INSURE "PBRE" IS SET...

```

T115 -- TEST SERIAL LINE REGISTER ADDRESSES

```

3227 030272 010067 150542      MOV    R0,DFPBR      ;...AND SAVE AS THE DEFAULT BAUD RATE.
3228 030276 000432      BR     DLTSTS        ; START 'EM UP.
3229
3230      ; COME HERE IF BUS TIME-OUT OR UNEXPECTED INTERRUPT.
3231
3232 030300      BADU1:
3233 030300 104423 030112      ERROR, S5           ;: *** BUS TIME-OUT ON SLU1 ADDRESS ***
3234 030304 000167 003270      JMP   DLDUN
3235 030310 104423 030142      BADU2: ERROR, S6           ;: *** BUS TIME-OUT ON SLU2 ADDRESS ***
3236 030314 000167 003260      JMP   DLDUN
3237 030320 104423 030364      BADRI: ERROR, DLTSTS      ;: *** UNEXPECTED RCVR INTERRUPT. ***
3238 030324 000167 003250      JMP   DLDUN
3239 030330 104423 030364      BADTI: ERROR, DLTSTS      ;: *** UNEXPECTED XMITR INTERRUPT. ***
3240 030334 000167 003240      JMP   DLDUN
3241
3242      ; FLOATING 1 AND 0 DATA TABLE, USED IN VARIOUS TESTS.
3243 030340      SYNC: .BYTE 015, 012      ; SYNC PAIR.
3244 030342 001 002 004 010 020 040 100 200      FLT1: .BYTE 001, 002, 004, 010, 020, 040, 100, 200
3245 030345 010 020 040
3246 030350 100 200
3247 030352 376 375 373 367 357 337 277 177      FLT0: .BYTE 376, 375, 373, 367, 357, 337, 277, 177
3248 030355 367 357 337
3249 030360 277 177
3250 030362 000000      .WORD 0 ; TERMINATOR.
3251
3252      DLTSTS:
3253      ;*****
3254      .SBTTL T116 -- TEST THAT BUS-RESET CLEARS THE RIGHT BITS
3255      ;*****
3256      T116: BEGINTEST, 116 ;: CHECK SWITCH REGISTER OPTIONS.
3257
3258      ; SET ALL WRITABLE BITS AND RESET. CHECK THAT ONLY
3259      ; RCVR-IE, XMIT-IE, MAINT, AND XMIT-BRK GET CLEARED.
3260
3261      MTPS #PR16 ; RAISE CPU
3262      MOV #177, BXCSR ; SET WRITABLE BITS IN XCSR...
3263      MOV #377, XBDF ;...AND XBDF...
3264      MOV #100, BRCSR ;...AND RCSR.
3265      MOV 240, 240 ; IN CASE WE NEED A DELAY.
3266      RESET 240, 240
3267      MOV BRCSR, R0 ; GET RCSR...
3268      MOV XBDF, R1 ;...XBDF (DUPPY READ)...
3269      MOV BXCSR, R2 ;...XCSR...
3270      MOV XBDF, R3 ;...AND XBDF.
3271      MOV DFPBR, BXCSR ; RESTORE DEFAULT XMIT...
3272      CLR BRCSR ;...AND RCVR STATUS.
3273      BIT #100, R0 ; NOW, DID RCVR-IE CLEAR ??
3274      BEQ 11 ;
3275      ERROR, T116 ;: *** BUS-RESET DIDN'T CLEAR RCVR-IE BIT IN RCSR ***
3276      BIT #105, R2 ; DID XIE, MAINT AND XBRK CLEAR ??
3277      BEQ 21 ;
3278      ERROR, T116 ;: *** BUS-RESET DIDN'T CLEAR XIE, MAINT, AND/OR XBRK ***

```

PC 30502 = BUS-RESET DIDN'T CLEAR XIE, MAINT, AND/OR XBRK.

```

3273 030506 120227 000272      20:  CMPB   R2,0272      ; ALL OTHERS SET (INCLUDING XMTRDY) ??
3274 030512 001402              BEQ     30
3275 030514 104423 030364      ERROR,  T116          ;; *** BUS-RESET CLEARED WRONG BITS IN XCSR ***
3276 030520 120327 177777      30:  CMPB   R3,0-1       ; ALL BITS STILL IN XBUF ??
3277 030524 001402              BEQ     40
3278 030526 104423 030364      ERROR,  T116          ;; *** BUS-RESET CLEARED BITS IN XBUF ***
3279 030532 106427 000000      40:  HTPS   @PRI0        ; LOWER THE CPU.
3280
3281
;*****
.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    10
      ERROR, T117          ;; *** UNDEFINED BITS SET IN RCVR STATUS ***
      10:  BIT    @13400,R1    ; RBUF, UNDEFINED BITS SET ??
      BEQ    20
      ERROR, T117          ;; *** UNDEFINED BITS SET IN RCVR DATA BUFFER ***
      20:  BIT    @177400,R2   ; XCSR, UNDEFINED BITS SET ??
      BEQ    30
      ERROR, T117          ;; *** UNDEFINED BITS SET IN XMTR STATUS ***
      30:  BIT    @177400,R3   ; XBUF, UNDEFINED BITS SET ??
      BEQ    40
      ERROR, T117          ;; *** UNDEFINED BITS SET IN XMTR DATA BUFFER ***
      40:  TSTB  @XCSR        ; XMTR READY ??
      BMI    50
      ERROR, T117          ;; *** XMITTER NOT READY ***
      ; .IIF P2, .PRINT .    ; *** DEBUG, NOP ***
      50:  SKP2
      JMP    DL0UN
.SBTTL

```

;JSD REV B



TRANSMITTERS

```

3306
3307 030652
3308
    030652 104420 000120
3309 030656 012700 030342
3310 030662 112001
3311 030664 010177 150162
3312 030670 017702 150156
3313 030674 120102
3314 030676 001402
3315 030700 104423 030664
3316 030704 112001
3317 030706 001366
3318
3319
    030710 104420 000121
3320 030714 106427 000340
3321 030720 052777 000001 150122
3322 030726 017700 150116
3323 030732 042777 000001 150110
3324 030740 017701 150104
3325 030744 016777 150070 150076
3326 030752 106427 000000
3327 030756 032700 000001
3328 030762 001002
3329 030764 104423 030710
3330 030770 032701 000001
3331 030774 001402
3332 030776 104423 030710
3333 031002
3334
3335
    031002 104420 000122
3336 031006 052777 000004 150034
3337 031014 017700 150030
3338 031020 042777 000004 150022
3339 031026 017701 150016
3340 031032 016777 150002 150010
3341 031040 032700 000004
3342 031044 001002
3343 031046 104423 031002
3344 031052 032701 000004
3345 031056 001402
3346 031060 104423 031002
3347 031064
3348
3349
    031064 104420 000123
3350 031070 052777 000002 147752
    
```

```

.SBTTL TRANSMITTERS
XITRS: ; 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.
10:   MOV R1,8XBUF ;LOAD DATA INTO "XMIT DATA BUF"...
      MOV 8XBUF,R2 ;...AND READ IT BACK.
      CPB R1,R2
      BEQ 20
      ERROR, 10 ; ** FLOATING 1 OR 0 INCORRECT IN XBUF **
20:   MOVB (R0)+,R1 ; GET NEXT CHAR...
      BNE 10 ;...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 #1,8XCSR ;SET "XMIT-BRK" BIT...
      MOV 8XCSR,R0 ;...AND SAVE IN R0.
      BIC #1,8XCSR ; CLEAR IT...
      MOV 8XCSR,R1 ;...AND SAVE IN R1.
      MOV DFPBR,8XCSR ; RESTORE DEFAULT XMITR.
      MTPS @PRIO ; AND PRIORITY.
      BIT #1,R0
      BNE 10 ;BR IF IT SET
      ERROR, T121 ; ** XMIT-BRK BIT FAILED TO SET **
10:   BIT #1,R1
      BEQ 20 ;BR IF IT CLEARED
      ERROR, T121 ; ** XMIT-BRK BIT FAILED TO CLEAR **
20:
;*****
.SBTTL T122 -- TEST THAT "XMIT-MAINT" CAN BE SET AND CLEARED
;*****
T122: BEGINTEST, 122 ; CHECK SWITCH REGISTER OPTIONS.
      BIS #4,8XCSR ;SET "XMIT-MAINT" BIT...
      MOV 8XCSR,R0 ;...AND SAVE IN R0.
      BIC #4,8XCSR ; CLEAR IT...
      MOV 8XCSR,R1 ;...AND SAVE IN R1.
      MOV DFPBR,8XCSR
      BIT #4,R0
      BNE 10 ;BR IF IT SET
      ERROR, T122 ; ** XMIT-MAINT BIT FAILED TO SET **
10:   BIT #4,R1
      BEQ 20 ;BR IF IT CLEARED
      ERROR, T122 ; ** XMIT-MAINT BIT FAILED TO CLEAR **
20:
;*****
.SBTTL T123 -- TEST THAT "XMIT-PBRE" CAN BE SET AND CLEARED
;*****
T123: BEGINTEST, 123 ; CHECK SWITCH REGISTER OPTIONS.
      BIS #2,8XCSR ;SET "XMIT-PBRE" BIT...
    
```

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

```

3351 031076 017700 147746          MOV    BXCSR,RO      ;...AND SAVE IN RO.
3352 031102 042777 000002 147740  BIC    #2,BXCSR     ; CLEAR IT...
3353 031110 017701 147734          MOV    BXCSR,R1     ;...AND SAVE IN R1.
3354 031114 016777 147720 147726  MOV    DFPBR,BXCSR
3355 031122 032700 000002          BIT    #2,RO
3356 031126 001002                    BNE    1#           ;BR IF IT SET
3357 031130 104423 031064          ERROR, T123       ;: *** XMIT-PBRE BIT FAILED TO SET ***
3358 031134 032701 000002 1#:    BIT    #2,R1
3359 031140 001402                    BEQ    2#           ;BR IF IT CLEARED
3360 031142 104423 031064          ERROR, T123       ;: *** XMIT-PBRE BIT FAILED TO CLEAR ***
3361 031146
3362
3363

```

```

;*****
.SBTTL      T124 -- TEST THAT "XMIT-PBR0" CAN BE SET AND CLEARED
;*****

```

```

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

```

```

;*****
.SBTTL      T125 -- TEST THAT "XMIT-PBR1" CAN BE SET AND CLEARED
;*****

```

```

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

```

```

;*****
.SBTTL      T126 -- TEST THAT "XMIT-PBR2" CAN BE SET AND CLEARED
;*****

```

```

T126:  BEGINTEST, 126      ;: CHECK SWITCH REGISTER OPTIONS.
        BIS    #40,BXCSR  ;SET "XMIT-PBR2" BIT...
        MOV    BXCSR,RO   ;...AND SAVE IN RO.
        BIC    #40,BXCSR  ; CLEAR IT...
        MOV    BXCSR,R1   ;...AND SAVE IN R1.
        MOV    DFPBR,BXCSR
        BIT    #40,RO
        BNE    1#         ;BR IF IT SET

```

PC 31356 = XMIT-PBR2 BIT FAILED TO SET.

```

3399 031356 104423 031312          ERROR, T126          ;; *** XMIT-PBR2 BIT FAILED TO SET ***
3400 031362 032701 000040      10:  BIT    #40,R1          ;BR IF IT CLEARED
3401 031366 001402              BEQ    20              ;; *** XMIT-PBR2 BIT FAILED TO CLEAR ***
3402 031370 104423 031312          ERROR, T126
3403 031374
3404
3405
;*****
.SBTTL      T127 -- TEST THAT "XMIT-IE" CAN BE SET AND CLEARED
;*****
T127:  BEGINTEST, 127          ;; CHECK SWITCH REGISTER OPTIONS.
      MTPS    @PRIO          ;RAISE CPU PRIORITY
      BIS    #100,EXCSR      ;SET "XMIT-IE" BIT...
      MOV    EXCSR,R0        ;...AND SAVE IN R0.
      BIC    #100,EXCSR      ; CLEAR IT...
      MOV    EXCSR,R1        ;...AND SAVE IN R1.
      MOV    DFPBR,EXCSR     ; RESTORE XMITR...
      MTPS    @PRIO          ;...AND CPU TOO.
      BIT    #100,R0
      BNE    10
      ERROR, T127          ;BR IF IT SET
      10:  BIT    #100,R1          ;; *** XMIT-IE BIT FAILED TO SET ***
      BEQ    20              ; BR IF IT CLEARED.
      20:  ERROR, T127          ;; *** XMIT-IE BIT FAILED TO CLEAR ***
;*****
.SBTTL      T130 -- TEST THAT "XMIT-RDY" CAN BE SET AND CLEARED
;*****
T130:  BEGINTEST, 130          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV    DFPBR,EXCSR
      CLR    R1              ; DELAY COUNT.
      CLR    R2              ; DITTO
      10:  TSTB  EXCSR          ;WAIT FOR "XMIT-RDY" TO SET
      BMI    20
      SOB    R1,10          ;KEEP TRYING.
      ERROR, T130          ;; *** XMIT-RDY FAILED TO SET ***
      20:  MOV    #0,EXBUF      ;TRANSMIT A NULL CHARACTER
      MOV    EXCSR,R0        ;READ XMIT STATUS REG
      BIT    @XMITRDY,R0     ; READY SHOULD BE GONE.
      BEQ    30              ; BR IF SO.
      ERROR, T130          ;; *** XMIT-RDY BIT FAILED TO CLEAR ***
      30:  TSTB  EXCSR          ;WAIT FOR "XMIT-RDY" TO SET
      BMI    40
      SOB    R2,30          ; KEEP TRYING.
      40:  ERROR, T130          ;; *** XMIT-RDY FAILED TO SET AFTER XMITTING CHARACTER ***
;*****
.SBTTL      T131 -- TEST THAT WE CAN INTERRUPT ON "XMIT-RDY"
;*****
T131:  BEGINTEST, 131          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV    DFPBR,EXCSR
      MOV    XVEC,R1         ; GET VECTOR POINTER.
      MTPS    2(R1)          ; RAISE CPU TO XMITR LEVEL.
      MOV    #20,(R1)        ; SET XMITTER VECTOR.
      10:  TSTB  EXCSR          ;WAIT FOR "XMIT-RDY"
      SPL    10

```

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

```

3447 031614 052777 000100 147226      BIS      #100,BXCSR      ;SET ENABLE "XMIT-IE" BIT
3448 031622 000240                      240                    ; INTERRUPT SHOULD BE HELD OFF...
3449 031624 000406                      BR      3#             ;...BR IF SO.
3450 031626 022626                      CMP      (SP), (SP)    ; IT WASN'T.
3451 031630 042777 000100 147212      BIC      #100,BXCSR    ; REMOVE "XMIT-IE"
3452 031636 104423 031560                      ERROR,  T131          ;; *** XMT-RDY INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
3453
3454 031642 012711 031670      3#:     MOV      #4,(R1)    ;CHANGE THE VECTOR.
3455 031646 106427 000000      MTPS     #PPIO         ;LOWER CPU PRIORITY
3456 031652 000240                      240                    ; INTERRUPT SHOULD HAVE COME IN.
3457 031654 042777 000100 147166      BIC      #100,BXCSR    ; IT DIDN'T, REMOVE "XMIT-IE" BIT
3458 031662 104423 031560                      ERROR,  T131          ;; *** XMT-RDY INTERRUPT NOT RECEIVED AT CPU LEVEL 0 ***
3459 031666 000415                      BR      6#
3460
3461      ; NOW, ON INTERRUPT, WE'RE HERE AT CPU LEVEL 4 (5 IF SLU2).
3462      ; LOWER CPU AGAIN AND CHECK THAT WE DON'T GET ANOTHER INTERRUPT.
3463
3464 031670 022626                      4#:     CMP      (SP), (SP)
3465 031672 012711 031706      MOV      #5,(R1)      ; CHANGE THE VECTOR.
3466 031676 106427 000000      MTPS     #PPIO         ;LOWER THE CPU PRIORITY AGAIN.
3467 031702 000240                      240                    ; SHOULD'N'T GET ANOTHER...
3468 031704 000406                      BR      6#             ;...OK, EXIT.
3469 031706 022626                      5#:     CMP      (SP), (SP)    ; 2ND INTERRUPT -- BAD NEWS.
3470 031710 042777 000100 147132      BIC      #100,BXCSR    ;REMOVE "XMIT-IE"
3471 031716 104423 031560                      ERROR,  T131          ;; *** XMT INTERRUPT ACK FAILED TO REMOVE THE REQUEST ***
3472
3473 031722 042777 000100 147120      6#:     BIC      #100,BXCSR    ; BE SURE THE ENABLE IS REMOVED.
3474 031730 012711 030330      MOV      #BADI,(R1)    ;RESET FALSE INTR VECTOR
3475
3476      ;*****
      .SBTTL      T132 -- TEST THAT EACH XMIT BAUD RATE SETS A DIFFERENT SPEED
      ;*****
3477 031734 104420 000132      T132:   BEGINTEST, 132      ;; CHECK SWITCH REGISTER OPTIONS.
3478
3479      ; FIRST DETERMINE THE RELATIVE SPEED OF EACH BAUD RATE.
3480      ; TRANSMIT A CHARACTER AND COUNT CPU BUS TRANSACTION CYCLES UNTIL
3481      ; TRANSMISSION IS COMPLETE (XMT-RDY ASSERTS).
3482      ; SAVE THE OBSERVED CYCLE COUNT IN THE SPEED TABLE.
3483      ; IF NO FLAG BY THE TIME THE COUNT OVERFLOWS (65536. CYCLES), SAVE
3484      ; THE ZERO, WHICH IMPLIES XMT RDY NEVER SET AT THAT SPEED.
3485
3486      ; NOTE: ALL FOLLOWING RECEIVER TESTS WILL USE THE SPEED VALUES
3487      ; OBSERVED HERE AS LOOP CONTROL COUNTERS.
3488      ; NOTE:NOTE: IF THERE'S A TTY ATTACHED, YOU'LL SEE SOME GARBAGE
3489      ; CHARACTERS ON THE SCREEN.
3490
3490 031740 012700 000002                      MOV      #2,R0        ; START WITH SPEED 0 (300 BAUD).
3491 031744 012701 001016                      MOV      #SPDO,R1     ; TABLE POINTER.
3492 031750 005002                      1#:     CLR      R2        ; CLEAR THE ACCUMULATOR.
3493 031752 010077 147072                      MOV      R0,BXCSR     ; SET PDR BITS.
3494 031756 012777 000000 147066      MOV      #0,BXBUF    ; XMIT A CHAR.
3495 031764 105777 147060      2#:     TSTB     BXCSR     ; ON READY, CHAR HAS BEEN COPIED...
3496 031770 100375 147052                      BPL      2#           ;...INTO THE SERIAL OUTPUT REGISTER.
3497 031772 012777 000000                      MOV      #0,BXBUF    ; XMIT AGAIN AND MEASURE THE TIME...
3498
3499      ;...UNTIL READY RETURNS WHICH IS THE...
3500 032000 105777 147044      3#:     TSTB     BXCSR     ;...SERIAL OUTPUT TIME.
      ; [ 4 ]

```

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

```

3501 032004 100403          BHI      40
3502 032006 062702 000010  ADD     #8,,R2
3503 032012 001372          BNE     30
3504 032014 010221          MOV     R2,(R1)+
3505
3506
3507 032016 020067 147016      CMP     R0,DFPBR
3508 032022 001002          BNE     50
3509 032024 010267 147012      MOV     R2,DFSPD
3510 032030 062700 000010      ADD     #10,R0
3511 032034 020027 000072      CMP     R0,#72
3512 032040 101743          BLOS   10
3513 032042 016777 146772 147000  MOV     DFPBR,BXCSR
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526 032050 012700 000072      MOV     #72,R0
3527 032054 012701 001034      MOV     #SPD7,R1
3528 032060 011102          MOV     (R1),R2
3529 032062 001002          BNE     110
3530 032064 104423 031734      ERROR, T132
3531 032070 020127 001016      CMP     R1,#SPD0
3532 032074 001421          BEQ     140
3533
3534 032076 014103          MOV     -(R1),R3
3535 032100 010304          MOV     R3,R4
3536 032102 006203          ASR     R3
3537 032104 160203          SUB     R2,R3
3538 032106 100001          BPL     120
3539 032110 003403          NEG     R3
3540
3541 032112 006204          ASR     R4
3542 032114 006204          ASR     R4
3543 032116 000240          NOP
3544 032120 000240          NOP
3545 032122 020304          CMP     R3,R4
3546 032124 003402          BLE     130
3547 032126 104423 031734      ERROR, T132
3548 032132 162700 000010      SUB     #10,R0
3549 032136 100350          BPL     100
3550 032140
3551

```

; [ 1 ] WE'RE DONE ON XMT-RDY.  
; [ 2 ] COUNT 8 CYCLES/LOOP...  
; [ 1 ] ...UNTIL DONE (OR OVERFLOW).  
; OK, R2 = TX TIME IN BUS CYCLES...  
; ...AT APPROX 3 USEC/COUNT...  
; ...SAVE IT IN THE SPEED TABLE.  
; CURRENT RATE = DEFAULT ???  
; SKIP IF NOT.  
; YES, SAVE AS DEFAULT SPEED.  
; INCREMENT PBR BITS.  
; LOOP FOR ALL 8 SETTINGS.  
; DONE, RESTORE DEFAULT XMTTR...  
; ...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.

; WORK FROM THE HIGHEST DOWN.  
; TABLE POINTER.  
; GET HIGHER (SMALLER) VALUE.

; \* \* \* XMT-RDY NEVER SET AT PBR IN R0 \* \* \*  
; IF PBR 0, THERE'S NOTHING LOWER...  
; ...SO JUST QUIT.

; GET NEXT LOWER (LARGER) VALUE.  
; SAVE A COPY FOR THE TOLERANCE.  
; NOW, 1/2 THE LARGER...  
; ...MINUS THE SMALLER = DELTA.

; MAKE ABSOLUTE IF NECESSARY.

; SET ALLOWABLE TOLERANCE...  
; ... = 25.0% OF THE LARGER.

; DELTA <= 25.0% ???  
; BR IF SO.

; \* \* \* SPEED DIFFERENTIAL ERROR \* \* \*  
; DEC RATE COUNT...  
; ...AND LOOP 'TIL DONE.

140:

.SBTTL

RECEIVERS

```

3553
3554 032140
3555
    032140 104420 000133
3556 032144 106427 000300
3557 032150 012777 000100 146666
3558 032156 017700 146662
3559 032162 005077 146656
3560 032166 017701 146652
3561 032172 032700 000100
3562 032176 001002
3563 032200 104423 032140
3564 032204 032701 000100
3565 032210 001402
3566 032212 104423 032140
3567 032216 106427 000000
3568
3569
    032222 104420 000134
3570
3571
3572
3573
3574
3575
3576
3577 032226 016777 146606 146614
3578 032234 052777 000004 146606
3579 032242 105777 146602
3580 032246 100373
3581 032250 016700 146566
3582 032254 006300
3583 032256 010001
3584 032260 010002
3585 032262 010003
3586 032264 077001
3587 032266 017700 146554
3588 032272 017700 146546
3589 032276 032700 004200
3590 032302 001402
3591 032304 104423 032222
3592
3593 032310 012777 000000 146534
3594 032316 000240
3595 032320 017700 146520
3596 032324 032700 004000
3597 032330 001003
3598 032332 077106
3599 032334 104423 032222
3600 032340 032700 000200
3601 032344 001402
3602 032346 104423 032222
3603 032352 017700 146466

```

```

.SBTTL RECEIVERS
RCVRS: ; RECEIVER TEST SECTION.
;*****
.SBTTL T133 -- TEST THAT "RCV-IE" CAN BE SET AND CLEARED
;*****
T133: BEGINTEST, 133 ; CHECK SWITCH REGISTER OPTIONS.
      MTPS @PRI6 ;RAISE CPU PRIORITY
      MOV @100,BRCR ;SET "RCV-IE" BIT...
      MOV BRCR,R0 ;...AND SAVE IN R0.
      CLR BRCR ; CLEAR IT...
      MOV BRCR,R1 ;...AND SAVE IN R1.
      BIT @100,R0
      BNE 10 ; BR IF IT SET.
      ERROR, T133 ; ** RCV-IE BIT FAILED TO SET **
10: BIT @100,R1
      BEQ 20 ; BR IF IT CLEARED.
      ERROR, T133 ; ** RCV-IE BIT FAILED TO CLEAR **
20: MTPS @PRIO

;*****
.SBTTL T134 -- TEST THAT "RCV-ACT" AND "RCV-DONE" CAN SET AND CLEAR
;*****
T134: BEGINTEST, 134 ; CHECK SWITCH REGISTER OPTIONS.
;
;THE TEST ENSURES THAT:
; 1. THE INTERNAL "MAINT" PATH IS FUNCTIONAL
; 2. RCV-ACT CLEARS BEFORE RCV-DONE SETS.
; 3. RCV-ACT CAN SET AND CLEAR
; 4. READING RCV DATA REGISTER CLEARS RCV DONE
;
      MOV DFPBR,BXCSR
      BIS @4,BXCSR ; SET MAINT BIT.
10: TSTB BXCSR ;INSURE WE'RE "XMIT-RDY"
      BPL 10
      MOV DFSPD,R0 ; SET A 2 CHAR (20 BITS) DELAY COUNT...
      ASL R0
      MOV R0,R1 ;...AND SAVE 3 COPIES FOR LATER.
      MOV R0,R2
      MOV R0,R3
      SOB R0,
      MOV BRBUF,R0 ; WAIT 20 BIT TIMES...
      MOV BRCR,R0 ;...AND DUMMY READ.
      BIT @RCVACT!RCVDUN,R0 ; GET RCVR STATUS.
      BEQ 20 ; BOTH ACTIVE AND DONE SHOULD BE CLEAR
      ERROR, T134 ; ** RCV-ACT AND/OR RCV-DUN ARE SET IN ERROR **
20: MOV @0,BXBUF ;NOW, XMIT A NULL CHARACTER
      MOV 240
30: MOV BRCR,R0 ;GET RCVR STATUS
      BIT @RCVACT,R0
      BNE 40 ;BR IF AND WHEN ACTIVE SETS.
      SOB R1,30
      ERROR, T134 ; ** RCV-ACT FAILED TO SET **
40: BIT @RCVDUN,R0 ;TEST THAT "RCV-DUN" IS NOT SET YET.
      BEQ 50
      ERROR, T134 ; ** RCV-DUN SET WHEN RCV-ACT WAS SET **
50: MOV BRCR,R0 ;READ RCVR STATUS AGAIN.

```

PC 32346 = RCV-DUN SET WHEN RCV-ACT WAS SET.

```

3604 032356 032700 004000          BIT      @RCVACT,RO
3605 032362 001403          BEQ      6@
3606 032364 077206          SOB      R2,5@
3607 032366 104423 032222          ERROR,  T134
3608 032372 017700 146446          6@:    MOV      @RCSR,RO
3609 032376 032700 000200          BIT      @RCVDUN,RO
3610 032402 001003          BNE     7@
3611 032404 077306          SOB      R3,6@
3612 032406 104423 032222          ERROR,  T134
3613 032412 017700 146430          7@:    MOV      @RBUF,RO
3614 032416 017700 146422          MOV      @RCSR,RO
3615 032422 032700 000200          BIT      @RCVDUN,RO
3616 032426 001402          BEQ      8@
3617 032430 104423 032222          ERROR,  T134
3618 032434          8@:
3619
3620
;*****
;SBTTL      T135 -- TEST THE "RCV-BRK", "RCV-ERR", AND "FR-ERR" BITS
;*****
032434 104420 000135          T135:   BEGINTEST, 135          ;; CHECK SWITCH REGISTER OPTIONS.
;
; GENERATE A FRAMING ERROR BY SETTING THE RCV-BRK BIT.
; ON SLU1 -- EXPECT THAT BRK INVOKES A MASKABLE LEVEL 7 HALT REQUEST.
;
3625 032440 013746 000140          1@:    MOV      @BHALT,-(SP)          ; SAVE BREAK VECTOR.
3626 032444 106427 000340          MTPS   @PRI7                  ; RAISE CPU.
3627 032450 016777 146364 146372          MOV      @DFPDR,@XCSR
3628 032456 052777 000004 146364          BIS      @4,@XCSR              ; SET MAINT BIT.
3629 032464 017700 146356          MOV      @RBUF,RO              ; DUMMY READ.
3630 032470 012767 032624 145442          MOV      @6@,BHALT            ; SET THE HALT VECTOR (FOR SLU1).
3631 032476 012737 000340 000142          MOV      @PRI7,@BHALT+2
3632 032504 016701 146332          MOV      @DFSPD,R1
3633 032510 006301          ASL     R1                      ; SET A 2 CHAR (20 BITS) DELAY.
3634 032512 052777 000001 146330          BIS      @XMTBRK,@XCSR         ; SET "XBRK"...
3635 032520 012777 000025 146324          MOV      @25,@XBUF            ;...START XMITTER...
3636 032526 077101          SOB     R1..                    ;...AND DELAY. IF SLU1, INTERRUPT...
3637          ;...SHOULD BE HELD OFF (CPU IS AT 7).
3638 032530 105777 146310          TSTB   @RCSR                  ; DONE SHOULD BE THERE BY NOW.
3639 032534 100402          BMI    2@                      ; BR IF SO.
3640 032536 104423 032444          ERROR,  1@
3641 032542 026727 146276 176540          2@:    CMP      @RCSR,@SLU2           ;...NO RCV-DUN AFTER XMIT WITH XMT-BRK SET...
3642 032550 001430          BEQ     7@                      ; IF SLU2...
3643          ;...JUST VERIFY STATUS/DATA.
3644          ; SLU1, VERIFY THAT "BHALT" INTERRUPT DOES THE RIGHT THINGS.
3645          ;
3646 032552 012767 032574 145360          3@:    MOV      @4@,BHALT            ; SO FAR, SO GOOD, CHANGE THE VECTOR...
3647 032560 106427 000300          MTPS   @PRI6                  ;...AND LOWER CPU TO 6...
3648 032564 000240          240
3649 032566 104423 032444          ERROR,  1@
3650 032572 000417          BR     7@                      ;...INTERRUPT SHOULD COME IN.
3651 032574 022626          4@:    CMP      (SP)+,(SP)+         ;...HALT INTERRUPT NOT RECEIVED AT CPU LEVEL 6...
3652 032576 012767 032614 145334          MOV      @5@,BHALT            ; OK, INT RECEIVED, FIX STACK.
3653 032604 106427 000300          MTPS   @PRI6                  ; NOW, CHANGE THE VECTOR...
3654 032610 000240          240
3655 032612 000407          BR     7@                      ;...LOWER CPU AGAIN...
3656 032614 022626          5@:    CMP      (SP)+,(SP)+         ;...AND SEE THAT WE DON'T GET ANOTHER.
3657 032616 104423 032444          ERROR,  1@                      ; TERRIFIC -- PROCEED.
; 2ND HALT TRAP -- TOUGH LUCK !!
;...DOUBLE HALT TRAP RECEIVED...

```

PC 32616 = DOUBLE HALT TRAP RECEIVED.

```

3658 032622 000403
3659 032624 022626
3660 032626 104423 032444
3661
3662
3663
3664 032632 017700 146210
3665 032636 012637 000140
3666 032642 032700 004000
3667 032646 001002
3668 032650 104423 032444
3669 032654 032700 100000
3670 032660 001403
3671 032662 032700 020000
3672 032666 001002
3673 032670
3674 032674 104423 032444
3675 032676 105700
3676 032676 001402
3677 032700 104423 032444
3678
3679
3680 032704 042777 000001 146136
3681 032712 016700 146124
3682 032716 077001
3683 032720 017700 146122
3684 032724 012777 000025 146120
3685 032732 105777 146106
3686 032736 100375
3687 032740 017700 146102
3688 032744 032700 124000
3689 032750 001402
3690 032752 104423 032704
3691 032756 120027 000025
3692 032762 001402
3693 032764 104423 032704
3694 032770 106427 000000
3695
3696
; *****
; .SBTTL ?136 -- TEST THAT "RCV-ERR" AND "OR-ERR" BITS SET AND CLEAR
; *****
T136: BEGINTEST, 136
MOV DFPBR, EXCSR
BIS #4, EXCSR
MOV BRBUF, RO
MOV #4, R1
MOV #0, EXBUF
TSTB EXCSR
BPL 2#
SOB R1, 1#
MOV DFPDR, RO
SOB RO, .
MOV BRBUF, RO
BPL 3#
BIT #ORERR, RO
; WE'RE HERE IF HALT WASN'T MASKED !!
; ** HALT TRAP INTERRUPT NOT MASKED BY CPU LEVEL 7 **
;
; BOTH LINES VERIFY CORRECT STATUS AND DATA BITS.
;
7#: MOV BRBUF, RO ; READ (IF SLU1, LOWERS BRK REQ)...
MOV (SP)+, @BHALT ; ...AND RESTORE BREAK VECTOR.
BIT #RCVBRK, RO ; RCV-BRK SHOULD BE SET.
BNE ERROR, 1#
; ** RCV-BRK BIT DIDN'T SET **
; SO SHOULD RCV-ERR...
8#: BIT #RCVERR, RO
BEQ 9#
BIT #FRERR, RO
BNE 10#
; ...AND FR-ERR.
;
; ** RCV-ERR AND/OR FR-ERR BITS DIDN'T SET **
; RCDV DATA SHOULD BE ZERO.
10#: TSTB RO
BEQ 11#
ERROR, 1#
; ** RECEIVED DATA NON-ZERO ON "BRK" **
;
; AND FINALLY, WE SHOULD BE ABLE TO RESTORE A NORMAL RECEIVER.
;
11#: BIC #XMITBRK, EXCSR ; NOW, CLEAR THE BREAK BIT...
MOV DFSPD, RO
SOB RO, .
MOV BRBUF, RO
MOV #25, EXBUF
; ...DELAY...
; ...DUPPY READ.
; XMIT A CHAR...
12#: TSTB EXCSR
BPL 12#
MOV BRBUF, RO
BIT #RCVERR|FRERR|RCVBRK, RO ; ERRORS SHOULD BE GONE.
BEQ 13#
; BR IF SO.
; ** RCV-ERR, FR-ERR, AND/OR BRK BITS DIDN'T CLEAR **
13#: CMPB RO, #25
BEQ 14#
; AND CORRECT DATA RECEIVED.
; ** DATA INCORRECT AFTER "BRK" SEQUENCE **
; DONE, LOWER THE CPU.
14#: HTPS #PRIO
; *****
; *****
032774 104420 000136
3697 033000 016777 146034 146042
3698 033006 052777 000004 146034
3699 033014 017700 146026
3700 033020 012701 000004
3701 033024 012777 000000 146020
3702 033032 105777 146012
3703 033036 100375
3704 033040 077107
3705
3706 033042 016700 145774
3707 033046 077001
3708 033050 017700 145772
3709 033054 100003
3710 033056 032700 040000
MOV DFSPD, RO
SOB RO, .
MOV BRBUF, RO
BPL 3#
BIT #ORERR, RO
; LOOP 4 TIMES...
; ...SHOULD CAUSE AN OVER-RUN ERROR.
; WAIT 1 ADDITIONAL CHAR TIME...
; ...AND READ RCV DATA BUFFER.
; BR IF NO "RCV-ERR".
; "OR-ERR" SHOULD BE SET AS WELL.

```



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

```

3711 033062 001002          BNE      40          ; BOTH SET -- TERRIFIC.
3712 033064          30:   ERROR,   T136          ;; *** RCV-ERR AND/OR OR-ERR NOT SET ON FORCED OVER-RUN ***
3713 033064 104423 032774
3714 033070 012777 000000 145754 40:   MOV      00,0XBUF          ; NOW XMIT AGAIN...
3715 033076 105777 145742 50:   TSTB   BRCSR          ; ...WAIT FOR RCVR DUN...
3716 033102 100375
3717 033104 017700 145736      MOV      BRBUF,RO          ; ...AND READ, ERROR SHOULD BE GONE.
3718 033110 032700 140000      BIT      @RCVERR|ORERR,RO
3719 033114 001402          BEQ      60          ; BR IF SO.
3720 033116 104423 032774      ERROR,   T136          ;; *** RCV-ERR AND/OR OR-ERR BITS FAILED TO CLEAR ***
3721 033122
3722
3723

```

```

;*****
.SBTTL      T137 -- TEST THAT WE CAN INTERRUPT ON "RCV-DUN"
;*****

```

```

T137:  BEGINTEST, 137          ;; CHECK SWITCH REGISTER OPTIONS.
      MOV      DFPBR,0XCSR
      BIS      04,0XCSR          ; SET MAINT BIT.
      MOV      RVEC,R1          ; GET RCVR VECTOR POINTER.
      MTPS    2(R1)          ; RAISE CPU TO RCVR LEVEL.
      MOV      020,(R1)       ; SET RCVR VECTOR.
      MOV      00,0XBUF       ; TRANSMIT A CHARACTER
      10:   TSTB   BRCSR          ; WAIT FOR "RCV-DUN"
      BPL      10
      BIS      0100,BRCSR       ; SET ENABLE "RCV-IE" BIT
      240
      BR      30              ; INTERRUPT SHOULD BE HELD OFF...
      20:   CMP      (SP), (SP)  ; ...BR IF SO.
      BIC      0100,BRCSR       ; BUT IT WASN'T.
      ERROR,   T137          ; REMOVE "RCV-IE"
      30:   MOV      040,(R1)    ; *** RCV-DUN INTERRUPT AT WRONG LEVEL (TOO HIGH) ***
      MTPS    @PRIO
      240
      BIC      0100,BRCSR       ; CHANGE VECTOR.
      ERROR,   T137          ; LOWER CPU PRIORITY
      BR      60              ; INTERRUPT SHOULD COME IN.
      40:   CMP      (SP), (SP)  ; BUT IT DIDN'T, REMOVE "RCV-IE" BIT
      MOV      050,(R1)       ; *** RCV-DUN INTERRUPT NOT RECEIVED AT ***
      MTPS    @PRIO
      BR      60

```

```

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

```

```

      40:   CMP      (SP), (SP)  ; CHANGE THE VECTOR.
      MOV      050,(R1)       ; LOWER THE CPU PRIORITY AGAIN.
      MTPS    @PRIO          ; SHOULDN'T GET ANOTHER...
      BR      60              ; ...OK, EXIT.
      50:   CMP      (SP), (SP)  ; 2ND INTERRUPT -- TOO BAD !!
      BIC      0100,BRCSR       ; REMOVE "RCV-IE"
      ERROR,   T137          ; *** INTERRUPT ACK FAILED TO REMOVE THE REQUEST ***
      60:   BIC      0100,BRCSR  ; INSURE THE ENABLE IS OFF.
      MOV      BRBUF,RO        ; READ
      MOV      @BADRI,(R1)     ; RESET FALSE INTR VECTOR

```

```

;*****
.SBTTL      T140 -- FLOAT 1 AND 0 THRU THE "MAINT" DATA LOOP (INTERNAL)
;*****

```

T140 -- FLOAT 1 AND 0 THRU THE "MAINT" DATA LOOP (INTERNAL)

```

3763 033316 104420 000140
3764
3765
3766
3767 033322 016777 145512 145520      MOV    DFPBR,BXCSR
3768 033330 052777 000004 145512      BIS    #4,BXCSR          ; SET MAINT BIT.
3769 033336 012700 030340                MOV    #SYNC,R0         ; DATA TABLE POINTER.
3770 033342 112077 145504                10:   MOVB  (R0),BXBUF      ; XMIT THE SYNC PAIR <CR>...
3771 033346 105777 145476                20:   TSTB  BXCSR
3772 033352 100375                        BPL    20
3773 033354 112077 145472                MOVB  (R0),BXBUF      ; ...AND <LF>.
3774 033360 105777 145460                30:   TSTB  BRCSR          ; WAIT FOR RCVR DONE...
3775 033364 100375                        BPL    30
3776 033366 127727 145454 000012        CPB   BRBUF,#12       ; AND PROCEED ON THE RCVD <LF>.
3777 033374 001371                        BNE    30
3778
3779 033376 112001                40:   MOVB  (R0),R1         ; NOW FLOAT DATA THRU THE LOOP.
3780 033400 010177 145446                50:   MOV   R1,BXBUF       ; XMIT A BYTE...
3781 033404 105777 145434                60:   TSTB  BRCSR
3782 033410 100375                        BPL    60
3783 033412 017702 145430                MOV   BRBUF,R2        ; ...AND READ IT BACK.
3784 033416 120102                        CPB   R1,R2
3785 033420 001402                        BEQ   70
3786 033422 104423 033400                ERROR, 50              ; *** RCVD DATA INCORRECT, MAINT DATA PATH FAILURE ***
3787 033426 105701                70:   TSTB  R1              ; TERMINATOR DONE ??
3788 033430 001362                        BNE   40              ; NOT YET, LOOP.
3789
3790

```

.SBTTL T141 -- FLOAT 1 AND 0 THRU THE LOOP-BACK (EXTERNAL, SLU2 ONLY)

```

3791 033432 104420 000141
3792 033436 026727 145402 177560
3793 033444 001453
3794 033446 016777 145366 145374      MOV    DFPBR,BXCSR      ; DEFAULT XMIT STATUS (NO MAINT).
3795 033454 017700 145366                MOV    BRBUF,R0         ; DUMMY READ.
3796 033460 016700 145356                MOV    DFSPD,R0
3797 033464 006300                ASL    R0
3798 033466 006300                ASL    R0
3799 033470 010001                MOV    R0,R1           ; 4 CHAR LOOP TIMER.
3800 033472 112777 000033 145352        10:   MOVB  #33,BXBUF       ; SAVE A COPY.
3801 033500 105777 145340                TSTB  BRCSR           ; XMIT AN "ESC".
3802 033504 100413                        BMI   30              ; IF RCVR-DUN SETS NOW, WE CAN ASSUME...
3803 033506 077004                        SOB   R0,10          ; ...THAT THE LOOP-BACK IS INSTALLED.
3804 033510 112777 000132 145334        20:   MOVB  #2,BXBUF       ; IF NOT, XMIT "Z".
3805 033516 105777 145322                TSTB  BRCSR          ; IF IT SETS NOW, WE HAVE A TTY...
3806 033522 100426                        BMI   DLDUN          ; ...RESPONDING TO THE "ESC Z" SEQUENCE.
3807 033524 077104                        SOB   R1,20          ; IF NEITHER, THE LOOP MUST BE OPEN.
3808 033526 104423 033432                ERROR, T141          ; *** NO RCV-DUN -- IS LOOP CONNECTOR INSTALLED ??? ***
3809 033532 000422                        BR    DLDUN          ; QUIT (WHILE WE'RE AHEAD) !!
3810
3811 033534 017700 145306                30:   MOV   BRBUF,R0       ; OK, DUMMY READ...
3812 033540 012700 030342                MOV   #FLT1,R0        ; ...SET TABLE POINTER...
3813 033544 112001                40:   MOVB  (R0),R1        ; ...AND FLOAT DATA THRU THE LOOP.
3814 033546 010177 145300                50:   MOV   R1,BXBUF       ; XMIT A BYTE...

```

PC 33526 = NO RCV-DUN -- IS LOOP CONNECTOR INSTALLED ???.

```

3815 033552 105777 145266      60:  TSTB  BRCSR
3816 033556 100375              BPL   60
3817 033560 017702 145262      MOV  BRBUF,R2      ;...AND READ IT BACK.
3818 033564 120102              CMPB  R1,R2
3819 033566 001402              BEQ   70
3820 033570 104423 033546      ERROR, S0          ;: *** RCV DATA INCORRECT, EXTERNAL DATA PATH FAILURE ***
3821 033574 105701      70:  TSTB  R1           ;: TERMINATOR DONE ??
3822 033576 001362              BNE   40           ;: NOT YET, LOOP.
3823
3824 033600 016777 145234 145242 DLDUN: MOV  DFPBR,8XCSR   ;: INSURE A DEFAULT XMITTER...
3825 033606 017700 145234      MOV  BRBUF,R0     ;:...AND RECEIVER.
3826 033612 012767 000006 144164      MOV  @ERRVEC+2,ERRVEC
3827 033620 004767 001232      CALL ENDSEG      ;: CHECK FOR LOOP-IN-SEGMENT...
3828
3829 033624 026727 145214 176540      CMP  RCSR,@SLU2   ;: ...RETURN AND FALL THRU IF NOT.
3830 033632 001402              BEQ   S7          ;: LINE 2 ??
3831 033634 000167 174302      JMP  S6          ;: BR IF SO...
3832
                                ;:...ELSE, GO 'ROUND ONCE.
                                .SBTTL

```

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

```

3834 .SBTTL SECTION 7 -- PARALLEL I/O PORT (8255) TESTS.
3835 ;
3836 ; PORTS A, B, AND C MUST BE INTERCONNECTED THRU A SPECIAL
3837 ; LOOP-BACK CONNECTOR. IF NOT THESE TESTS BREAK.
3838 ;
3839 PPH0= 233 ; CONTROL WORD TO SET MODE 0 (ALL INPUT).
3840 PPH1= 264 ; ANOTHER FOR MODE 1, A IN, B OUT, C<7:6> OUT.
3841 ;
3842 LEDBIT= 200 ; LED CONTROL BIT (0 = ON, 1 = OFF).
3843 INRDY= 040 ; INPUT DATA READY ( IDF ).
3844 INENA= 020 ; ENABLE INTERRUPT ON "INRDY"
3845 ININT= 010 ; INPUT INTERRUPT BIT.
3846 ;
3847 OUTENA= 004 ; ENABLE INTERRUPT ON "OUTRDY".
3848 OUTRDY= 002 ; OUTPUT PORT READY ( -OSF ).
3849 OUTINT= 001 ; OUTPUT INTERRUPT BIT.
3850 ;
3851 033640 012767 036453 002460 S7: MOV #SEG7,SEGN ; SEGMENT 7 TEXT.
3852 ;
3853 ;*****
.SBTTL T142 -- TEST PIO REGISTER ADDRESSES
;*****
3854 033646 104420 000142 T142: BEGINTEST, 142 ; CHECK SWITCH REGISTER OPTIONS.
3855 033652 000240 000240 240,240 ; *** TRY A RESET HERE ***
3856 033656 012700 176200 MOV #PIOA,R0 ; A IS 1ST OF 4 REGISTERS.
3857 033662 012701 000130 MOV #PIOB,R1 ; B IS 1ST OF 2 VECTOR PAIRS.
3858 033666 012702 000240 MOV #PIOP,R2 ; PRIORITY (5) FOR BOTH.
3859 033672 012767 033762 144104 MOV #BADPIO,ERRVEC ; SET TIME-OUT VECTOR.
3860 033700 012706 003776 MOV #STACK,SP
3861 033704 010067 145150 MOV RO,PPA ; SET REGISTER POINTERS.
3862 033710 005720 TST (R0); ; IF ANY TRAP -- TROUBLE !B!
3863 033712 010067 145144 MOV RO,PPB
3864 033716 005720 TST (R0);
3865 033724 005720 MOV RO,PPC
3866 033726 010067 145134 TST (R0);
3867 033732 005710 MOV RO,PPCM
3868 033734 010167 145132 TST (R0)
3869 033740 012721 033772 MOV R1,PPBV ; SAVE B VECTOR POINTER.
3870 033744 010221 033772 MOV #BADBI,(R1); ; SET TO CATCH UNEXPECTED INTERRUPTS.
3871 033746 010167 145116 MOV R2,(R1);
3872 033752 012721 034002 MOV R1,PPAV ; DITTO FOR PORT A VECTOR.
3873 033756 010221 MOV #BADAI,(R1);
3874 033760 000414 MOV R2,(R1);
3875 BR PIOTST
3876 ;
3877 ; COME HERE IF TIME-OUT OR UNEXPECTED PIO INTERRUPTS.
3878 033762 BADPIO:
033762 104423 033646 ERROR, T142 ; *** BUS TIME-OUT ON PIO ADDRESS ***
033766 000167 001050 JMP PIODUN
3880 033772 BADBI:
033772 104423 034012 ERROR, PIOTST ; *** UNEXPECTED PORT B INTERRUPT ***
033776 000167 001040 JMP PIODUN
3882 034002 BADAI:
034002 104423 034012 ERROR, PIOTST ; *** UNEXPECTED PORT A INTERRUPT ***
034006 000167 001030 JMP PIODUN
3884 034012 PIOTST:

```

PC 34002 = UNEXPECTED PORT A INTERRUPT.

```

3885
034012 104420 000143
3886
3887
3888
3889
3890
3891 034016 000240 000240
3892 034022 000005
3893 034024 005000
3894 034026 077001
3895 034030 117700 145024
3896 034034 120027 177777
3897 034040 001402
3898 034042 104423 034012
3899 034046
3900
3901
034046 104420 000144
034052 112777 000264 145006
3902
3903
3904 034060 112777 000016 145000
3905
3906 034066 112777 000014 144772
3907 034074 012700 000012
3908 034100 077001
3909 034102 117703 144752
3910 034106 117703 144752
3911 034112 120327 000002
3912 034116 001402
3913 034120 104423 034046
3914
3915 034124 005001
3916 034126 012702 177777
3917 034132 005000
3918 034134 110177 144722
3919 034140 117703 144720
3920 034144 032703 000002
3921 034150 001402
3922 034152 104423 034132
3923 034156 117703 144702
3924 034162 120327 000042
3925 034166 001403
3926 034170 077006
3927 034172 104423 034132
3928 034176 117702 144656
3929 034202 117703 144656
3930 034206 120327 000002
3931 034212 001402
3932 034214 104423 034132
3933 034220 120201
3934 034222 001402
3935 034224 104423 034132

;*****
;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           RO ; DELAY, CHIP SEEMS TO TAKE QUITE...
;          SOB           RO,; ...A WHILE (200USEC) TO GET "RESET".
;          MOVB          SPPA,RO ; READ PORT A (INPUT).
;          CPB           RO,0-1 ; LO BYTE ONLY -- SHOULD BE HIGH.
;          BEQ           10
;          ERROR,       T143    ;; *** PORT A RESET STATE INCORRECT ***
10:
;*****
;SBTTL      T144 -- TEST MODE 1 I/O THRU THE LOOP CONNECTOR (FLAG MODE)
;*****
T144:  BEGINTEST, 144          ;; CHECK SWITCH REGISTER OPTIONS.
;          MOVB          SPPH1,SPPCW ; SET MODE 1, A IN, B OUT, C<7:6> OUT.
;          MOVB          0<7_1.>!0,SPPCW ; CLEAR C(7) TO TURN ON THE TINY LED. ;JSD REV B
;          MOVB          016,SPPCW ; CLEAR C(7) TO TURN ON THE TINY LED. ;JSD REV B
;          MOVB          0<6_1.>!0,SPPCW ; CLEAR C(6) JUST FOR THE HELL OF IT. ;JSD REV B
;          MOVB          014,SPPCW ; CLEAR C(6) JUST FOR THE HELL OF IT. ;JSD REV B
;          MOV           010,,RO
;          SOB           RO,; DELAY ABOUT 50 USEC...
;          MOVB          SPPA,R3 ; ...AND DUFFY READ TO DROP "INDY".
;          MOVB          SPPC,R3 ; NOW, GET STATUS.
;          CPB           R3,0OUTRDY ; SHOULD HAVE THIS STATUS SET.
;          BEQ           10
;          ERROR,       T144    ;; *** PORT C MODE 1 STATUS INCORRECT ***
10:  CLR           R1          ; SEND A BINARY COUNT FROM R1...
;          MOV           0-1,R2 ; ...TO R2 THRU THE 8255.
20:  CLR           RO          ; LOOP DELAY.
;          MOVB          R1,SPPB ; XMIT DATA OUTPUT (THRU B).
30:  MOVB          SPPC,R3 ; GET STATUS.
;          BIT           0OUTRDY,R3 ; OUTRDY SHOULD BE GONE.
;          BEQ           40
;          ERROR,       20      ;; *** OUTRDY STILL SET AFTER OUTPUT XFER ***
40:  MOVB          SPPC,R3 ; GET STATUS AGAIN...
;          CPB           R3,0INDY!OUTRDY ; ...AND WAIT FOR THIS.
;          BEQ           50 ; WONDERFUL, MOVE ON.
;          SOB           RO,40
;          ERROR,       20      ;; *** STATUS INCORRECT AFTER OUTPUT XFER ***
50:  MOVB          SPPA,R2 ; DATA INPUT (THRU A).
;          MOVB          SPPC,R3 ; GET STATUS
;          CPB           R3,0OUTRDY ; EXPECT THIS STATUS.
;          BEQ           60 ; BR IF SO.
;          ERROR,       20      ;; *** STATUS INCORRECT AFTER INPUT XFER ***
60:  CPB           R2,R1 ; DATA LOOP INTACT ??
;          BEQ           70 ; BR IF SO.
;          ERROR,       20 ; *** DATA INCORRECT AFTER OUT/IN EXCHANGE ***

```

PC 34224 = DATA INCORRECT AFTER OUT/IN EXCHANGE.

```

3936 034230 105201          70:  INCB  R1
3937 034232 001337          BNE  20      ; LOOP 'TIL DONE.
3938
3939
;*****
.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  #10,(R4)      ; ...AND SET VECTOR.
      MOVB #<2,1.>11,BPPCH ; SET C<2> B INTR ENABLE.      ;JSD REV B
      MOVB #5,BPPCH      ; SET C<2> B INTR ENABLE.      ;JSD REV B
      240
      BR   20            ; OUTRDY IS ALREADY THERE, SO...
; ...INTERRUPT REQUEST SHOULD BE THERE...
; ...BUT HELD OFF BY THE CPU LEVEL.
; BUT IT WASN'T !!
      10:  CMP  (SP), (SP) ; ** PORT B INTERRUPT LEVEL INCORRECT **
      ERROR, T145
      20:  MOVB BPPC,R3    ; GET STATUS.
      CMPB R3, @OUTENA!OUTRDY!OUTINT
      BEQ  30
      ERROR, T145      ; ** STATUS INCORRECT WITH PORT B INTERRUPT PENDING **
; *****
      30:  MOV  #4,(R4)    ; OK, CHANGE VECTOR.
      MTPS #PRI4        ; LOWER CPU TO LEVEL 4...
      240
; ...INTERRUPT SHOULD COME IN.
      ERROR, T145      ; ** PORT B INTERRUPT NOT RECEIVED **
; *****
      40:  CMP  (SP), (SP) ; WE'RE HERE ON INTERRUPT, FIX STACK.
      MOV  #5,(R4)      ; CHANGE THE VECTOR...
      MTPS #PRI4        ; ...AND LOWER CPU AGAIN...
      240
; ...SHOULDN'T GET ANOTHER INTERRUPT.
      BR   60
      50:  CMP  (SP), (SP) ; BUT WE DID -- TROUBLE.
      ERROR, T145      ; ** DOUBLE PORT B INTERRUPT **
; *****
      60:  MOVB #<2,1.>10,BPPCH ; OK, CLEAR THE ENABLE...      ;JSD REV B
      MOVB #4,BPPCH      ; OK, CLEAR THE ENABLE...      ;JSD REV B
      MOVB #1,BPPB
      CLR  R0
; ...WRITE -- SHOULD LOWER INT REQUEST.
      70:  MOVB BPPC,R3    ; GET STATUS
      BIT  @OUTINT,R3
      BEQ  80
      SOB  R0,70
; BR WHEN REQUEST CLEARS.
      ERROR, T145      ; ** PORT B INT REQUEST DIDN'T CLEAR AFTER WRITE **
; *****
      80:  BITB @INRDY,BPPC
      BNE  90
      SOB  R0,80
; BR WHEN "INRDY" COMES UP...
; ...BUT DON'T WAIT FOREVER.
      90:  MOVB BPPA,R0    ; DUMMY READ.
      MOV  @BADBI,(R4)   ; RESET VECTOR.
; *****
.SBTTL      T146 -- TEST THE PORT A (INPUT) INTERRUPT
;*****
T146:  BEGINTEST, 146      ; CHECK SWITCH REGISTER OPTIONS.
      MOV  PPAV,R4        ; GET PORT A VECTOR POINTER.
3940 034234 104420 000145
3941 034240 016704 144626
3942 034244 106464 000002
3943 034250 012714 034266
3944 034254 112777 000005 144604
3945 034262 000240
3946 034264 000403
3947
3948 034266 022626
3949 034270 104423 034234
3950 034274 117703 144564
3951 034300 120327 000007
3952 034304 001402
3953 034306 104423 034234
3954
3955 034312 012714 034332
3956 034316 106427 000200
3957 034322 000240
3958 034324 104423 034234
3959 034330 000401
3960
3961 034332 022626
3962 034334 012714 034350
3963 034340 106427 000200
3964 034344 000240
3965 034346 000403
3966 034350 022626
3967 034352 104423 034234
3968
3969
3970 034356 112777 000004 144502
3971 034364 112777 000001 144470
3972 034372 005000
3973 034374 117703 144464
3974 034400 032703 000001
3975 034404 001403
3976 034406 077006
3977 034410 104423 034234
3978
3979 034414 132777 000040 144442
3980 034422 001001
3981 034424 077005
3982 034426 117700 144426
3983 034432 012714 033772
3984
3985
3986 034436 104420 000146
3987 034442 016704 144422

```

T146 -- TEST THE PORT A (INPUT) INTERRUPT

```

3987 034446 106464 000002      MTPS      2(R4)      ; RAISE CPU TO PIO LEVEL.
3988                                MOV      @<4_1.>11,BPPCW ; SET C<4> A INTR ENABLE.      ;JSD REV B
3989 034452 112777 000011 144406 ; MOV      @11,BPPCW      ; SET C<4> A INTR ENABLE.      ;JSD REV B
3990 034460 117703 144400      MOV      @PPC,R3
3991 034464 120327 000022      CMP      R3,@INENA!OUTRDY ; EXPECT TO SEE THIS.
3992 034470 001402      BEQ      10
3993 034472 104423 034436      ERROR,   T146      ;; *** STATUS INCORRECT ***
3994
3995 034476 012714 034520      10:      MOV      @20,(R4)      ; SET THE VECTOR.
3996 034502 005000      CLR      R0
3997 034504 012701 000123      MOV      @123,R1
3998 034510 110177 144346      MOV      R1,@PPB      ; SEND DATA (B => A) THRU THE LOOP.
3999 034514 077001      SOB      R0, .      ; INTERRUPT SHOULD BE HELD OFF.
4000 034516 000403      BR      30
4001 034520 022626      20:      CMP      (SP),.(SP)    ; BUT IT WASN'T.
4002 034522 104423 034436      ERROR,   T146      ;; *** PORT A INTERRUPT LEVEL INCORRECT ***
4003
4004 034526 117703 144332      30:      MOV      @PPC,R3      ; GET STATUS.
4005 034532 120327 000072      CMP      R3,@INRDY!INENA!ININT!OUTRDY
4006 034536 001402      BEQ      40      ; BR IF STATUS IS OK.
4007 034540 104423 034436      ERROR,   T146      ;; *** STATUS INCORRECT WITH PORT A INTERRUPT PENDING ***
4008
4009 034544 012714 034564      40:      MOV      @50,(R4)      ; CHANGE VECTOR.
4010 034550 106427 000200      MTPS      @PRI4      ; LOWER CPU TO LEVEL 4...
4011 034554 000240      240      ; ...INTERRUPT SHOULD COME IN.
4012 034556 104423 034436      ERROR,   T146      ;; *** PORT A INTERRUPT NOT RECEIVED ***
4013 034562 000401      SKP1
4014
4015 034564 022626      50:      CMP      (SP),.(SP)    ; WE'RE HERE ON INTERRUPT, FIX STACK.
4016 034566 012714 034602      MOV      @60,(R4)      ; CHANGE VECTOR...
4017 034572 106427 000200      MTPS      @PRI4      ; ...AND LOWER AGAIN...
4018 034576 000240      240      ; ...SHOULDN'T GET ANOTHER.
4019 034600 000403      BR      70
4020 034602 022626      60:      CMP      (SP),.(SP)    ; 2ND INTERRUPT, FIX STACK.
4021 034604 104423 034436      ERROR,   T146      ;; *** DOUBLE PORT A INTERRUPT ***
4022
4023
4024 034610 112777 000010 144250 ; 70:      MOV      @<4_1.>10,BPPCW ; OK, CLEAR THE ENABLE...      ;JSD REV B
4025 034616 117702 144236      70:      MOV      @10,BPPCW     ; OK, CLEAR THE ENABLE...      ;JSD REV B
4026 034622 005000      MOV      @PPA,R2      ; ...READ DATA (SHOULD LOWER FLAGS).
4027 034624 117703 144234      CLR      R0
4028 034630 032703 000010      80:      MOV      @PPC,R3      ; READ STATUS.
4029 034634 001403      BIT      @ININT,R3
4030 034636 077006      BEQ      90      ; BR WHEN INT REQUEST CLEARS.
4031 034640 104423 034436      SOB      R0,80
4032      ERROR,   T146      ;; *** PORT A INT REQUEST DIDN'T CLEAR AFTER READ ***
4033 034644 012714 034002      90:      MOV      @BADAI,(R4)   ; RESET PORT A VECTOR.
4034
4035      ; *****
      ;.SBTTL      T147 -- TEST THAT B PRIORITY IS HIGHER THAN A
      ; *****
4036 034650 104420 000147      T147:    BEGINTEST, 147      ;; CHECK SWITCH REGISTER OPTIONS.
4037 034654 016704 144212      MOV      PPSV,R4      ; GET B VECTOR POINTER.
4038 034660 106464 000002      MTPS      2(R4)      ; RAISE CPU TO DEVICE LEVEL.
4039 034664 012714 034760      MOV      @30,(R4)      ; SET B VECTOR.
4040 034670 012764 034750 000004      MOV      @20,4(R4)     ; SET A VECTOR.
      ; MOV      @<2_1.>11,BPPCW ; ENABLE B...      ;JSD REV B

```

T147 -- TEST THAT B PRIORITY IS HIGHER THAN A

```

4041 034676 112777 000005 144162      MOVB    @5,BPPCW      ; ENABLE B...           ;JSD REV B
4042                                     MOVB    @<4_1.>!1,BPPCW ; ...AND A INTERRUPTS. ;JSD REV B
4043 034704 112777 000011 144154      MOVB    @11,BPPCW     ; ...AND A INTERRUPTS. ;JSD REV B
4044 034712 112777 000001 144142      MOVB    @1,BPPB       ; OUTPUT DATA...
4045 034720 005000                                     CLR     R0
4046 034722 132777 000040 144134 10:   BITB    @INRDY,BPPC   ; ...AND WAIT 'TIL DONE.
4047 034730 001001                                     BNE    .+4
4048 034732 077005                                     SOB    R0,10         ; KEEP-ALIVE.
4049
4050 034734 106427 000200                                     MTPS   @PRI4         ; LOWER CPU, B SHOULD COME IN FIRST.
4051 034740 000240                                     240
4052 034742 104423 034650                                     ERROR, T147         ;: *** NEITHER ONE INTERRUPTED -- WHAT THE HELL !! ***
4053 034746 000415                                     BR     @
4054 034750 022626                                     CMP    (SP)+,(SP)+   ; WE'RE HERE IF A CAME IN FIRST.
4055 034752 104423 034650                                     ERROR, T147         ;: *** A INTERRUPTED BEFORE B -- BAD NEWS !! ***
4056 034756 000411                                     BR     @
4057
4058 034760 022626                                     30:   CMP    (SP)+,(SP)+   ; WE'RE HERE IF B INTERRUPTED FIRST.
4059 034762 012764 035002 000004      MOV     @4,4(R4)      ; CHANGE THE A VECTOR...
4060 034770 106427 000200                                     MTPS   @PRI4         ; ...AND LET IT IN NOW.
4061 034774 000240                                     240
4062 034776 104423 034650                                     ERROR, T147         ;: *** A INT NOT RECEIVED AFTER B INT ***
4063
4064                                     ;40:   MOVB    @<2_1.>!0,BPPCW ; CLEAR THE ENABLES.           ;JSD REV B
4065 035002 112777 000004 144056 40:   MOVB    @4,BPPCW     ; CLEAR THE ENABLES.           ;JSD REV B
4066                                     ;     MOVB    @<4_1.>!0,BPPCW ;                               ;JSD REV B
4067 035010 112777 000010 144050      MOVB    @10,BPPCW    ;                               ;JSD REV B
4068 035016 112777 000001 144036      MOVB    @1,BPPB       ; LOWER REQUESTS.
4069 035024 117700 144030                                     MOVB   @PPA,R0
4070 035030 012714 033772                                     MOV     @BADSI,(R4)   ; RESET VECTORS.
4071 035034 012764 034002 000004      MOV     @BADAI,4(R4)
4072
4073 035042 012767 000006 142734 P10DUN: MOV    @ERRVEC+2,ERRVEC
4074 035050 004767 000002                                     CALL   ENDSEG
4075 035054 000410                                     BR     ENDPAS        ; CHECK FOR LOOP-IN-SEGMENT...
4076                                     .SBTTL              ; ...RETURN AND DO ENDPASS IF NOT.

```



END OF PASS ROUTINES.

```

4078
4079
4080
4081
4082
4083
4084
4085
4086
4087 035056 005767 000730
4088 035062 001001
4089 035064 000207
4090
4091 035066 104007
4092 035070 016700 001232
4093 035074 000402
4094 035076 012700 035256
4095 035102 104003
4096 035104 012700 035267
4097 035110 104003
4098 035112 005267 146664
4099 035116 026727 146656
4100 035122 000000
4101 035124 001002
4102 035126 005267 146652
4103 035132 016767 146642 177762
4104 035140 016700 146636
4105 035144 104424
4106 035146 012700 035303
4107 035152 104003
4108 035154 016700 146620
4109 035160 104424
4110 035162 104007
4111 035164 032767 010000 143004
4112 035172 001402
4113 035174 004767 000124
4114 035200 106427 000000
4115 035204 005000
4116 035206 012701 000004
4117 035212 077001
4118 035214 077102
4119
4120 035216 015700 000042
4121 035222 001405
4122 035224 000240
4123 035226 004710
4124 035230 000240 000240 000240
4125
4126 035236 016700 000550
4127 035242 001002
4128 035244 012700 010336
4129 035250 012706 003776
4130 035254 000110
4131
4132 035256 015 012 116
      035261 113 130 101
      035264 102 060 000
    
```

```

.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.
;
ENDSEG: TST INSEG ; LOOP-IN-SEGMENT ??
        BNE 10 ; YES.
        RETURN ; JUST RETURN IF NOT.

10: EMT+7 ; <CR LF>
    MOV SEGN,RO ; CURRENT SEGMENT DONE.
    SKP2
ENDPAS: MOV #EOP,RO ; ALL SEGMENTS DONE.
        EMT+3
        MOV #EOP1,RO
        EMT+3 ; END PASS.
        INC TPASS ; BUMP TOTAL PASS COUNT.
        CMP ERRCNT,(PC) ; ANY ERRORS THIS PASS ??
        0
        BNE 20 ; SKIP IF SO.
        INC GPASS ; NO, BUMP CONSECUTIVE-GOOD PASS COUNT.
        MOV ERRCNT,10 ; UPDATE SAVED COUNT.
        MOV TPASS,RO
        TRAP+24 ; PASS COUNT...
        MOV #EOP2,RO
        EMT+3
        MOV ERRCNT,RO
        TRAP+24 ; ...AND TOTAL ERROR COUNT.
        EMT+7 ; <CR LF>
        BIT #10000,SMR ; ERROR SUMMARY ??
        BEQ 30
        CALL ERRSUM ; PRINT IT IF SO.
        MTPS #PRIO ; LOWER THE CPU.
        CLR RO
        MOV #4,R1
        SOB RO,, ; ENDPASS DELAY...
        SOB R1,40 ; ...ABOUT 1 SECOND.
        ;
        MOV #M42,RO
        BEQ 50 ; BR IF NOT CHAINING.
        NOP ; OR RESET.
        CALL (RO) ; SYSMAC STYLE CHAIN RETURN.
        240,240,240
        ;
        MOV INSEG,RO ; GET ITERATION ADDRESS.
        BNE 60
        MOV #DOALL,RO ; IF NONE, BACK TO SQUARE 1.
        MOV #STACK,SP ; RESET STACK...
        JMP (RO) ; ...AND LOOP FOREVER.

EOP: .ASCIZ '<15><12>'MKXABO'
    
```

END OF PASS ROUTINES.

4133 035267 054 040 105 EOP1: .ASCIZ ', END PASS '

035272 116 104 040

035275 120 101 123

035300 123 040 000

4134 035303 054 040 124 EOPE: .ASCIZ ', TOTAL ERRORS '

035306 117 124 101

035311 114 040 105

035314 122 122 117

035317 122 123 040

035322 000

.EVEN

4135

4136

4137

4138

4139

4140

4141 035324 012702 004006

4142 035330 005712

4143 035332 001414

4144 035334 012700 035366

4145 035340 104003

4146 035342

4150 035342 104010

035344 012200

035346 104424

035350 104010

035352 012200

035354 104424

4151 035356 104007

4152 035360 005712

4153 035362 001367

4154 035364 000207

4155

4156 035366 015 012 011

035371 105 122 122

035374 120 103 011

035377 040 124 111

035402 115 105 123

035405 015 012 000

4157

.EVEN

\*\*\*\*\*

; ERROR SUMMARY REPORT.

; CALLED IN "END-PASS" IF SWR<12> IS SET.

ERRSUM: MOV @EPCTBL,R2

TST (R2)

BEG 2;

; BR IF NO ERROR DATA LOGGED.

MOV @ERHEAD,R0

ENT\*3

; HEADER.

1:

ENT\*10

:: <TAB>

MOV (R2)+,R0

TRAP\*24

:: PRINT (R2)+

ENT\*10

:: <TAB>

MOV (R2)+,R0

TRAP\*24

:: PRINT (R2)+

ENT\*7

; <CRLF>

TST (R2)

BNE 1;

; LOOP 'TIL DONE.

2:

RETURN

ERHEAD: .ASCIZ <15><12><11>'ERRPC'<11>' TIMES'<15><12>

END OF PASS ROUTINES.

4158  
 4159  
 4160  
 4161  
 4162  
 4163 035410 005767 143422  
 4164 035414 001020  
 4165 035416 012767 035442 142360  
 4166 035424 012700 174020  
 4167 035430 005010  
 4168 035432 000240  
 4169 035434 010067 143376  
 4170 035440 000403  
 4171  
 4172 035442 005067 143370  
 4173 035446 022626  
 4174 035450 012767 000006 142326  
 4175 035456 016700 143354  
 4176 035462 000207

```

;*****
; SUBROUTINE TO CHECK FOR Q-BUS EXERCISOR.
; RETURN QBX CSR1 ADDRESS IN R0 IF IT'S THERE.
;
QBXIN:  TST    QBXA      ; QBX INSTALLED ??
        BNE    30        ; BR IF SO.
        MOV    #10,ERRVEC ; DON'T KNOW YET, SET TRAP CATCHER...
        MOV    @QBXCSR,R0 ; ...AND A POINTER.
        CLR    (R0)      ; CLEAR QBX CSR1 (IF POSSIBLE)...
        240        ; ...TRAP TO 10 IF NOT.
        MOV    R0,QBXA   ; IT'S THERE, SET FLAG = CSR ADDRESS...
        BR     20        ; ...AND RETURN.
;
10:     CLR    QBXA      ; NOT THERE, CLEAR FLAG.
        CMP    (SP)+,(SP)+ ; FIX STACK.
        MOV    @ERRVEC+2,ERRVEC ; RESET ERROR VECTOR.
        MOV    QBX,R0    ; RETURN CSR POINTER OR ZERO.
        30:
        RETURN

```

TRAP HANDLER

```

4178
4179
4180
4181
4182 035464 010046
4183 035466 010146
4184 035470 016600 000004
4185 035474 116000 177776
4186 035500 010001
4187 035502 162701 000017
4188 035506 100001
4189 035510 005001
4190 035512 006301
4191 035514 016107 035520
4192
4193 035520 035534
4194 035522 035564
4195 035524 035606
4196 035526 035616
4197 035530 036014
4198 035532 036602
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211 035534 016601 000006
4212 035540 042701 177760
4213 035544 020001
4214 035546 001003
4215 035550 062766 000004 000004
4216 035556 012601
4217 035560 012600
4218 035562 000002
4219
4220
4221
4222
4223
4224 035564 104421
4225 035566 012601
4226 035570 012600
4227 035572 017667 000000 000524
4228
4229 035600 062716 000002
4230 035604 000002

```

```

.SBTTL TRAP HANDLER
;*****
; TRAP HANDLER
;
TRAPHAN: MOV     R0,-(SP)      ; SAVE CALLERS RO...
          MOV     R1,-(SP)      ; ...AND R1.
          MOV     4(SP),R0      ; GET TRAP PC.
          MOVB    -2(R0),R0     ; GET TRAP CODE.
          MOV     R0,R1        ; WORKING COPY.
          SUB     #17,R1
          BPL     10           ; BR IF TRAP+20 OR HIGHER.
          CLR     R1           ; TRAP+0 THRU TRAP+17 ARE COMMON.
10:      ASL     R1           ; SHIFT UP TO WORD OFFSET...
          MOV     20(R1),PC     ; ...AND DISPATCH THERE.

20:      CCBIT                    ; TRAP0 - TRAP17 = CC BIT CHECK.
          BGNTST                   ; TRAP20 = BEGIN NEW TEST.
          CHKSMR                   ; TRAP21 = CHECK SMR.
          NEWSMR                   ; TRAP22 = GET NEW SMR.
          LOGERR                   ; TRAP23 = LOG AND REPORT ERROR.
          ITOA                    ; TRAP24 = CONVERT INTEGER TO ASCII AND PRINT.
          ;
          ; ADD MORE AS NEEDED.
          ;
;*****
; CONDITION CODE BIT CHECKER.
; CALL: TRAP+N, WHERE N IS THE CONDITION CODE TO EXPECT (0 - 17).
;
; PSW TO EXAMINE IS AT 6(SP), N IS IN R0.
; RETURN TO CALL+2 IF CC N.E. N
; RETURN TO CALL+6 OTHERWISE.
;
CCBIT:   MOV     6(SP),R1        ; GET PSW TO CHECK.
          BIC     #17,R1        ; KEEP CC BITS ONLY.
          CMP     R0,R1
          BNE     10           ; BR IF CC BITS INCORRECT.
10:      ADD     #4,4(SP)       ; OK, ADJUST RETURN PC.
          MOV     (SP)+,R1
          MOV     (SP)+,R0
          RTI

;*****
; BEGIN TEST ROUTINE.
; CALL VIA TRAP+20
;
BGNTST:  CHECKSMR
          MOV     (SP)+,R1      ; RESTORE CALLERS REGISTERS.
          MOV     (SP)+,R0
          MOV     8(SP),TSTNUM ; SAVE TEST NUMBER...
          ; ...FOR THE ERROR HANDLER...
          ADD     #2,(SP)      ; ...BUMP RETURN PC...
          RTI                 ; ...AND RETURN.

```

TRAP HANDLER

```

4232
4233
4234
4235
4236
4237 035606 104005
4238 035610 120027 000007
4239 035614 001024
4240
4241 035616 005737 000042
4242 035622 001021
4243 035624 012700 035674
4244 035630 104003
4245 035632 016700 142340
4246 035636 104424
4247 035640 012700 035705
4248 035644 104003
4249 035646 104000
4250 035650 104011
4251 035652 000405
4252 035654 120067 142316
4253 035660 001016
4254 035662 010067 142310
4255 035666 012601
4256 035670 012600
4257 035672 000002
4258
4259 035674 015 012 123
035677 127 122 040
035702 075 040 000
4260 035705 040 040 116
035710 103 127 040
035713 075 040 000

4261
4262
4263
4264
4265 035716 010067 142254
4266 035722 042700 177770
4267 035726 006300
4268 035730 016067 035772 000054
4269 035736 012700 004000
4270 035742 012701 002000
4271 035746 005020
4272 035750 077102
4273 035752 016700 000034
4274 035756 001002
4275 035760 012700 010336
4276 035764 012706 003776
4277 035770 000110
4278 035772 000000
4279 035774 010336 024030 027052
4280 036002 027746 030112 030142
4281 036010 033640
4282 036012 000000
4283

```

```

;*****
; SWITCH REGISTER HANDLER.
; CALLED VIA TRAP-21 (CHECKSMR) AND TRAP-22 (GETSMR).
;
CHKSMR: .ENABL  LSB
        EMT-3
        CHPB   RO,07
        BNE    10
; CHECK KEYBOARD ( => RO ).
; <FG> ??
; JUST RETURN IF NOT.

NEWSMR: TST    0042
        BNE    10
        MOV    @OSMR,RO
; CHAINING ??
; DON'T IF SO.
        EMT-3
; OLD SMR...
        MOV    SMR,RO
        TRAP-24
; ...= THIS.
        MOV    @NSMR,RO
        EMT-3
; NEW SMR ...
        EMT-0
; ...AND WAIT FOR INPUT.
        EMT-11
; PARSE OCTAL NUMBER.
        BR     10
; JUST EXIT IF NO CHANGE.
        CHPB   RO,SMR
; NEW SEGMENT SELECTED ??
        BNE    NEWSSEG
; BR IF SO.
        MOV    RO,SMR
; OTHERWISE, SAVE NEW SMR...
        MOV    (SP)+,R1
; ...RESTORE...
        MOV    (SP)+,RO
        RTI
; ...AND RETURN TO CALLER.

OSMR:   .DSABL  LSB
        .ASCIZ <15><12>'SMR = '

NSMR:   .ASCIZ ' NEW = '

        .EVEN

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

```



TRAP HANDLER

```

036206 104010      EMT+10      ; ; <TAB>
036210 010400      MOV          R4,R0      ; ; PRINT R4
036212 104424      TRAP+24      ; ; <TAB>
036214 104010      EMT+10      ; ; <TAB>
036216 010500      MOV          R5,R0      ; ; PRINT R5
036220 104424      TRAP+24      ; ; <TAB>
036222 104010      EMT+10      ; ; <TAB>
036224 016700 000072 MOV          ESP,R0      ; ; PRINT ESP
036230 104424      TRAP+24      ; ; <TAB>
036232 104010      EMT+10      ; ; <TAB>
4330 036234 104007      EMT+7        ; ; <CRLF>
4331 036236 012601      10: MOV          (SP),R1      ; ; RESTORE REGISTERS.
4332 036240 012600      MOV          (SP),R0
4333 036242 005767 141730 TST          SMR          ; ; HALT-ON-ERROR ( SMR<15> ) ??
4334 036246 100012      BPL          20          ; ; BR IF NOT.
4335 036250 013767 177564 000042 MOV          @0177564,SAVTTY ; ; YES, SAVE CONSOLE TTY STATUS...
4336 036256 042737 000004 177564 BIC          @4,@0177564 ; ; ...AND INSURE "MAINT" IS NOT SET.
4337 036264 000000      EHALT        ; ; ERROR HALT.
4338 036266 016737 000026 177564 MOV          SAVTTY,@0177564 ; ; ON PROCEED, RESTORE TTY STATUS.
4339 036274 032767 001000 141674 20: BIT          @1000,SMR      ; ; LOOP-ON-ERROR ( SMR<9> ) ??
4340 036302 001003      BNE          30          ; ; BR IF SO.
4341 036304 062716 000002      ADD          @2,(SP)      ; ; NO, ADJUST RETURN PC...
4342 036310 000002      RTI         ; ; ...AND RETURN TO CALLER.
4343 036312 017616 000000      30: MOV          @0(SP),@0(SP) ; ; REPLACE RETURN PC WITH LOOP ADDRESS...
4344 036316 000002      RTI         ; ; ...AND DO IT.
4345
4346 036320 000000      SAVTTY: 0
4347 036322 000000      ESP: 0
4348 036324 000000      TSTNUM: 0
4349 036326 036330      SEGN: SEG1
4350 036330 124 061 061      SEGN: SEG1 ; POINTS TO ONE OF THE FOLLOWING:
036333 040 111 116      SEGN: .ASCIZ 'T11 INSTRUCTION'
036336 123 124 122
036341 125 103 124
036344 111 117 116
036347 000
4351 036350 124 061 061      SEG2: .ASCIZ 'T11 TRAP/INTERRUPT'
036353 040 124 122
036356 101 120 057
036361 111 116 124
036364 105 122 122
036367 125 120 124
036372 000
4352 036373 114 117 103      SEG3: .ASCIZ 'LOCAL RAM'
036376 101 114 040
036401 122 101 115
036404 000
4353 036405 114 117 103      SEG4: .ASCIZ 'LOCAL ROM'
036410 101 114 040
036413 122 117 115
036416 000
4354 036417 123 105 122      SEG5: .ASCIZ 'SERIAL LINE 1'
036422 111 101 114
036425 040 114 111
036430 116 105 040
036433 061 000
4355 036435 123 105 122      SEG6: .ASCIZ 'SERIAL LINE 2'

```

TRAP HANDLER

	036440	111	101	114	
	036443	040	114	111	
	036446	116	105	040	
	036451	062	000		
4356	036453	120	101	122	SEG7: .ASCIZ 'PARALLEL I/O PORT'
	036456	101	114	114	
	036461	105	114	040	
	036464	111	057	117	
	036467	040	120	117	
	036472	122	124	000	
4357	036475	040	106	101	FINTST: .ASCIZ ' FAILS IN TEST '
	036500	111	114	123	
	036503	040	111	116	
	036506	040	124	105	
	036511	123	124	040	
	036514	000			
4358	036515	015	012	040	HDR: .ASCII <15><12>' PC+2'<11>
	036520	120	103	053	
	036523	062	011		
4359	036525	040	040	120	.ASCII ' PSW'<11>
	036530	123	127	011	
4360	036533	040	040	122	.ASCII ' R0'<11>
	036536	060	011		
4361	036540	040	040	122	.ASCII ' R1'<11>
	036543	061	011		
4362	036545	040	040	122	.ASCII ' R2'<11>
	036550	062	011		
4363	036552	040	040	122	.ASCII ' R3'<11>
	036555	063	011		
4364	036557	040	040	122	.ASCII ' R4'<11>
	036562	064	011		
4365	036564	040	040	122	.ASCII ' R5'<11>
	036567	065	011		
4366	036571	122	066	050	.ASCIZ 'R6(SP)'<15><12>
	036574	123	120	051	
	036577	015	012	000	
4367					.EVEN



TRAP HANDLER

```

4369
4370
4371
4372
4373
4374
4375
4376
4377 036602 016600 000002
4378 036606 012701 036640
4379 036612 005011
4380 036614 104030
4381 036616 005767 000016
4382 036622 001403
4383 036624 012700 036640
4384 036630 104003
4385 036632 012601
4386 036634 012600
4387 036636 000002
4388
4389 036640      061      062      063
      036643      064      065      066
      036646      000
4390
4391
4392 036650 036650
4393
4394      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 XXDP+ 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              ; CONVERT I TO A.
      TST     24
      BEQ     14          ; IF V1.0, IT'S ALREADY BEEN PRINTED.
      MOV     @2,R0        ; IF V1.1, PRINT IT NOW.
      EMT+3
14:   MOV     (SP)+,R1     ; RESTORE...
      MOV     (SP)+,R0
      RTI                ;...AND RETURN.
24:   .ASCIZ  /123456/
      .EVEN
#LSTAD: .
      .END BEGIN

```

## SYMBOL TABLE

ADC80	012536	EHALT	= 000000	MODE0	015224	GBXCSR	= 174020	SLU2P	= 000240
ADC81	020146	EMTRP	010140	MODE1	015270	GBXIN	035410	SLU2V	= 000120
ADCO	014142	EMTVEC	= 000030	MODE2	015364	GBXVEC	= 000110	S0B1	023354
ADC1	022122	ENDPAS	035076	MODE3	015506	RAMAD1	= 160010	SPD0	= 001016
ADDO	014656	ENDSEG	035056	MODE4	015664	RAMAD2	= 100000	SPD1	= 001020
ADD1	022672	EOP	035256	MODE5	016112	RAMS1	= 003764	SPD2	= 001022
ADR	= 001000	EOPE	035303	MODE6	016322	RAMS2	= 024000	SPD3	= 001024
ADR1	= 001002	EOP1	035267	MODE7	016456	RBUF	= 001046	SPD4	= 001026
ADR2	= 001004	EPCTBL	= 004006	MOV0	013010	RCSR	= 001044	SPD5	= 001030
ALLCC	010604	ERHEAD	035366	MOV1	020430	RCVACT	= 004000	SPD6	= 001032
ASL80	012364	ERRCNT	= 004000	NBIT	010434	RCVBRK	= 004000	SPD7	= 001034
ASL81	017760	ERROR	= 104423	NEGB0	012144	RCVDUN	= 000200	STACK	= 003776
ASLO	013770	ERRSUM	035324	NEGB1	017524	RCVERR	= 100000	START	= 172000
ASL1	021734	ERRVEC	= 000004	NEGO	013550	RCVIE	= 000100	SUB0	015004
ASR80	012436	ESP	036322	NEG1	021464	RCVRS	032140	SUB1	023156
ASR81	020036	FINTST	036475	NEWSEG	035716	REGS	010630	SWAB0	014466
ASR0	014042	FLT0	030352	NEWSMR	035616	RESRVD	= 000010	SWAB1	022462
ASR1	022012	FLT1	030342	NOBIT	010374	RESTAR	= 172004	SMR	= 000176
BADAI	034002	FPLUS	027062	NOP1	= 000260	RESVEC	= 000010	SXT0	014414
BADBI	033772	FRERR	= 020000	NSMR	035705	ROL80	012224	SXT1	022404
BADPIO	033762	GETSMR	= 104422	ODT	= 172000	ROL81	017610	SYNC	030340
BADRI	030320	GPASS	= 004004	ODTBRK	= 170000	ROL0	013630	S1	010336
BADTI	030330	HDR	036515	ODTIN	= 000140	ROL1	021560	S2	024030
BADU1	030300	HELLO	010300	ODTRE	= 172004	ROMAD1	= 170000	S3	027052
BADU2	030310	INCB0	011714	ORERR	= 040000	ROMAD2	= 164000	S4	027746
BEGIN	010000	INCB1	017214	OSMR	035674	ROMS1	= 002000	S5	030112
BEGINT	= 104420	INCO	013320	OUTENA	= 000004	ROMS2	= 004000	S6	030142
BEVNT	= 000100	INC1	021212	OUTINT	= 000001	ROR80	012302	S7	033640
BGNTST	035564	INENA	= 000020	OUTRDY	= 000002	ROR81	017672	TEMP	= 001006
BHALT	= 000140	INDINT	= 000010	PI0A	= 176200	ROR0	013706	TEMP1	= 001010
BIC80	011610	INIT	010000	PI0AV	= 000134	ROR1	021646	TEMP2	= 001012
BIC81	017006	INIT1	010072	PI0B	= 176202	RVEC	= 001054	TEMP3	= 001014
BICO	013162	INDY	= 000040	PI0BV	= 000130	R6	= 0000006	TN6	= 000147
BIC1	020744	INSEG	036012	PI0C	= 176204	R6TST	= 024042	TPASS	= 004002
BPTVEC	= 000014	IOTVEC	= 000020	PI0DUN	035042	R7	= 0000007	TPSM	015126
BRANCH	010350	IT0A	036602	PI0NC	= 176206	SAVTTY	036320	TPSM1	023454
BTMRD	023572	JMP1	010712	PI0P	= 000240	SBC80	012650	TRAP4	035464
CBIT	010514	JMP2	010752	PI0TST	034012	SBC81	020264	TRAP4	= 167776
CCBIT	035534	JMP3	011006	PNTERR	036062	SBC0	014254	TRPVEC	= 000034
CHECKC	= 104400	JMP4	011056	PPA	= 001060	SBC1	022240	TRTVEC	= 000014
CHECKS	= 104421	JMP5	011112	PPAV	= 001070	SEGN	036326	TST80	011422
CHKSMR	035606	JMP6	011162	PPB	= 001062	SEG1	036330	TST81	016604
CLNZ	= 000254	JMP7	011234	PPBV	= 001072	SEG2	036350	TSTNUM	036324
CMP80	011476	JSRTST	011322	PPC	= 001064	SEG3	036373	T1	010344
CMP81	016722	KXT11	= 000004	PPCW	= 001066	SEG4	036405	T10	011316
CMP0	013104	K0	= 001016	PPH0	= 000233	SEG5	036417	T100	025646
CMP1	020516	K1	= 001020	PPH1	= 000264	SEG6	036435	T101	025764
COM80	012052	K2	= 001022	PRI0	= 000000	SEG7	036453	T102	026102
COM81	017414	K3	= 001024	PRI1	= 000040	SENV	= 000273	T103	026276
COM0	013456	K4	= 001026	PRI2	= 000100	SEVC	= 000263	T104	026376
COM1	021362	K5	= 001030	PRI3	= 000140	SKP1	= 000401	T105	026460
DFPBR	= 001040	K6	= 001032	PRI4	= 000200	SKP2	= 000402	T106	026526
DFSPD	= 001042	K7	= 001034	PRI5	= 000240	SKP3	= 000403	T107	026614
DLDUN	033600	LEDBIT	= 000200	PRI6	= 000300	SLU1	= 177360	T11	011416
DLSET	030170	LOGEND	= 007776	PRI7	= 000340	SLU1P	= 000200	T110	026742
DLTSTS	030364	LOGERR	036014	PRVEC	= 000024	SLU1V	= 000060	T111	027200
DOALL	010336	MFPT	= 000007	GBXA	= 001036	SLU2	= 176540	T112	027316

SYMBOL TABLE

T113	027506	T136	032774	T25	013452	T47	017410	T70	024210
T114	030012	T137	033122	T26	013624	T5	010746	T71	024466
T115	030170	T14	011710	T27	013764	T50	017604	T72	024652
T116	030364	T140	033316	T3	010624	T51	017754	T73	024750
T117	030536	T141	033432	T30	014136	T52	020142	T74	025122
T12	011472	T142	033646	T31	014410	T53	020424	T75	025240
T120	030652	T143	034012	T32	014652	T54	020512	T76	025356
T121	030710	T144	034046	T33	015122	T55	020740	T77	025474
T122	031002	T145	034234	T34	015220	T56	021206	VBIT	010462
T123	031064	T146	034436	T35	015360	T57	021356	XBUF	= 001052
T124	031146	T147	034650	T36	015302	T6	011052	XCSR	= 001050
T125	031230	T15	012046	T37	015660	T60	021554	XPTBRK	= 000001
T126	031312	T16	012220	T4	010706	T61	021730	XPTIE	= 000100
T127	031374	T17	012360	T40	016106	T62	022116	XPTRDY	= 000200
T13	011604	T2	010600	T41	016316	T63	022400	XPTRS	030652
T130	031466	T20	012532	T42	016452	T64	022666	XORO	014552
T131	031560	T21	013004	T43	016600	T65	023450	XOR1	022554
T132	031734	T22	013100	T44	016716	T66	023566	XVEC	= 001056
T133	032140	T23	013156	T45	017002	T67	024036	ZBIT	010544
T134	032222	T24	013314	T46	017210	T7	011156	ILSTAD	036650
T135	032434								

. ABS. 036652 000  
000000 001

ERRORS DETECTED: 3

VIRTUAL MEMORY USED: 8608 WORDS ( 34 PAGES)

DYNAMIC MEMORY: 20060 WORDS ( 77 PAGES)

ELAPSED TIME: 00:04:28

CNKXAB.BIC,CNKXAB.LST/-SP-CNKXAB.P11

•